Single-layer Perceptron

Activity #4

G022 | CSci 141 | Intelligent Systems

Vaughn Cedric L. Araneta

BSCS-3

Date of Submission: December 18, 2025

Instructor: Prof. Jonah Flor O. Maaghop

About

QLogicae SimNapse is a Boolean, Single Perceptron desktop simulation software. To be brief, users can modify a perceptron's parameters, such as input, output, and maximum learning epoch, and observe how it teaches itself to predict the correct output. This project was built using the C++20 programming language and is designed to be run on the Windows 11 platform.

Purpose

Similar to any endeavor, big or small, strategy is useless without purpose – a goal. The goal of SimNapse is purely educational, allowing users to experiment with a perceptron's parameters, inputs, and outputs, and observe how the weights and assumed predicted outputs change over time to fit the desired output set by the users.

Usability aside, this project also serves as a side-experiment for the 'QLogicae CLI' project, which is a multi-purpose software development tool used to build and deploy C++20 applications. SimNapse essentially helps in giving insights into how 'QLogicae CLI' performs throughout SimNapse's development lifecycle, from project initialization to release.

Features

SimNapse is a relatively simple project, but it is nothing without value. The core features are as follows:

1. Customizable Input and Output Parameters
    - The ability to teach a perceptron to predict the outputs of AND and OR operations via customized input and expected output values.
2. Real-Time Learning
    - When training begins, users can watch it improve its accuracy, in real-time.
3. Evaluation History
    - Assessing just the final accuracy may be lacking - you can also view the learning results, weights and biases, through time, from the first and last iteration.

Scope and Limitations

Developing machine learning simulators takes time and effort, similar to any form of work. As much as the development team can see further potential for development, we will need to comply with realistic limitations and a few rules to maximize time and effort spent on making this project come to fruition. Having said that, this does not mean that our efforts have been wasted to this day; more expressive, efficient, reliable and sophisticated applications will take on the experience gained from developing projects like SimNapse.

The following limitations will include:

1. The application can be run on Windows 11. It's predecessor, Windows 10, may have to be tested out by any curious users since the development team could not confirm the application's stability when run in such an operating system. We only possess Windows 11.

Pre-requisites

To avoid technical issues, please assess your system if it meets or exceeds the following requirements:

1. Operating System – Windows 11
2. RAM - At least 1 GB
3. Storage - At least 80 MB

Installation

The SimNapse application installer can be found via the GitHub link - https://github.com/CedricDeVon/qlogicae_simnapse. All that is required is navigating to the aforementioned link, clicking 'Initial Release', under the 'Release' section on the right sidebar. Finally, under 'Initial Release', click on the '.exe' item under the 'Assets' section.

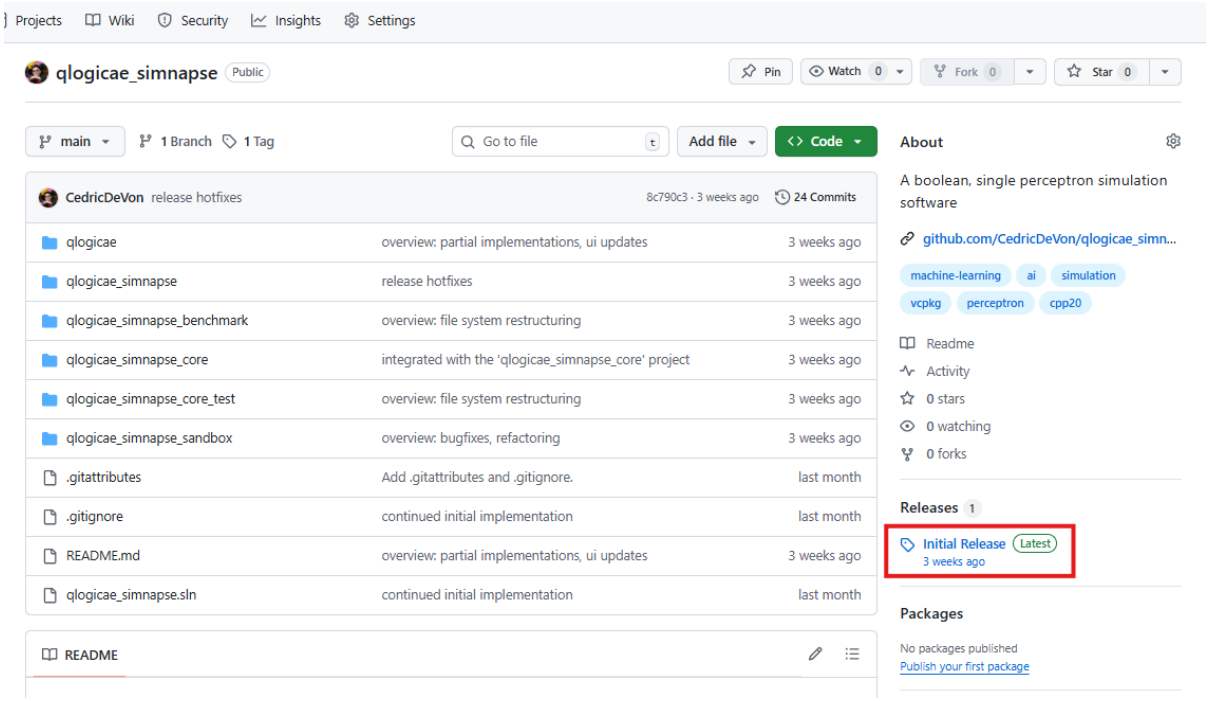

Figure No. 1 – The SimNapse GitHub Homepage

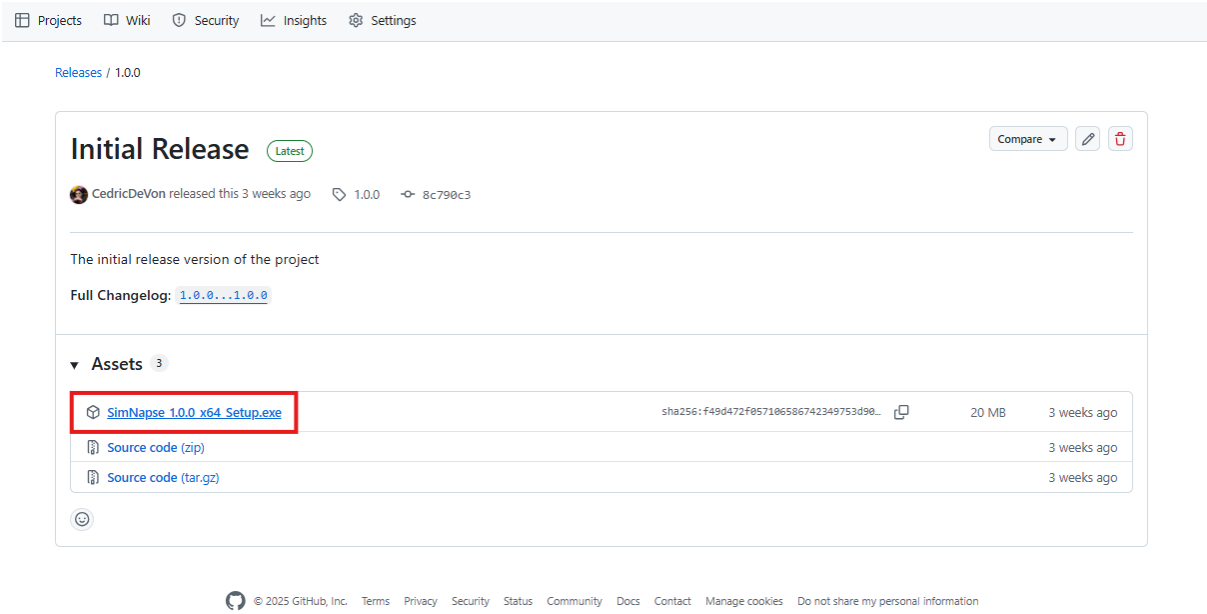

Figure No. 2 – The SimNapse 'Initial Release' Page

When download is complete, double-click on the installer, a Windows Installer Wizard will show on your screen. Next, the following sections: License Agreement, Select Destination Location, Select Additional Tasks, and Ready to Install will be displayed, in order. For convenience, all installation options are pre-defined during development. You can click 'Next' all throughout, until the application is successfully installed.

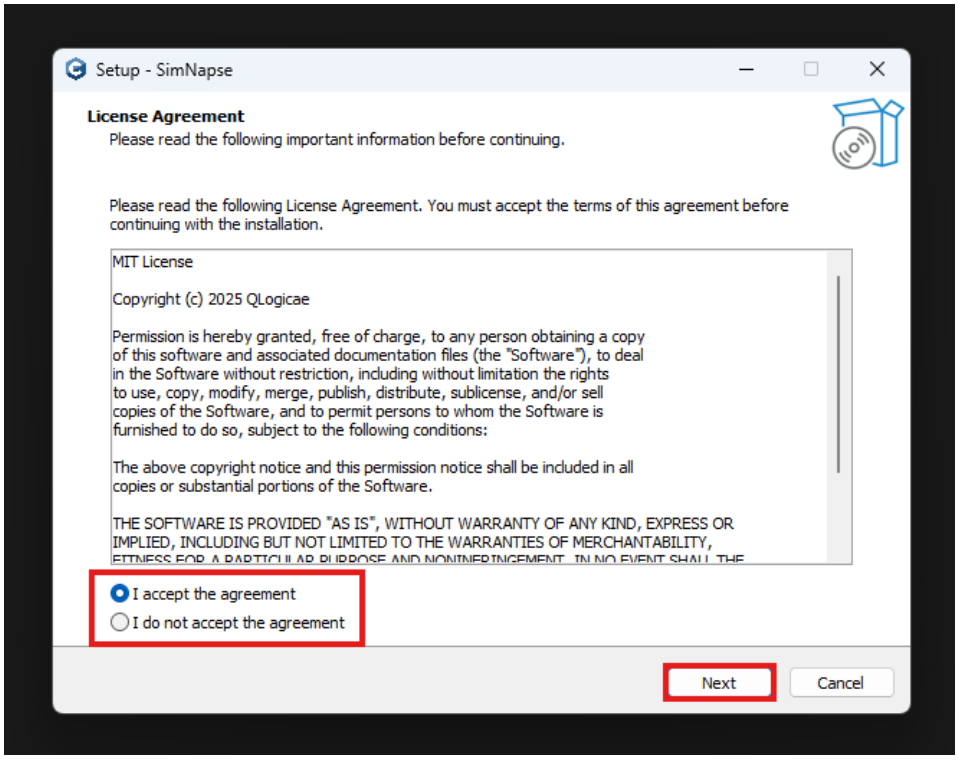

Figure No. 3 – SimNapse Setup Wizard, License Agreement

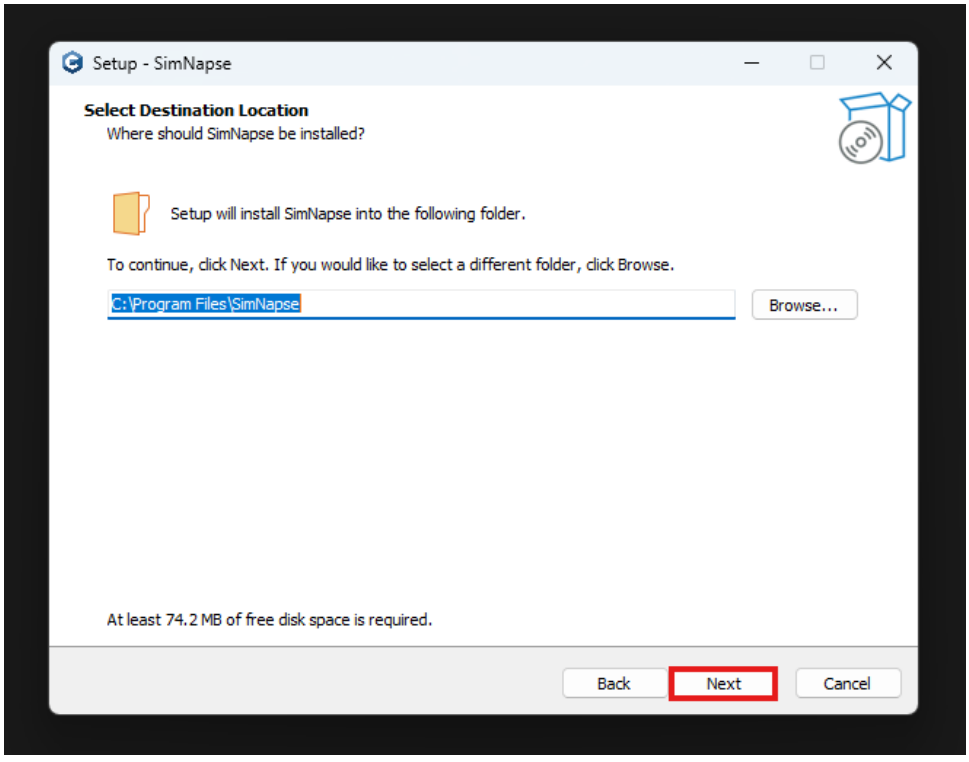

Figure No. 4 – SimNapse Setup Wizard, Select Destination Location

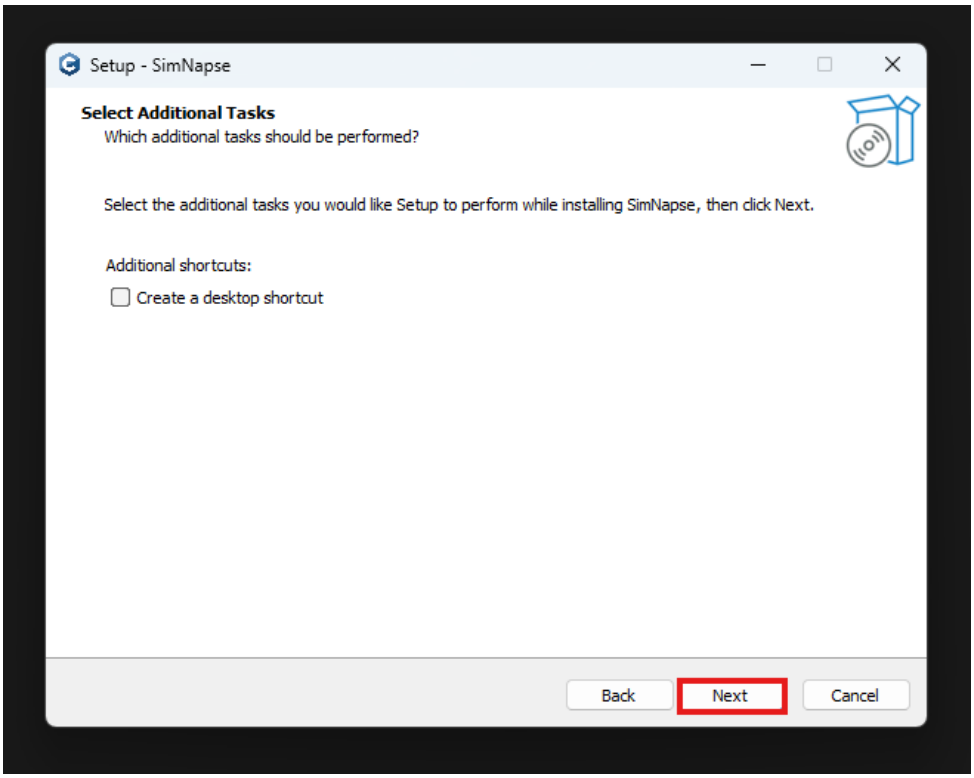

Figure No. 5 – SimNapse Setup Wizard, Select Additional Tasks

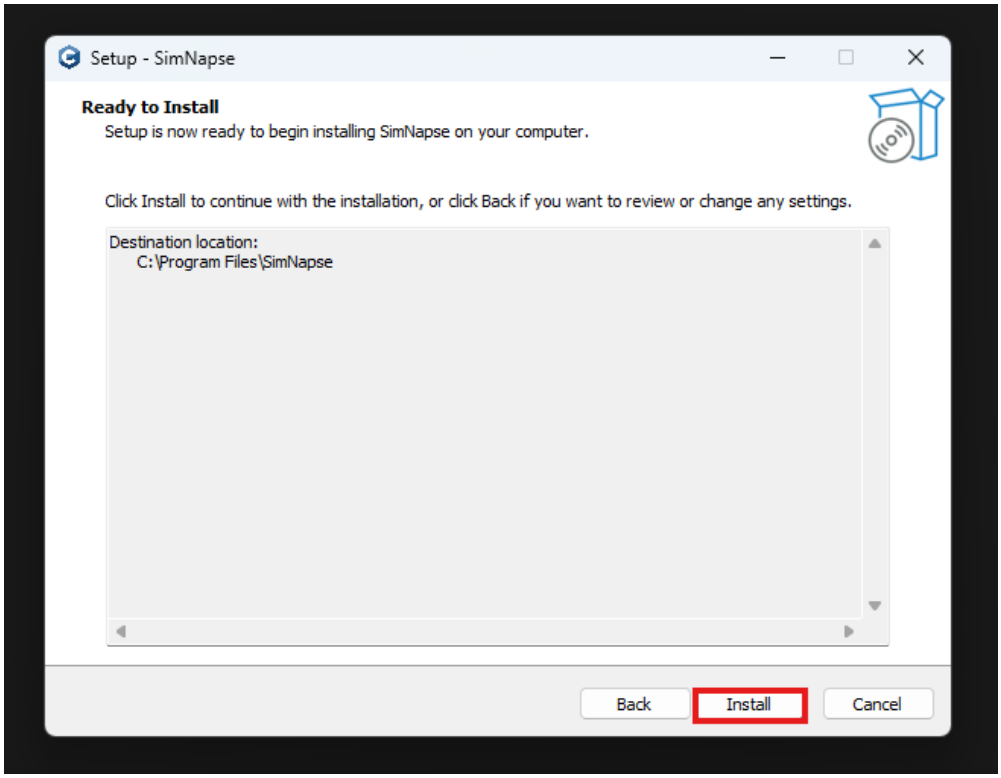

Figure No. 6 – SimNapse Setup Wizard, Ready to Install

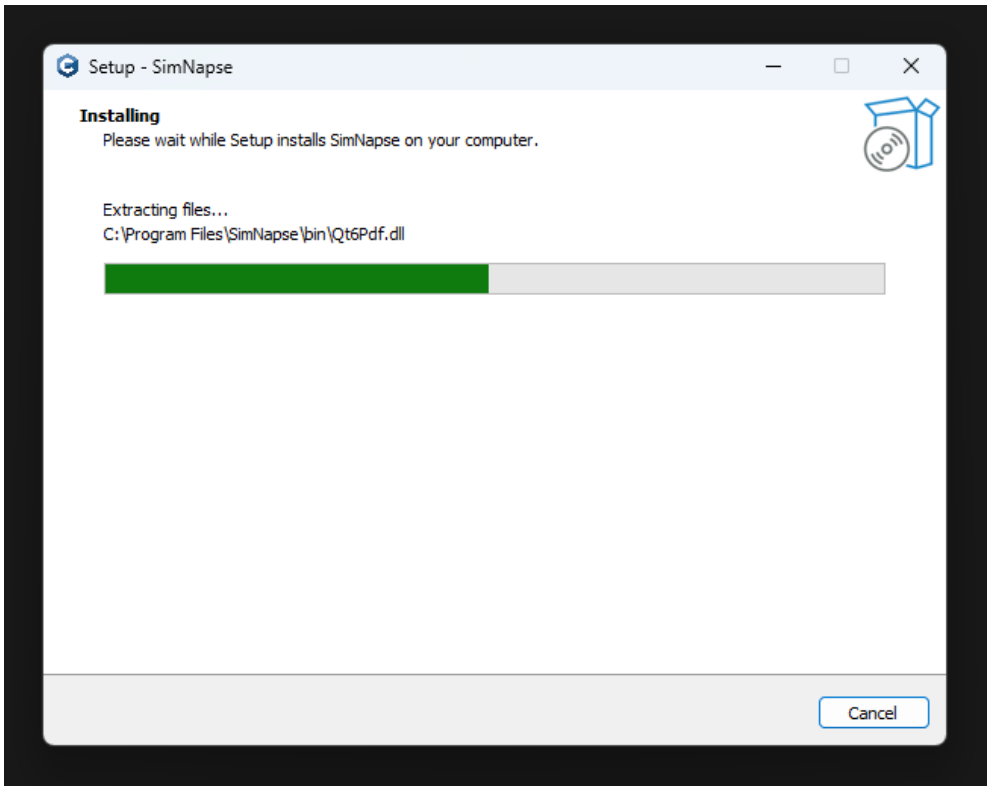

Figure No. 7 – SimNapse Setup Wizard, Installing

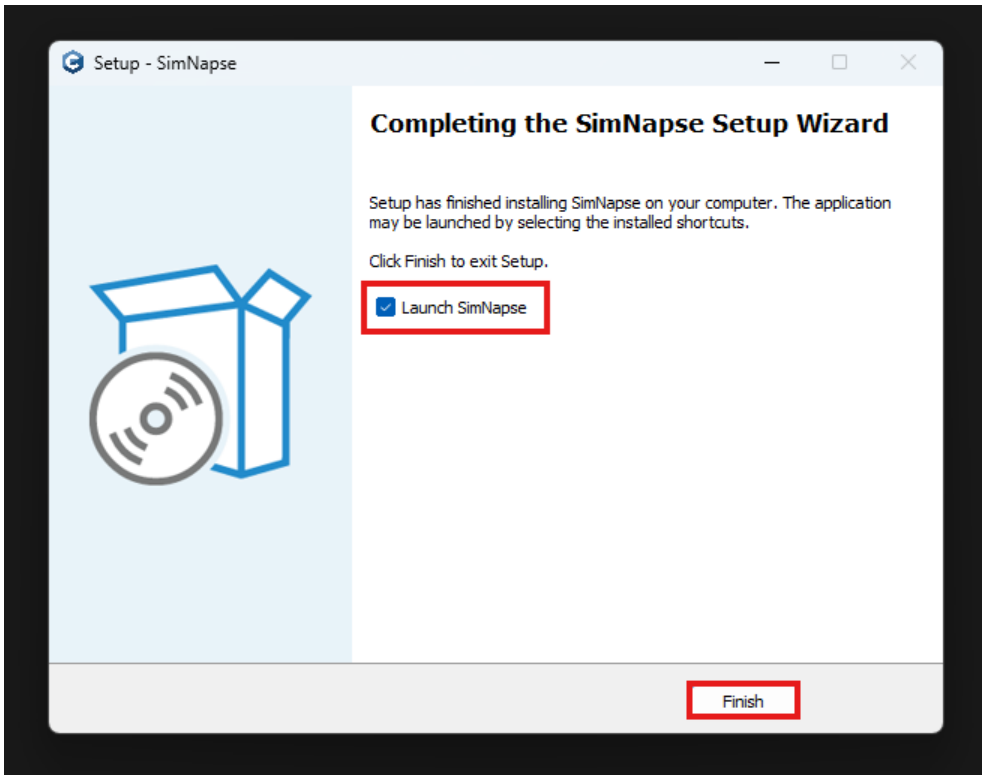

Figure No. 8 – SimNapse Setup Wizard, Completion

Usage

Installing and executing the application will eventually lead you to this display – SL Perceptron Evaluation. From here, you can set several parameters such as the 'Final Epoch', 'Input 1', 'Input 2', and 'Output'. When ready, you can click the 'Learn' button to start the simulation.

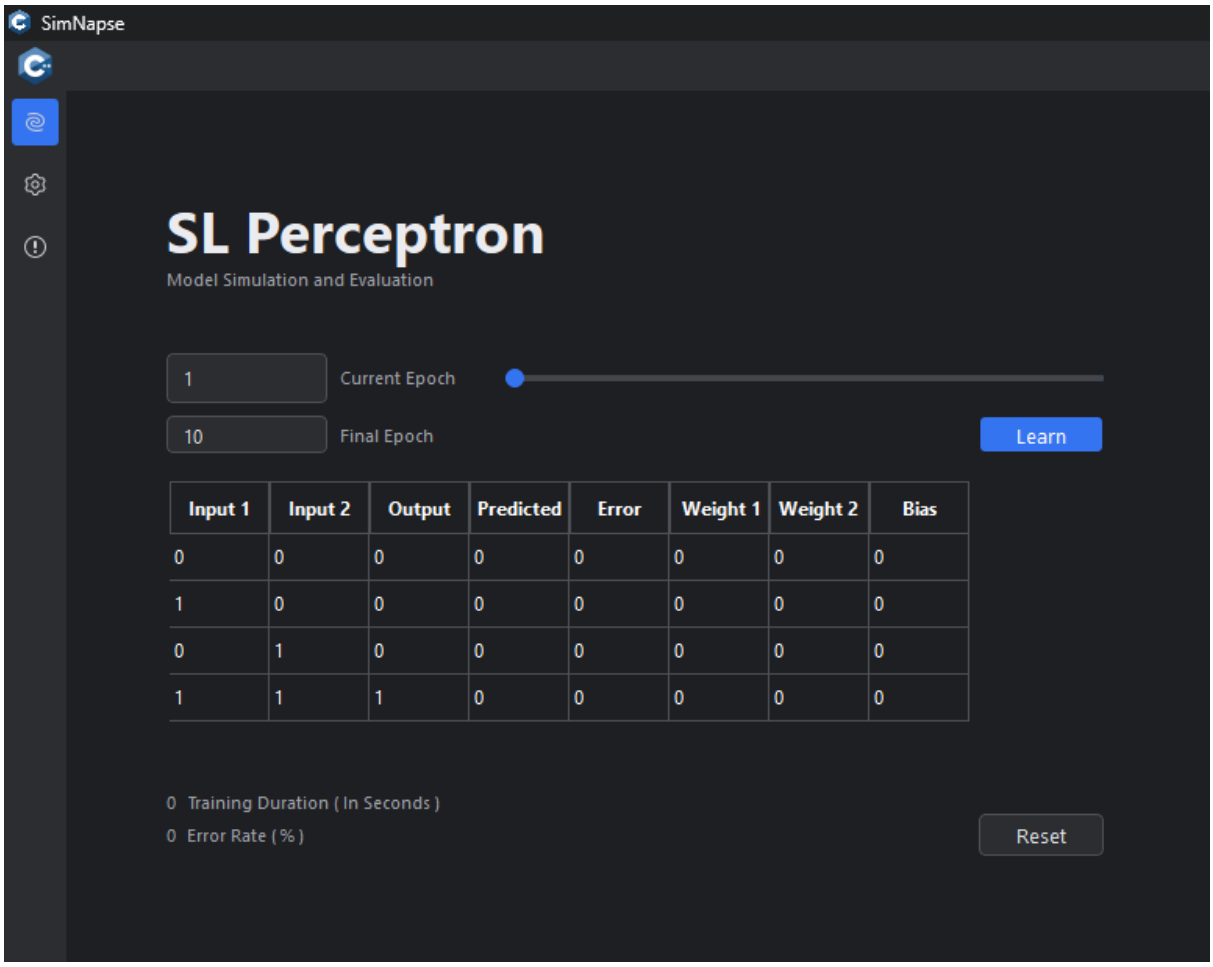

Figure No. 9 – SimNapse Evaluation | Pre-Learn Values

One of the main factors which will affect the learning performance is the 'Final Epoch' value. Users can set it to 100,000 if they would like, however, training result values such as: 'Predicted', 'Error', 'Weight 1', 'Weight 2', and 'Bias', usually remain stationary after epoch no. 5 to 7.

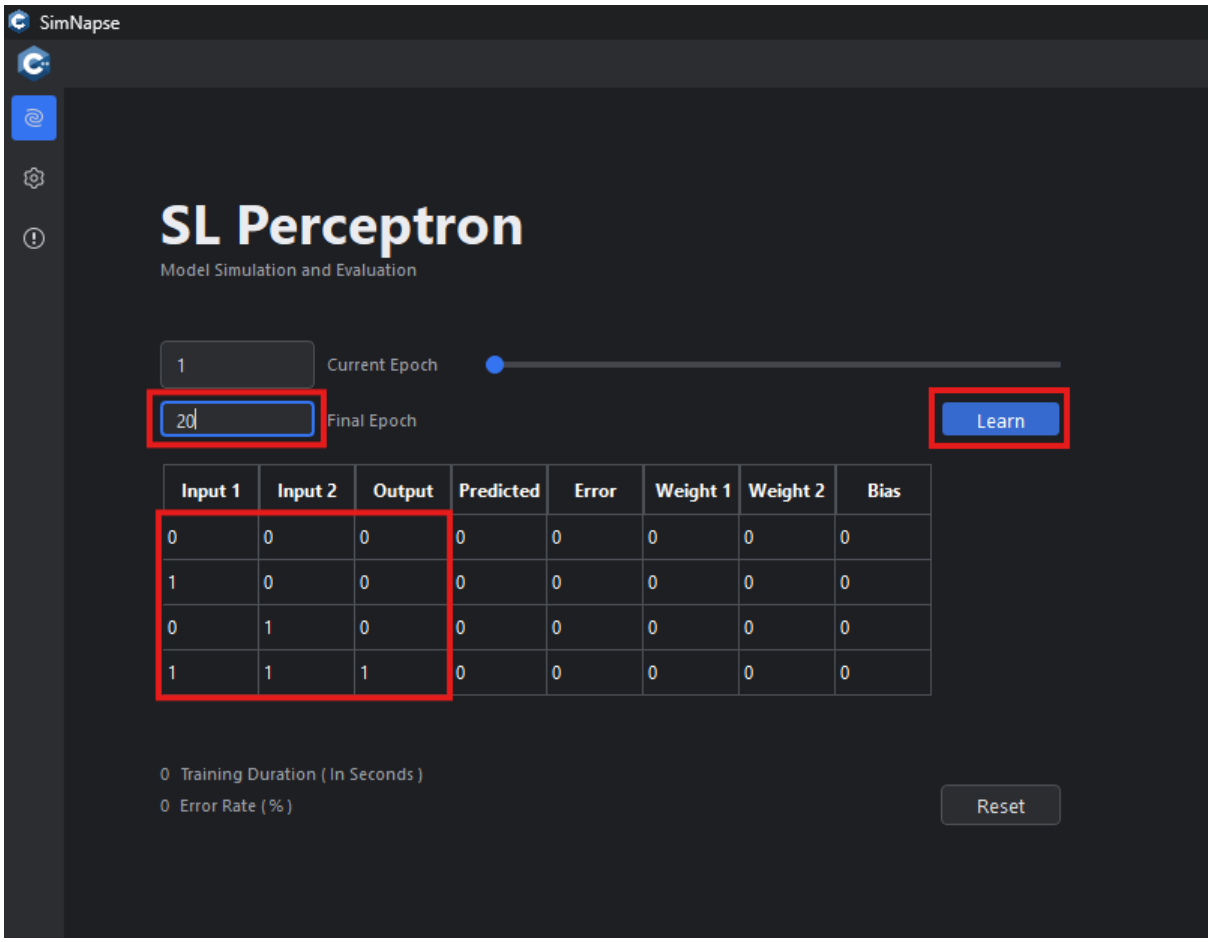

Figure No. 10 – SimNapse Evaluation | Post-Learn Values

As stated previously, users can also possess the ability to view each training result value on each epoch. That can be observed by interacting with the blue-colored slider at the top-right of the result table. Finally, to start back from scratch, you can simply click on the 'Reset' button on the bottom-right of the result table.

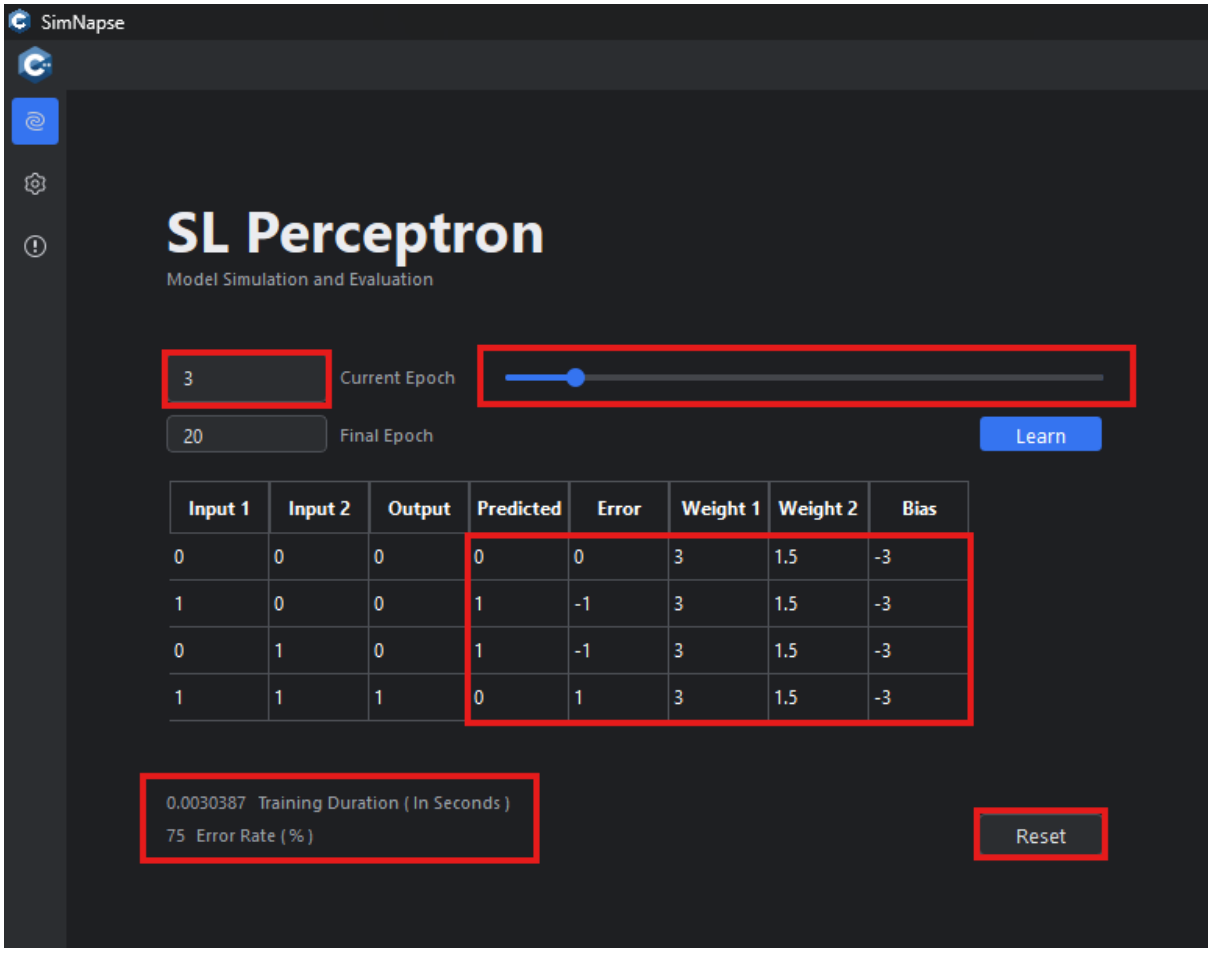

Figure No. 11 – SimNapse Evaluation | 3rd Epoch Values

Perceptron Model Architecture

The following content will include context regarding what the perceptron model would look like, in its logical form:

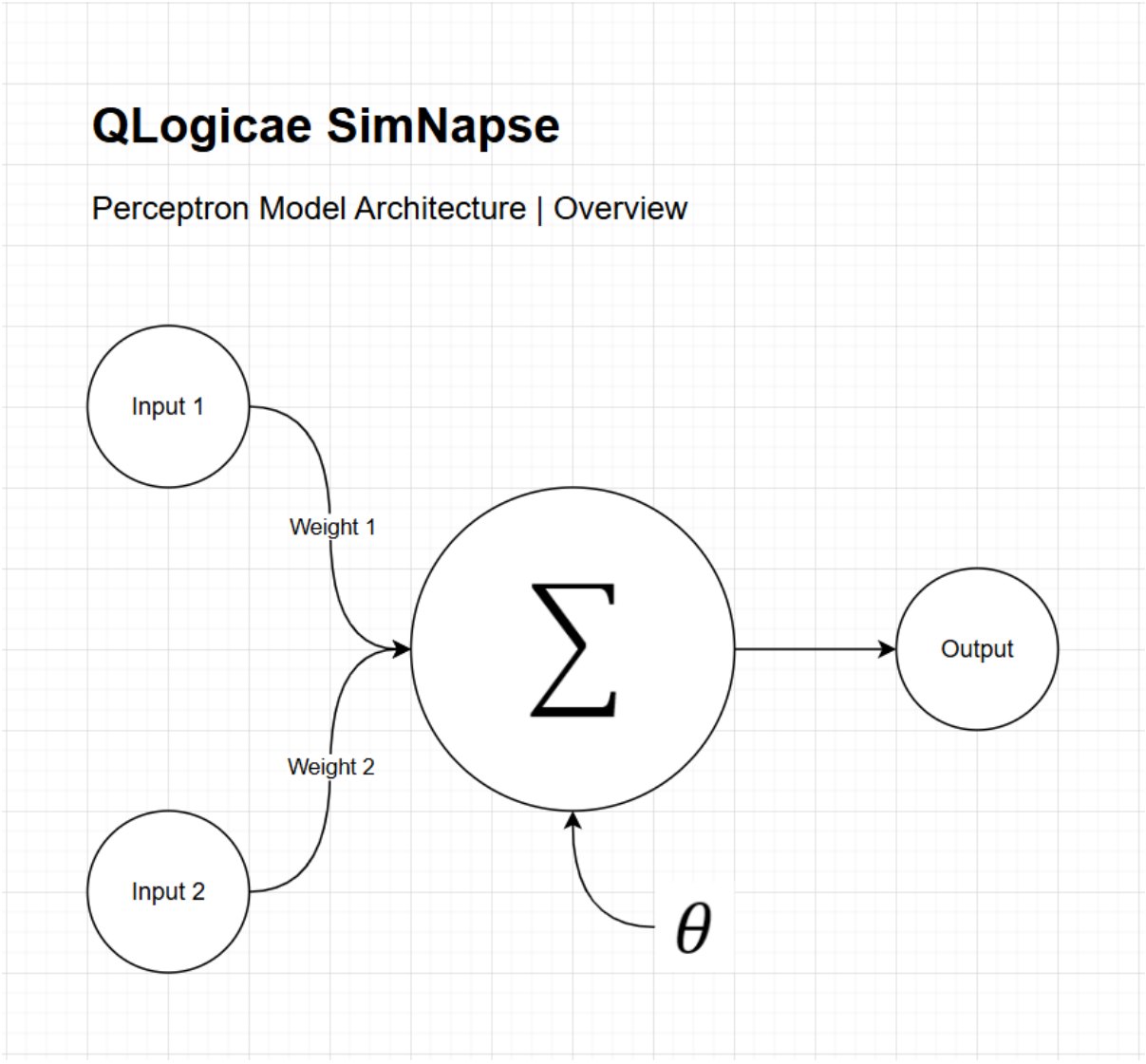

Figure No. 12 – The SimNapse Perceptron Model Architecture

Methodology

The Waterfall software development methodology has been chosen and applied for the following reasons:

1. To develop a single-version application, with potential bugfixes after the initial release.
2. To optimize time management and work efficiency, for a relatively small-sized project.

Requirements

One of the first phases of such a project methodology will involve identifying what type of application the development team is producing, the purpose of the project, and the reality of the scope and limitations to be compromised.

Design

Next, this stage will involve further research into the theoretical and practical concepts of the perceptron model, software architecture, and software design timeline.

Development

The implementation of the C++20 core library, unit test suite, and Qt desktop application will take place during this stage, respectively. Before the development of the desktop application commences, the core library must be implemented and thoroughly unit tested first before it can be accepted for integration with the Qt desktop application.

Unfamiliar user interface components within the Qt framework will be experimented during this stage.

Testing

As for now, the current and practical means of Quality Assurance include GTest Unit Testing and Acceptance Testing from the developers themselves. The latter method involves typical, manual interaction with the SimNapse application, from installation to perceptron training and evaluation.

Relevant unit tests yielded a 98.73% success rate based on the ‘QLogicae SimNapse Core Test’ project.

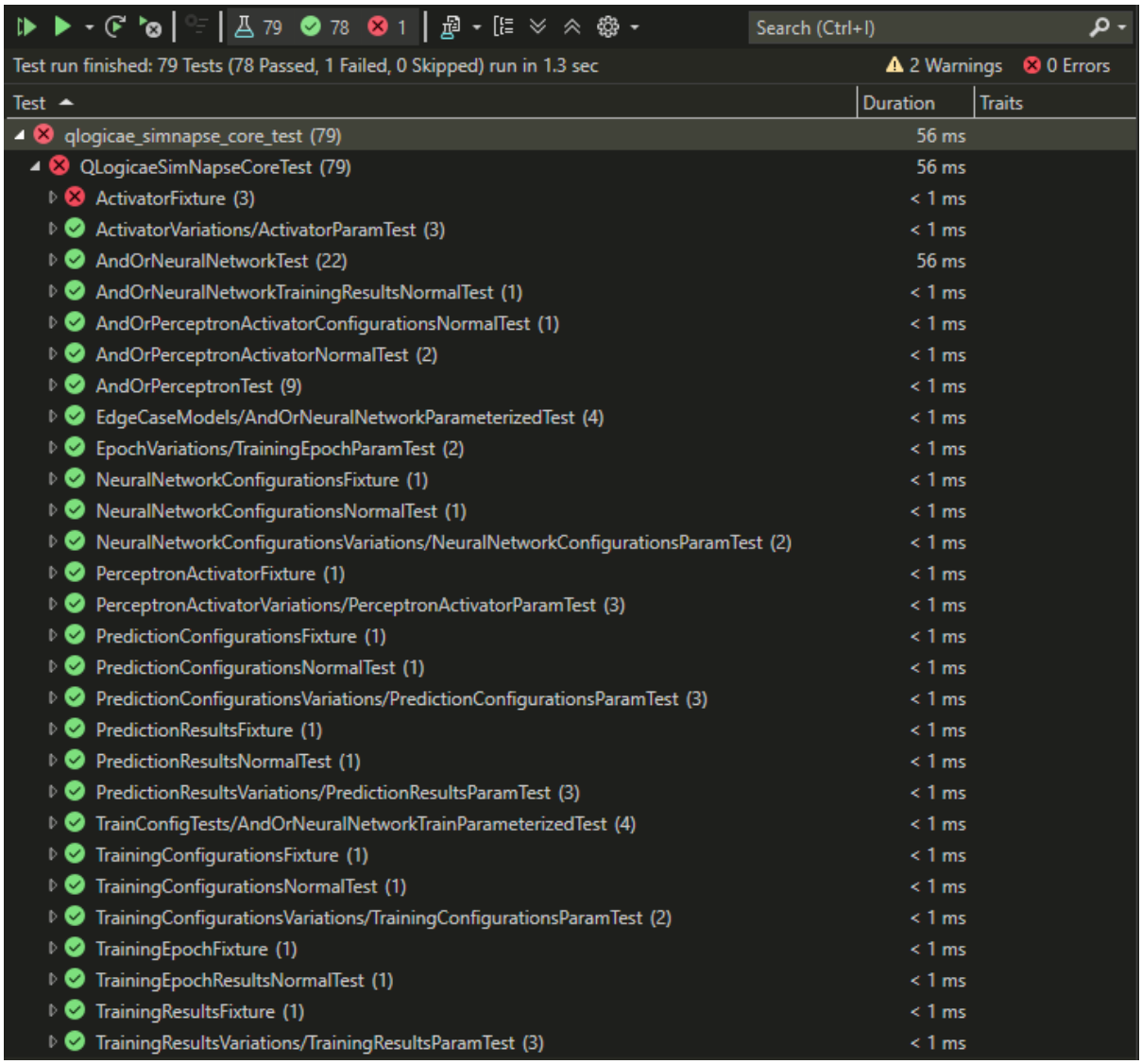

Figure No. 13 - 'QLogicae SimNapse Core Test' Results

Deployment

The production build itself will be compressed and packaged inside a Windows Setup Wizard via the Inno Setup console application. Executable deployment itself will be hosted via GitHub (visit the Installation section for more information).

Maintenance

Recently found bugs and documentation updates will take place during the maintenance project management phase. No new features will be implemented until further notice by the development team.

Development Tools and Technologies

The relevant items to be listed within this section are as follows:

1. **Programming Languages**
    - C++20
        - Commonly utilized for performance-critical software applications.
2. **IDEs**
    - Visual Studio 2022
        - One of the reasonable IDEs used for developing C++ applications.
        - Integrates well with the Vcpkg package manager.
3. **Frameworks**
    - Qt
        - A C++ framework for developing desktop applications.
    - QLogicae Core
        - A custom-built C++ framework, containing wrapper code of 3rd party packages, for developing desktop applications.
4. **Package Manager**
    - Vcpkg
        - A package manager for developing C++ applications.

5. **Benchmarking**
    - Nanobenchmark
        - A C++ library for nanosecond-level, runtime benchmarking analysis.
6. **Unit Testing**
    - GTest
        - A C++ library for unit testing.
        - Comes pre-installed together with Visual Studio 2022.
7. **Deployment**
    - Inno Setup
        - A console application for creating custom Windows Setup Wizards for given software builds.
    - QLogicae CLI
        - A multi-purpose, software development tool for building and deploying C++ applications integrated with the QLogicae Framework and Visual Studio 2022.

Software Architecture

This section will provide context as to what the Visual Studio 2022 project is structured:

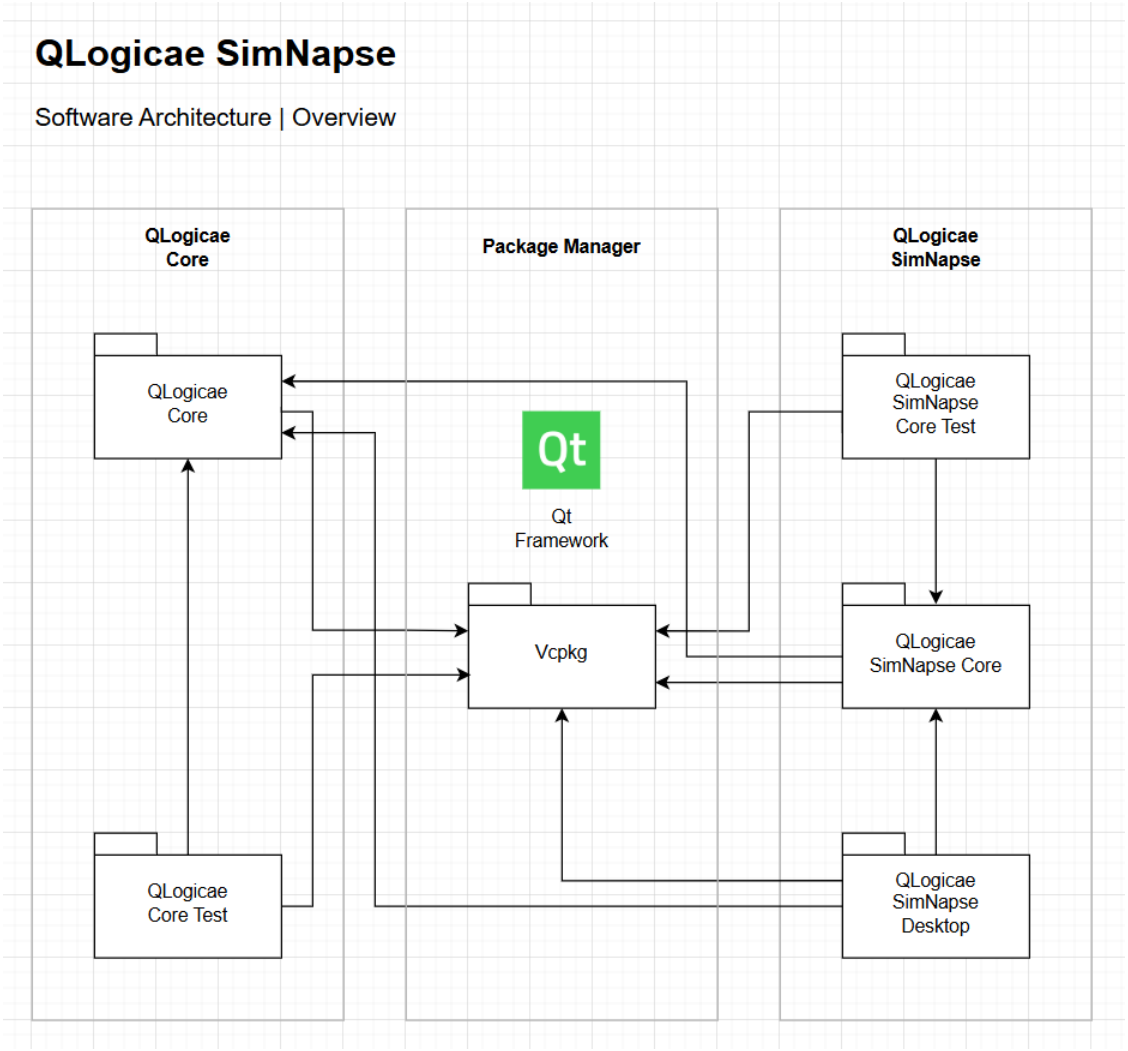

Figure No. 14 – The SimNapse Software Architecture

To start, a 'Solution' is simply a set of Visual Studio 2022-recognizes projects. The SimNapse project utilizes two 'Solutions' of projects for software development – QLogicae SimNapse Proper, and QLogicae Core. 3rd party C++ packages and libraries will be utilized within the Vcpkg package manager.

The projects found within the ‘QLogicae SimNapse’ solution include ‘QLogicae SimNapse Core’, a C++ static library implemented to contain the Single Perceptron-related code and its functionalities. ‘QLogicae SimNapse Core Test’, which contains GTest testing code for the ‘QLogicae SimNapse Core’ project itself. Finally, the ‘QLogicae SimNapse Desktop’ project represents the actual desktop application implementation itself, which depends on ‘QLogicae Core’ and ‘QLogicae SimNapse Core’, respectively.

The ‘QLogicae Core’ solution includes a wrapper library for multiple Vcpkg packages with the purpose of reducing code duplication and implementing a common software API to be used in QLogicae-integrated projects. SimNapse is one of such applications. ‘QLogicae Core’ is the designated C++ static library, while ‘QLogicae Core Test’ is implemented to test the former.

References

1. *Chapter 6 – Artificial Neural Network*
2. *Chapter 7 – Single Layer Perceptron*
3. *Chapter 8 – Multilayer Perceptron*
4. *CedricDeVon. (n.d.-d). GitHub - CedricDeVon/qlogicae_simnapse: A boolean, single perceptron simulation software. GitHub. https://github.com/CedricDeVon/qlogicae_simnapse*
5. *CedricDeVon. (n.d.-a). GitHub - CedricDeVon/qlogicae_cli at feature. GitHub. https://github.com/CedricDeVon/qlogicae_cli/tree/feature*
6. *CedricDeVon. (n.d.). GitHub - CedricDeVon/qlogicae-core at feature. GitHub. https://github.com/CedricDeVon/qlogicae-core/tree/feature*
7. *CedricDeVon. (n.d.-c). GitHub - CedricDeVon/qlogicae_opera. GitHub. https://github.com/CedricDeVon/qlogicae_opera*
8. *GoogleTest User's Guide. (n.d.). GoogleTest. https://google.github.io/googletest/*
9. *QT | Tools for each stage of software development lifecycle. (n.d.). https://www.qt.io/*
10. *vcpkg - Open source C/C++ dependency manager from Microsoft. (n.d.). https://vcpkg.io/en/index.html*
11. *Sdwheeler. (n.d.). Install PowerShell on Windows - PowerShell. Microsoft Learn. https://learn.microsoft.com/en-us/powershell/scripting/install/install-powershell-on-windows?view=powershell-7.5*
12. *Visual Studio IDE - AI for coding debugging and testing. (2025, December 8). Visual Studio. https://visualstudio.microsoft.com/vs/*