

Final Documentation

Project 2: MATLAB to Python Application Translation
Version 1.2

Team members:

Garrett Cook

Lucas Towers

Jasmine Mishra

Jose Pena Revelo

Contents

Introduction	3
Scope & Charter	3
1. Identification	3
2. Software Description	3
3. Objectives	4
4. Stakeholder List	5
5. Scope	5
6. Milestones	5
7. Requirements	6
7.1. Functional Requirements	6
7.2. Non-Functional Requirements	7
7.3. Technical Requirements	7
7.4. User Requirements	7
7.5 Additional Requirements	8
8. Assumptions	8
9. High-Level Risks	9
Design	9
1. User Groups	9
10.1. Usage scenarios	9
10.2. Use Case Diagrams	11
2. System Architecture	12
3. Diagrams	13
3.1 Data Flow Diagrams	13
3.2. Sequence Diagrams	17
4. Technical Specification	22
5. Test Plan	22
Project Management	25
Work Breakdown Structure	25
Burnup and burndown	29

Features/Work packages completed	30
Software Implementation Status	33
Test Report	33
Unit Tests	33
Usability Testing Results	35
Requirements Delivered	38
Known Bugs	40
Lessons Learned	41
Deployment	41
Repository	41
Finding the code	42
User Manual	43
Installation	43
Workflow, testing, and deployment environments	44
Appendix A - Record of Changes	47

Introduction

There are three MATLAB tools created by the National Research Council's National Institute for Nanotechnology which are used to analyze and calibrate electron microscope data: qEELS, NanoMi Optics, and Tomography Alignment Software. The goal of this project is to translate the MATLAB software to Python and maintain the same functionality.

Scope & Charter

1. Identification

Project Name:

NRC Electron Microscope Tools

Sponsor Details:

Misa Hayashida, Research Officer

National Research Council's National Institute for Nanotechnology at the University of Alberta

2. Software Description

There are three MATLAB tools created by the NRC laboratory used to analyze and calibrate electron microscope data. The client wants to switch the software to Python because it is more portable, user friendly, and readable, among various other advantages. Descriptions of the three software that will be created in Python are as follows:

qEELS software calibrates energy loss axis (eV/pixel) and transfer axis (microrad/pixel) by fitting peaks of surface plasmon and bulk plasmon in q-EELS patterns. qEELS stands for momentum transfer electron energy loss spectroscopy. Users will load an image of a q-EELS pattern, edit the contrast of the image, indicate points on the image, edit settings, and request calculations.

NanoMi Optics allows users to plan, visualize, and optimize NanoMi transmission electron microscope settings. The user can adjust each lens setting and view the effect of the adjustments on the electron beams and magnification. The electron microscope settings are displayed in a diagram and are downloadable in csv format. The settings calculated using the software are intended for later use on an actual NanoMi transmission electron microscope.

Tomography Alignment Software is used to find and track particles in three dimensions as they spin around a holder that is rotated in space at set increments.

In the Tomography Alignment software, the user will be able to:

1. Load images.
2. Contrast adjustment across all images.
3. Filter, binning, bulk image shift and bulk image rotation.
4. Inter-image alignment via cross-correlation to maintain stability.
5. Particle detection.
6. Automatic particle tracking with manual adjustment.
7. Track optimization and tomography.
8. Save results.

3. Objectives

- Understand and document the legacy MATLAB implementations.
- Translate MATLAB features to Python.
- Maintain functionality and outputs from the legacy project in new software.
- Design a familiar user interface similar to the legacy implementation.
- Create straightforward manuals/documentation that is understandable for non-programmers.
- Expose our client to formal software engineering practices.
- Test and verify.
- Software will run as fast or faster than the legacy implementation.

4. Stakeholder List

- Misa Hayashida
- National Research Council
- University of Alberta
- University of British Columbia Okanagan
- Project Team
- Potential future users

5. Scope

In scope

- Creating the tools
- Use input/data provided by the client in our tools
- Testing
- Documentation

Out of Scope

- Using an actual electron microscope

6. Milestones

1. Charter, Scope, and Requirements document - 5 June
2. Minimum Viable Product Presentation - 6 July
3. Final Prototype and Presentation - 17 August
4. Port the software
 - a. Port qEELS
 - i. GUI, backend, tests
 - b. Port NanoMi optics
 - i. GUI, backend, tests
 - c. Port Tomography Alignment Software
 - i. GUI, backend, tests
5. Document software

7. Requirements

7.1. Functional Requirements

- New software will function the same as the MATLAB with acceptable numerical outputs.
- qEELS
 - Software will accept q-EELS pattern image and feature locations.
 - Software will adjust contrast on the q-EELS pattern image.
 - Software will detect fitted peaks of bulk and surface plasmon.
 - Bulk plasmon, upper surface plasmons, and lower surface plasmons.
 - Software will calculate the calibrated energy loss axis and transfer axis.
 - Software will save results.
- NanoMi optics
 - Software will accept desired parameters from the user.
 - Software will calculate and display settings in a diagram of a NanoMi electron transmission microscope.
 - Software will calculate optimized optics settings.
 - Software will save results.
- Tomography Alignment Software
 - Software will accept many images.
 - Software will adjust contrast on images.
 - Software will transform images.
 - Software will perform inter-image alignment via cross correlation to maintain stability.
 - Software will automatically determine the location of selected particles across images.
 - Software will accept manual adjustments to location of particles.
 - Software will optimize and align the sequence of images.
 - Software will save results.

7.2. Non-Functional Requirements

- Performance
 - Portable and ability to run on multiple operating systems
- Development
 - Deliver by early August
 - Cannot use paid resources
 - Use Kanban workflow using GitHub Projects
 - Use git workflow with GitHub
- Quality
 - Code must be linted and in a consistent format

7.3. Technical Requirements

- Legacy software must be run and analyzed
- MATLAB dependencies must be replaced
- New software must be written in Python
- New software must use well-supported Python packages
- New software must be thoroughly tested
- The GUI will be separated from backend
- Avoid GPL and maintain correct licensing

7.4. User Requirements

- Users will have the ability to learn how to use software quickly from documentation.
- Users who used the legacy software can pick up new software easily.
- qEELS
 - User will load a q-EELS pattern image into the software.
 - User will adjust the contrast of the image.
 - User will indicate the location of bulk plasmon, upper surface plasmons, and lower surface plasmons on the image.
 - User will indicate that the software will detect fitted peaks of surface plasmon and bulk plasmon.
 - User will indicate the material.
 - User will request calibrated energy loss axis and transfer axis from software.
 - User will save results on their machine

- NanoMi optics
 - User inputs desired lens parameters for optics.
 - User will request optimized lens settings from the software.
 - User will view results, zoom in, zoom out, and change views on the diagram.
 - User will save results on their machine
- Tomography Alignment Software
 - User will load images into the software.
 - User will perform automatic and/or manual contrast adjustment across all images.
 - User will perform filter, binning, bulk image shift and bulk image rotation.
 - User will request coarse alignment.
 - User will indicate the location of particles in images for tracking.
 - User will manually adjust particle tracking.
 - User will request optimized aligned sequence images from the software.
 - User will view saved results.

7.5 Additional Requirements

Requirements that were added during development:

- qEELS - contrast adjustment
- qEELS - material selection
- NanoMi Optics - change upper lens mode

8. Assumptions

- Supplied with the legacy scripts
- The supplied legacy script works as intended to begin with

9. High-Level Risks

- Developers get sick/travel
- Client is unavailable
- Poor team synergy
- Bad estimation of task complexities
- Poor team/client communication
- Understanding MATLAB
- Not completing the project, leading to failure to create a product
- Time constraints
- Issues with current code base

Design

1. User Groups

The user group for this project are the National Research Council nanotechnology researchers at the University of Alberta. They are very knowledgeable about nanotechnology science, but they are not experts at software engineering. The users have experience with the three softwares in MATLAB, so they are familiar with the layout of the MATLAB software GUI and usage.

10.1. Usage scenarios

Name:	qEELS
Description:	Calculate calibrated energy loss from a q-EELS pattern image.
Flow of Events:	<ol style="list-style-type: none">1. The user opens the software.2. The user uploads the q-EELS pattern image.3. The user indicates the location of features on the q-EELS pattern image.4. The user indicates that the software will detect fitted peaks of surface plasmon and bulk plasmon.5. The user requests the calibrated energy loss axis and

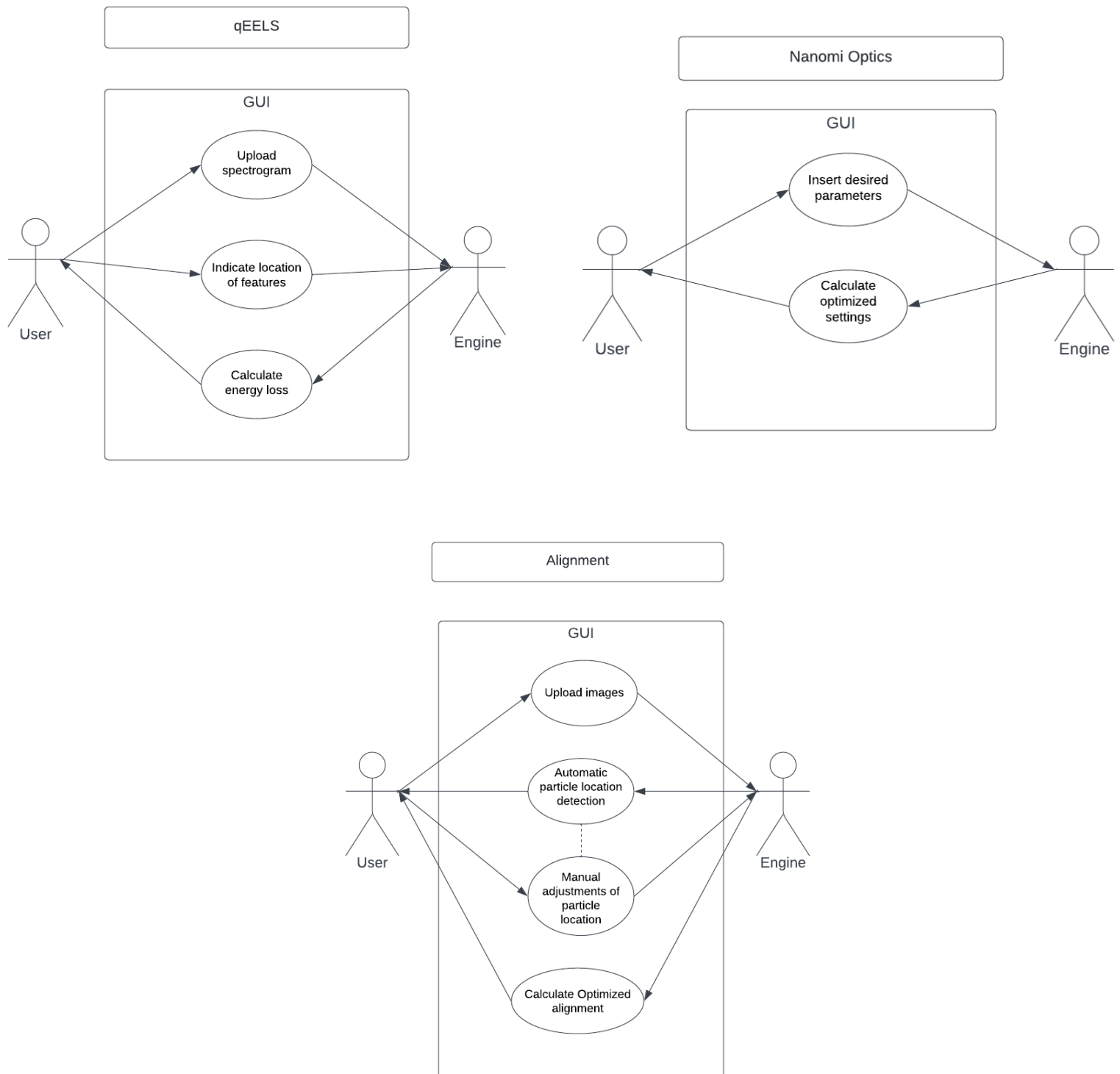
	transfer axis.
Pre-conditions:	<ul style="list-style-type: none"> • The user must have the software open. • The user must have an appropriate q-EELS pattern image. • The user has a knowledge of q-EELS pattern images.
Post-Conditions:	The energy loss axis and transfer axis are calculated and displayed in an image.

Name:	NanoMi Optics
Description:	Optimize electron microscope optics settings.
Flow of Events:	<ol style="list-style-type: none"> 1. The user opens the software. 2. User inputs their desired parameters for optics 3. User request optimized lens settings 4. User views results in the diagram.
Pre-conditions:	<ul style="list-style-type: none"> • The user must have the software open. • The user has knowledge of electron microscope settings.
Post-Conditions:	Optimized settings are displayed to the user.

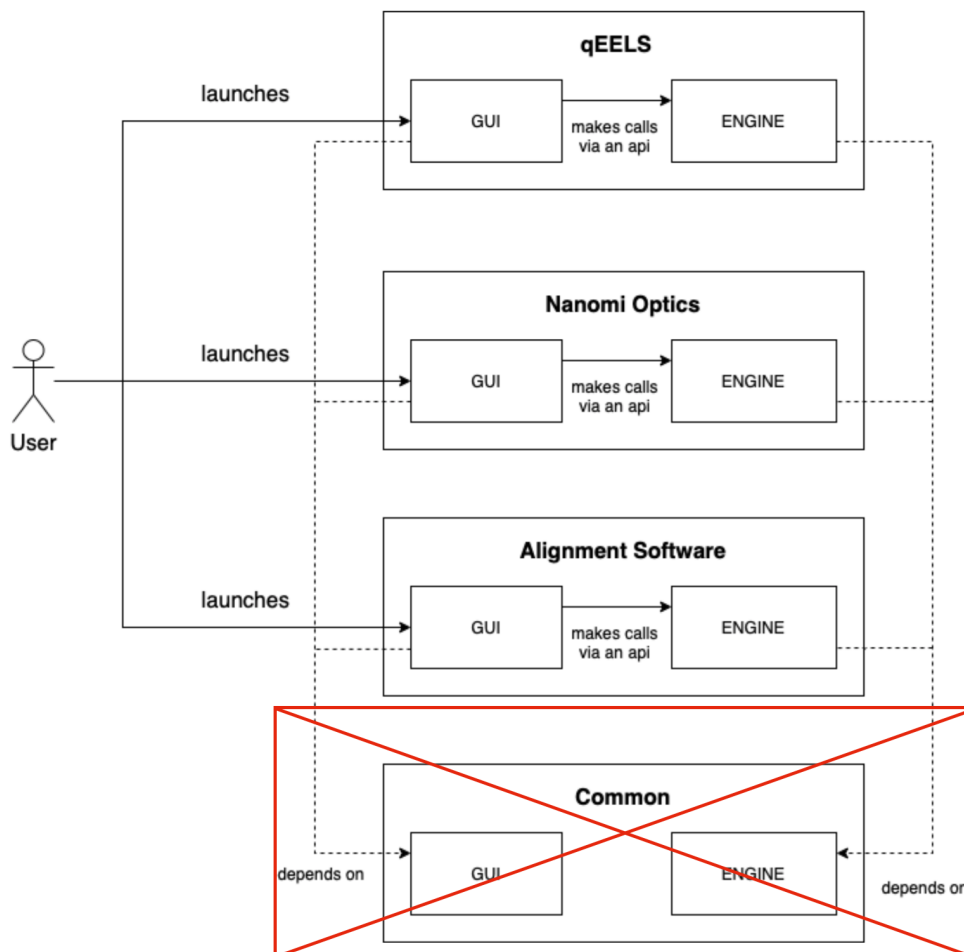
Name:	Tomography Alignment Software
Description	Align nanoparticle images.
Flow of Events:	<ol style="list-style-type: none"> 1. The user opens the software. 2. The user uploads nanoparticle images. 3. The user selects the location of the particle on the image. 4. The user requests aligned sequence images from software.
Pre-conditions:	<ul style="list-style-type: none"> • The user must have the software open. • The user has appropriate nanoparticle images. • The user has knowledge of nanoparticle images.
Post-Conditions:	An aligned sequence of images is optimized and calculated.

10.2. Use Case Diagrams

In all three pieces of software, the user will interact via GUI. The calculations and/or optimization will occur in the engine and report results back to the user in the GUI, by graph, image, or numerical values.



2. System Architecture



Each of the three software being converted will have its own top-level package. Code from specific software should not import functionality from another software.

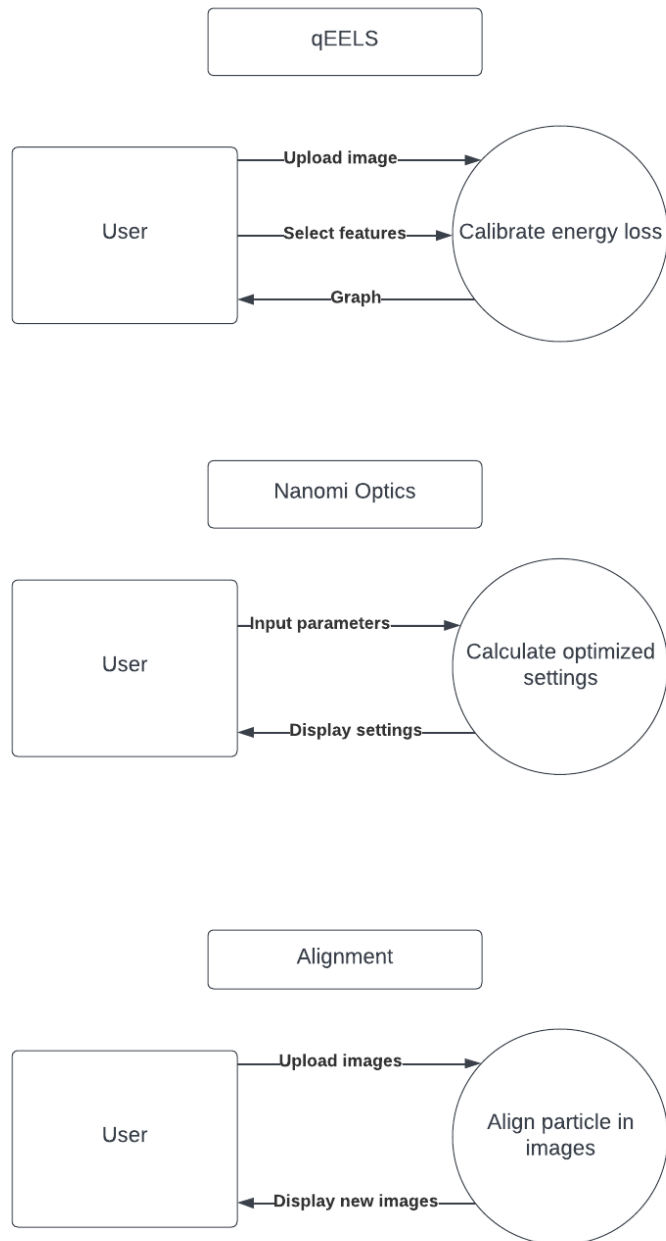
Each **engine** should include all of the core logic and processing behind the software. The engine packages must be thoroughly tested. The engine packages must also have zero dependencies on the *gui* packages or gui libraries. Each **gui** package should not include any data processing and should instead delegate all of that in calls to its respective engine. Each type of call made from a *gui* to an *engine* package should be tested. The *gui* should focus only on handling user interaction.

The common package was planned to hold common logic used between the software, but was removed during development to make each software completely independent per client's request.

3. Diagrams

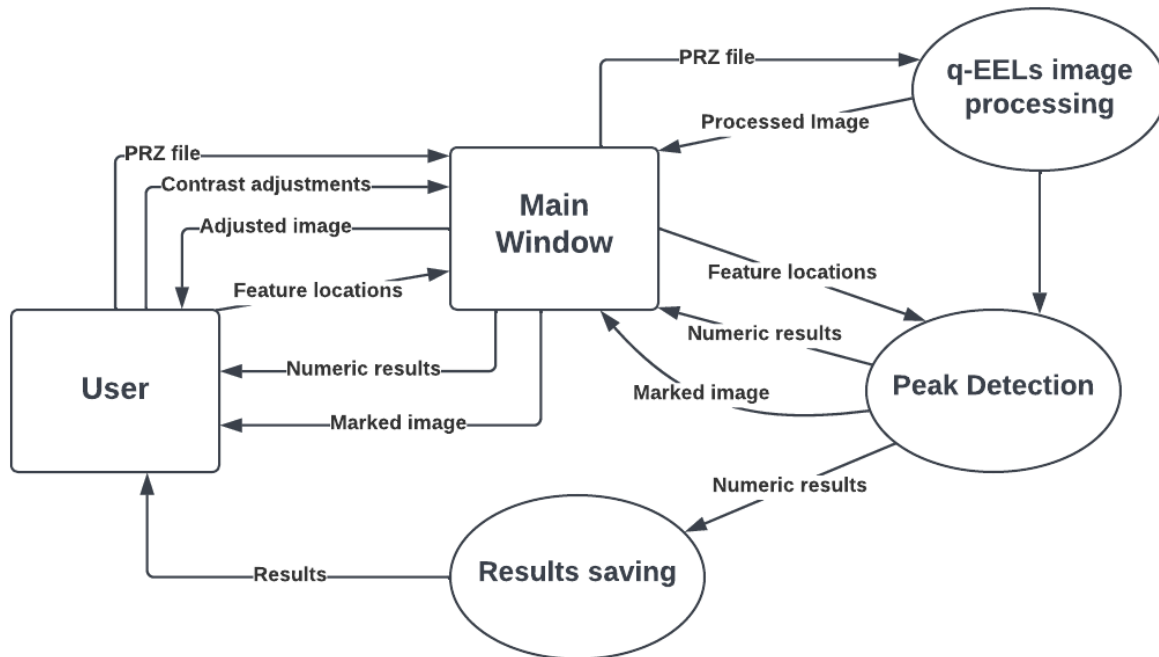
3.1 Data Flow Diagrams

Level 0 Data Flow Diagrams (High-Level):



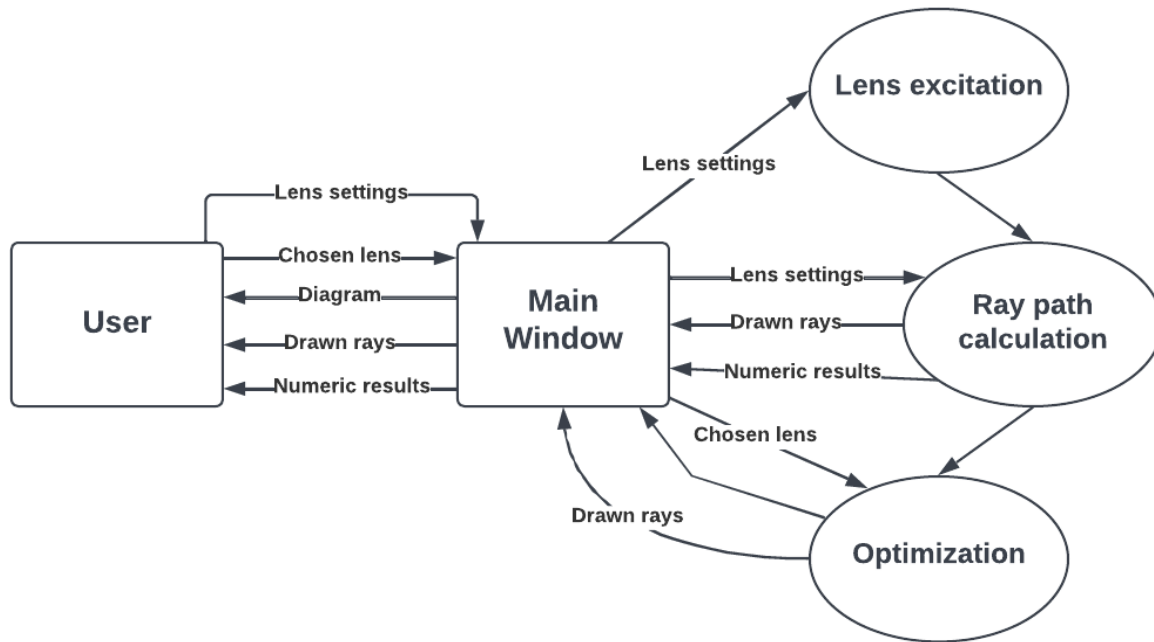
These high level diagrams show the basic interactions of data flow through each software.

Level 1 Data Flow Diagrams:
qEELS:



This diagram shows how data flows through the qEELS software. The user interacts mainly with the main window, uploading, inputting, and receiving data. The inputted data from the main window flows to the three main processes of the software which are q-EELS image processing, peak detection, and results saving. The uploaded PRZ file from the user is processed and displayed back to the user. The inputted feature locations are used for peak detection and the results/marked images are displayed back to the user. The results saving uses the results from peak detection and saves them on the user's device.

NanoMi Optics:



This diagram shows how data flows through the NanoMi Optics software. The user interacts mainly with the main window by changing the settings for the lenses. This lens setting data from the main window flows to the three main processes of the software which are lens excitation, ray path calculation, and optimization. Lens excitation uses the lens settings data in a calculation for finding C_f and U_R values (types of focal lengths) of the lens. The ray path calculation uses the lens setting data to calculate where the rays will be drawn on the diagram and numerical results about all of the lenses will be displayed in the main window to the user. The optimization uses the chosen lens by the user to optimize the path of the rays and draws them on the diagram for the user.

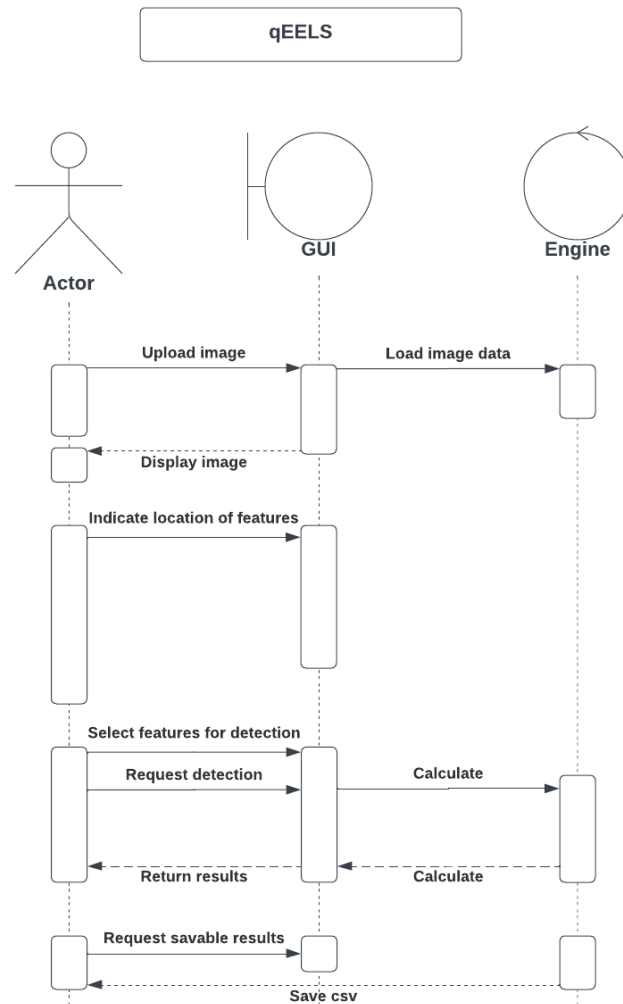
Tomography Alignment Software:



This diagram shows how data flows through the Tomography Alignment Software. The user interacts with the Alignment Software via various steps, which all have their own process and data flow too. The loading step uses the user's DM3 images and loads them into the main window. The contrast step takes the user's contrast adjustment and adjusts the images, then displays them back into the main window. The transform step takes the user's transformation adjustments and adjusts the images, then displays them back into the main window. The coarse alignment step is triggered by the user and performs an alignment on the images that are displayed in the main window. The data from coarse alignment is used in the automatic tracking step and the manual tracking step. The user then indicates the particles to the software, which is used in both auto and manual tracking, and the particle locations are displayed to the user. The data from auto and manual tracking are used in the optimization step which forms aligned images that are displayed in the main window. During the entire process, the user can preview the data in the main window.

3.2. Sequence Diagrams

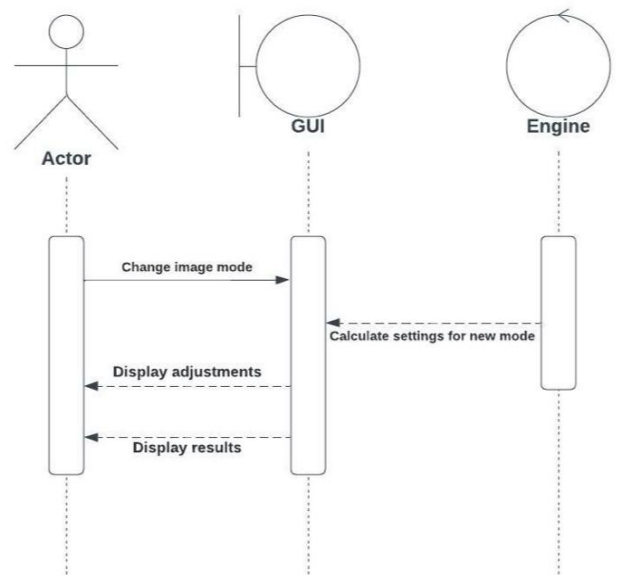
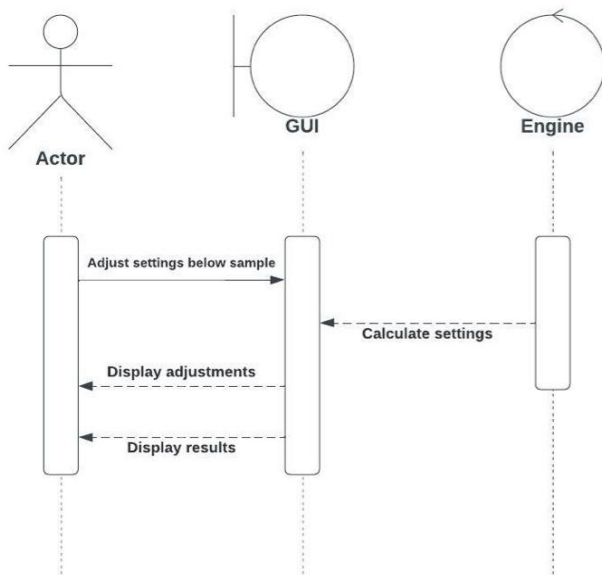
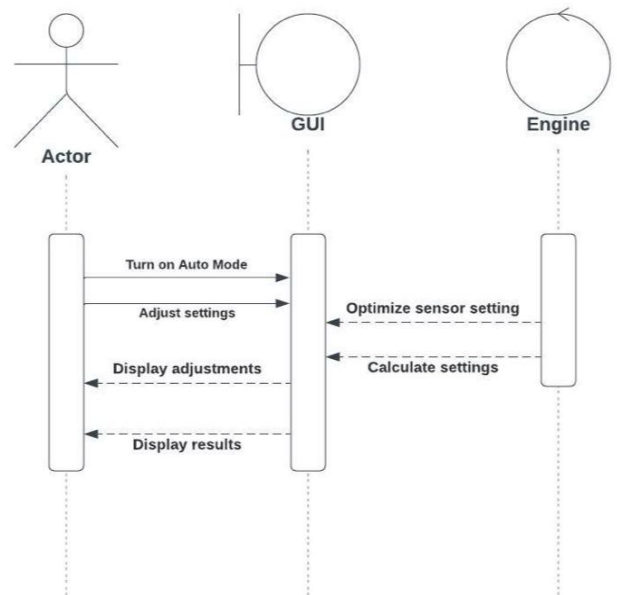
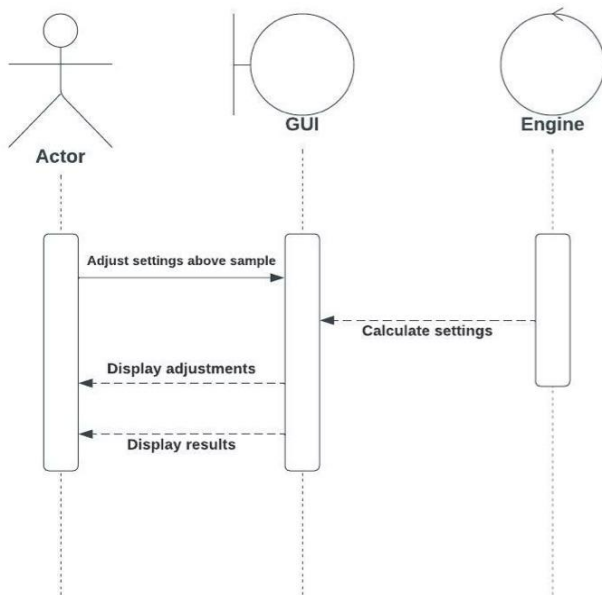
Sequence diagram of the user interface and the operations in the qEELS software:



There are several steps in calibrating energy loss, which include:

1. **Upload image:** uploading the image of the spectrogram to the software
 - a. **Display Image:** the image will be displayed on the GUI
2. **Indicate location of features:** the user will indicate the location of each feature on the image
3. **Select features for detection:** the user will select which features they would like to include in the detection process
4. **Request detection:** the user will request the calculation of the energy loss based on the selected features
 - a. **Return results:** the GUI will display the results as an image to the user
5. **Request savable results:** the user is now able to save the results in csv

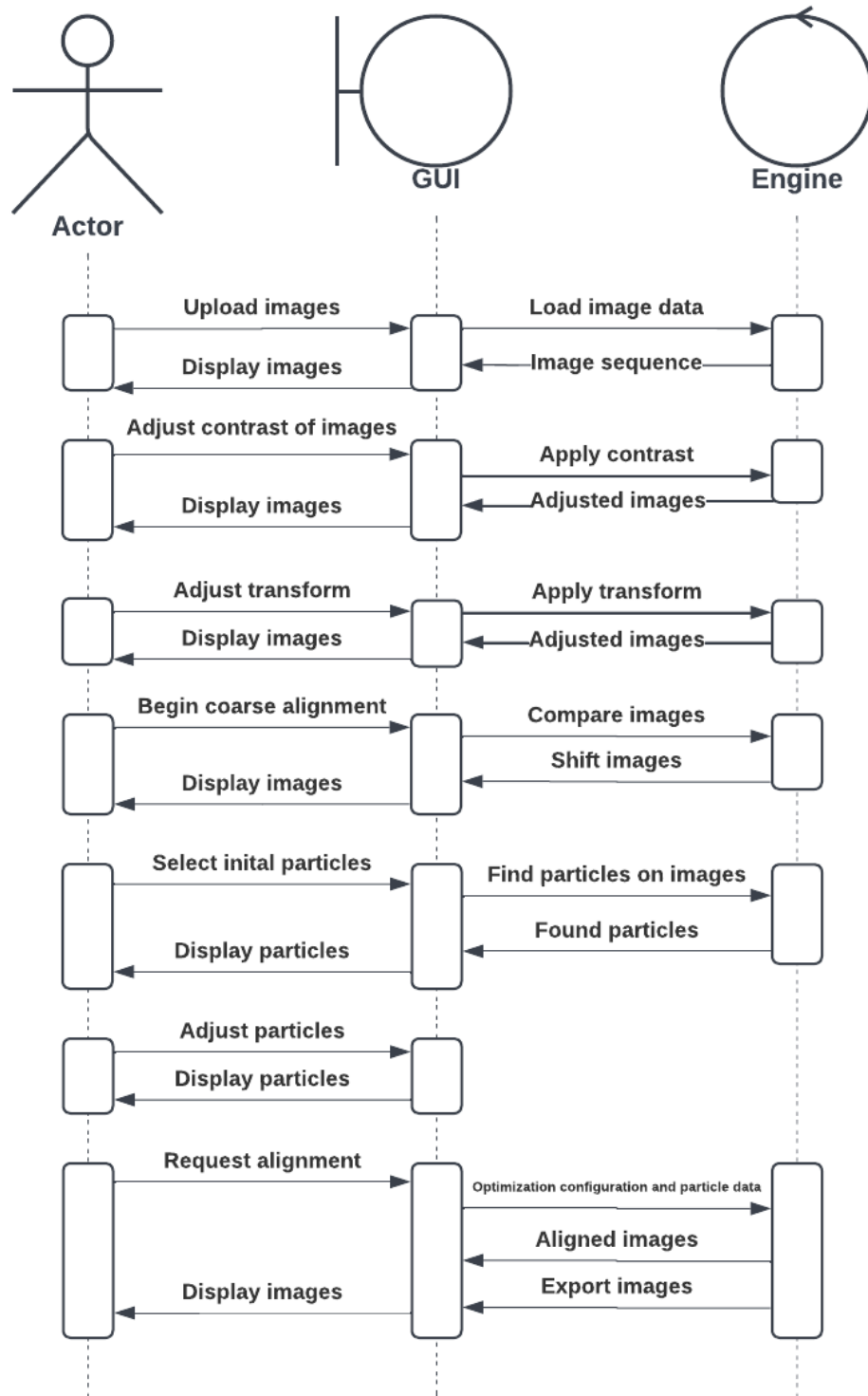
Sequence diagram of the user interface and the operations in the NanoMi Optics software. Note that the actions performed by the user in the NanoMi Optics software can be done in any preferred order, which is why there are several diagrams.



There are several steps in calculating optimized electron microscope settings, which include:

- **Adjust settings above sample:** changing the focal lengths of the sensors that sit above the sample
- **Select image mode:** the user is able to select diffraction or image mode to view the electron microscope settings
- **Turn on Auto mode:** turning on auto mode enables one of the sensors' (below the sample) settings' to be optimized while other sensors are being adjusted
- **Adjust settings below sample:** changing to focal lengths of the sensors that sit below the sample
- **Display results:** display of changing settings, and/or the optimized settings in auto mode
- **Display adjustments:** display of changing features on the diagram of the electron microscope

Sequence diagram of the user interface and the operations in Tomography Alignment Software:



There are several steps in aligning nanoparticle images, which include:

1. **Upload images:** uploading the images of the nanoparticles to the software
 - a. Images are displayed to the user
2. **Adjust contrast of images:** adjust the contrast of the images for better viewing
 - a. Updated images are displayed to the user
3. **Adjust transform:** transform the images for better viewing
 - a. Updated images are displayed to the user
4. **Begin coarse alignment:** coarse alignment aligns the images
 - a. Updated images are displayed to the user
5. **Select initial particles:** the user selects all the particles in the image to be found/tracked on all the images
 - a. The particle tracking is displayed on the images to the user
6. **Adjust particles:** user adjusts the locations of the particles on the images
 - a. The particle tracking is displayed on the images to the user
7. **Request alignment:** when all parameters are set, the user requests optimized alignment
 - a. **Display alignment:** the optimized aligned images are displayed to the user

4. Technical Specification

The following lists are not exhaustive and can be expanded as the need for functionality is discovered.

Programming Languages:

- Python

Python Libraries:

- Tkinter (GUI)
- Matplotlib (Chart Rendering)
- Scipy (Optimization and other algorithms)
- Numpy (Efficient numeric arrays)
- Pillow (Image processing and transforms)
- Flake8 (Linting)
- Pytest (Unit Testing)

Supporting Technologies:

- Github
- Github Actions (CI/CD)
- Github Projects (Project Management)
- Visual Studio Code

5. Test Plan

Unit Testing

We will take several measures to ensure that the code we produce is of good quality. We will set up Github Actions workflows to run all tests automatically when code changes are proposed. In our repository, there will be a few restrictions so that we can double-check our code. Pushing code directly to the main branch will be blocked. Attempting to merge a pull request without all the tests passing will be blocked. All pull requests will require an approving review form from at least one

other team member before merging. Pull requests will not be approved unless they include tests that demonstrate appropriate functionality. We will write our unit tests with Pytest.

To make our project testable, we will structure our project with testing in mind. The project's GUI will be difficult to test without excessive mocking because there are not any well-supported frameworks for testing tkinter GUIs. Because of this, anything tightly integrated with the GUI will also be difficult to test. Therefore, we will isolate backend logic from the GUI to make sure that it can be tested thoroughly. The backend and GUI portions of each program will be in separate Python packages. The backend will not be allowed to make any calls or references to the GUI or import any GUI related package. The GUI must only make calls to the backend via clearly defined and well-tested interfaces.

Unit testing will be used to verify all operations of the engines of the softwares. This includes all core math operations and all file reading and writing. For math operations test cases will be crafted based on known input and output extracted from the MATLAB software. The same input will be fed into the Python engine and output will be asserted against MATLAB's output. For floating-point results, tolerances will need to be set for assertions to allow outputs that are not exactly the same, but functionally identical.

Acceptance Testing

Acceptance testing will be used to ensure the client accepts changes to the GUI and Workflow, as well as that to ensure the software is functioning correctly at a high level. The client will be shown or asked to perform a task with software. Then we will ask the client if the software performed acceptably and whether they would like any changes. This will be done in weekly meetings and logged in a spreadsheet. This is particularly important for major GUI deviations from legacy software. Additionally there will be a final acceptance for each software to determine whether it is an acceptable replacement for the legacy software.

Usability Testing

Usability testing will be used to test how usable the software is with new users. As the user group for this software has used the legacy software before, they are already accustomed to using similar software. Testing the software on new users allows for exposure of usability problems that would not be found during acceptance testing. The System Usability Scale (SUS) will be used to gather information about usability as well as observing the users.

Project Management

Work Breakdown Structure

TASK	Lucas	Jasmine	Garrett	Jose	Total
	Estimated hours / Actual Hours				
Development environment					
Poetry	3 / 2	0 / 0	0 / 0	0 / 4	
GitHub actions	0 / 1	0 / 0	0 / 0	3 / 0	
Set up GitHub	0 / 1	0 / 0	3 / 0	0 / 0	
CI/CD	0 / 1	3 / 0	0 / 0	0 / 0	
Dev environment estimate total	3	3	3	3	12
Dev environment actual total	4	0	0	6	10
qEELS					
Run, document and analyze legacy qEELS	0 / 0	8 / 0	0 / 8	0 / 0	
Develop test scenarios	6 / 0	0 / 0	0 / 12	0 / 0	
Implement tests	10 / 0	0 / 0	0 / 14	0 / 0	
Determine MATLAB functionality can be replaced by python libraries	0 / 0	8 / 0	0 / 10	0 / 0	
Implement loading of image	0 / 0	24 / 0	0 / 16	0 / 0	
Implement GUI widget layout	0 / 0	0 / 0	0 / 14	14 / 0	
Implement rendering of image	0 / 0	0 / 0	14 / 14	0 / 0	

Implement clicking to indicate features on image	0 / 0	0 / 0	26 / 18	0 / 0	
Implement peak detection and fitting backend	0 / 8	0 / 0	0 / 14	26 / 0	
Integrate backend with GUI	20 / 0	0 / 0	0 / 18	0 / 0	
Validate new qEELS with Misa	1 / 1	1 / 1	1 / 1	1 / 1	
qEELS estimate Total	41	41	41	41	164
qEELS Actual Total	9	1	130	1	141
NanoMi Optics					
Run, document and analyze legacy NanoMi Optics	0 / 0	0 / 14	12 / 0	0 / 10	
Develop test scenarios	0 / 0	8 / 3	0 / 0	0 / 11	
Implement tests	0 / 0	12 / 3	0 / 0	0 / 11	
Determine MATLAB functionality can be replaced by python libraries	0 / 0	0 / 7	0 / 0	10 / 11	
Implement GUI widget layout	0 / 1	26 / 35	0 / 0	0 / 3	
Implement lens diagram rendering	0 / 0	0 / 8	0 / 0	36 / 30	
Implement lens settings optimization backend	48 / 0	0 / 0	0 / 0	0 / 50	
Integrate backend with GUI	0 / 0	0 / 0	34 / 0	0 / 34	
Validate new NanoMi Optics with Misa	1 / 1	1 / 1	1 / 1	1 / 1	
NanoMi Optics Estimate Total	49	47	47	47	190
NanoMi Optics Actual Total	2	71	1	124	197

Tomography Alignment Software					
Run, document, analyze legacy Tomography Alignment Software	16 / 10	0 / 0	0 / 0	0 / 1	
Develop test scenarios	0 / 10	0 / 0	12 / 0	0 / 0	
Implement tests	0 / 12	0 / 0	14 / 0	0 / 0	
Determine dependency footprint	0 / 4	4 / 0	0 / 0	0 / 0	
Determine MATLAB functionality can be replaced by python libraries	0 / 12	6 / 0	0 / 0	0 / 0	
Implement loading of many images	0 / 13	8 / 0	0 / 0	0 / 0	
Implement GUI widget layout	10 / 2	0 / 0	0 / 0	0 / 21	
Implement image rendering and frame switching	0 / 13	0 / 0	0 / 0	12 / 0	
Implement image contrast adjustment	5 / 6	0 / 0	0 / 0	0 / 0	
Implement manual clicking and indication of particle location across frames	0 / 16	0 / 0	16 / 0	0 / 0	
Implement automatic detection of particle location across frames backend	0 / 20	24 / 0	0 / 0	0 / 0	
Implement optimization backend	0 / 16	0 / 0	0 / 0	30 / 0	
Integrate backend with GUI	10 / 8	0 / 0	0 / 0	0 / 0	
Validate new Tomography Alignment Software with Misa	1 / 1	1 / 1	1 / 1	1 / 1	
Alignment Software Estimate Total	42	43	43	43	171
Alignment Software Actual Total	143	1	1	23	168

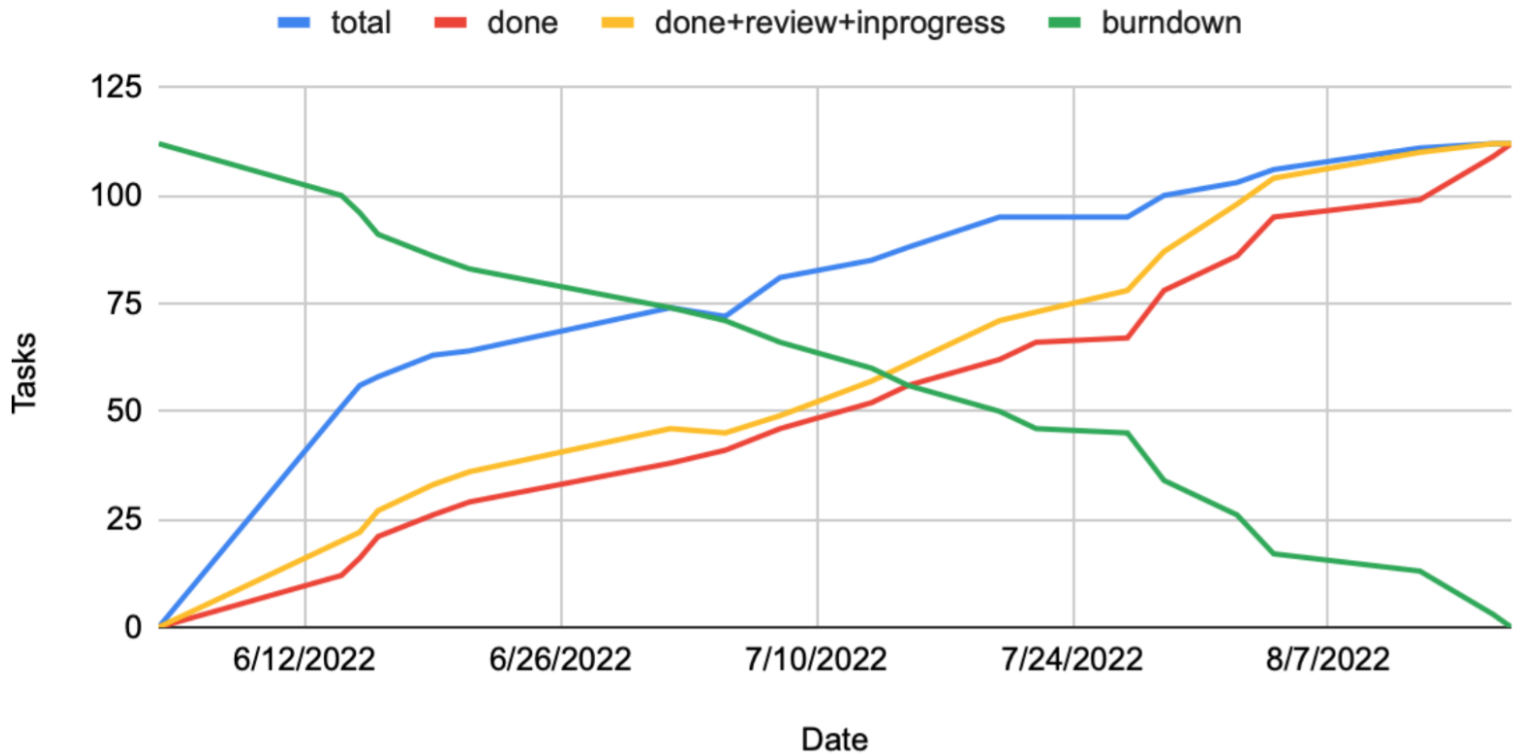
Documentation					
Determine what format documentation should be in	1 / 0	0 / 3	0 / 0	0 / 0	
Document new qEELS usage	0 / 0	0 / 22	0 / 0	12 / 0	
Document new Nanomi Optics usage	0 / 0	12 / 23	0 / 0	0 / 0	
Document new Tomography Alignment Software usage	0 / 0	0 / 30	12 / 0	0 / 0	
Document technical architecture and library usage	10 / 1	0 / 21	0 / 1	0 / 1	
Validate new Documentation with Misa	1 / 0	1 / 0	1 / 0	1 / 0	
Documentation Estimate Total	12	13	13	13	51
Documentation Actual Total	1	98	1	1	101
Estimated Total	147	147	147	147	588
Actual Total hours	~197	~172	~139	~169	~674
Estimated Weekly Average (9 weeks)	16.3	16.3	16.3	16.3	16.3
Actual Weekly Average (9 weeks)	~22	~19	~15.5	~19	~18.9

The total hours reflect the total times that each group member has logged.

Burnup and burndown

Project 2: NRCEMT Burnup

Based on Github Projects board



Features/Work packages completed

<div> <div>Combined</div> <div>qEELS</div> <div>Nanomi</div> <div>Alignment</div> <div>Technicals</div> <div>Completed</div> <div>In Progress</div> <div>New view</div> </div>						
Title	Assignees	Size	Milestone			
1 ✓ Create alignment user manual	coffeehousejazz	medium	Alignment Software			
2 ✓ Integrate transformation engine with GUI	luctowers	medium	Alignment Software			
3 ✓ Implement cross-correlation alignment in engine	luctowers	small	Alignment Software			
4 ✓ Integrate coarse alignment with GUI	luctowers	medium	Alignment Software			
5 ✓ Implement discovery of file sequences with naming example_001.ext -> example_999.ext	luctowers	small	Alignment Software			
6 ✓ Implement basic image processing and contrast adjustment engine	luctowers	small	Alignment Software			
7 ✓ Implement reading of DM3 image files	luctowers	medium	Alignment Software			
8 ✓ Implement writing of DM3 image files	luctowers	medium	Alignment Software			
9 ✓ Create a function to load DM3 image data given a file path	luctowers	x-small	Alignment Software			
10 ✓ Implement coarse alignment in Alignment Software engine	luctowers	medium	Alignment Software			
11 ✓ Implement non-functional GUI for automatic particle tracking	josepena97	medium	Alignment Software			
12 ✓ Implement non-functional GUI for manual particle tracking	josepena97	small	Alignment Software			
13 ✓ Implement non-functional GUI for optimization window	josepena97	small	Alignment Software			
14 ✓ Integrate loading and rendering sequence of DM3 images	luctowers	medium	Alignment Software			
15 ✓ Implement non-functional GUI for contrast adjustment popup-window	josepena97	small	Alignment Software			
16 ✓ Fix DM3 loading wrong image resolution	luctowers	x-small	Alignment Software			
17 ✓ Implement histogram rendering and controls for contrast adjustment window	luctowers	medium	Alignment Software			
18 ✓ Implement automatic particle tracking engine	luctowers	large	Alignment Software			
19 ✓ Integrate automatic particle tracking	luctowers	large	Alignment Software			
20 ✓ Implement alignment optimization engine	luctowers	large	Alignment Software			
21 ✓ Integrate contrast adjustment window GUI without histogram	josepena97	small	Alignment Software			
22 ✓ Implement manual tracking graphs	luctowers	medium	Alignment Software			
23 ✓ Implement manual tracking controls	luctowers	medium	Alignment Software			
24 ✓ Integrate optimization with GUI	luctowers	large	Alignment Software			
25 ✓ Refactor alignment software to output necessary data to csv and resume from previous run	luctowers	large	Alignment Software			
26 ✓ Implement interpolation for automatic particle tracking	luctowers	small	Alignment Software			
27 ✓ Adding docstrings to alignment software	luctowers	small	Alignment Software			
28 ✓ Make improvements to alignment software windowing system	luctowers	small	Alignment Software			
29 ✓ Add a reset button to transform window	luctowers	x-small	Alignment Software			
30 ✓ Implement non-functional GUI for the main window in Alignment Software	josepena97	medium	Alignment Software			
31 ✓ Add manual contrast adjustment to alignment software	luctowers	small	Alignment Software			
32 ✓ tech documentation for Alignment Software	coffeehousejazz	large	Documentation			
33 ✓ tech documentation for qEELS	coffeehousejazz	medium	Documentation			
34 ✓ tech documentation for Nanomi	coffeehousejazz	large	Documentation			
35 ✓ make template/guide for our manuals/documentation	coffeehousejazz	small	Documentation			
36 ✓ scope + charter final	coffeehousejazz	small	Documentation			

37	✓ design doc final	coffeehousejazz ▾	medium ▾	Documentation ▾
38	✓ Improve readme, installation and dev instructions	luctowers ▾	x-small ▾	Documentation ▾
39	✓ learn good technical writing practices	coffeehousejazz ▾	small ▾	Documentation ▾
40	✓ user manual template	coffeehousejazz ▾	x-small ▾	Documentation ▾
41	✓ qeels user manual drafts	coffeehousejazz ▾	small ▾	Documentation ▾
42	✓ Nanomi user manual drafts	coffeehousejazz ▾	small ▾	Documentation ▾
43	✓ alignment user manual drafts	coffeehousejazz ▾	small ▾	Documentation ▾
44	✓ tech documentation draft nanomi	coffeehousejazz ▾	x-small ▾	Documentation ▾
45	✓ Create Nanomi user manual	coffeehousejazz ▾	medium ▾	Nanomi Optics ▾
46	✓ Display lens magnifications and tests	josepena97 ▾	medium ▾	Nanomi Optics ▾
47	✓ Refactoring Nanomi GUI	josepena97 ▾	? size unknown ▾	Nanomi Optics ▾
48	✓ Integrate units dropdown menu	josepena97 ▾	small ▾	Nanomi Optics ▾
49	✓ Refactoring Upper Lenses ray path math errors and ray path tests	josepena97 ▾	large ▾	Nanomi Optics ▾
50	✓ Analyze legacy math code for bottom lenses	josepena97 ▾	small ▾	Nanomi Optics ▾
51	✓ Integrate GUI controls with upper lenses engine	josepena97 ▾	medium ▾	Nanomi Optics ▾
52	✓ slider and on/off button for lower sliders	coffeehousejazz ▾	x-small ▾	Nanomi Optics ▾
53	✓ add function for turning on/off lens	coffeehousejazz ▾	small ▾	Nanomi Optics ▾
54	✓ Plan oop design for nanomi lenses	josepena97 ▾	large ▾	Nanomi Optics ▾
55	✓ Implement upper lens (above the sample) adjustment calculations	coffeehousejazz... ▾	small ▾	Nanomi Optics ▾
56	✓ make upper beam sliders work!	coffeehousejazz ▾	small ▾	Nanomi Optics ▾
57	✓ Implement widgets GUI on the 'Settings above sample' window	coffeehousejazz ▾	medium ▾	Nanomi Optics ▾
58	✓ save and reset buttons	josepena97 ▾	medium ▾	Nanomi Optics ▾
59	✓ Implement optimization for auto setting on 'Projective' lens	josepena97 ▾	medium ▾	Nanomi Optics ▾
60	✓ Implement optimization for auto setting on 'Intermediate' lens	josepena97 ▾	? size unknown ▾	Nanomi Optics ▾
61	✓ Implement optimization for auto setting on 'Objective' lens	josepena97 ▾	? size unknown ▾	Nanomi Optics ▾
62	✓ Integrate Nanomi optimizations (auto setting) with GUI	josepena97 ▾	medium ▾	Nanomi Optics ▾
63	✓ Refactoring ray plotting and tests	josepena97 ▾	large ▾	Nanomi Optics ▾
64	✓ Understanding upper lens math	josepena97 ▾	? size unknown ▾	Nanomi Optics ▾
65	✓ Implement results window GUI	coffeehousejazz ▾	small ▾	Nanomi Optics ▾
66	✓ draw electron beam above the sample	coffeehousejazz ▾	large ▾	Nanomi Optics ▾
67	✓ Implement widgets GUI on the 'Settings below sample' window	coffeehousejazz ▾	small ▾	Nanomi Optics ▾
68	✓ revise code to the Tkinter standards	coffeehousejazz ▾	x-small ▾	Nanomi Optics ▾
69	✓ Diagram layout GUI - drawing the lenses/boxes	coffeehousejazz ▾	medium ▾	Nanomi Optics ▾
70	✓ Display results in the results window	josepena97 ▾	medium ▾	Nanomi Optics ▾
71	✓ Integrate GUI controls with lower lenses engine	josepena97 ▾	medium ▾	Nanomi Optics ▾
72	✓ Implement bottom lenses to Lens class	josepena97 ▾	medium ▾	Nanomi Optics ▾

73	✓ Integrate backend upper lens (above the sample) calculations with GUI	josepena97	small	Nanomi Optics
74	✓ Create qEELS user manual	coffeehousejazz	medium	qEELS
75	✓ adding docstrings to qeels engine	veengren-s	small	qEELS
76	✓ fix axis flipping	veengren-s	small	qEELS
77	✓ Implement qEELS GUI widget layout	veengren-s	medium	qEELS
78	✓ Integrate qEELS engine with GUI	veengren-s	medium	qEELS
79	✓ Impliment contrast adjustment	luctowers	small	qEELS
80	✓ impliment spectrogram modifications	veengren-s	small	qEELS
81	✓ impliment surface plasmon calculations	veengren-s	medium	qEELS
82	✓ implement bulk plasmon calculation	veengren-s	medium	qEELS
83	✓ Implement loading of spectrogram	veengren-s	x-small	qEELS
84	✓ Implement rendering of spectrogram in qEELS	veengren-s	small	qEELS
85	✓ Impliment qEELS UI Logic	veengren-s	medium	qEELS
86	✓ Implement clicking to indicate features on spectrogram in qEELS	veengren-s	medium	qEELS
87	✓ Implement box drawing around plasmons	veengren-s	medium	qEELS
88	✓ Impliment results saving	veengren-s	medium	qEELS
89	✓ fix double boxes	veengren-s	small	qEELS
90	✓ Implement peak detection and core calculations in qEELS	veengren-s	size unknown	qEELS
109	✓ tech documentation draft alignment	coffeehousejazz	x-small	
110	✓ tech documentation draft qEELS	coffeehousejazz	x-small	
111	✓ test-o-rama quizzes	coffeehousejazz	small	
112	✓ test-o-rama tasks	coffeehousejazz	small	

Software Implementation Status

Test Report

Unit Tests

```
platform darwin -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: /Users/luctowers/Documents/ubco/cosc499/matlab-to-python-application-translation-project2-nrc/qeels
collected 15 items

tests/test_peak_detection.py ..... [ 80%]
tests/test_prz.py .. [ 93%]
tests/test_results.py . [100%]

===== 15 passed in 1.87s =====

collected 9 items

tests/test_lens_static_methods.py .. [ 22%]
tests/test_lower_lenses_math.py . [ 33%]
tests/test_optimization.py . [ 44%]
tests/test_upper_lenses_math.py ... [ 77%]
tests/test_ur_conversion.py .. [100%]

===== 9 passed in 0.37s =====

collected 36 items

tests/test_csv_io.py .... [ 11%]
tests/test_dm3.py ... [ 19%]
tests/test_file_discovery.py ... [ 27%]
tests/test_img_io.py ... [ 36%]
tests/test_img_processing.py ..... [ 69%]
tests/test_optimization.py ..... [ 88%]
tests/test_particle_tracking.py .... [100%]

===== 36 passed in 2.76s =====
```

Acceptance Tests & Unit Test report

REQUIREMENTS	Type of Test UN: Unit Testing A: Acceptance Testing	Pass or Fail P: pass F: fail	Contributor GC: Garrett Cook JM: Jasmine Mishra JP: Jose Pena Revelo LT: Lucas Towers
Functional Requirements			
qEELS			
Software will load the spectrogram from a PRZ file	UN	P	GC
Software will render points and boxes of indicated features on the spectrogram	A	P	GC
Software will detect fitted peaks of surface and bulk plasmon	UN	P	GC
Software will calculate calibrated energy loss axis and transfer axis	UN	P	GC
Software will output results to CSV format	UN	P	GC
Software will be functionally equivalent to legacy software	A	P	GC

Nanomi Optics			
Software will render lenses on diagram	A	P	JM
Software will calculate upper beam ray	UN	P	JP
Software will calculate beam ray through projective, intermediate, and objective lenses	UN	P	JP
Software will render upper beams on diagram	A	P	JM, JP
Software will render lower beams on diagram	A	P	JP
Software will display results in a table	A	P	JM, JP
Software will calculate optimized settings for projective lens	UN	P	JP
Software will calculate optimized settings for intermediate lens	UN	P	JP
Software will calculate optimized settings for objective lens	UN	P	JP
Software will be at least as functional as legacy software	A	P	JP
Alignment Software			
Software will load DM3 images	UN	P	LT
Software will perform automatic contrast adjustment	UN	P	LT
Software will perform translation, rotation and scaling of frames	UN	P	LT
Software will perform coarse alignment of frames with cross-correlation	UN	P	LT
Software will automatically detect the location of particles with kernel convolution	UN	P	LT
Software will calculate optimized alignment of frames	UN	P	LT
Software will output intermediary alignment info to csv format	UN	P	LT
Software will write DM3 images	UN	P	LT
Software will be at least as functional as legacy software	A	P	LT
User Requirements			
qEELS			
User of legacy software will be comfortable with the new GUI	A	P	GC
User can click on the spectrogram to indicate location of features	A	P	GC
Nanomi Optics			
User of legacy software will be comfortable with the new control GUI	A	P	JM, JP
User of legacy software will be comfortable with the new results GUI	A	P	JM, JP
Alignment Software			
User of legacy software will be comfortable with the new main window GUI	A	P	JP

User of legacy software will be comfortable with the new contrast adjustment GUI	A	P	JP
User of legacy software will be comfortable with the new transformation window GUI	A	P	JP,LT
User of legacy software will be comfortable with the new automatic tracking window GUI	A	P	JP,LT
User of legacy software will be comfortable with the new manual tracking window GUI	A	P	LT
User can adjust particle location manually using manual tracking window	A	P	LT
User can export images to DM3 format	A	P	LT

Usability Testing Results

Through a user testing event, data was gathered on the three softwares' usability through the users following a set of tasks and then completing a questionnaire on their experience. The questionnaire that was created for our software is based on the System Usability Scale (SUS), an industry standard quick and reliable tool to help measure usability. The questions resemble the SUS standard questions with slight modifications. The questions that were included in our questionnaire are as follows:

1. I think that if I were a nanotechnology researcher, I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that nanotechnology researchers would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The users will answer how much they agree to each statement on a scale of 1-5 with 1 being "I disagree" and 5 being "I agree."

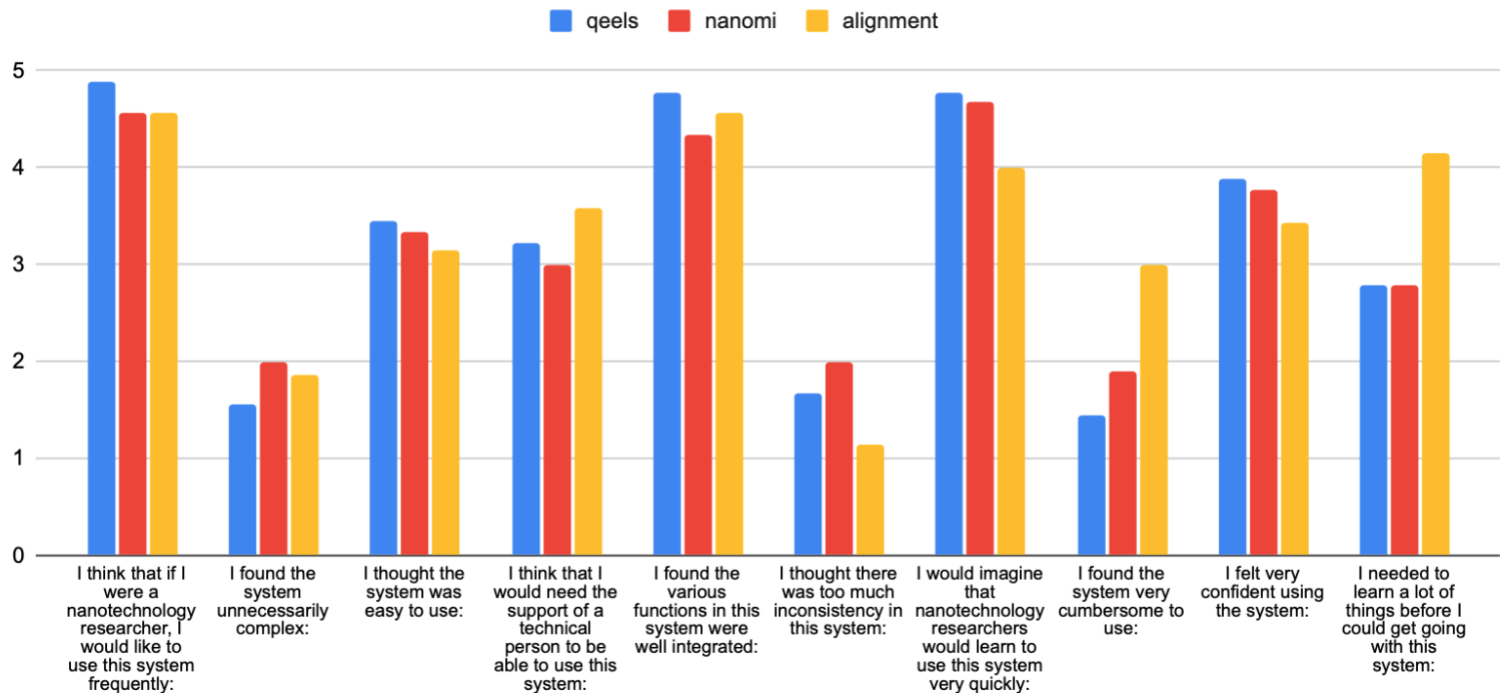
Each questions' answer contributes to a calculation of an overall score to measure usability. The scores are calculated on a 1-100 scale, with research showing that a score over 68 as above average usability and under 68 as below average.

Results:

Software	Score
qEELS	77.78
NanoMi Optics	72.50
Tomography Alignment Software	65.00

Results for each question:

User Testing Results



In addition to the SUS, observations were made while the participants were using the software:

qEELS

- If users click the main detect button without selecting detect buttons for each set of points, all the points will get reset.
- If users select a material in the material listbox before selecting points on the image, the material selection will be undone.

NanoMi Optics

- Users wanted to zoom in and out of the diagram with a pinching motion on the trackpad, or use the zoom icon just by clicking and not drawing a box around the area intended to zoom.

Tomography Alignment Software

- Issues with window management. When one window is fullscreen, it was difficult for the users to keep track of which windows were open.

Fixes that were made to the software in response to usability testing:

- Fix issue with detection on qEELS.
- Changed qEELS material selection listbox to a dropdown menu.
- New alerts added to Alignment Software to help users manage windows.

Final Thoughts:

The three softwares are not intended for use by the general population, so analysis of results from the usability testing reflect that. Despite the small user group, the three software got reasonable SUS scores.

Tomography Alignment Software was the only software that did not have a passing score. However, this was due to the difficulty of the users understanding of how to complete tasks because of their incomplete knowledge of nanotechnology. The scores on the quiz reflects this, as users on average had low confidence and agreed that they need to learn a lot of things before being able to use the software.

However, since this software is intended to perform complex operations by a small user group, the usability of the software is still up to standards.

Another consideration is that testing users did not have the opportunity to test the legacy MATLAB software in order to compare it to the new software. The new software's user interface is based on the legacy software so that current users of the legacy software would have an easy time transitioning to the new software. If the users during testing also learned the legacy software, it is likely that they would have different opinions on usability. This is the reason that there were no changes made to the zoom function in NanoMi Optics, despite the observation that many users were attempting to use a pinching motion. The current zoom function is familiar among the current MATLAB software users and allows for more precise zooming capabilities.

Requirements Delivered

All core requirements that were planned in the Scope & Charter and Design Document were delivered. Some additional requirements were made during development by the client. Also, several extra functionalities were delivered as well,

which made improvements to the existing software, but were not required by the client.

Core requirements:

Performance

- All three software have the same or better performance than the original MATLAB software, with new designs and additional features implemented.
- Portable and able to run on multiple operating systems.

Development

- Delivered by early August.
- No paid resources used.
- Kanban workflow used with GitHub Projects.
- Git workflow used with GitHub.

Quality

- Code is linted and in a consistent format using Flake8.
- All code has been reviewed.

Testing

- Every function in the engines is tested.
- All user interactions are tested with acceptance testing and usability testing.
- GitHub actions are used to test all code before committing.

qEELS

- Software accepts q-EELS pattern image and feature locations.
- Software adjusts contrast on the q-EELS pattern image.
- Software detects fitted peaks of surface and bulk plasmon.
- Software calculates the calibrated energy loss axis and transfer axis.
- Software saves results.

NanoMi optics

- Software accepts desired parameters from the user.
- Software calculates and displays settings in a diagram of a NanoMi electron transmission microscope.
- Software calculates optimized optics settings.
- Software saves results.

Tomography Alignment Software

- Software accepts many images.

- Software adjusts contrast on images.
- Software transforms images.
- Software performs inter-image alignment via cross-correlation to maintain stability.
- Software automatically determines the location of selected particles across images.
- Software accepts manual adjustments to location of particles.
- Software optimizes and aligns the sequence of images.
- Software saves results.

Technical Requirements

- Legacy software run and analyzed.
- MATLAB dependencies replaced.
- Software written in Python
- Software uses well-supported Python packages.
- Software thoroughly tested.
- The GUI separated from the backend.
- GPL and correct licensing maintained.

Users

- There is thorough documentation that users can learn how to use the software from.
- Users who used the legacy software can pick up new software easily.

Additional requirements:

These requirements were added to the project upon request from the client.

qEELS

- Contrast Adjustment.
- Material Selection.

NanoMi Optics

- Changing between Cf and UR mode for lenses above sample.

Tomography Alignment Software

- Manual contrast adjustment

Extra functions:

Improvements and additional functions in each software.

qEELS

- Improved peak detection.
- Improved optimization.

NanoMi Optics

- Improved GUI.
- Improved optimization.

Tomography Alignment Software

- Improved GUI.
- Improved Coarse Alignment.
- Interpolation for Automatic Detection.
- Interpolation for Manual Detection.
- Improved optimization.
- Restore previous session.

Known Bugs

Interacting with tkinter GUI widgets on a non-main thread is technically undefined behavior. In our code, we used threads in-order to make sliders behave responsively even when the results of their actions have not yet been processed. In very rare cases, this can cause a crash. We have not observed this happen, and it appears to be a non-issue in practice.

Lessons Learned










During the project, many lessons were learned among all team members including:

- Working in a professional setting with a client.
- Software development with Kanban.
- Using a Git workflow for version control and collaboration.
- Using new tools including MATLAB, tKinter, SciPy, etc.
- Writing complete testing.
- Code review.
- Technical writing.
- Time management.

Deployment

Repository

The GitHub repository contains all of the code and its history. In this project, GitHub was used for collaboration, version control, workflows, and project management. The following is a description of the organization of the main branch on this project's Github:

	<code>.github/workflows</code>	Add main branch constraint back	4 days ago
	<code>.vscode</code>	Fix vscode launch configuration	2 days ago
	<code>alignment-software</code>	Add colspanspan to manual contrast frame	8 hours ago
	<code>docs</code>	Add summary and logs for august 12th	2 days ago
	<code>nanomi-optics</code>	Merge pull request #271 from UBCO-COS...	23 hours ago
	<code>qeels</code>	Merge pull request #272 from UBCO-COS...	3 hours ago
	<code>.gitignore</code>	Ignore pyinstaller spec files	2 months ago
	<code>.pylintrc</code>	Add basic pylintrc	2 months ago
	<code>README.md</code>	Reword readme	3 days ago

workflows - sets up workflows for Github actions testing, linting, and creating executable files

.vscode - contains launch files for VS code

alignment-software - code for Tomography Alignment Software

docs - contains all documentation

nanomi-optics - code for NanoMi optics

qeels - code for qEELS

.gitignore - tells Git which files to ignore when committing

.pylintrc - used for linting

README.md - overview of project, setup.

Finding the code

All code and executable files can be found on the GitHub public repository.

Download the code here:

matlab-to-python-application-translation-project2-nrc/**qeels/**

matlab-to-python-application-translation-project2-nrc/**nanomi-optics/**

matlab-to-python-application-translation-project2-nrc/**alignment-software/**

User Manual

There are user manuals for each of the software in PDF format, which can be found in the GitHub repository:

matlab-to-python-application-translation-project2-nrc/**docs/user-manuals/**

There is also technical documentation for each of the software in Markdown and PDF, which can be found in the GitHub repository:

matlab-to-python-application-translation-project2-nrc/**docs/technical-documentation/**

Installation

Windows Install

1. A compatible version of **Python** (Python ≥ 3.9) must be installed.
 - a. See: <https://www.python.org/downloads/windows/>
2. Extract the full source code to a folder on your machine.
 - a. Go to the GitHub repository, and click the green button that says "Code" then **Download Zip**.
 - b. Extract the zip file. (Unzip the file)

- c. See:
<https://github.com/UBCO-COSC-499-Summer-2022/matlab-to-python-application-translation-project2-nrc.git>
- 3. Navigate into the **NanoMi Optics folder**.
- 4. Run the `install.py` file in the folder with Python. This installs the software.
 - a. You can do this by right clicking on the file and click **with Python**.
- 5. Run the `main.py` file in the folder with Python. This runs the software.
 - a. You can do this by right clicking on the file and click **with Python**.

Mac/Linux Install

1. A compatible version of Python (Python ≥ 3.9) must be installed.
 - a. Using a package manager such as `brew` or `apt` is recommended.
 - i. `brew install python@3.9`
 - ii. `sudo apt install python3.9`
2. Install supporting libraries for Tcl/Tk.
 - a. `brew install python-tk`
 - b. `sudo apt install python3-tk`
3. Extract the full source code to a folder on your machine.
 - a. Go to the GitHub repository, and click the green button that says "Code," then click **Download Zip**.
 - b. Go to the GitHub repository, and click the green button that says "Code" then **Download Zip**.
 - c. Extract the zip file. (Unzip the file)
 - d. See:
<https://github.com/UBCO-COSC-499-Summer-2022/matlab-to-python-application-translation-project2-nrc.git>
4. Navigate into the **NanoMi Optics folder**.
5. Run the `install.py` file in the folder with Python. This installs the software.
 - a. You can do this by right clicking on the file and click **with Python**.
6. Run its `main.py` file in the folder with Python. This runs the software.
 - a. You can do this by right clicking on the file and click **with Python**.

Executables:

The user can also run the three software using executable files (EXE file).

Executables are easily run on Windows machines. Simply download the software's file onto your machine, double-click on the file in your directory, and the software will begin.

Workflow, testing, and deployment environments

Workflows and testing

We used GitHub actions to set up all of our workflows and testing. Every time the code was pushed onto our GitHub repository, GitHub actions would run all of the tests and linting on our code. The code was not allowed to be pushed onto the main branch unless all of the code passed the tests.

Deployment Setup

Development Environment:

1. Install Poetry for Python
2. Clone the repository and cd into it
3. Install dependencies with poetry install
4. Enter a virtual shell with poetry shell

Enter the Poetry virtual shell during every development session to use supporting libraries, lint, and test.

Useful Commands:

- Run all tests: `pytest`
- Lint all code: `flake8 nrcemt`

Alternative Development:

Another, easier way to develop this software is to do the following:

1. After installing the software to your machine, open the main folder with all the code in VS Code
2. Edit the files in VS code to make changes

3. Run the specific software you want by running the main.py file in the software folder with Python.
4. **OR** run the software you want by clicking **Run**, then choose **Start Debugging** the VS Code. This way, you can add breakpoints in the code during development. Change the software that runs by clicking on the blue bar at the bottom of your VS Code window and choose the software to run.

Setup considerations:

These software use tools that developers need to download on their machine. If you are a developer, you must verify that your machine has permission to download the various tools required. If you are unsure whether your machine has permission, please contact your IT support service.

Appendix A - Record of Changes

Record of Changes:

Version	Date	Author	Description of Changes
1.0	08/10/2022	Jasmine Mishra	First draft done
1.1	08/15/2022	Lucas Towers	Additions + edits
1.2	08/17/2022	Jasmine Mishra	Final document completed