CS482/495/496 Software Project: Artificial Axon Pipeline

Chloe Miranda and Jonathan Dargakis

2026-01-30

CS495 Software Project Proposal:
Artificial Axon Labs' Data Pipeline Jonathan Dargakis and Chloe Miranda 2026-01-30

1 Client Information

By sharing this client information and the rest of this document, you are stating that this client has provided this project as something they want (not something you created and asked if they wanted), and that they are interested in having you complete this project for your capstone.

1. Client name: Dr.Alexis Lowe
2. Client title: Senior Scientist
3. Client email address: alexislowe816@gmail.com
4. Client employer: Artificial Axon Labs
5. How you know the client: I was mutually connected with them through a colleague based on our shared interest in computational medicine.

2 Project Description

2.1 Overview

Artificial Axon Labs specializes in studying artificial neurons and their myelination using high-resolution 3D imaging. Currently, Dr. Lowe and Dr. Jagelskia rely on a highly fragmented and inefficient workflow to process these image datasets. In its present form, their pipeline spans Python scripts, Jupyter notebooks, and ImageJ each responsible for different stages of preprocessing, rendering, and analysis. Switching between these tools for every dataset is tedious and time-consuming, and even minor adjustments often require manually editing code or scripts. Because each dataset may require unique parameter settings, researchers must repeatedly tune and reconfigure files in different environments, significantly slowing down their ability to produce usable figures and scientific insights.

Additionally, the current workflow suffers from decentralized practices between team members. Dr. Lowe and Dr. Jagelskia process their data using different methods, scripts, and organizational structures, resulting in inconsistent outputs and further complicating collaboration. The lack of a unified system not only increases cognitive load but also makes reproducibility and long-term data management more difficult.

The purpose of this project is to design and implement a dedicated software application that consolidates this entire image-processing pipeline into a single, cohesive system, as well as utilizes computational techniques to speed up the process of rendering their 3D data. The goal is not to eliminate scientific control, but to provide a streamlined interface where researchers can upload files, adjust analysis parameters, view results, and export processed images without navigating three separate programs. By integrating configurable image-analysis tools and automated processing routines, the application aims to reduce repetitive work while preserving the flexibility required for scientific experimentation.

A key requirement expressed by the client is the ability to maintain a history of processed datasets. Both researchers emphasized the importance of comparing multiple renderings, tracking different configurations, and easily retrieving past results for publication-quality figures. The application will therefore include a structured history or "edit-tracking" system that allows users to revisit previous outputs or apply past settings to new datasets.

Ultimately, this project addresses a real and immediate bottleneck in the client's research process. By centralizing tools, reducing manual overhead, and supporting reproducible workflows, the software will significantly improve efficiency and consistency in analyzing artificial neuron imagery—benefits that are vital for a rapidly growing research lab.

2.2 Key Features

2.3 File Management: image files are organized by wells/sections of the 3d printed axon.

- upload files to app
- automated file organization based on naming conventions
  - option for manually organizing files

2.4 Run configuration and history:

Runs of file processing will have options, and those choices can be recorded and reused for future runs.

- Each run will also have some sort of metadata attached to it, including the configuration used, date, and ID
- Configurations will consist of:
  - input and output folder paths
  - image data type: 2D or 3D
  - microscope type: Keyence or Olympus
  - Channel settings
  - number of field of views per well
- Note that for 3D data, additional inputs are required
  - number of frames per z-stack
  - cherry picking frames for the run
  - distance between frames (micrometers)
  - Yes or no: run wrapping analysis algorithm

Automated Image Processing:

Due to the nature of their work, our clients' pipeline will remain relatively intact, utilizing a combination of bash, python and ImageJ to process these images but now with user inputs and consolidated into one run.

Result management:

- after running the scripts, there should be four outputs:
  - Masked tiff files for each field of view
  - data files of each field of view
  - summary data of each well used in experiment

  - tiff file of mylein overlap

- These four outputs should be consolidated into two separate csv files, one for well data and one for field of view data.
- For 3D data, there should also be an extra csv for the outputs of wrapping analysis.
- The metadata and run configuration should also be recorded and output somewhere, likely as either a text file or in the app itself

2.5 Why this Project is Interesting

This project is interesting because it serves as a bridge between biology and computer science, applying computational tools to address real problems in modern biomedical research. The software directly supports the study of artificial neurons and myelination—processes that are essential for understanding neural development, neurodegenerative diseases, and the behavior of complex neural systems. By building tools that manage large image datasets and extract meaningful information from 3D microscopy images, the project contributes to a field where efficient data handling is often a major bottleneck.

Beyond the scientific relevance, this project is enticing because it involves solving multifaceted engineering challenges. It blends file system design, image processing, data visualization, and user-facing tool development. A challenge I'm excited to tackle with this project is the UI design. Our clients need to have quite a lot of control over what's being processed so it will be interesting to figure out a UI that allows that control without revealing too much of the backend and overwhelming them with information. Developing robust software that researchers can rely on requires disciplined software engineering practices, thoughtful UX, and careful consideration of performance and scalability—skills that align strongly with the goals of a capstone.

Finally, this project matters because the underlying research matters. Improving tools for analyzing 3D neuron imagery has immediate value for the professionals using them and the scientific questions they are trying to answer. The work that Artificial Axon Labs is doing is already groundbreaking. Good data processing pipelines can accelerate the pace of discovery, reduce human error, and make previously impractical analyses feasible. A capstone that enables real researchers to work more efficiently—and potentially contributes to a better understanding of neural health and disease—has both intellectual and practical significance.

2.6 Summary of Efforts to Find a Project

(Not necessary for 482) I was mutually connnected with Dr.Lowe through a community crochet night. Other connections I've reached out to did not work out or were not able to offer anything.

[Most CS495 projects end here. The sections below are for CS482 and CS496 software projects].

3 Requirements

3.1 Non-Functional Requirements

ID	Title	Category	Description
NFR1	Color scheme	Visual / Design	The application shall use a dark red and white color scheme.
NFR2	Error messages	Visual / Design	The system shall display clear and informative error messages whenever a script fails during pipeline execution.
NFR3	Editable codebase	Accessibility	The codebase shall be easily editable to support future requirements and extensions.

ID	Title	Category	Description
NFR4	Analysis consistency	Results	The system shall always produce standard wrapping analysis results using the original formula for all 3D data.
NFR5	Exporting on error	Visual / Design	Options to export scripts as text files and save settings shall be presented whenever an error of both user and system occur.
NFR6	User confirmations	Visual / Design	The system shall display confirmation dialogs when starting, ending, or deleting a process.
NFR7	Pipeline performance	Performance	The system shall execute pipeline sessions within a reasonable time under normal hardware conditions for both 2D and 3D pipelines.
NFR8	File management safety	Reliability	The system shall not modify or delete original input files; all changes shall occur within system-managed copies.
NFR9	Reproducibility	Reproducibility	The system shall produce identical outputs when executed multiple times using the same inputs and configuration settings.
NFR10	User instructions	Visual / Design	The system shall provide tooltips or guidance for all major configuration options and user operations.

note: In all user stories and system requirements, unless the pipeline is explicitly stated, assume that both 2D and 3D pipelines are included.

3.2 User Stories

ID	Title	Points	Description
U1	Data upload	1	As a user, I want to upload data files or folders so that the application can access data for processing.
U2	Setting inputs	2	As a user, I want to input settings for data processing so that I can customize how data is processed depending on the experiment.
U3	Channel selection	1	As a user, I want to select one or multiple channels in a 2D experiment so that I can perform simple or complex experiments.
U4	Image preview after organization	2	As a user, I want to preview selected images after organization so that I can determine which images should be processed.
U5	Selecting images after organization	1	As a user, I want to select organized images for a run so that I can clean up experimental image datasets without reorganizing files.
U6	Deleting images after organization	1	As a user, I want to delete organized images for a run so that I can clean up experimental image datasets without reorganizing files.
U7	2D or 3D pipelines	1	As a user, I want to choose between 2D and 3D image pipelines so that I can correctly process different types of image data.

ID	Title	Points	Description
U8	Intensity threshold	1	As a user, I want to set an intensity threshold or enable automatic selection so that I can subtract debris or noise from images.
U9	Field of view amount	1	As a user, I want to specify the number of fields of view per well so that I can define how many wells are used in an experiment.
U10	Channel selection	1	As a user, I want to select which channels are used so that I can identify stains and suppress debris.
U11	Microscope selection	1	As a user, I want to select which microscope was used so that the system can distinguish between Keyence and Olympus data.
U12	Creating setting configuration	2	As a user, I want to create configuration profiles for data processing sessions so that I can reuse them for similar experiments.
U13	Reading setting configuration	2	As a user, I want to view saved configuration profiles so that I can reuse previous settings.
U14	Update setting configuration	2	As a user, I want to update saved configurations so that I can modify them to meet new experimental needs.
U15	Delete setting configuration	2	As a user, I want to delete saved configurations so that I can manage and declutter my configuration list.
U16	Z-stack frames	1	As a user, I want to enter the number of frames per z-stack so that I can define the structure of 3D image data.
U17	Frame distance	1	As a user, I want to enter the distance between frames in micrometers so that I can define spacing in 3D image data.
U18	Text file generation	2	As a user, I want to generate text files of scripts so that I can manually edit and run them if the system fails.
U19	Preview configuration	1	As a user, I want to preview metadata of saved configurations so that I can identify the correct configuration before selecting it.
U20	Preview cherry-pick images	1	As a user, I want to preview images before selecting them for 3D processing so that I can avoid skewed data.
U21	Session start	1	As a user, I want to start a data processing session so that selected images and settings execute as a single run.
U22	Session end	2	As a user, I want to stop or end a running session so that I can terminate or finalize processing when needed.
U23	Session history	2	As a user, I want to view metadata from past sessions so that I can review and document previous runs.
U24	Choosing configurations	1	As a user, I want to select a saved configuration before starting a run so that I can save time.

ID	Title	Points	Description
U25	Auto-fill	1	As a user, I want to automatically run the pipeline after selecting a configuration so that I can avoid stepping through each stage.
U26	Rerun	1	As a user, I want to auto-fill settings from a previous configuration so that I can rerun experiments with minimal changes.
U27	Pause a run	1	As a user, I want to pause or stop a run so that I can halt execution if incorrect settings were chosen.
U28	Image recovery	1	As a user, I want to undo image exclusions so that I can recover images mistakenly removed from the 3D pipeline.
U29	Progress monitor	1	As a user, I want to see which stage the pipeline is currently executing so that I can track processing progress.
U30	Export failed files	3	As a user, I want to export scripts with previous settings so that I can manually run the pipeline if the system fails.
U31	Preview results	2	As a user, I want to preview results before ending a session so that I can verify outputs.
U32	Preview graphs	5	As a user, I want to generate graphs before exporting results so that I can visualize and verify data.
U33	SSH access for files	1	As a user, I want to access my files via SSH so that I can retrieve them directly.
U34	Mask creation	2	As a user, I want to create masks for both 2D and 3D images so that I can improve visibility and analysis quality.
Total Points			**51**

3.3 System Requirements

ID	Title	Points	Description
S0	File organization	2	The pipeline will create a uniform file structure when accepting raw TIFF and OIR files.
S1	File renaming	1	The pipeline will automatically rename image data files to create uniform scaffolding for future processing steps.
S2	Field of view aggregation	1	The 2D pipeline will aggregate data from all fields of view into a single summary file when the pipeline is complete.
S3	Wrapping analysis	1	The 3D pipeline will generate a wrapping analysis when the pipeline is complete.
S4	Ezra's algorithm	2	The 3D pipeline will run Ezra's algorithm when selected in the settings.
S5	Start date and time	1	The pipeline will automatically record the starting date and time of a data processing session when a session begins.
S6	Script runtime	1	The pipeline will automatically record the runtime of scripts executed during a session, only when a script is run.

ID	Title	Points	Description
S7	End date and time	1	The pipeline will record the date and time when a session finishes after producing output and being ended by the user.
S8	Debris calculation	1	The 2D pipeline will calculate debris present in images when the 2D pipeline is selected.
S9	Overlapping calculation	2	The pipeline will calculate overlapping myelin present in images when either the 2D or 3D pipeline is selected.
S10	Myelin growth calculation	2	The pipeline will calculate myelin growth on artificial axons found in images after overlapping myelin is calculated.
S11	Calculation results	4	The pipeline will consolidate calculation results into appropriate CSV files after all calculations are performed.
S12	Ezra's results	1	Ezra's algorithm will consolidate its results with the main CSV file after being selected and all other calculations are completed.
S13	Session history	1	The pipeline will record which pipeline steps were executed and which steps failed during a session run.
S14	Input validation	1	The system will validate all user-provided inputs to prevent invalid or out-of-range values.
S15	Graceful failure handling	2	The pipeline will allow partial results to be preserved if a processing stage fails.
Total Points			**25**

4 System Design

4.1 Architecture

This project adopts a hybrid architecture that combines Pipeline and Model–View–Controller (MVC) patterns to balance modular data processing with a clean, maintainable user interface structure.

The pipeline architecture is used to structure the core processing workflow, where data flows through a series of well-defined stages. Each stage performs a specific transformation or analysis and passes its output to the next stage. This approach improves modularity, simplifies debugging, and allows individual components to be modified or extended without affecting the rest of the system. It also allows our client to have a high-level understanding of their own data processing session when it runs.

The MVC architecture is applied to the user interface and front-end components, particularly for managing and reusing saved configurations. The Model represents application data and configuration states, the View handles user-facing presentation, and the Controller manages user interactions and coordinates updates between the Model and View. The controller will also manage file retrieval for any selection and preview thumbnails. This separation of concerns improves maintainability, supports clearer organization of UI logic, and enables future expansion of features with minimal coupling.

Together, these architectural patterns provide a scalable foundation that supports both efficient data processing and a flexible, user-friendly interface.

4.2 Diagrams

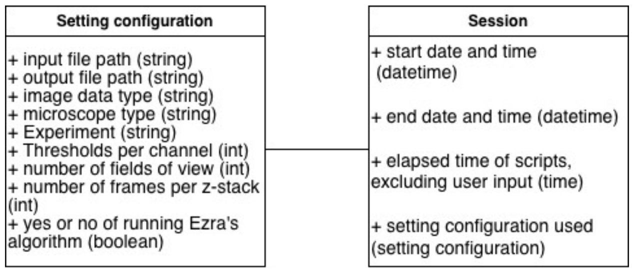

Setting configuration
+ input file path (string)
+ output file path (string)
+ image data type (string)
+ microscope type (string)
+ Experiment (string)
+ Thresholds per channel (int)
+ number of fields of view (int)
+ number of frames per z-stack (int)
+ yes or no of running Ezra's algorithm (boolean)
Session
+ start date and time (datetime)
+ end date and time (datetime)
+ elapsed time of scripts, excluding user input (time)
+ setting configuration used (setting configuration)

Figure 1: Class diagram of proposed objects, setting configurations and settings

4.3 Technology

This project will be reflecting their original pipeline's code The primary programming language is Python, which serves as the core language for application logic, orchestration, and integration between components. Bash scripts are used to automate pipeline execution and manage system-level tasks. We'll also be using Python's datetime library to store date and time objects for session history.

For the user interface, the project uses Tkinter, a lightweight Python library for building graphical user interfaces. Integration between Python and ImageJ is achieved using pyimagej, allowing ImageJ functionality to be accessed and controlled directly from Python code, no need to use ImageJ as a separate application. This enables seamless coordination between image processing tasks and the rest of the application logic.

All testing is conducted using pytest which provides a structured and automated testing framework to validate individual components and ensure overall system correctness. Data generated and consumed by the application is stored locally as per the client's request as well as general sanity.

4.4 Coding Standards

general coding standards

- Code should prioritize readability and clarity over clever or overly compact implementations.
- Follow consistent naming conventions across all files, functions, and variables.
- Avoid duplication by reusing functions and scripts where possible.
- All code must be version-controlled and include meaningful commit messages.
- All commits must be linked to a respective task in github via its commit message
- use UPPER_SNAKE_CASE for constants
- use lower_snake_case for local variables
- keep python and bash scripts in separated directories
- have meaningful file names

python coding standards

- use snake_case for variables and functions
- use PascalCase for class names
- include docstrings for all modules, classes, and public functions.
- include explicit error handling with try/except statements

bash scripting standards

- make sure all scripts begin with a shebang.
- avoid hard-coded paths. Opt for variables instead

testing and validation standards

- unless you physically can't, all code should be tested
- tests should be repeatable and not depend on an external state unless explicitly simulated to do so

4.5 Data

The main data structures that will be stored are setting configurations and sessions:
setting configuration are comprised of:

- input file path (string)
- output file path (string)
- image data type (string)
- microscope type (string)
- Experiment (string)
- Thresholds per channel (int)
- number of fields of view (int)
- number of frames per z-stack (int)
- yes or no of running Ezra's algorithm (boolean)

a session will contain the following:

- start date and time (datetime)
- end date and time (datetime)
- elapsed time of scripts (excluding user input)
- setting configuration used (configuration object)

all data objects will be stored as simple txt files in local file storage, not utilizing a database.

4.6 UI Mocks

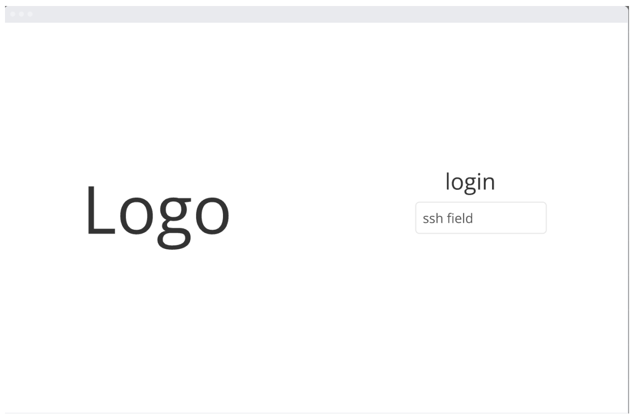


Figure 2: login page for ssh

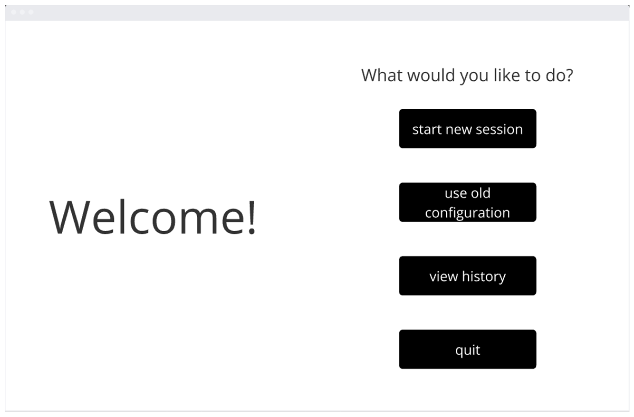


Figure 3: landing page after login

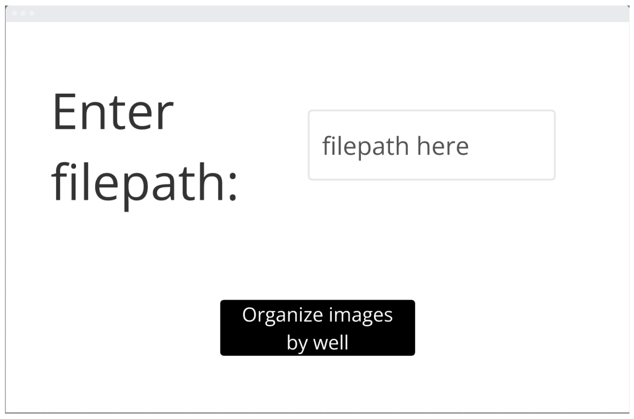


Figure 4: image data filepath input menu to access image data files for session

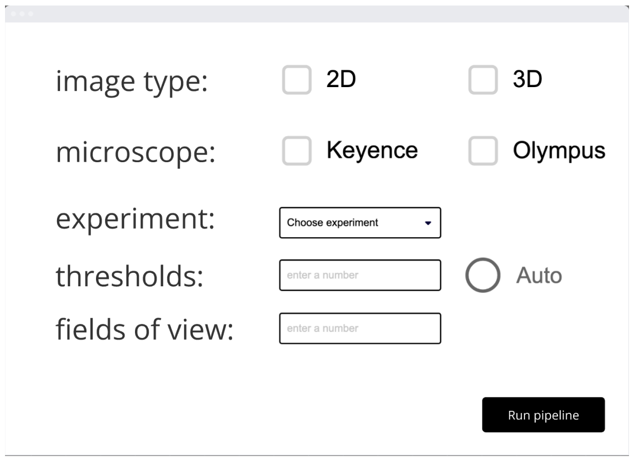


Figure 5: 2D setting inputs

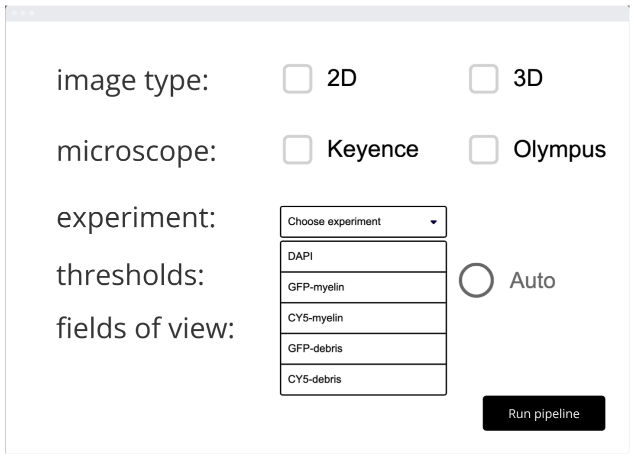


Figure 6: dropdown menu example using 2D image

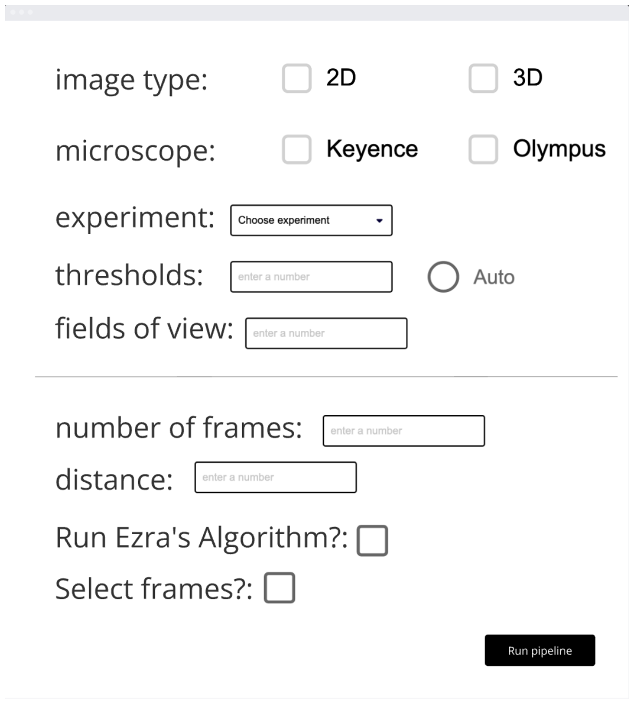


Figure 7: expanded 3D image data settings

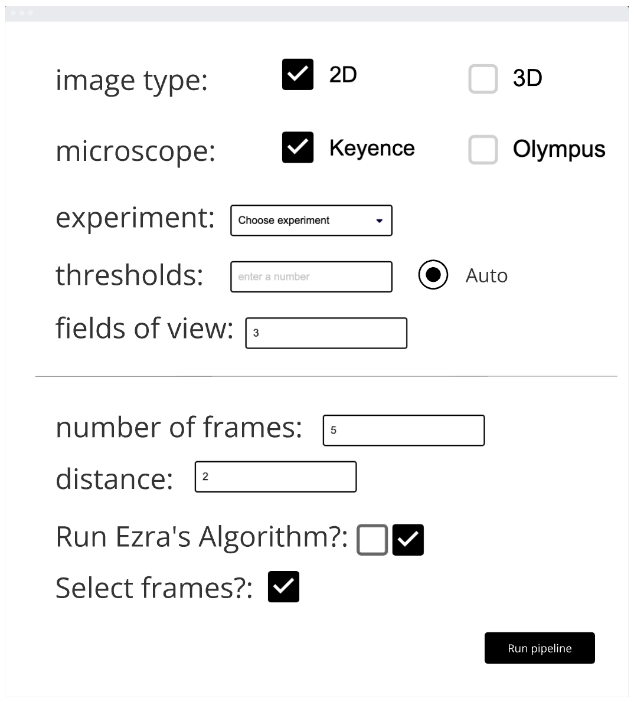


image type: [✓ 2D] 3D

microscope: [✓ Keyence] Olympus

experiment: Choose experiment

thresholds: enter a number [✓ Auto]

fields of view: 3

number of frames: 5

distance: 2

Run Ezra's Algorithm?: ☐ ✓

Select frames?: ✓

Figure 8: example layout of auto-filled settings

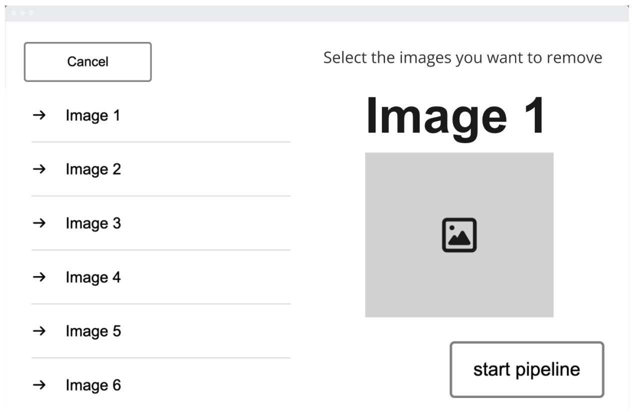


Figure 9: image preview menu

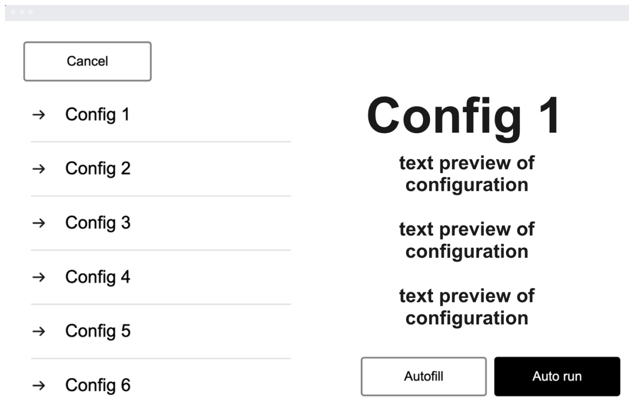

Cancel
Config 1
Config 2
Config 3
Config 4
Config 5
Config 6
Config 1
text preview of configuration
text preview of configuration
text preview of configuration
Autofill
Auto run

Figure 10: configuration preview menu

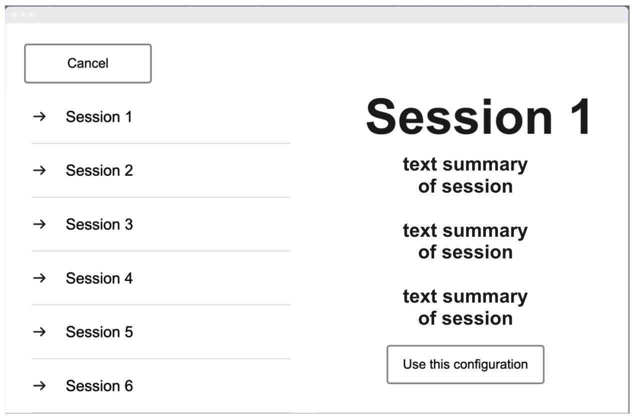


Figure 11: session history menu

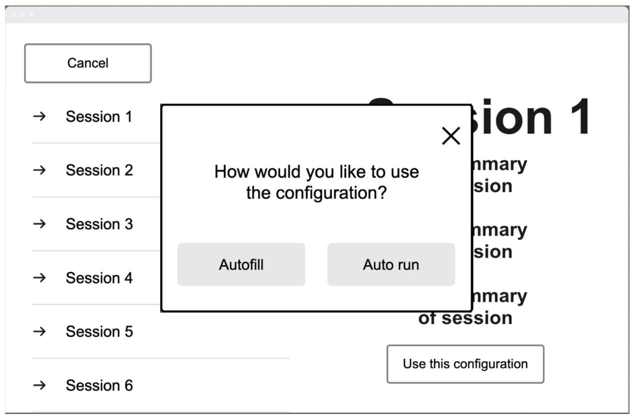
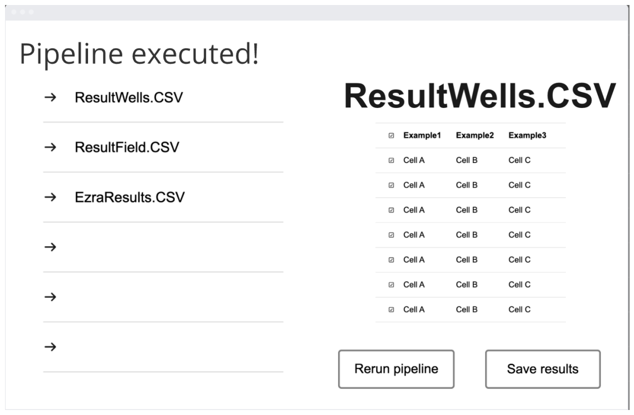


Figure 12: Session history preview with popup for starting a session with contained configuration



Example1	Example2	Example3
Cell A	Cell B	Cell C
Cell A	Cell B	Cell C
Cell A	Cell B	Cell C
Cell A	Cell B	Cell C
Cell A	Cell B	Cell C
Cell A	Cell B	Cell C
Cell A	Cell B	Cell C

Figure 13: result preview with example result files

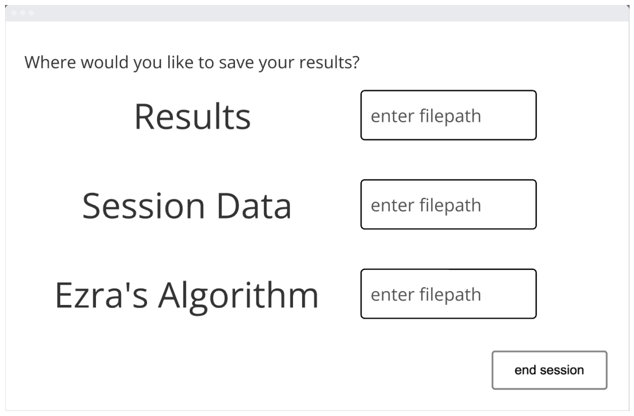


Figure 14: result saving menu

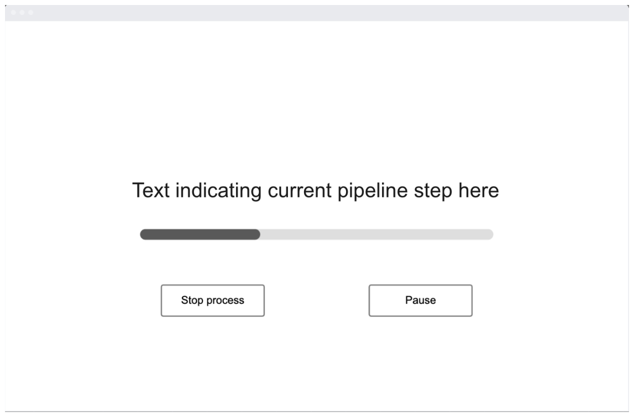


Figure 15: loading screen for running pipeline example

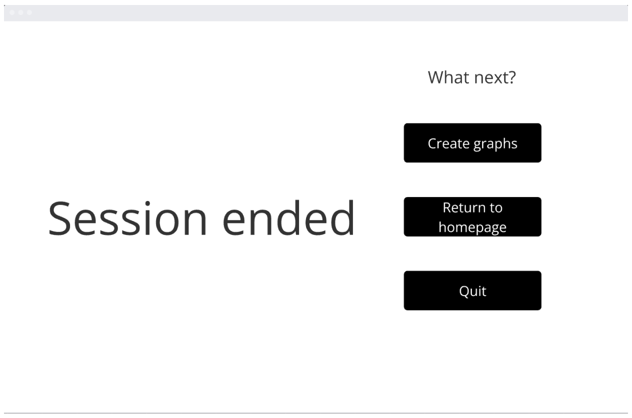


Figure 16: session end screen

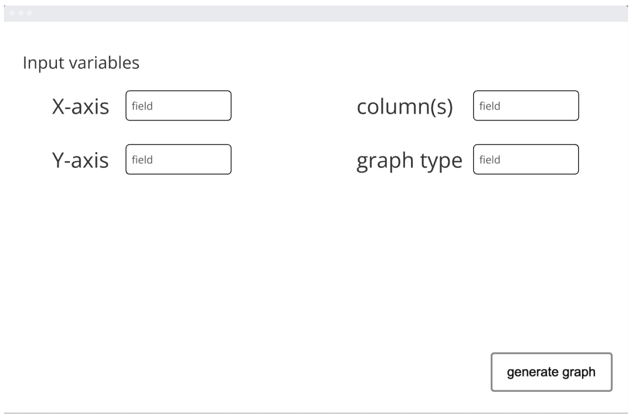


Figure 17: graph creation menu

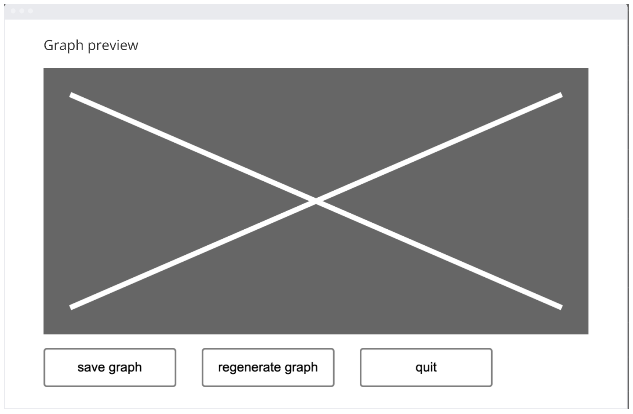


Figure 18: graph preview

5 Iterations

5.1 iteration 1: Pipeline foundations (Delivery: Feb 10th)

ID	Title	Points	Description
U1	Data upload	1	Upload data files or folders for processing.
U2	Setting inputs	2	Input customizable data-processing settings.
U7	2D or 3D pipelines	1	Choose between 2D and 3D pipelines.
U11	Microscope selection	1	Select which microscope produced the data.
U21	Session start	1	Start a data-processing session.
S0	File organization	2	Create a uniform file structure for raw image files.
S1	File renaming	1	Automatically rename files for uniform scaffolding.
S5	Start date and time	1	Record the start time of a session.
S14	Input validation	1	Validate all user-provided inputs.
Total Points		**11**	

5.2 iteration 2: image processing (Delivery: Feb 24th)

ID	Title	Points	Description
U3	Channel selection	1	Select one or more channels for 2D experiments.
U4	Image preview	2	Preview images after organization.
U5	Select images	1	Select organized images for a run.
U6	Delete images	1	Delete organized images without reorganizing files.
U8	Intensity threshold	1	Set or auto-select an intensity threshold.
U9	Field of view amount	1	Specify number of fields of view per well.
U33	SSH access	1	Access files directly via SSH.
S2	FOV aggregation	1	Aggregate all FOVs into a single summary file.
S8	Debris calculation	1	Calculate debris when the 2D pipeline is selected.
Total Points		**10**	

5.3 iteration 3: configuration management (Delivery: March 17th)

ID	Title	Points	Description
U10	Channel selection (stains)	1	Select channels used for staining and debris reduction.
U12	Create configuration	2	Create reusable processing configurations.
U13	Read configuration	2	View saved configurations.
U14	Update configuration	2	Modify saved configurations.
U15	Delete configuration	2	Remove unused configurations.
U16	Z-stack frames	1	Specify frames per z-stack.
U17	Frame distance	1	Specify distance between frames.
S3	Wrapping analysis	1	Generate wrapping analysis for 3D data.
S4	Ezra's algorithm	2	Run Ezra's algorithm when selected.
Total Points		**14**	

5.4 iteration 4: 3D image processing (Delivery: March 31st)

ID	Title	Points	Description
U19	Preview configuration	1	Preview configuration metadata before selection.
U22	Session end	2	End or stop a running session.
U23	Session history	2	View metadata of previous sessions.
U24	Choose configuration	1	Select a saved configuration before a run.
U25	Auto-fill	1	Automatically run the pipeline using a configuration.
U26	Rerun	1	Reuse settings from a previous run.
U27	Pause a run	1	Pause or stop a running pipeline.
U28	Image recovery	1	Recover images excluded from processing.
U29	Progress monitor	1	View current pipeline execution stage.
S6	Script runtime	1	Record runtime of executed scripts.
S7	End date and time	1	Record session completion time.
S13	Session history	1	Record executed and failed pipeline steps.
S15	Graceful failure handling	2	Preserve partial results if a stage fails.
Total Points		**16**	

5.5 iteration 5: final outputs (Delivery: April 14th)

ID	Title	Points	Description
U18	Text file generation	2	Generate editable script text files.
U20	Preview cherry-pick images	1	Preview images before 3D selection.
U30	Export failed files	3	Export scripts with previous settings after failure.
U31	Preview results	2	Preview results before ending a session.
U32	Preview graphs	5	Generate graphs to visualize results.
U34	Mask creation	2	Create masks for 2D and 3D images.
S9	Overlapping calculation	2	Calculate overlapping myelin.
S10	Myelin growth calculation	2	Calculate myelin growth metrics.
S11	Calculation results	3	Consolidate outputs into CSV files.
S12	Ezra's results	1	Merge Ezra's results into final output.
Total Points		**23**	

5.6 Iteration/Sprint 1

5.6.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

5.6.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.6.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

Figure 19 shows the test coverage

`figures/placeholder.png`

Figure 19: Iteration 1 test coverage report

5.6.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

5.7 Iteration/Sprint 2

5.7.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

5.7.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.7.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

5.7.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

5.8 Iteration/Sprint 3

5.8.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

5.8.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.8.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

5.8.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

5.9 Iteration/Sprint 4

[CS496 has 5 sprints. CS482 only has only 3 sprints (remove Iterations 4 and 5 from this doc if you are writing a doc for 482]

5.9.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

5.9.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.9.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

5.9.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

5.10 Iteration/Sprint 5

5.10.1 Planning

[Which stories did you plan for this iteration/sprint. Add the total points for this plan. You can also explain the reason behind your planning, and what major feature(s) your team is focusing on delivering by completing these stories. You may use a table for a summary display of the planning, but elaborate in text more detail in your focus and feature plan.]

5.10.2 Work Done

[Which stories did you complete in this iteration/sprint. Which ones did you partially complete? Who worked on which story? You may elaborate in paragraph(s) to add more detail about the work done.]

5.10.3 Testing Coverage

[Testing is very important. Show your coverage here. Is this coverage good enough? Explain why you think so. Is it not good enough? Explain a plan to increase the coverage. You may also elaborate on why some artifacts do not undergo much testing. If the testing changed from the last iteration, explain the reasons.]

5.10.4 Retroespective & Reflection

[What were the pitfalls, challenges, and issues you had in this iteration? How can you address them to improve the process in the next iteration? Did anything not go according to plan? Why so and how to avoid the same mistake? Write a personal reflection on what you learned in this iteration (even if a small technical thing like Database storage).]

6 Final Remarks

6.1 Overall Progress

[Have you completed everything? If so, present evidence on how you brought value to your client, and the overall client satisfaction. Otherwise, estimate how much progress you done and how long it would take to finish this project. Be concrete about your progress, you know how many story points your software is, how many points you completed (this shows your progress). You also how many points your team delivers at each iteration, therefore you can estimate how many more iterations it would take to finish the leftover points (show the math).]

6.2 Project Reflection

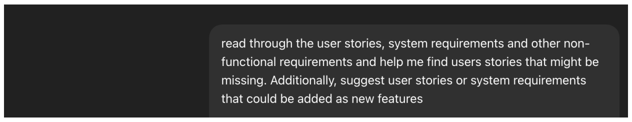
[Your personal reflection on the project. What lessons did you learned. What would you have done differently? How can you do better work in future projects? You may write this as a team or per person (or both — if all your iterations were team reflections, then it would be better to write individual reflections here)]

Appendix

[Appendix section if needed]

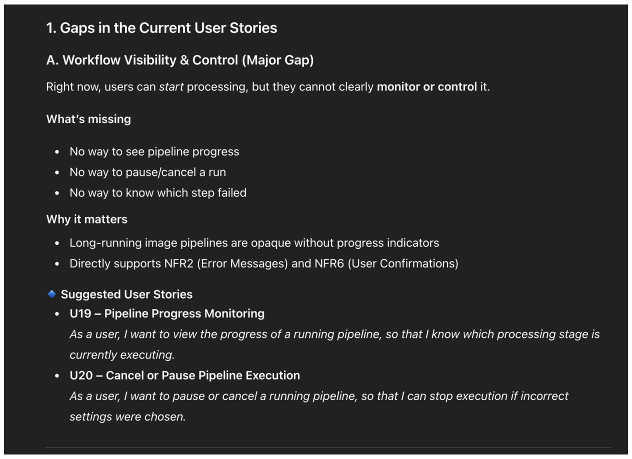
AI usage

AI was used to revise user stories



read through the user stories, system requirements and other non-functional requirements and help me find users stories that might be missing. Additionally, suggest user stories or system requirements that could be added as new features

Figure 20: user story message with ChatGPT



1. Gaps in the Current User Stories

A. Workflow Visibility & Control (Major Gap)

Right now, users can *start* processing, but they cannot clearly **monitor or control** it.

What's missing

- No way to see pipeline progress
- No way to pause/cancel a run
- No way to know which step failed

Why it matters

- Long-running image pipelines are opaque without progress indicators
- Directly supports NFR2 (Error Messages) and NFR6 (User Confirmations)

🔹 **Suggested User Stories**

- **U19 – Pipeline Progress Monitoring**
  As a user, I want to view the progress of a running pipeline, so that I know which processing stage is currently executing.
- **U20 – Cancel or Pause Pipeline Execution**
  As a user, I want to pause or cancel a running pipeline, so that I can stop execution if incorrect settings were chosen.

Figure 21: user story response from ChatGPT

6.3 reflection

In terms of helpfulness, it was actually really helpful in finding gaps. It looked at my existing user stories and caught some inherent user stories that were missing. For example, we had start and stop session stories but there was no pause story. It also provided pre-made resulting user stories, which I did not use, and it overestimated points sometimes.