# Design Document: Process Monitoring Utility

## 1. Introduction

* This document's goal is to describe the design and implementation specifics of a command line utility for monitoring and terminating processes in accordance with predetermined criteria.
* Project overview A process name, its maximum lifespan (in minutes), and a monitoring frequency (also in minutes) are the three arguments that the utility expects.

## 2. Requirements

* The utility has to start checking on processes at the designated interval.
* The tool should stop the process and make a note to the log if a process of interest runs longer than the permitted time.
* Even if there isn't a process running at any given time, the utility should still be kept under observation.
* When a certain keyboard button (such as 'q') is pushed, the utility should terminate.

## 3. Design

* Architecture: The tool will be made to run from the command line.
* User Interface: The command line interface is how the user will communicate with the tool.
* Components:
  + Line of Command Parser: Charged for checking the format of the input arguments and parsing them.
  + Process Monitor: Keeps track of the designated process and terminates it if it runs longer than permitted.
  + Logger: Records pertinent data, such as process specifics and actions conducted.

## 4. Workflow

* Parse Command Line inputs:
  + From the supplied inputs, the command line parser component will validate and extract the process name, maximum lifetime, and monitoring frequency.
* Start Monitoring:
  + Based on the given frequency, the utility will start monitoring the process.
  + It will continually check to see if the process is running and if its maximum lifetime has been reached.
* Kill Process and Log:
  + The tool will kill the process and log the relevant record if it is discovered that a process has exceeded the maximum lifetime.
* Continue Monitoring:
  + Even if there are no processes running, the program will keep checking in case any later processes start up.
* Stop Monitoring:
  + Pressing the specific keyboard button ('q') causes the utility to stop monitoring and quit.

## 5. NUnit Test

* Test Case 1: Command line argument parsing and validation.
* Test Case 2: Invalid Command line argument parsing and validation.
* Test Case 3: Validate the process killing mechanism.
* Test Case 4: Ensure the logging functionality records the expected information.

### Code Coverage with NUnit Tests

* Test Case 1:
  + Check to see if the utility successfully interprets and validates the supplied command-line inputs, such as the process name, maximum lifespan, and monitoring frequency.
* Test Case 2:
  + Make sure that missing or incorrect arguments are gently handled with the relevant error messages.
* Test Case 3:
  + Confirm that the utility correctly terminates the process by creating a dummy process that runs longer than the allowed lifespan.
  + Make sure there are no unforeseen errors or exceptions when the process is terminated.
* Test Case 4:
  + Verify that the utility logs the required data, including the specifics of the processes and the actions taken (such as process termination).
  + Verify the accuracy of the log entries by validating their content.

## 6. Test Environment Setup

1. Test environment includes Nuit framework, Visual Studio 2022 and necessary nugget packages required.
2. Following are the steps involved in creating test environment-
   1. Create Nunit test project using Visual Studio.
   2. Add reference of project needed to test.
   3. Write test cases necessary for the program.

## 7. Conclusion

1. I adhered to all coding best practices and standards, such as writing methods and invoking them when necessary, which resulted in code optimization and code efficiency maintenance for the process monitoring tool.
2. Included all test cases required for the utility to function properly.