ElyteOS: Software for Automated Electrolyte Preparation, Measurement, and Data Management

Yuhan Chen[a], Hongyi Lin[b], Tianyi Zhang[b], Adarsh Dave[c], Jared Mitchell[d], Jay Whitacre[c,e], Venkatasubramanian Viswanathan[a,b]

[a]*Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan, 48109, United States*
[b]*Department of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan, 48109, United States*
[c]*Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, United State*
[d]*Stratus Materials Inc., 6901 Lynn Way, Suite 320 Pittsburgh, Pennsylvania, 15208*
[e]*Wilton E. Scott Institute for Energy Innovation, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, United State*

Abstract

ElyteOS is a graphical user interface written in Python 3.8.3 which enables the automation of the processes of electrolyte preparation, measurement, data storage, and data visualization. It provides a user-friendly interface and acts as a framework for automating lab equipment with different commands as well as managing the procedure of the experiments. Meanwhile, ElyteOS automatically saves the experimental data in a database and provides data searching and visualization tools for researchers. Therefore, ElyteOS not only facilitates electrolyte researches but also provides potential in optimizing battery performance and advancing battery technologies.

Keywords: Laboratory automation, Graphical user interface, Database, Data visualization, Electrolyte, Battery

Metadata

Nr.	Code metadata description	Metadata
C1	Current code version	v1.0.1
C2	Permanent link to code/repository used for this code version	`https://github.com/BattModels/Clio.git`
C3	Permanent link to Reproducible Capsule	None
C4	Legal Code License	MIT
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	Python 3.8.3, SQLite
C7	Compilation requirements, operating environments & dependencies	See requirements.txt in Github repository
C8	If available Link to developer documentation/manual	None
C9	Support email for questions	cyhalek@umich.edu

1. Motivation and significance

Batteries are crucial in energy storage and transportation.[1] As one of the most commonly used types of batteries, lithium ion batteries transport Li ions between the anodes and cathodes through liquid electrolytes during operation.[2] Finding the optimal formulations of solvents and salts in electrolytes can significantly improve functionality of batteries, e.g. higher conductivity electrolyte can enable faster charging,[3] electrolytes with lower melting point can enable low temperature operation,[4] electrolytes with high flash point reduce flammability and improve safety.[5] A typical commercial Li-ion battery electrolyte is a formulation of more than two solvents, one or two salts and more than two additives.[6] Therefore, optimization of electrolyte formulations is laborious work as there exists a huge number of different combinations of possible solvents and salts.[7] The adoption of lab automation in various fields, including biology [8], material science[9], and optical physics [10], has demonstrated its potential for scientific discovery and optimization. Incorporating lab automation into electrolyte research to optimize electrolyte formulation could revolutionize the field, unlocking faster breakthroughs.[11]
In a previous work, we introduced a robotic platform named 'Clio' which makes the optimization process more efficient by integrating a balance, a potentiostat and a viscometer, along with accessories including a thermometer, valves, and pumps, to automate the electrolyte property measurement. Combining the automation pipeline with Bayesian optimization, we demonstrated optimization in a known chemical space of solvents (ethylene carbonate (EC), ethylmethyl carbonate

(EMC), and dimethyl carbonate (DMC)) and $LiPF_6$ as salt. This demonstrated the potential for an acceleration factor of 11.5 relative to manually conducting measurements.[11]

Clio has some fundamental limitations for interoperability which we address in this work. Firstly, the software control of Clio is not easily maintainable as it is controlled through LabView.[11, 12] Secondly, Clio is operated through command-line Python. The lack of a GUI has reduced the accessibility of Clio, as it requires a higher learning curve to new users.[12] Thirdly, Clio does not provide a status management process, which makes it difficult to track experiments, especially when encountering errors. Lastly, Clio does not incorporate any built-in data visualization function. After making batches of measurements, Clio stores all the information in an AWS server, but users need to download the data and plot the data manually.

Motivated by the improvements a new software could bring to further accelerate electrolyte autonomous experimentation, we have written a software named ElyteOS to solve these limitations. The software keeps all the functions of Clio and simultaneously provides solutions to the problems discussed above. Firstly, ElyteOS is now fully written in Python, bringing flexibility in maintenance and easy usage. Secondly, ElyteOS offers a GUI, which helps users with minimal computer science background to control Clio easily. In addition, the GUI incorporates log pages to check the status of the equipment, which is crucial in the tracking and handling of errors. It also has pages to track the inventory and the potential good data. The GUI also provides tabular and graphical data visualization for users to easily analyze the experiment data. The detailed functionalities of the software are introduced in the next section.

2. Software description

Overview: programming language selection ElyteOS is a software aimed at automating the process of electrolyte preparation, measurement, data storage and visualization. The measurement is conducted by Clio, a robotic system controlled by ElyteOS, which consists of a series of commutable instruments for electrolyte preparation and measurement. Previously, Clio was controlled via the Labview platform. In this work, Python is chosen over Labview as the primary language for ElyteOS and Clio due to its modularity, shareability, maintainability, and support for version control.[11]

Primary Frameworks The rebuilt Clio uses the PySerial framework to communicate to a potentiostat, a viscometer, a balance, two pumps, and three 10-port valves by serial signal. It also uses the HIDapi framework to send HID commands to a thermometer and some relays (which then controls 3-way valves to direct flow

and a DC motor to stir the electrolyte mixture). The detailed structure is provided in figure 1.

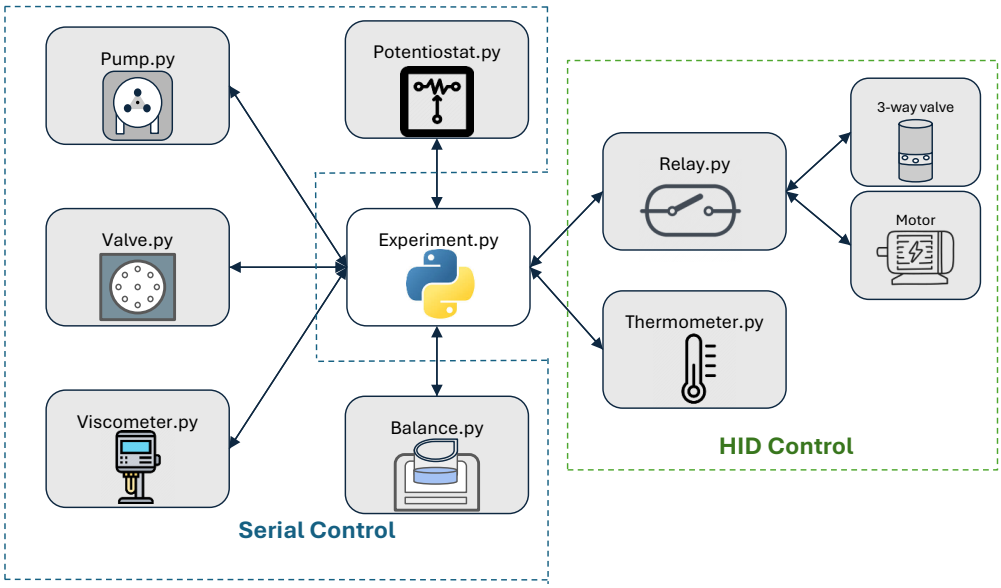

Figure 1: Overall structure of Clio: Experiment.py serves as a manager to call the surrounding python files, which thereby send commands to the equipments. The balance, the potentiostat, the pumps, the 10-port valves, and the viscometer are controlled by serial commands, and the relay (which then controls the 3-way valves and the motor) and the thermometer are controlled by HID commands.

On the software end, ElyteOS uses PyDash library as the primary framework. PyDash wraps JavaScript in Python and facilitates back-end and front-end connection. This reduces the complexity of the software, as it only requires importing PyDash package without any dependencies, instead of using JavaScript and connecting to python via Flask.[13]

Overall Architecture ElyteOS is launched with the python file Main.py, with a total of six pages, as shown in 2, with four belonging to the experiment section and two to the database section. The detailed structure is provided in S1, in supplementary information. The screenshots of the pages are also provided in pictures S2, S3, S4, S5, S6, S7 in supplementary information.

2.1. Experiment section

2.1.1. Experiment architecture

The experiment section consists of four pages, each having a specific function:

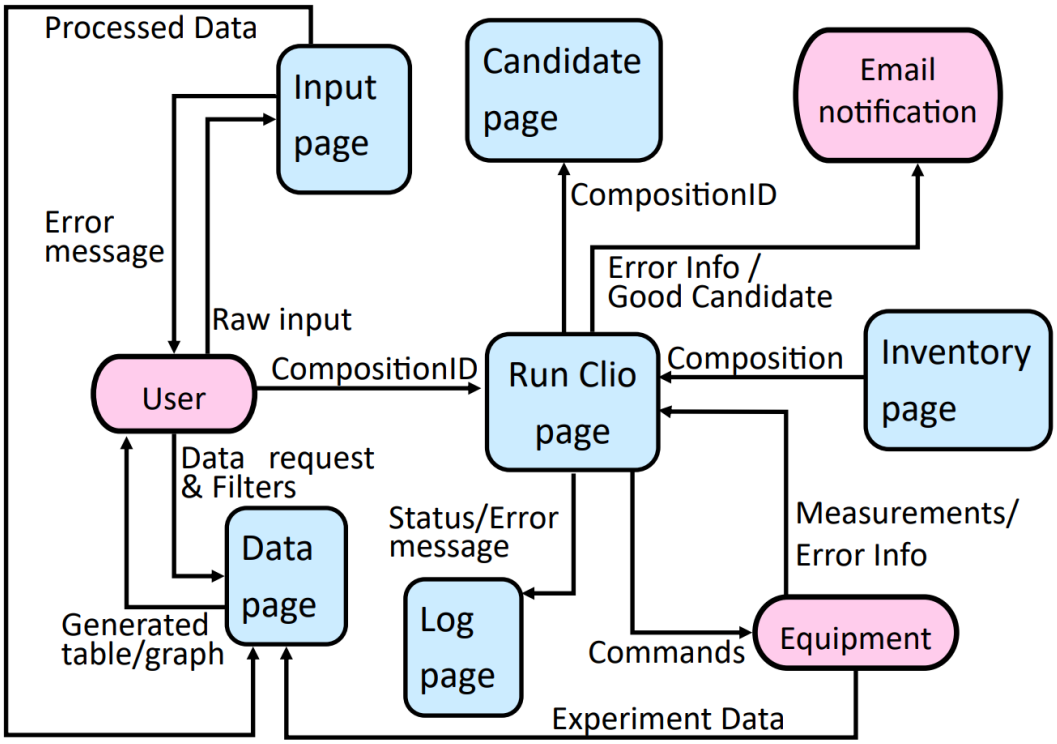

Figure 2: Pages of ElyteOS and their information communication: When conducting experiments, the user submits a compositionID to the Run Clio Page. Then, the Run Clio Page requests inventory information from Inventory Page, conduct experiment by calling the equipments, and log equipment status and errors on Log page. After running the experiment, the data is saved into Data Page, and candidate page keeps compositions with results meeting the user-selected benchmarks. Email notifications will also be sent when a such candidate is found, or an error happened in the experiment. The user can then look for visualizing experiment results on Data Page after running experiments. The user can also directly input data on Input Page without running experiments. The blue colored items represent a page, and the pink colored items represent the elements outside the software.

- **Run-Clio Page**: Serves as the primary interface for the organization of experiments.
- **Inventory Page**: Tracks, stores and displays remaining materials stored in Clio inventory, and stores all information in a CSV file.
- **Candidate Page**: Records and displays of promising electrolyte candidates with favorable electrolyte properties specified by the users, and all information is stored in a CSV file.
- **Log Page**: Monitors and documents the status of each piece of equipment in Clio during experiments.

2.1.2. Experiment functionalities

Experiment token It is important to determine the details of the experiment–in this case, the composition of the electrolyte needed to be measured. Therefore, ElyteOS uses tokens to keep track of the compositions. A single experiment token is represented by a composition ID with four sections. The first section specifies the types of solvents used; the second section indicates the mass fraction of each type of solvent in the mixture; the third section identifies the types of salts used; and the final section represents the molality of each salt in the mixture. Sections are separated by a vertical bar, and items within the same section are separated by an underscore. For example, an electrolyte solution composed of 60% dimethyl carbonate (DMC) and 40% ethyl methyl carbonate (EMC) as solvents, lithium hexafluorophosphate ($LiPF_6$) and lithium bis(fluorosulfonyl)imide (LiFSI) as salts with molalities 1 mol/kg and 1.5 mol/kg, respectively, would have the CompositionID: DMC_EMC|60_40|LiPF6_LiFSI|1_1.5. ElyteOS uses the encoded information to prepare the mixture and measure its properties[11]

Experiment status The experiment status is also important as it mainly tracks the Clio status and determines the starting of ongoing experiments. It is governed by two variables: is_running and error_status. The is_running variable checks whether an experiment is in progress and is represented with two possible values: "No experiment is running" and "running experiments. The error_status variable checks whether Clio encounters an error with values corresponds to the specific error Clio encounters. If Clio does not encounter an error, the error_status variable is set to "No errors".

Both experiment tokens and experiment statuses are crucial in the experiment processes, which are addressed in the following paragraphs.

Processes The experiment flow consists of three processes: the User process, the Manager process, and the Experiment process. The three processes run independently, except for the commands sent by the User process and the Manager process to launch or terminate the experiment process. The processes are managed by the Multiprocess Package in Python.

User process The User process mainly interacts with the users, who provide the tokens, i.e. the compositionIDs. Then, the process verifies the validity of IDs with an ID validator (which will be covered in later sections). After the verification of IDs, the process adds the tokens into the experiment queue. The flowchart of User process is provided in S8 in supplementary information.

In addition to adding the experiment tokens, the User process also controls the error status of the experiment. During the process of an experiment, the user has the option to click on the "stop Clio" button, which sets the status of the experiment from "No errors" into "Clio is Locked", and halts the experiment process. Furthermore, when Clio encounters an error during an experiment, the

experiment process will halt automatically and update the error status to reflect the error type Clio encountered. In such a case, the user can click the "trouble resolved" button to reset the error status to "No errors" and restart the experiment.
Manager process The Manager process primarily operates an interval, which checks every 0.1 seconds to determine whether to launch the experiment process. If the "is_running" variable indicates no experiment is running, the "error_status" variable shows no errors, and the experiment queue is not empty, the experiment process pops the first token from the experiment queue, passes the token into the experiment process, sets the "is_running" variable to be "Running experiments", and starts the experiment process. Otherwise, it will not perform operations until the next check is made. The flowchart of the Manager process is provided in S9 in supplementary information.
Experiment process The experiment process is the core of the execution of the experiment, guided by the operation manager. It starts by receiving the token, or CompositionID, and then parses it into the information for the inventory manager, which then computes, logs, and returns the required quantities for each component in the inventory. If the inventory is infeasible for the composition to be prepared, the inventory manager returns an error. Otherwise, it updates the inventory after the composition is prepared.
After that, the operation manager calls the equipment manager, which repeatedly generates commands to control the equipment. The status of the equipment is logged after each execution of a command. In case of errors in any piece of equipment, the experiment manager logs the errors, stops the experiment process, and returns the error. The operation manager then sends users an email notification regarding the error message so that the errors can be resolved promptly.
If an experiment completes without errors, the operation manager sends the measured information to the candidate manager, which evaluates whether the electrolyte meets the desired requirement to determine whether to store or discard the composition. Whenever an electrolyte with desired properties is found, the operation manager sends an email notification to the users.
Finally, the operation manager stores the measured data of the composition in the database and sets the is_running variable into "No experiment is running" regardless of whether the composition has the desired properties, completing the whole measurement process. The flowchart of Experiment process is provided in S10 in supplementary information.

2.2. Database section

2.2.1. Database architecture

As described above, ElyteOS itself provides a database to store the experiment data by offering two pages for the database in addition to the pages for the ex-

periment section. Currently, the database uses SQLite as the primary database to store all data from the experiment.

Why SQL Currently, databases can be categorized into two types: structural (SQL) and nonstructural (NoSQL, such as MongoDB). A structural database is chosen as the primary database for ElyteOS due to the simple and tabular structure (The detailed format will be covered in the next section). Additionally, the advantages of structured query language perfectly satisfy the requirements of the database. The advantages include safety, which ensures data reliability and the ACID property; standardization, which facilitates softwares to adapt to other structural databases; strong queries capabilities, which help to perform complex operations including data filtering; and space efficiency, which allows the software to store a large volume of experimental data.

The lack of SQL's flexibility can be resolved by allowing the database to have multiple tables to accommodate more complex structures. This approach effectively resolves the limitation of SQL and meets the requirements of the ElyteOS database.

Experiment Data Structures As SQL requires structured data, it is important to breakdown the components of an electrolyte experiment data point, which is listed below:

ID: Unique identifier 32-byte integer generated through special algorithms

CompositionID: String

Density, Conductivity, Viscosity, Mass, Volume, Resistance, Temperature: all 64-bit floats

Date: datetime object

Trial Number: 64-bit integer

In order to facilitate data filtering (which will be addressed in later paragraphs), the data structure is modified when writing the data into the database. The variables stored inside the database can be categorized into six groups: dependent variables, independent variables, solvent mass percentage, salt molality, solvent molar ratio, and salt molar ratio. Density, Conductivity, Viscosity, Mass, Volume, and Resistance are dependent variables. Meanwhile, Temperature, Date and Trial Number are independent variables.

The remaining categories are not stored directly from data points. Instead, they are parsed from the compositionID, which contains information about the electrolyte components, including a tabular representation of the percentage of solvents and molality of salts. Tables S1, S2, S3, S4 and S5 illustrate the overall structure of the database.

In the tables above, the same ID in each table is associated with the same piece of data record. When performing data retrieval, the tables are joined on the IDs to display the complete experiment records.

2.2.2. Database functionalities

Data insertion Although Clio is designed to upload experiment data into the database automatically, ElyteOS allows the users to input the experiment data. The users can either manually enter data directly with the given text fields or upload a CSV file for the software to parse. After submission, the data validator verifies the validity of the input.

The validator verifies a piece of data records by checking whether it satisfies all of the following criteria. Firstly, the validator checks whether all numeric inputs fall within the correct range (typically above zero). Secondly, it ensures that the compositionID satisfies correct format. Specifically, each compositionID must consist of four sections with the required separators. Thirdly, the number of solvents must match the number of the corresponding percentages, and the number of salts must match the number of the corresponding molalities. Moreover, the percentage values of solvents mass ratios must be between 0 and 100 and sum up to 100.

After the validation process, the data is parsed into the correct format and a query generator generates a series of queries to insert the data into a database. In order to ensure atomicity (i.e., avoid partial or inconsistent uploads), the changes will only be committed after all the queries are executed.

Data filtering After entering the data into the database, it is important to retrieve the data, which contains two steps: data filtering and visualization. Data filtering is important as retrieving all stored data and manually identifying necessary ones is impractical. Therefore, ElyteOS offers a data filtering function, enabling users to extract necessary data under predefined criteria.

The filtering criteria are simple: users can define whether a set of given variables falls within particular ranges. For each variable, the users can check the corresponding checkbox to include it in the visualization. After selection, a two-column table is generated for each selected variable: one column contains the ID, the other contains the corresponding variable's values.

After defining criteria for individual variables, users can specify the relationships between the criteria. ElyteOS enables the users to select whether the criteria associated with a certain type of variable should have "and" or "or" relations. The relationships determine whether the corresponding tables are inner-joined (and-relationships) or outer-joined (or-relationships).

Data visualization: After a successful join, ElyteOS allows the user to select whether to display the joined table or graphs. Displaying a table is straightforward as PyDash can directly display tables through dataframes. For graphs, however, more complex algorithms is required.

Displaying the graph: Plotly is used as the primary package for plotting the data, as it is compatible with dash and convenient for users to analyze the data interactively. Plotly supports 2D and 3D graphs, therefore it allows at the maximum

of three axes (i.e., two independent variables and one dependent variable). When only one independent variable is selected, a series of 2D plots will be generated, each representing the selected independent variable plotted against each dependent variable. When multiple independent variables (specifically, the ones related to solvents and salts) are selected, ElyteOS looks for all pairwise combinations of the variables, and each pair is plotted against all selected dependent variables. For example, if the user selects 5 independent variables and 3 dependent variables, ElyteOS generates a total of $3 \times 5 \times (5-1)/2 = 30$ plots.
In summary, the ElyteOS software achieves the pipeline of lab automation by managing the experiments, inventory, equipments, and storing the data for filtering and retrival as well. The automation processes reduce the time for the users to manually perform experiments and manage data.

3. Illustrative examples

The software functions described above are crucial in electrolyte research and in the development of the automation system itself. The next part provides two examples: a grid search example and a calibration example. The grid search example demonstrates how the experiment system and the database system interact to advance electrolyte research, while the calibration example shows the importance of the database in constructing the automation pipeline.

3.1. Electrolyte Experiment

The importance of GUI in running experiments and performing data analysis can be demonstrated in a grid search example on the molality of the salt in aqueous electrolytes. In this experiment, the relationship between the molality of Li_2SO_4 salt in water and the resulting ionic conductivity is tested. A total of six compositions are selected, which include 0.43 mol/kg, 0.86 mol/kg, 1.29 mol/kg, 1.72 mol/kg, 2.15 mol/kg, and 2.58 mol/kg. Each composition is measured twice.
The measurement starts by creating a CSV file containing a list of CompositionIDs. The general format of the IDs follows: H2O|100|Li2SO4|x, where x denotes the selected molality of Li_2SO_4. Each ID is repeated twice. The CSV file for this experimental campaign is provided in tables titled `CompositionIDs_to_run.csv`.
In this experiment, the ionic conductivity threshold is set at 55 mS/cm, indicating electrolyte candidates with good ionic transport capabilities. After running the experiments, the compositions that meet the criteria (H2O|100|Li2SO4|1.72, H2O|100|Li2SO4|2.15, and H2O|100|Li2SO4|2.58) are displayed on the good candidate page, and all experimental results are stored in the database. During data retrieval, the filtering and sorting criteria applied to the database are depicted in figure 3. The generated table is provided in tables titled `Grid_search_result.csv`,

the generated plots are provided in supplementary information figures S13 and S14, and the processed graph is shown in figure 4.

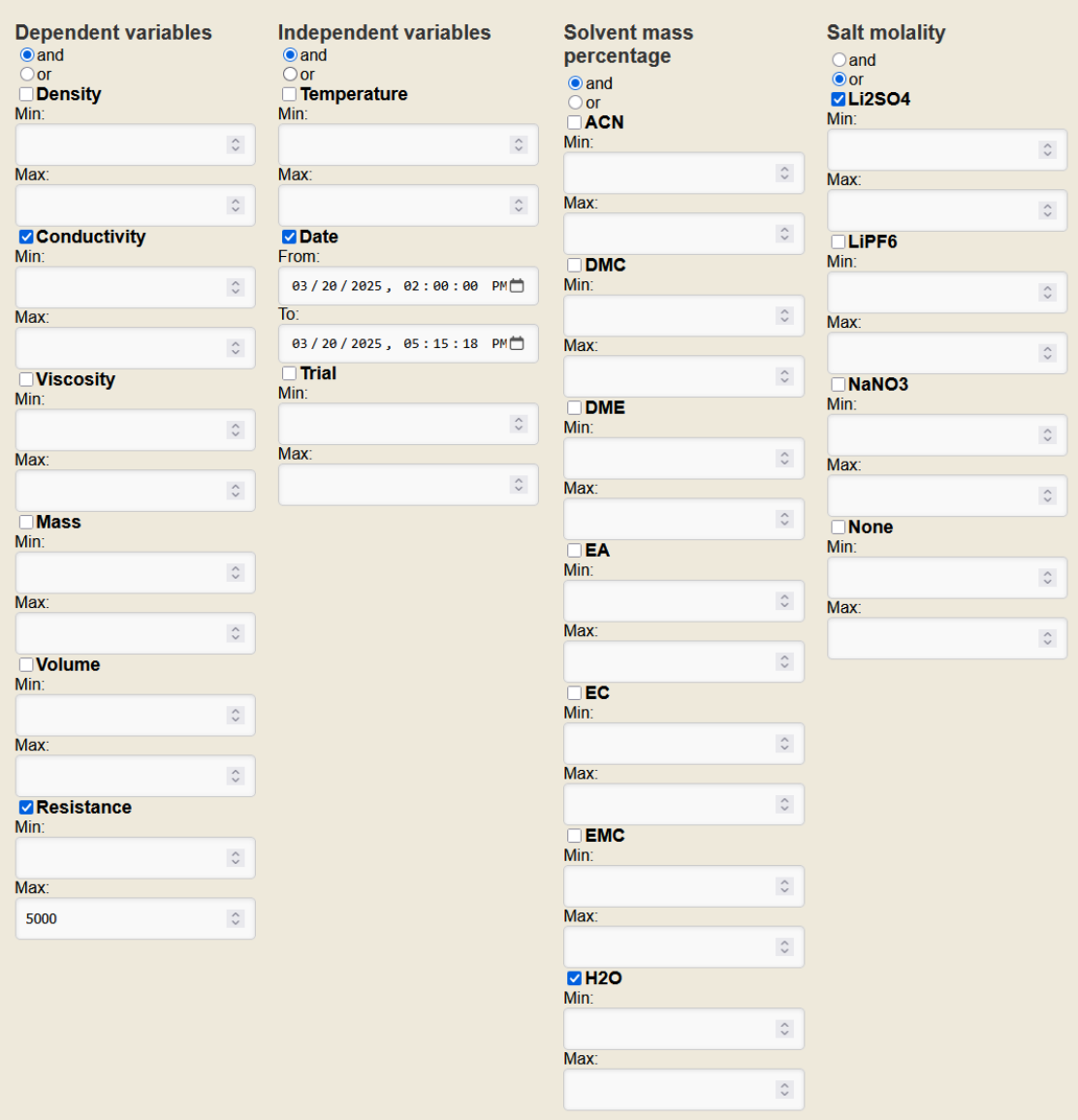

Figure 3: Li_2SO_4 and H_2O are selected because they are the components of the electrolytes. A filter is applied to the time period when the experiment was performed. Conductivity and resistance are also checked. Additionally, a filter on resistance less than 5000 Ω is applied to eliminate any possible incorrect measurements.

In figure 4, the conductivity of the electrolyte increases with the molality of Li_2SO_4, peaks at approximately 58 ms/cm when Li_2SO_4 molality reaches around 2.15 mol/kg, and then decreases as the molality continues to increase. The compo-

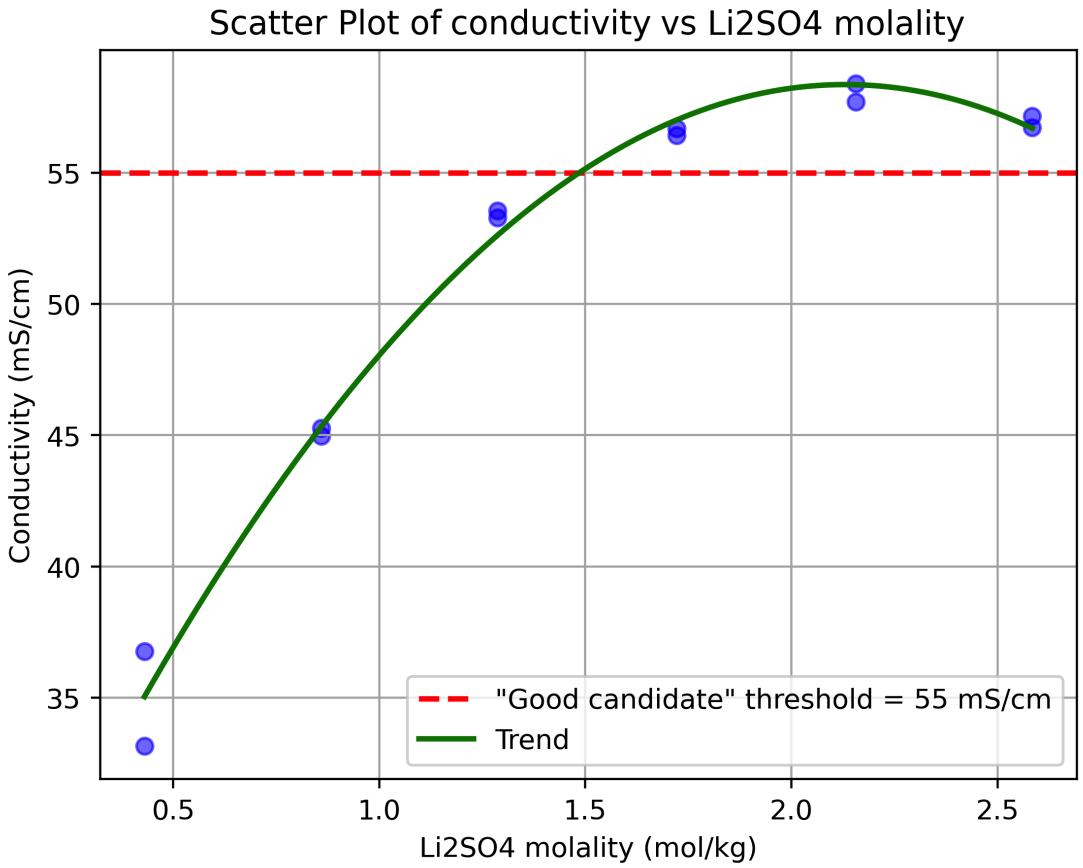

Figure 4: Scatter plot for conductivity vs Li_2SO_4 molality: Electrolytes with Li_2SO_4 molality of 1.72, 2.15, and 2.58 (in mol/kg) are considered good candidates.

sition H2O|100|Li2SO4|2.15 is identified as the recommended composition as it has the highest conductivity.

In our previous work, Clio, we have demonstrated acceleration in automating the experiment processes of electrolyte research.[11] Building upon this foundation, ElyteOS further accelerates this approach by providing data visualization tools to support users performing data analysis and drawing conclusions. In addition, it provides notifications to inform users when a candidate electrolyte meets the selected criteria. Furthermore, ElyteOS informs users when an error occurs for timely troubleshooting. To facilitate the debugging process, ElyteOS provides log files to track the status of the equipment. Lastly, the inventory page helps users plan the experiments and replace the samples in time to avoid any shortage of samples during the experiments. In general, ElyteOS accelerates the experiment process by facilitating all aspects of research processes and avoiding delays caused by errors.

3.2. Flow rate selection

Before achieving full automation, calibration is required to ensure that all laboratory equipment works correctly. For density measurement calibration, the mass measurement and actual volume pumped into the balance are both important. The accuracy of the pump depends heavily on the selected flow rate. Therefore, several flow rates are chosen when pumping the liquid (specifically, 5 ml/min, 4 ml/min, 3 ml/min, 2.5 ml/min, 2 ml/min, and 0.5ml/min) to calibrate on February 3 and 4, 2025. The flow rate greater than 5.5 ml / min is above the maximum flow rate of the pump, while the flow rate less than 0.5 ml / min may cause the measurements to be time inefficient. After calibration, the results can be obtained through the database. The filters applied for checking calibration result are shown in figure 5. The results are provided in `DI_Calibration_results_Feb_3.csv` and `DI_Calibration_results_Feb_4.csv`.

According to the results, the flow rate of 4 ml/min is preferred, as it has the lowest bias and a relatively low variance. To ensure the reliability of Clio, the flow rate of 4 ml/min is chosen for the final flow rate to measure the density.

4. Impact

While our previous work, Clio, has demonstrated how lab automation can accelerate battery electrolyte [11], its new software, ElyteOS, further facilitates the research procedure. By providing a GUI with self-explanatory elements, ElyteOS reduces the learning curve to the users. With the logger for experimental status and inventory manager, ElyteOS ensures that the lab automation process is reliable and traceable, enabling users to debug errors and monitor experiment processes. In addition, the database and visualization tools in ElyteOS provide an all-in-one process for users conducting experiments, enabling them to analyze the data, draw scientific conclusions, and even perform further studies (for example, using data from the database and performing machine learning to predict the properties of electrolytes) more easily.[14]

As lab automation is gaining prominence in multiple scientific domains, ElyteOS demonstrates an adaptable approach to improve the accessibility, robustness, and efficiency of experiments. A similar approach is applicable beyond Clio or battery electrolyte research[15], which helps researchers from various fields greatly by streamlining the research process.[16]

5. Conclusions

ElyteOS provides a user-friendly Python-based GUI that drives an autonomous experimentation pipeline for electrolyte preparation, testing, and data management. Through ElyteOS, users can configure the experimental workflow and orchestrate

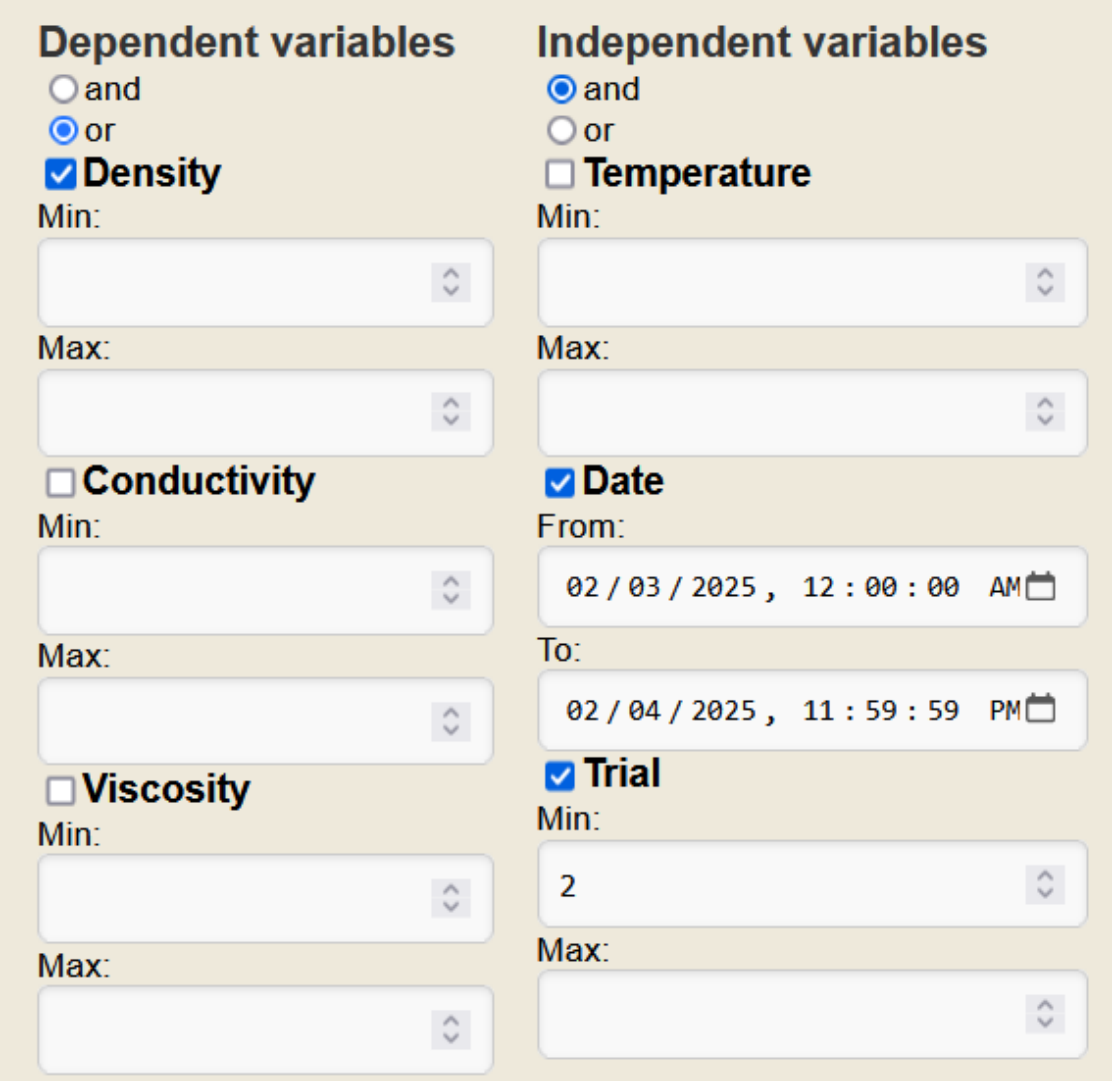

Figure 5: Filters applied for Calibration: A filter on date is applied based on the time the calibration happened. Another filter with Trial number greater than 2 is applied to avoid any underlying effect when starting a new set of experiments. Density is also checked to determine the effect of flow rates on the accuracy and precision of density measurement.

different types of instrument used in battery electrolyte research. The equipment manager, inventory manager, and candidate manager of the GUI can help users conveniently track the status of the experiment process and effectively resolve issues. The integration of the database with filtering and visualization tools also helps researchers visualize and analyze the data in an efficient process. In general, the development of ElyteOS provides better accessibility to researchers with an accelerated process with reduced workload in the research, optimization, and

discovery of electrolytes for advancing battery technologies.

Acknowledgments

The authors acknowledge funding from Toyota Research Institute for this work.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve the readability of this article. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] G. G. Njema, R. B. O. Ouma, J. Kibet, A review on the recent advances in battery development and energy storage technologies, Journal of Renewable Energy (2024). doi:10.1155/2024/2329261.

[2] G. A. Giffin, The role of concentration in electrolyte solutions for non-aqueous lithium-based batteries., Nature Communications (2022). doi:10.1038/s41467-022-32794-z.

[3] N. Gao, S. Kim, P. Chinnam, E. J. Dufek, A. M. Colclasure, A. Jansen, S.-B. Son, I. Bloom, A. Dunlop, S. Trask, K. L. Gering, Methodologies for Design, Characterization and Testing of Electrolytes that Enable Extreme Fast Charging of Lithium-ion Cells, Energy Storage Materials 44 (2022) 296–312. doi:10.1016/j.ensm.2021.10.011.
URL https://www.sciencedirect.com/science/article/pii/S2405829721004748

[4] J. Xu, J. Zhang, T. P. Pollard, Q. Li, S. Tan, S. Hou, H. Wan, F. Chen, H. He, E. Hu, K. Xu, X.-Q. Yang, O. Borodin, C. Wang, Electrolyte design for Li-ion batteries under extreme operating conditions, Nature 614 (7949) (2023) 694–700, publisher: Nature Publishing Group. doi:10.1038/s41586-022-05627-8.
URL https://www.nature.com/articles/s41586-022-05627-8

[5] S. Hess, M. Wohlfahrt-Mehrens, M. Wachtler, Flammability of Li-Ion Battery Electrolytes: Flash Point and Self-Extinguishing Time Measurements, Journal of The Electrochemical Society 162 (2) (2015) A3084, publisher: IOP Publishing. doi:10.1149/2.0121502jes.
URL https://iopscience.iop.org/article/10.1149/2.0121502jes/meta

[6] S. S. Zhang, A review on electrolyte additives for lithium-ion batteries, Journal of Power Sources 162 (2) (2006) 1379–1394. doi:10.1016/j.jpowsour.2006.07.074.
URL https://www.sciencedirect.com/science/article/pii/S0378775306017538

[7] D. Aurbach, Y. Talyosef, B. Markovsky, E. Markevich, E. Zinigrad, L. Asraf, J. S. Gnanaraj, H.-J. Kim, Design of electrolyte solutions for li and li-ion batteries: a review, Electrochimica Acta (2004). doi:10.1016/j.electacta.2004.01.090.

[8] C. Liang, J. Liu, W. Peng, B. Wang, F. Yang, W. You, Y. Wang, Aquags: An integrated gui pipeline for genomic selection in aquaculture breeding, SoftwareX (2024). doi:10.1016/j.softx.2024.101770.

[9] Y. Fei, B. Rendy, R. Kumar, O. Dartsi, H. P. Sahasrabuddhe, M. J. McDermott, Z. Wang, N. Szymanski, L. N. Walters, D. Milsted, Y. Zeng, A. Jain, G. Ceder, Alabos: A python-based reconfigurable workflow management framework for autonomous laboratories, Digital Discovery (2024). doi:10.1039/d4dd00129j.

[10] J. Soto-Perdomo, J. Morales-Guerra, J. D. Arango, S. M. Villada, P. Torres, E. Reyes-Vera, Optigui datacollector: A graphical user interface for automating the data collecting process in optical and photonics labs, SoftwareX (2023). doi:10.1016/j.softx.2023.101521.

[11] A. Dave, J. Mitchell, S. Burke, H. Lin, J. Whitacre, V. Viswanathan, Autonomous optimization of non-aqueous li-ion battery electrolytes via robotic experimentation and machine learning coupling, Nature Communications (2022). doi:10.1038/s41467-022-32938-1.

[12] W. Zhang, L. Hao, V. Lai, R. Corkery, J. Jessiman, J. Zhang, J. Liu, Y. Sato, M. Politi, M. E. Reish, R. Greenwood, N. Depner, J. Min, R. El-khawaldeh, P. Prieto, E. Trushina, J. E. Hein, Ivoryos: an interoperable operating system for flexible self-driving laboratories, Nature Communications (2025). doi:10.1038/s41467-025-60514-w.

[13] D. Gilland, Pydash 8.0.3 documentation, version 8.0.3, accessed on July 22, 2024 (2024).
URL https://pydash.readthedocs.io/en/latest/index.html

[14] S. Zhu, B. Ramsundar, E. Annevelink, H. Lin, A. Dave, P.-W. Guan, K. Gering, V. Viswanathan, Differentiable modeling and optimization of non-

aqueous li-based battery electrolyte solutions using geometric deep learning, Nature Communications (2024). doi:10.1038/s41467-024-51653-7.

[15] J. T. Yik, L. Zhang, J. Sjölund, H. Xu, P. H. Svensson, K. Edström, E. J. Berg, Automated electrolyte formulation and coin cell assembly for high-throughput lithium-ion battery research, Digital Discovery (2023). doi:10.1039/d3dd00058c.

[16] I. Oh, M. A. Pence, N. G. Lukhanin, O. Rodríguez, C. M. Schroeder, J. Rodríguez-López, The electrolab: An open-source, modular platform for automated characterization of redox-active electrolytes, Device (2023). doi:10.1016/j.device.2023.100103.

Supplementary information

Additional Figures

- Figure S1 shows the overall structure and data flow between the python files.
- Figures S2, S3, S4, S5, S6, S7 show the overall structure and data flow between the python files.
- Figure S8, S9, S10, S11 shows the process of each data flow happens in the software.
- Figure S12 shows the calibration results of the pumps.
- Figure S13 and S14 show the calibration results of the pumps.

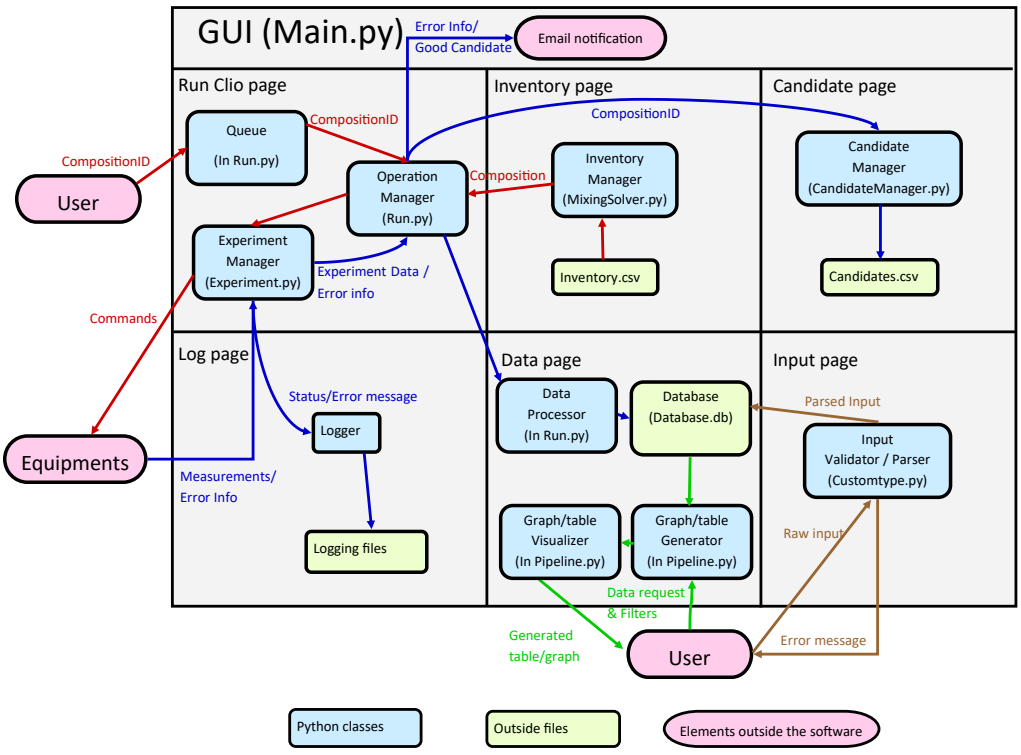

Figure S1: File structures of the software: The arrows with the same color denotes the same data flow when a user is operating on the interface. All six pages are contained in the Main.py Python file.

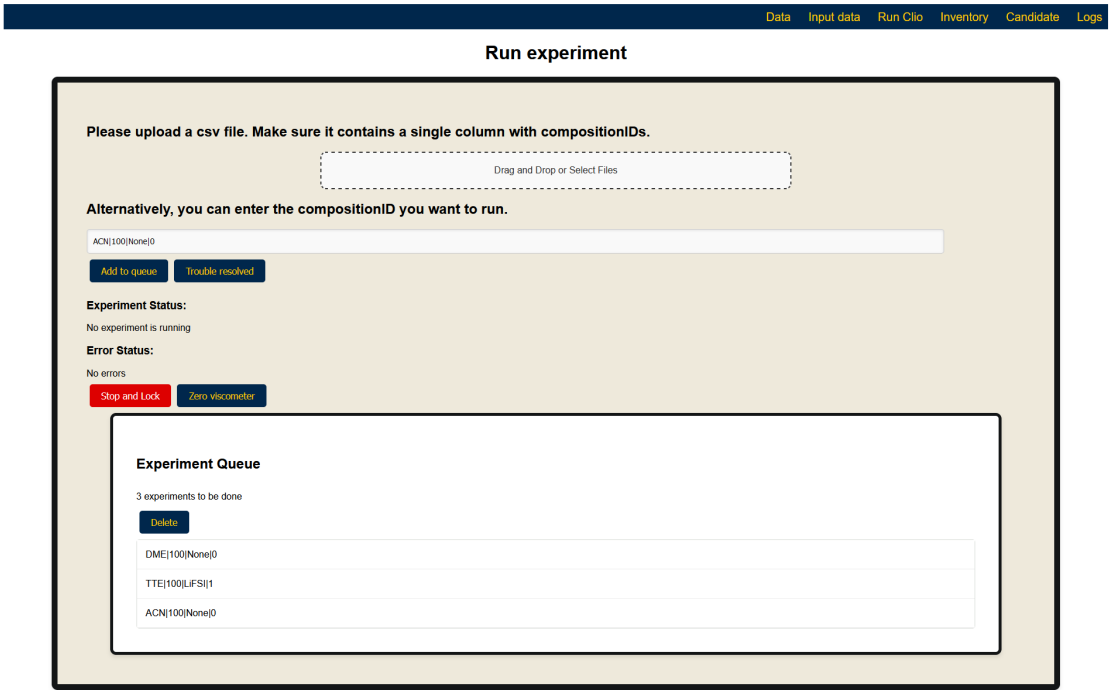

Figure S2: Run Clio page of GUI: The user can either upload the CSV file to add a list of CompositionIDs into the experiment queue, or enter the CompositionID manually. The experiment queue reflects the ongoing compositions to be tested. The experiment status indicates whether Clio is running the experiment, and the error status indicates whether Clio encounters an error or not.

Additional Tables

Tables S1, S2, S3, S4, S5, show the structure of the database.

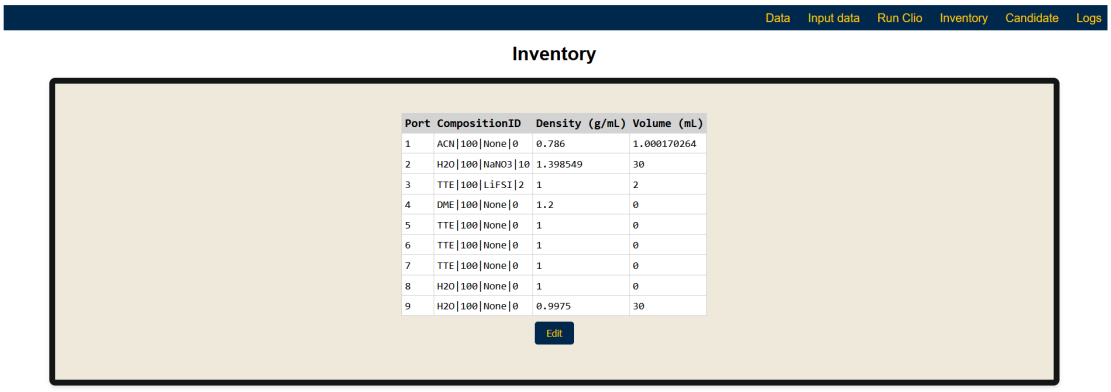

Figure S3: Inventory page of GUI: The page automatically updates the compositions of electrolytes and their respective remaining volumes in each bottle connected to the 10-port valves. The user can edit the inventory with the "edit" button.

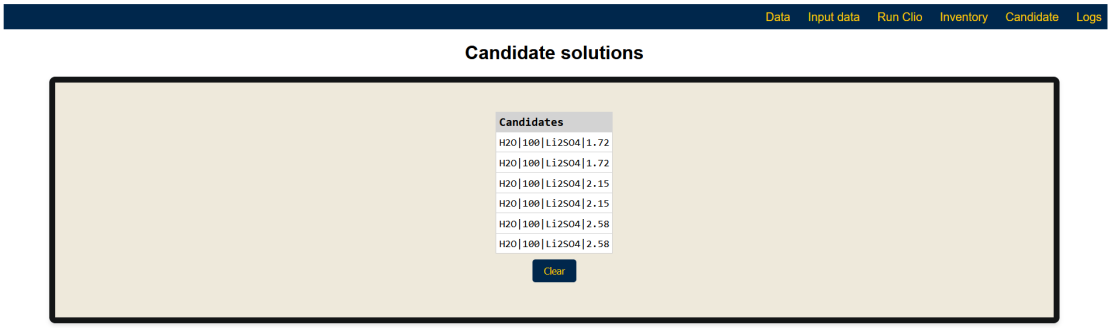

Figure S4: Candidate page of GUI: The page lists the list of compositions of the candidates meeting the criteria. The user can clear the composition list with the "clear" button.

ID	Density (g/ml)	Conductivity (mS/cm)	Viscosity (cP)	Temp. (°C)	Date	Trial
A	1.00	2.00	3.00	4.00	2025-02-03T00:00:00	1
B	5.00	6.00	7.00	8.00	2025-02-04T00:00:00	2

Table S1: Experiment Table: The table contains the record ID, the density, the conductivity, the viscosity, the temperature, the date, and the trial number. The compositionID is not directly stored inside this table. Instead, it is parsed and stored in the next four tables. Please note that the tables only serve to demonstrate the structures and do not reflect the actual information inside the database.

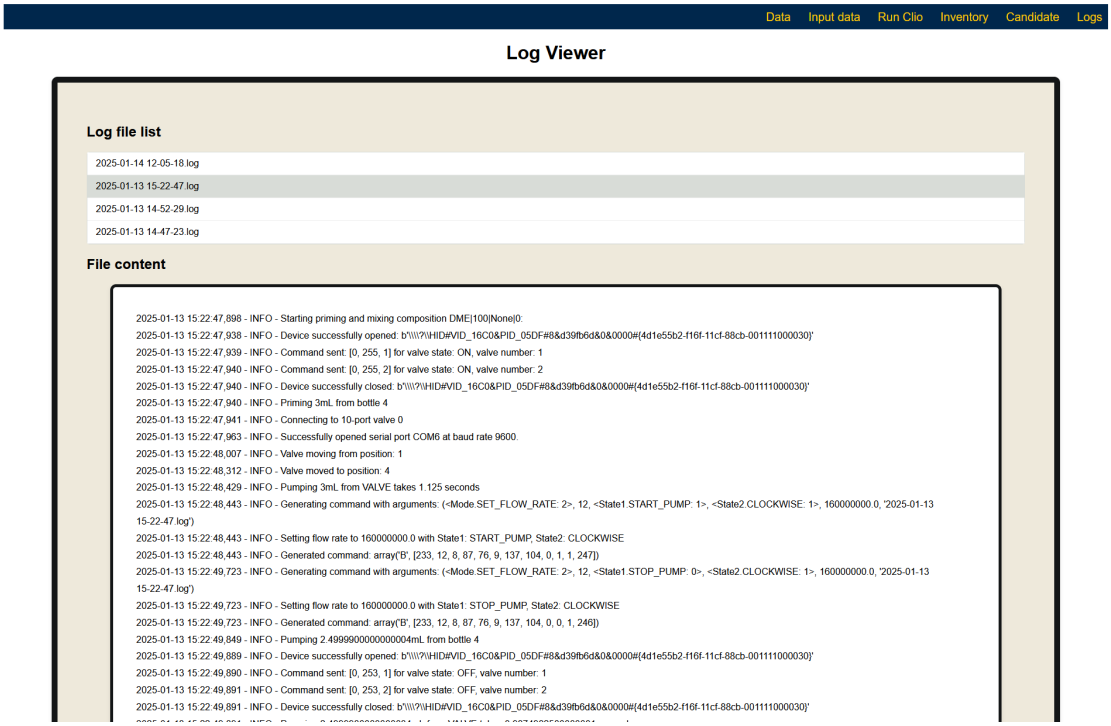

Figure S5: Log page of GUI: The page loads and displays the log files made by the loggers to let the users keep track of the status of the experiment. The log files are sorted and named based on the starting time of the experiments.

ID	Solvent	Percentage
A	DMC	50
A	EMC	50
B	DMC	25
B	EMC	75

Table S2: Solvent Mass Percentage Table

ID	Salt	Molality
A	LiPF6	1.5
A	LiFSI	1.5
B	LiPF6	2.5

Table S3: Salt Molality Table

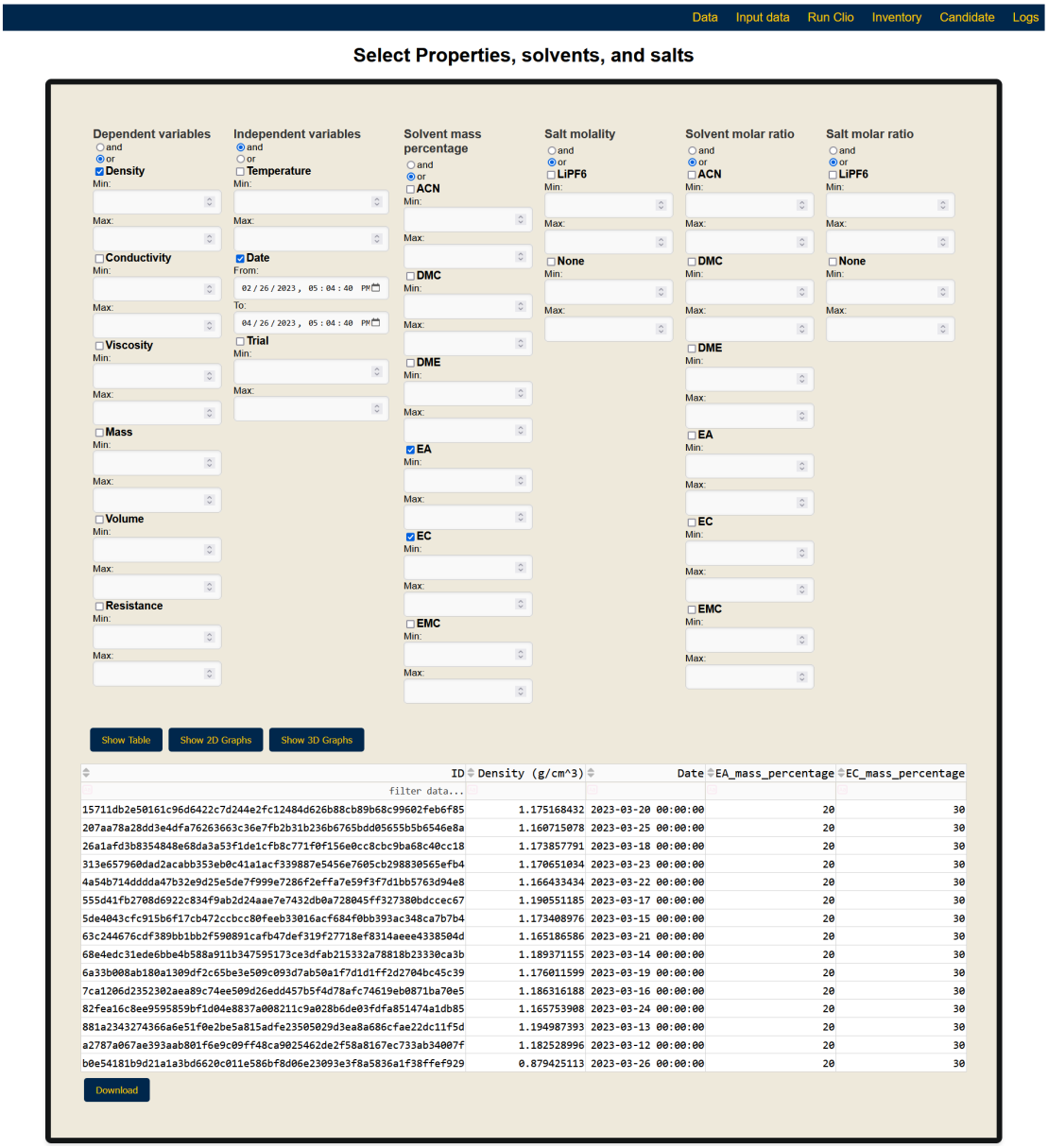

Figure S6: Data page of GUI: The items on the top half of the page show the variables. The users can select and filter the variables to visualize the data. Then, the users can download the generated tables or graphs using the download button.

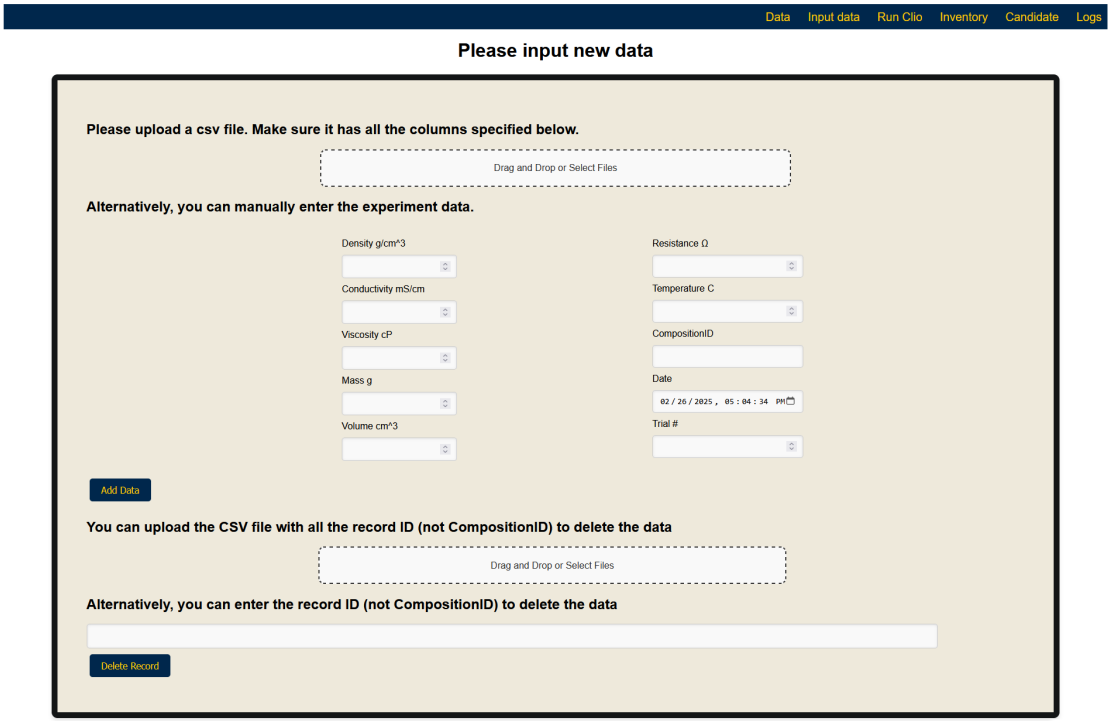

Figure S7: Data page of GUI: The user can either upload a CSV file to the database directly, or enter data manually with the textboxes provided. The user can also delete pieces of record by entering their id.

ID	Solvent	Molar ratio
A	DMC	0.416
A	EMC	0.360
B	DMC	0.222
B	EMC	0.577

Table S4: Solvent Molar Ratio Table

ID	Salt	Molar ratio
A	LiPF6	0.112
A	LiFSI	0.112
B	LiPF6	0.200

Table S5: Salt Molar Ratio Table

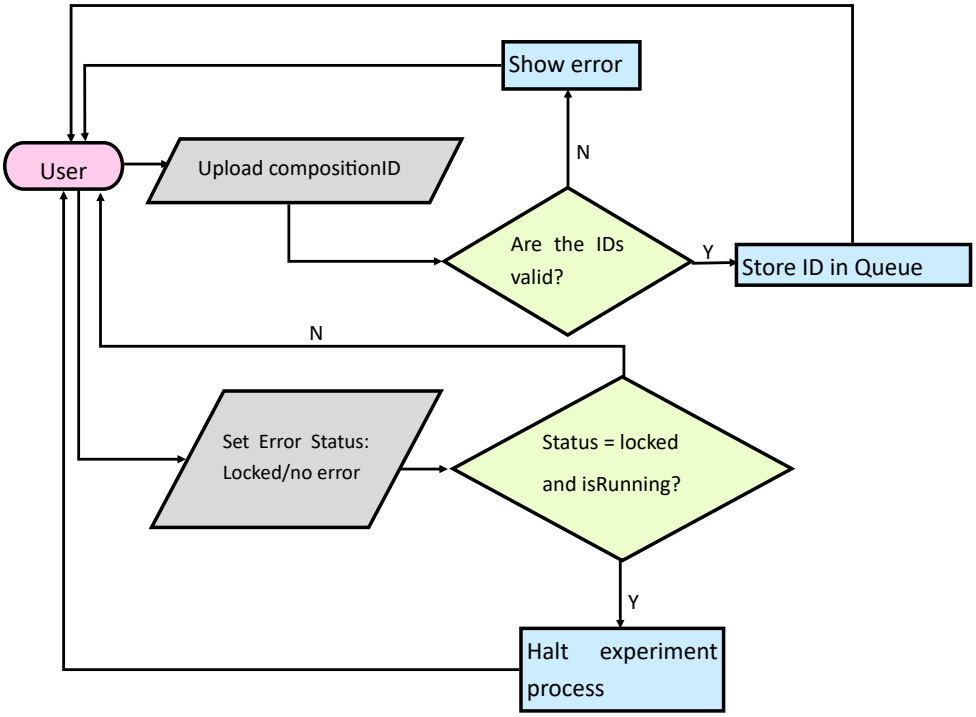

Figure S8: User flow of GUI: The user can upload the compositionID to the experiment operator (in Run.py) which will check whether the compositionID is valid. If yes, the ID will be stored in Queue. The user can also explicitly change the experiment status by using the buttons on the GUI to halt or resume running experiments.

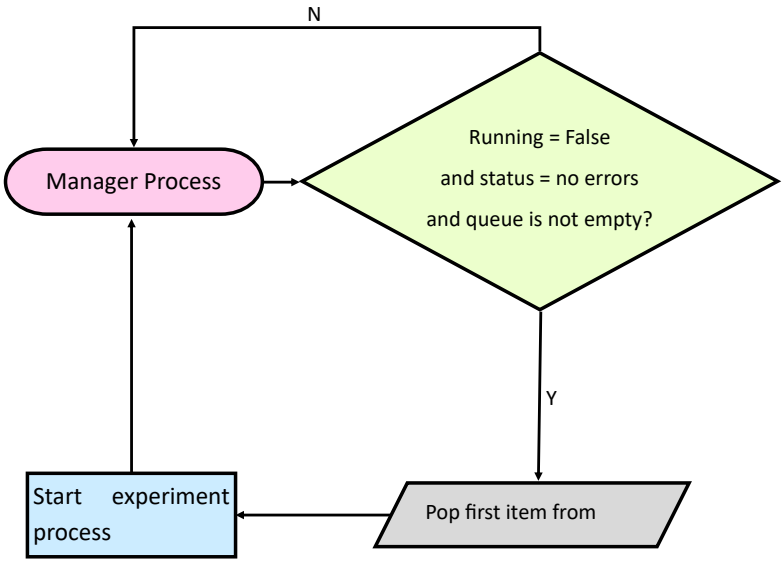

Figure S9: Manager flow of GUI: It repeatly checks whether it is time to start a new experiment.

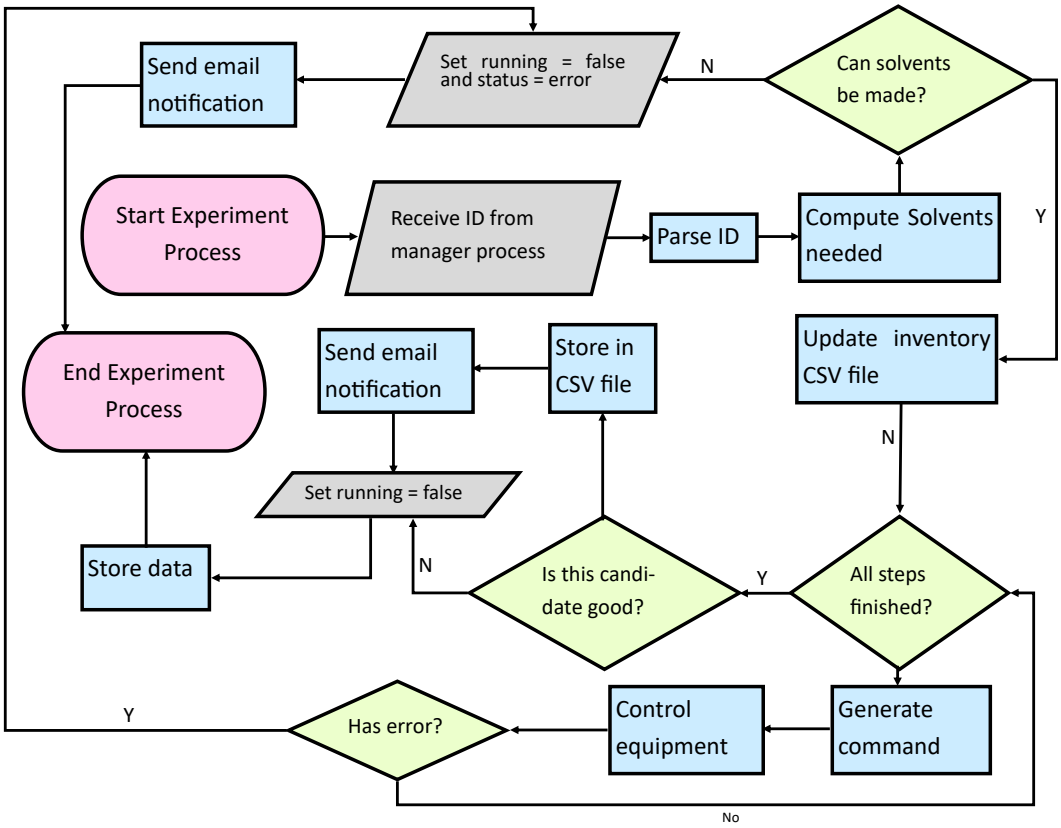

Figure S10: Experiment flow of GUI: The experiment flow parses the compositionID, prepares the sample with the inventory, controls the equipment, and store the data. The experiment flow also keeps track of the inventory, the candidates, and the status of the equipments.

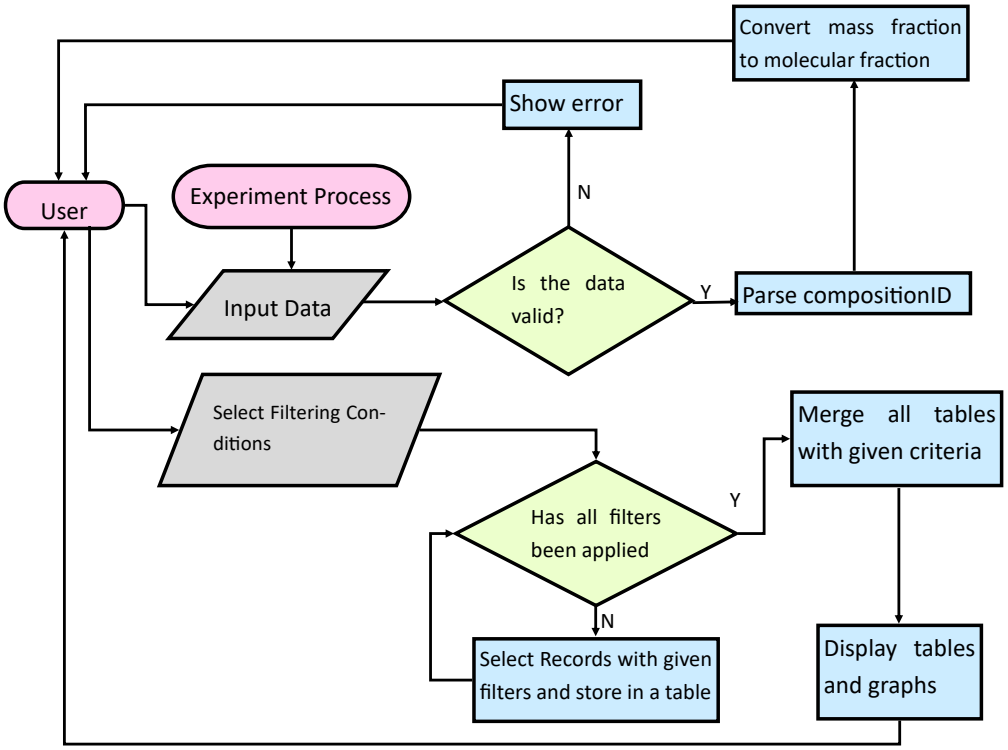

Figure S11: Data flow of GUI: The user or the experiment process inputs data, and the validator process ID and store the data. Also, the user can apply filters to fetch data from the database.

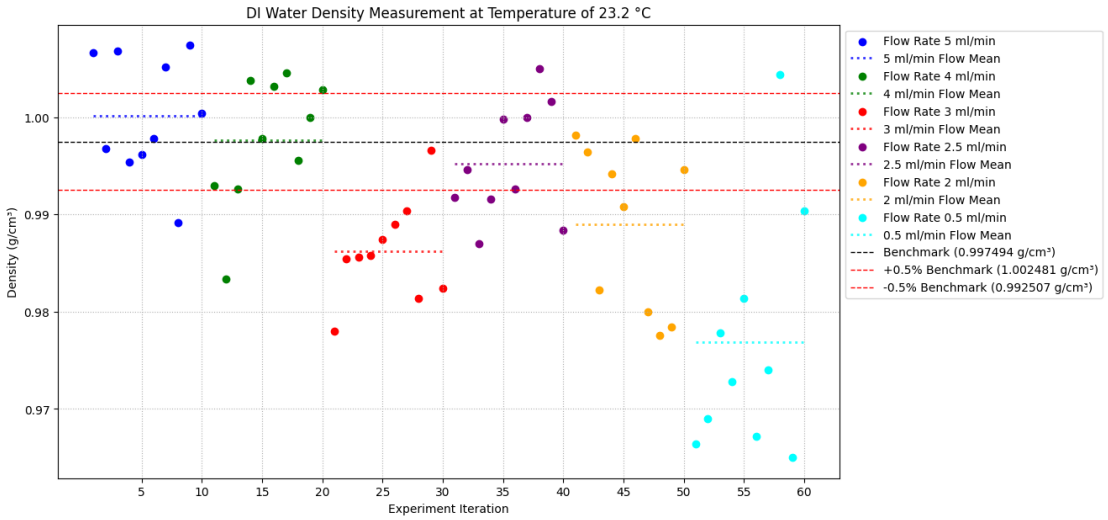

Figure S12: Calibration results for different flow rates with distilled water: As the flow rate decreases, the bias and the variance both increase. When the flow rate is chosen to be 4ml/min, both the bias and the variance are low. Therefore, the flow rate 4ml/min is preferred for density measurement in DI water.

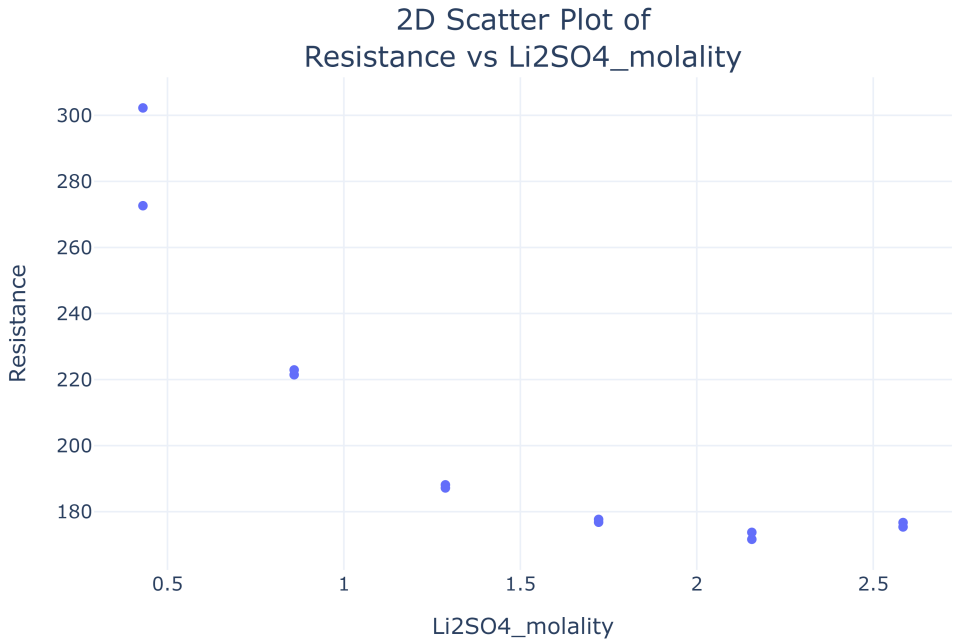

Figure S13: Scatter plot for Resistance vs Li_2SO_4 molality: The plot is automatically generated by ElyteOS. The chosen molalities (in mol/kg) include 0.43, 0.86, 1.29, 1.72, 2.15, and 2.58.

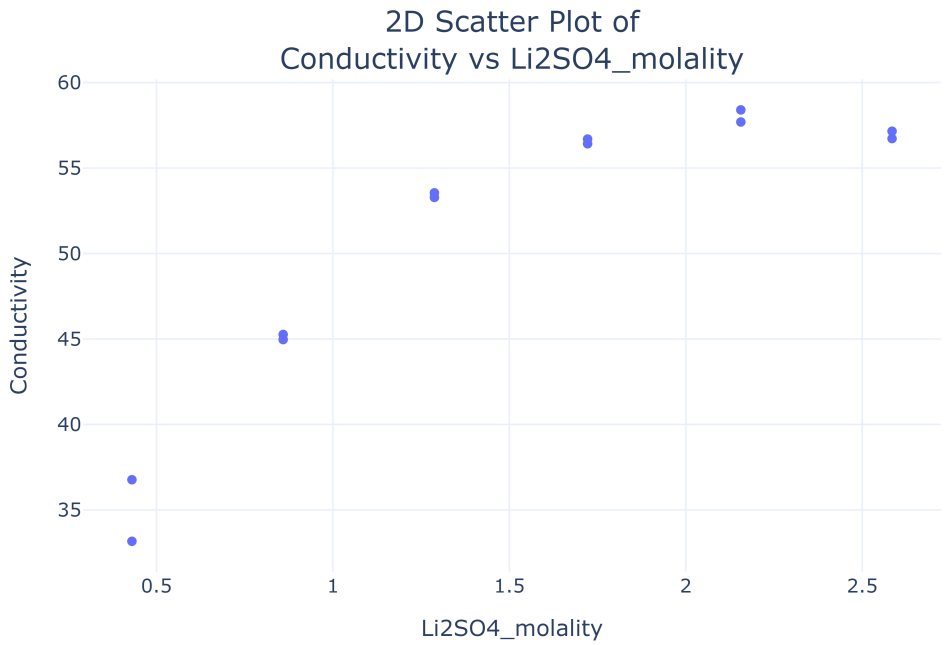

Figure S14: Scatter plot for conductivity vs Li_2SO_4 molality: The plot is also automatically generated by ElyteOS. The conductivity of the electrolyte is inversely proportional to the resistance inside the given chamber. The product of the resistance and conductivity of the same electrolyte in the same cell is the cell constant, which is measured to be 10.0236965757 cm^{-1}.