# Semester Long Internship Report

On

## GUI Development for OpenModelica simulator

Submitted by

**Garima**

**Thamaraimanalan M**

Under the guidance of

**Prof.Kannan M. Moudgalya**
Chemical Engineering Department
IIT Bombay

June 17, 2025

# Acknowledgment

We would like to express our gratefulness to the FOSSEE team for giving us this opportunity to learn, implement, test and understand the applications of technology in a project of industrial scale. From implementing code standarization, to debugging features and understanding the importance of documentation and structured application development, these 4 months have been a wonderful experience.

We would like to express our gratitude to Prof. Kannan M. Moudgalya for providing us with such an opportunity to work on such an impactful project.

We would also like to express our appreciation to our mentors during these 4 months of work, Mr. Sumanto Kar and Mr. Nikhil Sharma for their guidance at each step of our work. Nikhil Sir's lessons in Modelica language helped us in navigating through some of the complex debugging sessions and optimization process while Sumanto Sir's tips helped us in makin sure the development process was structured and smooth.

it was a journey in the making, a chance to collaborate with sharp minds and learn from some of the best. We both hope to take the lessons we leanrt during our time on this project forward in our professional career and look forward to contributing to more such innovative, impactful and ground-level software development wherever our path takes us. Thank you FOSSEE for giving us this chance and thank you for everything.

# Contents


**1 Introduction** **3**

1.0.1 OpenModelica . . . . . 3

1.0.2 Technologies Utilized . . . . . 3

**2 Problem Statement** **5**

**3 UI Design** **6**

3.1 UI Compilation . . . . . 6

3.2 GUI Walkthrough . . . . . 7

3.2.1 Compound Selection . . . . . 7

3.2.2 Backend: Compound.mo File Modification . . . . . 8

3.2.3 Thermodynamic Method Selection . . . . . 8

3.2.4 Operations Selection . . . . . 9

3.2.5 Input Parameter Page . . . . . 10

3.2.6 Info Button Feature . . . . . 11

3.2.7 Process Optimization . . . . . 12

3.2.8 Simulation Execution . . . . . 13

3.2.9 Simulation Output and Result Files . . . . . 13

3.2.10 Scalar Variable Extraction and Plot Setup . . . . . 15

3.2.11 Plotting Simulation Results with OMPlot . . . . . 16

3.2.12 Logging and Debugging Support . . . . . 19

3.2.13 Theme Customization (Dark / Light Mode) . . . . . 20

**4 Conclusion and Future Scope** **21**

**Bibliography** **22**

# Chapter 1

# Introduction

### 1.0.1 OpenModelica

OPENMODELICA is an open-source Modelica-based1 modeling and simulation environment intended for industrial and academic usage. Its long-term development is supported by a non-profit organization – the Open Source Modelica Consortium (OSMC)[?].

The goal with the OpenModelica effort is to create a comprehensive Open Source Modelica modeling, compilation and simulation environment based on free software distributed in binary and source code form for research, teaching, and industrial usage. We invite researchers and students, or any interested developer, to participate in the project and cooperate around OpenModelica, tools, and applications. Open Source Modelica Consortium supports its development. It runs on Windows, Linux, and Mac operating systems. FOSSEE, IIT Bombay has taken up the initiative of promoting FLOSS ( Free/Libre and Open Source Software), for education. We, the OpenModelica team at FOSSEE, IIT Bombay, promote the use of OpenModelica as being accessible and readily available. The goal of this project is to enable the students and faculty of various colleges/institutes/universities across India to use Free/Libre and Open Source Software tools for all their modeling and simulation purposes, thereby improving the quality of instruction and learning and to avoid expensive licenses of commercial modeling and simulation packages for research and education.

The FOSSEE project is part of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Human Resource Development (MHRD), Government of India.

Our project software, a OpenModelica Graphical User Interface based Simulator is an extension of Openmodelica OMEdit.

### 1.0.2 Technologies Utilized

- **Language:** Python

- **Libraries/Framework:** PyQt6, Matplotlib, Pyzipper, os, shutil, sys, regex, xml.etree.ElementTree, subprocess, pathlib
- **Tools:** Qt Designer, Visual Studio Code, Git and GitHub, OMPlot

# Chapter 2

# Problem Statement

Modeling and simulating chemical processes using OpenModelica typically requires manual editing of Modelica files and executing command-line scripts. This workflow is not user-friendly, especially for students or engineers without prior programming experience. Additionally, managing simulation parameters, modifying thermodynamic models, and visualizing results often involves switching between multiple tools, increasing the chances of human error and reducing productivity.

There is a need for an intuitive graphical user interface (GUI) that simplifies the simulation workflow by:

- Allowing users to select chemical compounds from a database of 433 compounds.
- Automatically generating and modifying Modelica files based on user input.
- Providing step-by-step configuration of thermodynamics and operations.
- Managing simulation parameters and executing simulations with a single click.
- Visualising results using built-in plotting tools like OMPlot.

This project aims to bridge the gap between complex simulation tools and user accessibility by developing a GUI-based automation layer over OpenModelica, reducing manual effort and enhancing usability.

# Chapter 3

# UI Design

The user interface for the OpenModelica-based application was designed using **Qt Designer**, a visual tool for creating intuitive and responsive GUIs. The '.ui' files generated by Qt Designer were integrated into the Python application using the **PyQt6** framework. This approach allowed for efficient UI prototyping and rapid development of user-friendly components.

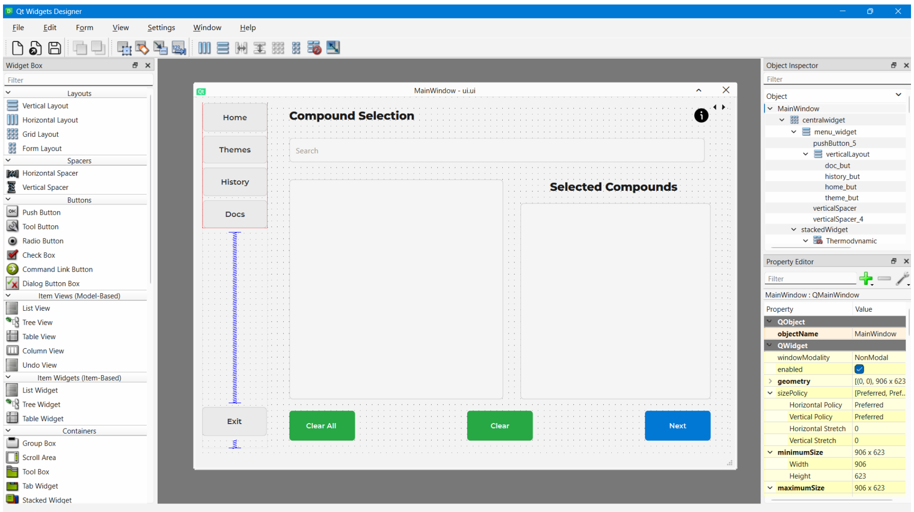



Figure 3.1: Designing the UI in Qt Designer

## 3.1 UI Compilation

The '.ui' files created using Qt Designer were converted to Python files using the `pyside6-uic` tool. This process generates Python code that can be directly imported and used within the application.

```
pyside6-uic input.ui -o output.py
```

## 3.2 GUI Walkthrough

### 3.2.1 Compound Selection

In this step, users can select two or more chemical compounds from the Chemsep-Database using the left-hand list view. The user can either scroll or type the name to search the compounds. The selected compounds are displayed on the right-hand side. Once the desired compounds are selected, there are two options for removing them, either one by one with clear button or all the selected compounds using clear all button. Clicking the **Next** button proceeds to the next stage of the application.

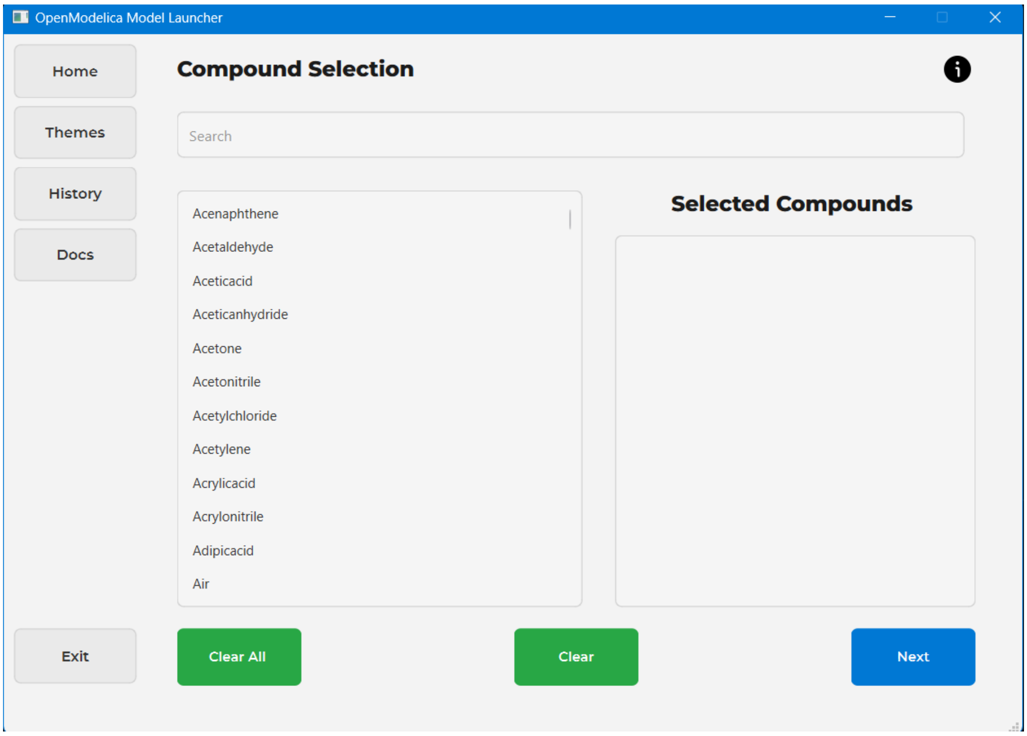



Figure 3.2: Compound Selection Screen

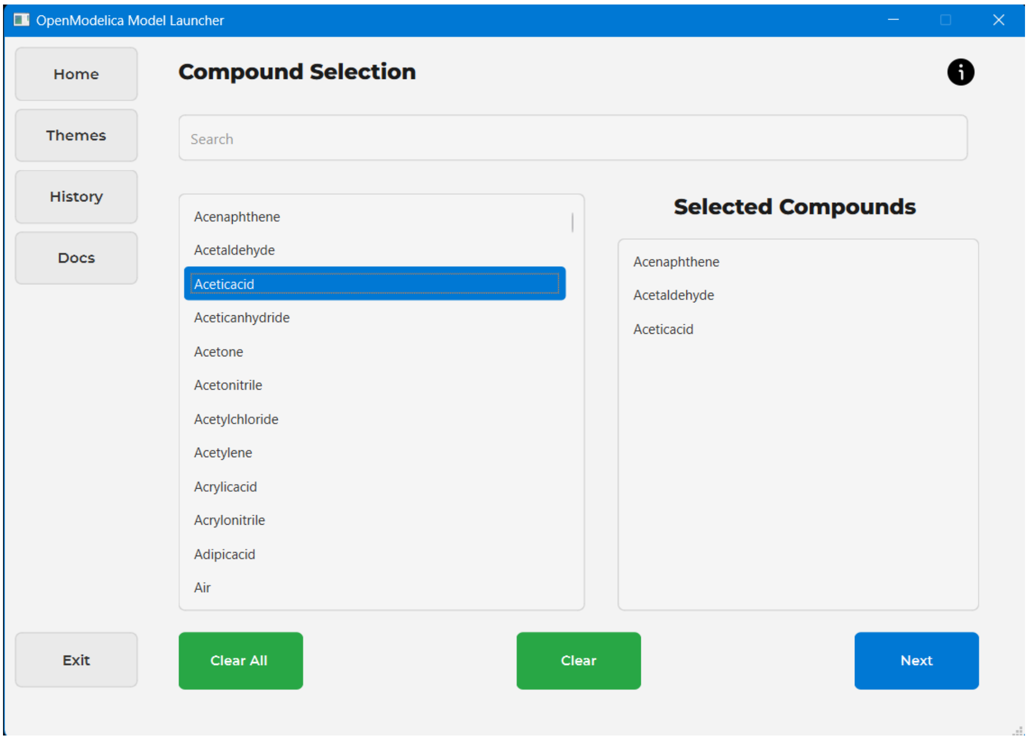



Figure 3.3: Compound Selecting

### 3.2.2 Backend: Compound.mo File Modification

Once the user clicks the **Next** button after selecting compounds, the application dynamically writes the selected compound names to a Modelica-compatible file named `Compounds.mo`.

```
Example content written to Compounds.mo:
compoundList = {"Acenaphthene", "Acetaldehyde", "Aceticacid"};
```

### 3.2.3 Thermodynamic Method Selection

In this stage, the user selects a thermodynamic property method (e.g., NRTL, UNIQUAC, Rault's Law) from a dropdown list. This selection is critical as it defines how phase equilibria and other thermodynamic calculations are handled in the simulation.

Upon selecting the method, the application updates the `Batch_Rectifier.mo` file using Python. The chosen method is inserted or replaced in the Modelica model, ensuring the simulation reflects the selected thermodynamic behavior.

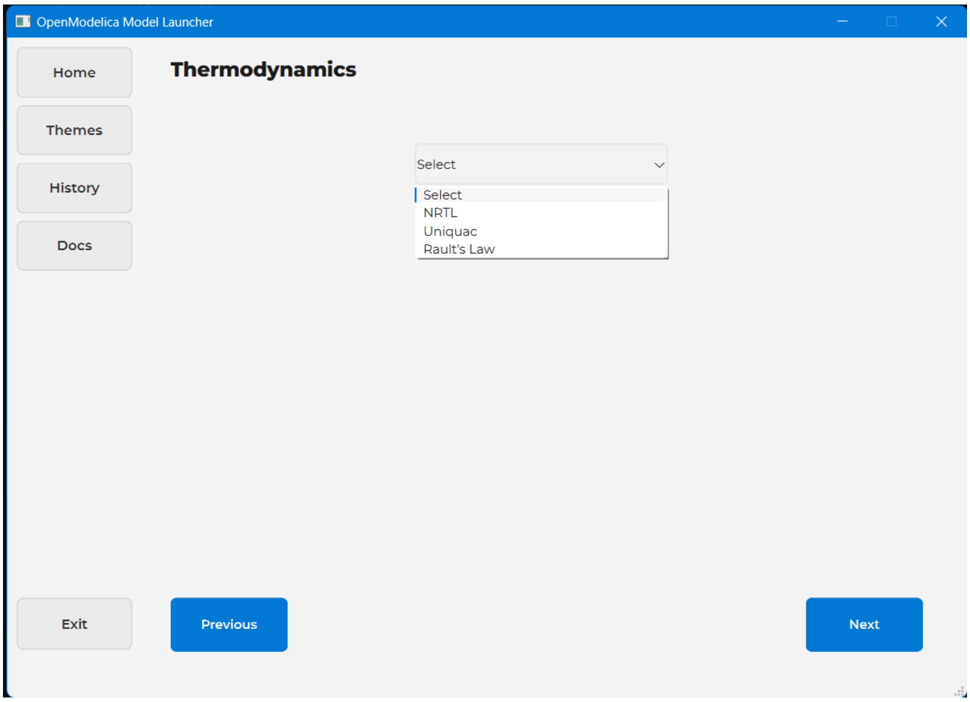



Figure 3.4: Thermodynamic Method Selection Screen

### 3.2.4 Operations Selection

Currently, the only operation supported in the application is the **Batch_Rectifier**. Once selected, the simulation setup continues using the relevant parameters and configuration for this operation.

Below are screenshots showing the operation selection interface and the initial configuration screen for the batch rectifier.

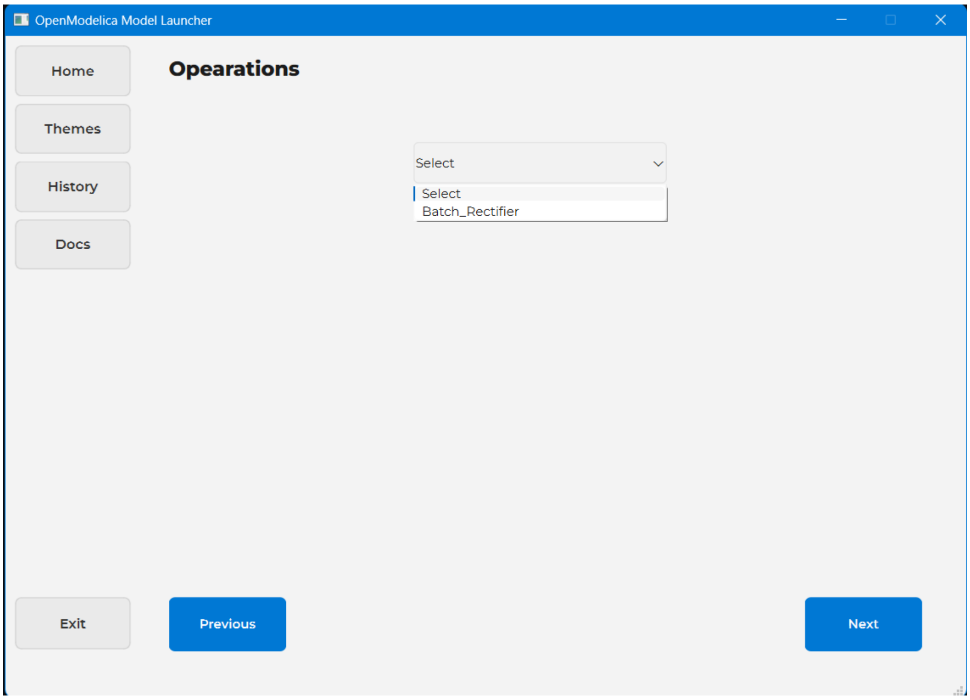



Figure 3.5: Operation Selection Screen

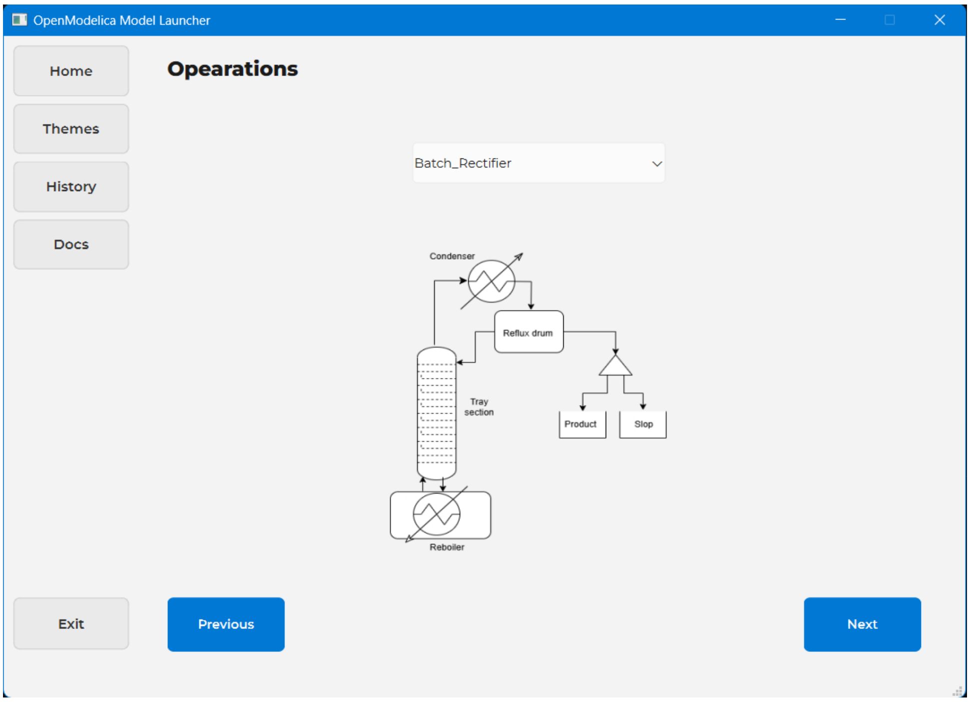



Figure 3.6: Batch Rectifier Configuration

### 3.2.5 Input Parameter Page

In this stage, the user is prompted to enter the simulation parameters required for the operation simulation process. These inputs are necessary for running the OpenModelica simulation with the correct initial and boundary conditions.

The input fields available in the GUI include:

- **Start Time** – Enter start time in seconds
- **Stop Time** – Enter stop time in seconds
- **HB0** – Enter feed amount
- **Ha** – Enter initial accumulator hold up
- **Hc** – Enter condenser hold up
- **PB** – Enter reboiler pressure
- **PC** – Enter condenser pressure
- **QR** – Enter heat duty
- **R** – Enter reflux ratio (value between 0 and 1)
- **XB0** – Enter initial charge feed composition

All values entered by the user are validated for correct format and range. Once submitted, these values are written into the operation model file to parameterize the Modelica simulation.

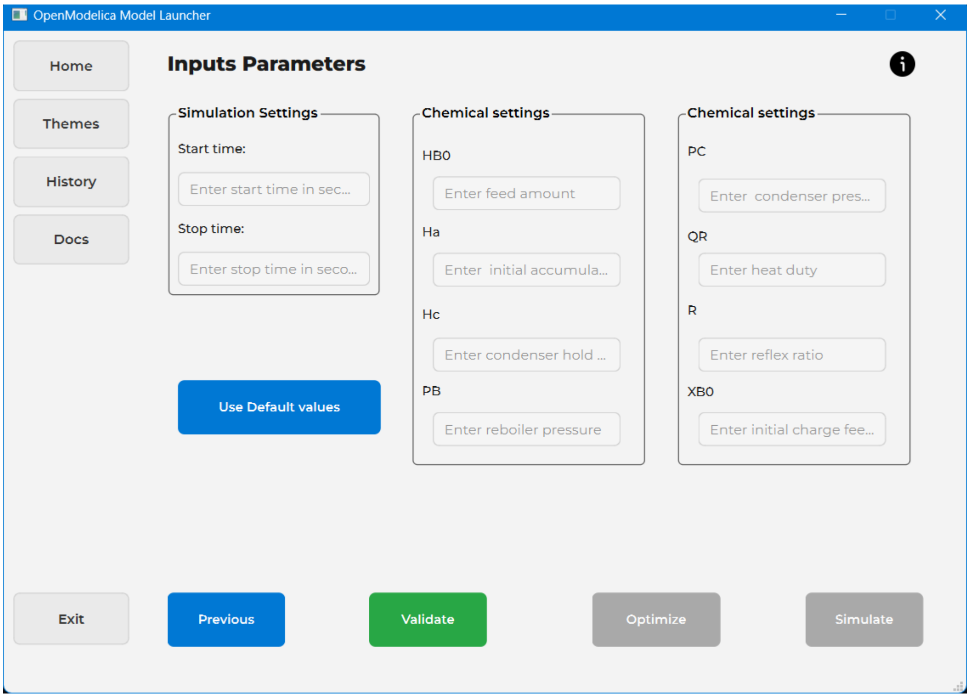



Figure 3.7: Input Parameter Entry Screen

### 3.2.6 Info Button Feature

As shown in Figure 3.9, to improve usability, the GUI includes an **Info** button ( Figure 2.10 ) beside the parameter input fields. When clicked, a helpful tool-tip or pop-up appears that contains a brief explanation of each parameter.

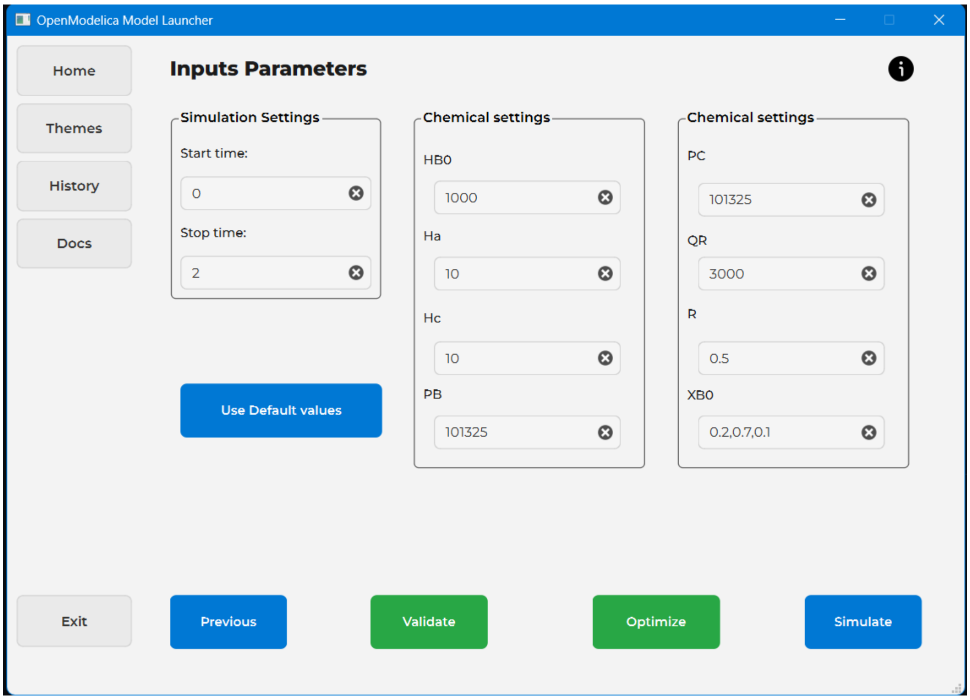



Figure 3.8: Input Parameter after Validation

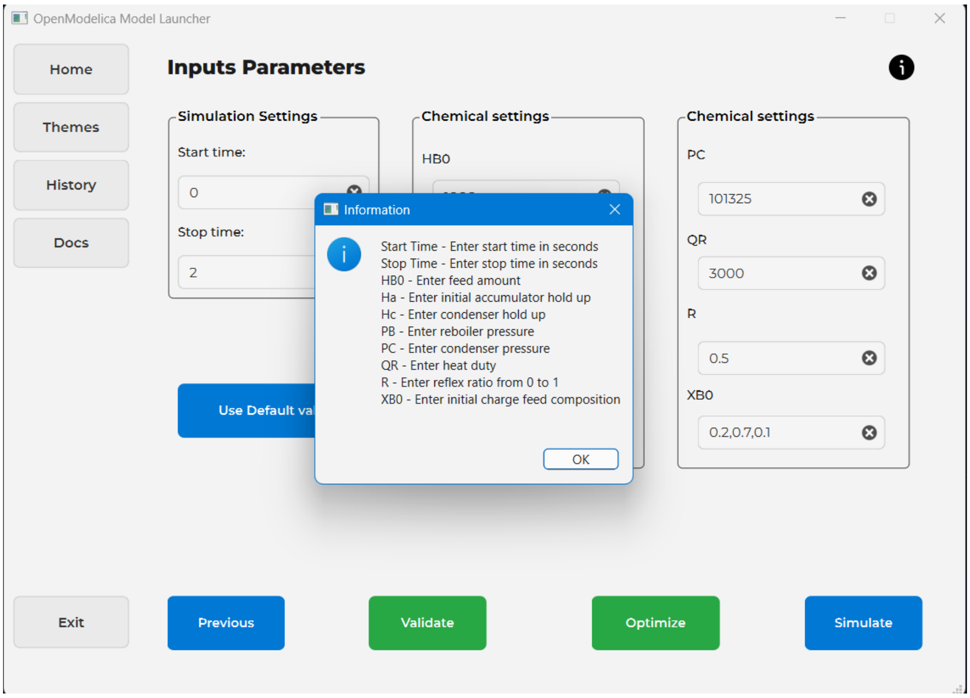



Figure 3.9: Info Popup Explaining Simulation Parameters

### 3.2.7 Process Optimization

The **Optimize** button gives the user the option to further optimize the process, with select parameters for a better result. Currently, the user has the option of `BASIC` optimization. The user has the option of choosing the boundary value for purity of Reflux ration R as `x_min` and `x_max` which are used for constraint-based optimization.

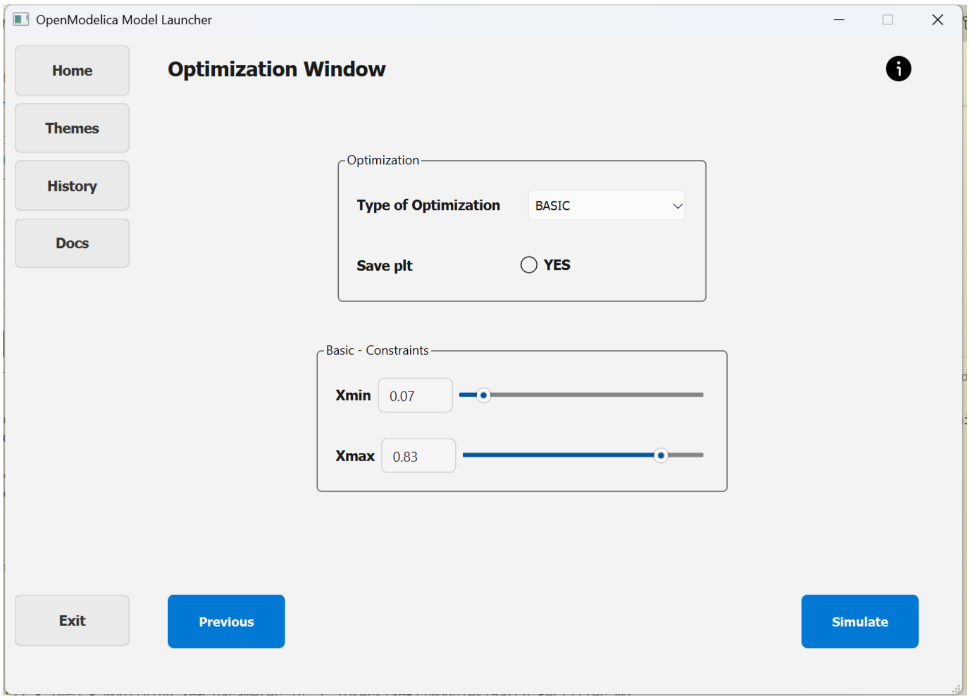



Figure 3.10: Info Popup Explaining Simulation Parameters

### 3.2.8 Simulation Execution

After entering the required parameters, the user can start the simulation by clicking the `Simulate` button. This triggers the execution of a `.mos` script using OpenModelica's command-line interface `omc.exe` in the backend.

The GUI internally uses Python's `subprocess` module to invoke OpenModelica. The relevant Modelica script (e.g., `simulate.mos`) is executed through the following command:

```
result = subprocess.run(
    [self.omc_path, self.script_content],
    capture_output=True,
    text=True,
)
```

Upon execution, OpenModelica compiles the model, runs the simulation, and generates output files as `.mat` for further analysis.

### 3.2.9 Simulation Output and Result Files

After the simulation completes, OpenModelica generates several output files in the temporary directory. The most important of these are:

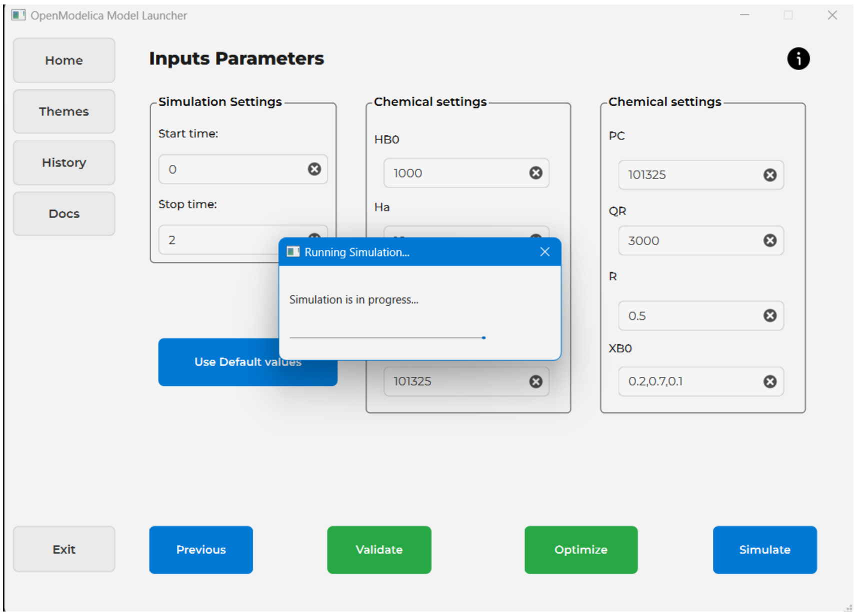


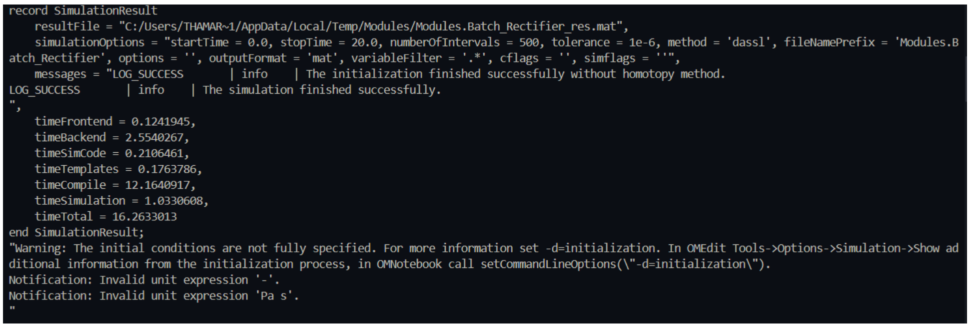

Figure 3.11: Simulation Triggered via GUI

- `Modules.Batch_Rectifier_res.mat` – contains simulation result data (used for plotting and post-processing).
- `Modules.Batch_Rectifier_res.xml` – contains metadata about the variables and structure of the model.

An example simulation result record is shown below:

```
record SimulationResult
    messages = "LOG_SUCCESS | The simulation finished successfully.",
    timeFrontend = 0.124s,
    timeBackend = 2.554s,
    timeCompile = 12.164s,
    timeSimulation = 1.033s,
    timeTotal = 16.263s
end SimulationResult;
```

These files are saved in a system temporary directory and are used by the GUI for plotting and exporting data.



```
record SimulationResult
    resultFile = "C:/Users/THAMAR~1/AppData/Local/Temp/Modules/Modules.Batch_Rectifier_res.mat",
    simulationOptions = "startTime = 0.0, stopTime = 20.0, numberOfIntervals = 500, tolerance = 1e-6, method = 'dassl', fileNamePrefix = 'Modules.B
atch_Rectifier', options = '', outputFormat = 'mat', variableFilter = '.*', cflags = '', simflags = ''",
    messages = "LOG_SUCCESS       | info    | The initialization finished successfully without homotopy method.
LOG_SUCCESS       | info    | The simulation finished successfully.
",
    timeFrontend = 0.1241945,
    timeBackend = 2.5540267,
    timeSimCode = 0.2106461,
    timeTemplates = 0.1763786,
    timeCompile = 12.1640917,
    timeSimulation = 1.0330608,
    timeTotal = 16.2633013
end SimulationResult;
"Warning: The initial conditions are not fully specified. For more information set -d=initialization. In OMEdit Tools->Options->Simulation->Show ad
ditional information from the initialization process, in OMNotebook call setCommandLineOptions(\"-d=initialization\").
Notification: Invalid unit expression '-'.
Notification: Invalid unit expression 'Pa s'.
"
```

Figure 3.12: Simulation Output Files and Log Messages

### 3.2.10 Scalar Variable Extraction and Plot Setup

Once the simulation is complete, the application parses the `Modules.Batch_Rectifier_res.xml` file to extract all available **scalar variables** for plotting. These variables include time-dependent values such as compositions, pressures, temperatures, holdups, and flow rates.

The extracted variables are displayed in a `ListView` within the GUI. Users can select one or more variables from this list to visualize them graphically.

The XML parsing is done using Python's `xml.etree.ElementTree` library. The application scans the `<ScalarVariable>` tags and extracts relevant metadata such as:

- Variable name
- Description (if available)
- Variability (continuous, parameter, constant)
- Unit

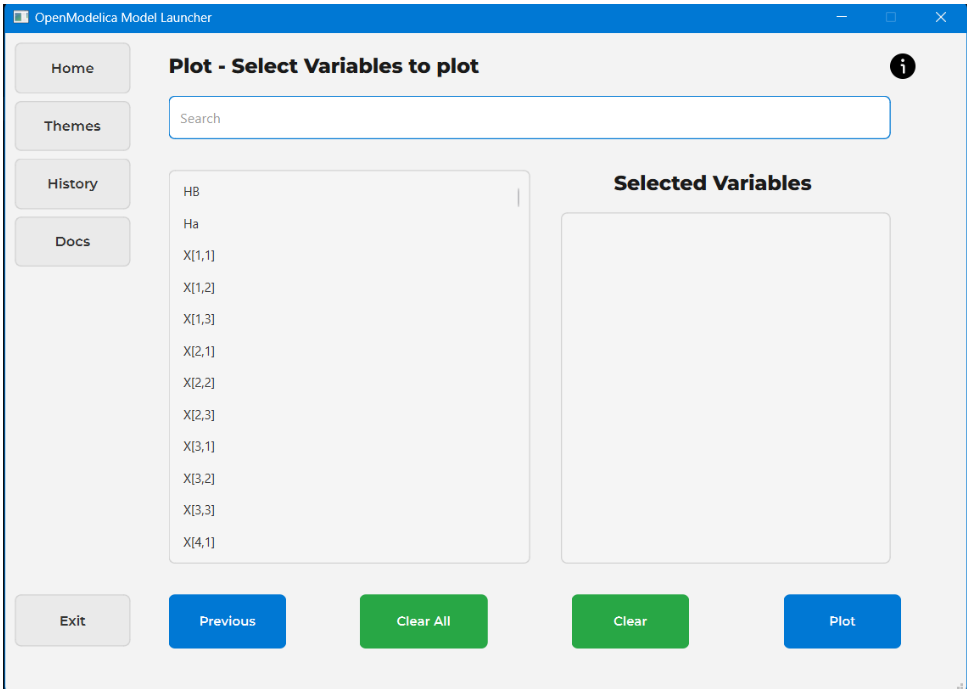



Figure 3.13: List of Scalar Variables Extracted from XML File

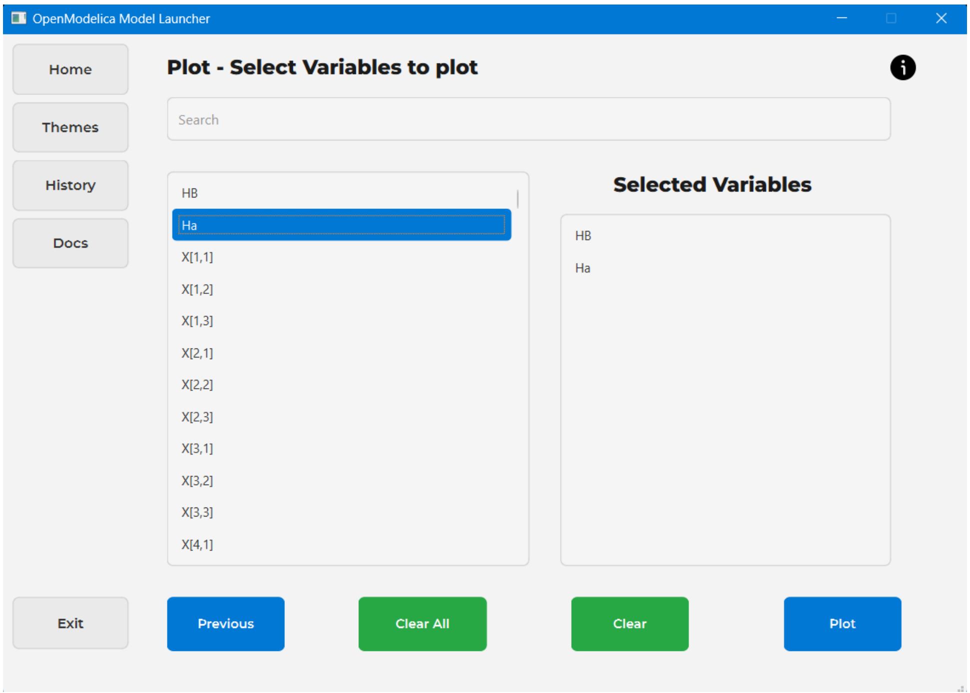



Figure 3.14: List of selected Scalar Variables

### 3.2.11 Plotting Simulation Results with OMPlot

After selecting one or more scalar variables from the list, the user can click the **Plot** button. This action triggers a subprocess that launches `OMPlot`, a built-in plotting tool of OpenModelica, to visualize the results stored in the `.mat` file.

The GUI invokes OMPlot using the following Python code:

```
result = subprocess.Popen(
    [om_plot, "--plot", f"--filename={self.file_path}"] + self.variables,
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
    text=True,
)
```

Where:

- `om_plot` is the path to the `OMPlot.exe` executable.
- `self.file_path` refers to the full path of the generated `.mat` file.
- `self.variables` is a list of selected variable names passed as command-line arguments.

OMPlot reads the `.mat` result file and plots the specified variables over the simulation time interval. This provides a clear and interactive visual representation of the system's dynamic behavior.

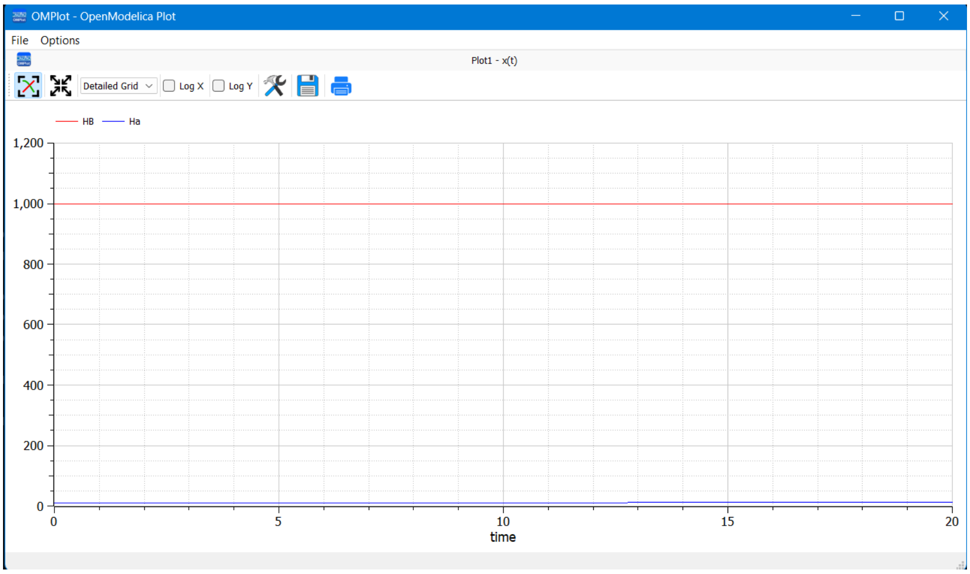



Figure 3.15: OMPlot Showing Simulation Results for Selected Variables

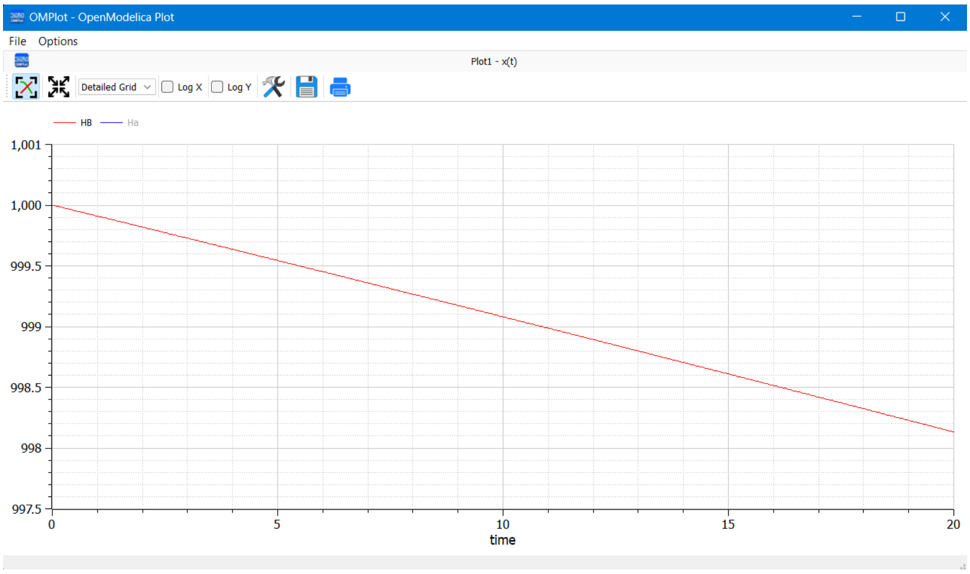



Figure 3.16: OMPlot Showing Simulation Results for Selected Variables

### 3.2.12 Logging and Debugging Support

To enhance transparency and aid in debugging, the GUI provides real-time console logging as well as persistent log file generation. This ensures that users can monitor the progress of each stage — from file generation to simulation and plotting — and revisit the logs in case of failures or warnings.

- **Console Logging:** A dedicated logging window is embedded within the GUI, displaying all backend activity such as file generation, script execution, subprocess outputs, and warnings in real time.
- **Log File Generation:** The GUI creates a log file (e.g., `modelica.log`) during every simulation run. This log contains detailed output from `omc.exe`, including:
  - Simulation configuration and options
  - Compilation and simulation messages
  - Time taken for each stage (frontend, backend, compilation, simulation)
  - Warnings and error messages, if any
- **Error Diagnostics:** Warnings such as incomplete initial conditions or unit mismatches are captured and displayed both in the console and the log file, helping users quickly identify and correct issues.

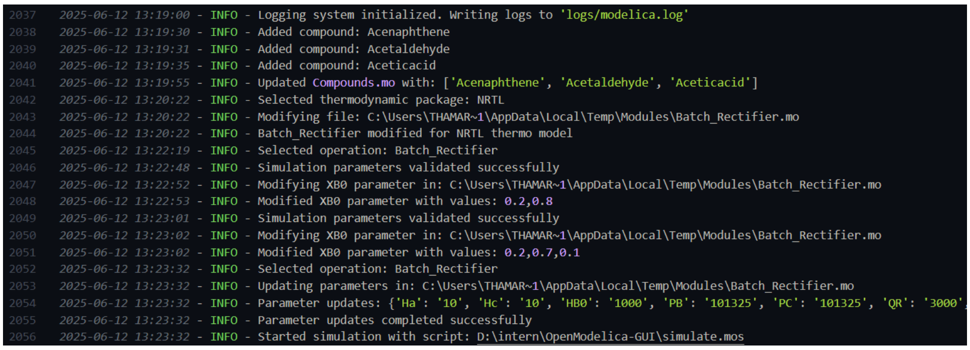



Figure 3.17: Console Log Window Displaying Real-Time Simulation Output

### 3.2.13 Theme Customization (Dark / Light Mode)

Users can toggle between **Dark Mode** and **Light Mode** dynamically during runtime.

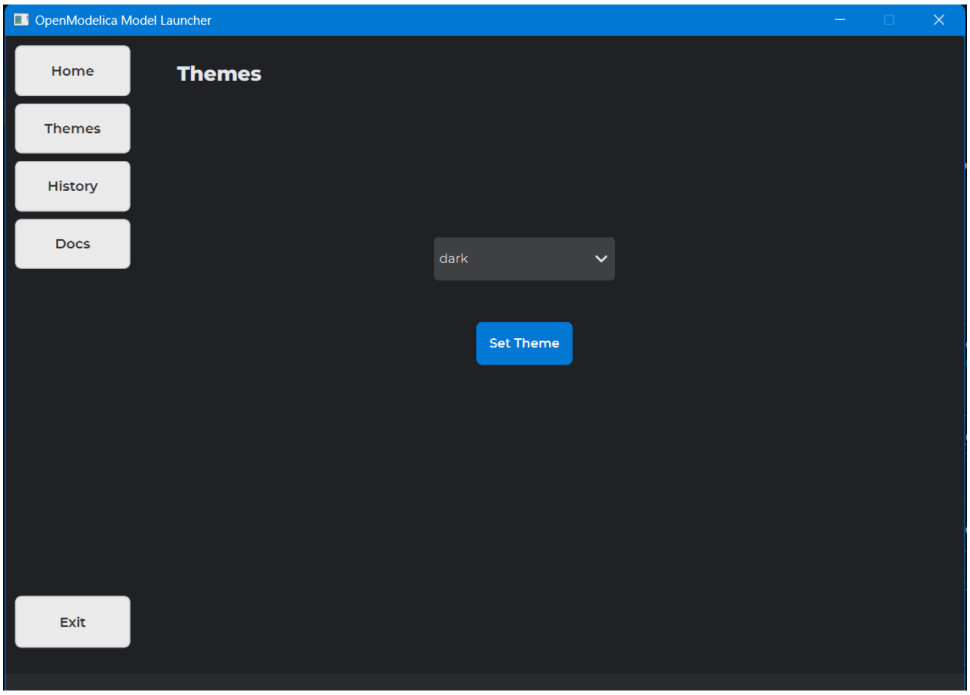



Figure 3.18: Dark Theme

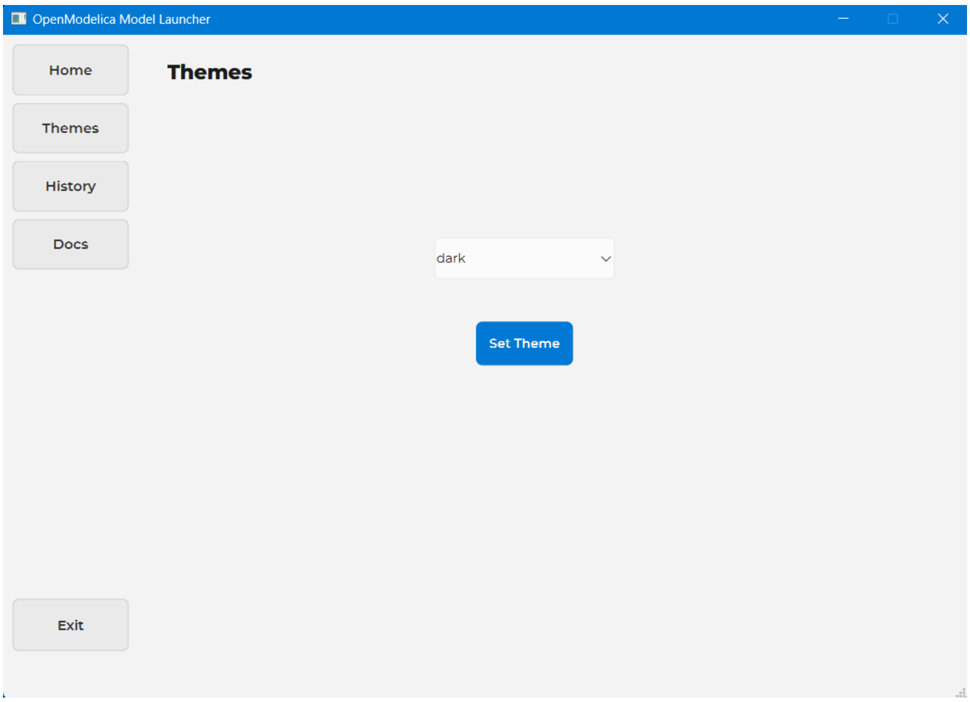



Figure 3.19: White Theme

# Chapter 4

# Conclusion and Future Scope

The development of a GUI for OpenModelica-based batch distillation simulation has successfully streamlined an otherwise complex and manual process. By integrating multiple functionalities — including compound selection, thermodynamic setup, model operation selection, parameter input, simulation execution, real-time logging, and result visualization — the application provides a unified and user-friendly interface for process simulation.

This project reduces the dependency on manual scripting and file manipulation, making Modelica-based simulations accessible to students, researchers, and engineers without deep programming knowledge. The addition of features like OMPlot integration, theme customization, and logging further improves usability, debugging, and aesthetics.

Through this internship, a comprehensive understanding of PySide6 for GUI development, OpenModelica integration, and process simulation workflows was gained. The outcome is a modular, extensible application that can serve as a strong foundation for future features such as support for additional unit operations, export options, and advanced result analytics.

# Bibliography


[1] OpenModelica Official Documentation.
URL: https://openmodelica.org/documentation

[2] Qt for Python (PySide6) Official Documentation.
URL: https://doc.qt.io/qtforpython/

[3] Qt Designer Manual - Qt Documentation.
URL: https://doc.qt.io/qt-6/qtdesigner-manual.html

[4] Matplotlib Documentation.
URL: https://matplotlib.org/stable/contents.html

[5] Pyzipper — A Python module for encrypted zip files.
URL: https://pypi.org/project/pyzipper/

[6] Python Docs: subprocess module.
URL: https://docs.python.org/3/library/subprocess.html

[7] Python Docs: xml.etree.ElementTree module.
URL: https://docs.python.org/3/library/xml.etree.elementtree.html

[8] Python Docs: re — Regular expression operations.
URL: https://docs.python.org/3/library/re.html

[9] Python Docs: pathlib — Object-oriented filesystem paths.
URL: https://docs.python.org/3/library/pathlib.html

[10] FOSSEE, IIT Bombay - Official Website.
URL: https://fossee.in

[11] OpenModelica OMEdit GUI Tool.
URL: https://openmodelica.org/tools/omedit

[12] OpenModelica OMPlot Tool.
URL: https://openmodelica.org/tools/omplot

[13] qdarktheme: A modern dark/light theme for Qt.
URL: https://pypi.org/project/qdarktheme/