# Network Diagnostics Tool Report

## Tool Design

My goal was to create, as tasked, a lightweight network diagnostics tool which collects and stores network and system statistics in an SQL database for future data analysis and visualization.

I wanted the tool to also be platform-independent, so that at any given time additional support for a previously unsupported platform could be added easily.

StatisticsMonitorFactory

DiagnosticsController

* IStatisticsMonitor \* m\_systemMonitor
* IStatisticsMonitor \* m\_networkMonitor
* DatabaseManager\* m\_dbManager

IStatisticsMonitor

INetworkStatisticsMonitor

ISystemStatisticsMonitor

DatabaseManager

SystemMonitorLinux

DatabaseUtils

NetworkMonitorLinux

CommandExecutor

**IStatisticsMonitor** interface provides a generic abstraction for monitoring statistics. This ensures flexibility and extensibility for various types of system and network monitoring tasks.

**ISystemStatisticsMonitor** and **INetworkStatisticsMonitor** interfaces inherit from IStatisticsMonitor, defining specific functionalities for system and network monitoring.

**SystemMonitorLinux** and **NetworkMonitorLinux** are concrete implementations tailored for Linux systems, defining platform-specific behavior while adhering to their interfaces.

**StatisticsMonitorFactory** centralizes the creation of monitor objects, ensures proper instantiation based on runtime requirements (platform, configuration detection).

**CommandExecutor** executes system-specific commands.

**DatabaseManager** class handles data persistence, uses methods from the **DatabaseUtils** class for low-level database operations. This modularity improves code readability and reusability.

**DiagnosticsController** is the central coordinator and is responsible for the tool's operation. It integrates a SystemMonitor and a NetworkMonitor for data collection. Is also responsible for handling the DatabaseManager instance.

## Challenges Faced

* Integrating both C++ and batch/bash scripts in the tool development
* Deciding which programming languages to use for each stage (monitoring, storing, parsing, loading into the database)
* Coordinating the execution of the scripts and storing the data required careful planning
* Supporting both Windows and Linux OS, different commands and output formats made the parsing process challenging
* Various possible system and network statistics to monitor, need to be parsed differently
* Not enough time to properly plan the tool’s design, changed design several times during the assignment
* Converting from string format in CSV to numeric type in the database
* Selecting which analyzations to perform on the dataset
* Handling file paths after reorganizing the project directory
* Understanding and correctly implementing potential network issues and the correlation between network and system metrics
* Making the tool as user-friendly as possible, with clear and precise instructions

## Insights

* Modular Design: Building modular components enabled better integration and reusability.
* Cross-Platform Compatibility: Abstracting platform-specific details simplified the development and allows for extensibility (adding support for new platforms is straightforward)
* Visualization: Using external tools such as python for data visualization was simple and did not require complicating the core tool.

## Tool Enhancement

* Add user configurability: enable setting the intervals and time frame of data collection
* Add user interactivity: allow users to define which statistics they want to monitor
* Display network statistics in console log or user interface
* Use external open-source libraries to simplify data parsing