QUALITY DESIGN DOCUMENT

1. Introduction and Project Overview

The System monitor is a central monitoring system built using the client-server model to collect system information from multiple nodes in a network.
The motivation behind collecting system information from nodes on a network comes from a need to allocate tasks and resources to different nodes on the network in an efficient manner. In this way, the server (administrative node) on the network can keep track of the system information of all nodes, ensure that no single node is overloaded and no node is idle/wasted.

Such a monitoring system is a key component in order to achieve increased efficiency, stability and profitability over the network.
One of the ways to achieve increased productivity is using a central monitoring system to gather data from all node periodically, conduct an analysis of the system information gathered and provide it to the user in a concise, consolidated form.

This analysis can assist the administration of the network in:

1. Finding different methods for enhancing a system's performance.
2. Improved decision-making to enhance resource optimization through the examination of system information.
3. Reviewing systems' data and identifying anomalies as soon as possible to prevent any further problems or complications.
4. Optimizing the overall functioning of the nodes and the network

The goal of this project is to develop a central monitoring system based on a client server architecture in which the server would collect system information periodically from the client nodes that connect to it over the TCP/IP network, and then analyse the data received from client machines.

2. Requirements

The requirements on the client end and server end are as follows:

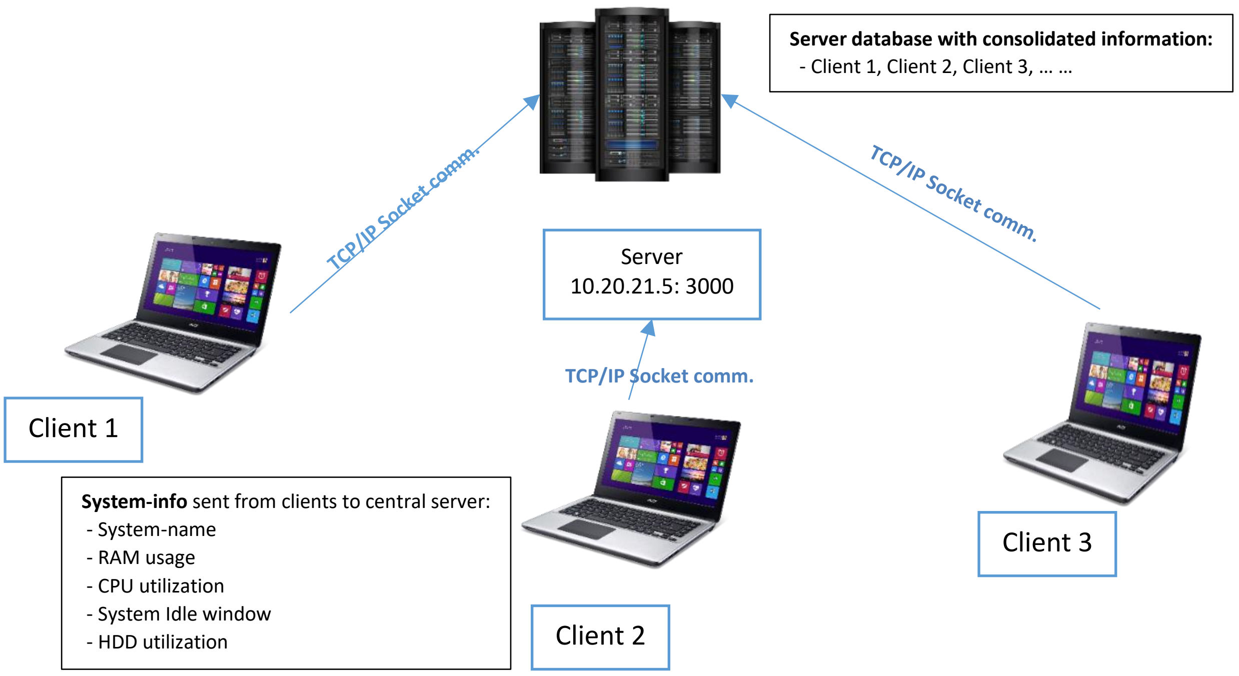
Client Side:

1. The client should be able to periodically fetch relevant data by interacting with the Windows system.
2. The data to be fetched from the system includes – System-name, RAM usage, CPU utilization, System Idle window, HDD utilization, Network statistics, etc.
3. After fetching this data, it should be able to send it using TCP/IP to a server system.
4. The data should be sent periodically to the server.

Server Side:

1. The server should be able to receive the data sent by the client periodically over the TCP/IP network.
2. It should then store the received data into a database.
3. Relevant data from the database should be fetched and displayed when the corresponding command is given.
4. The server should be able to interact with, i.e. receive and store data, from multiple clients.

A diagram representing the requirements mentioned is as below:



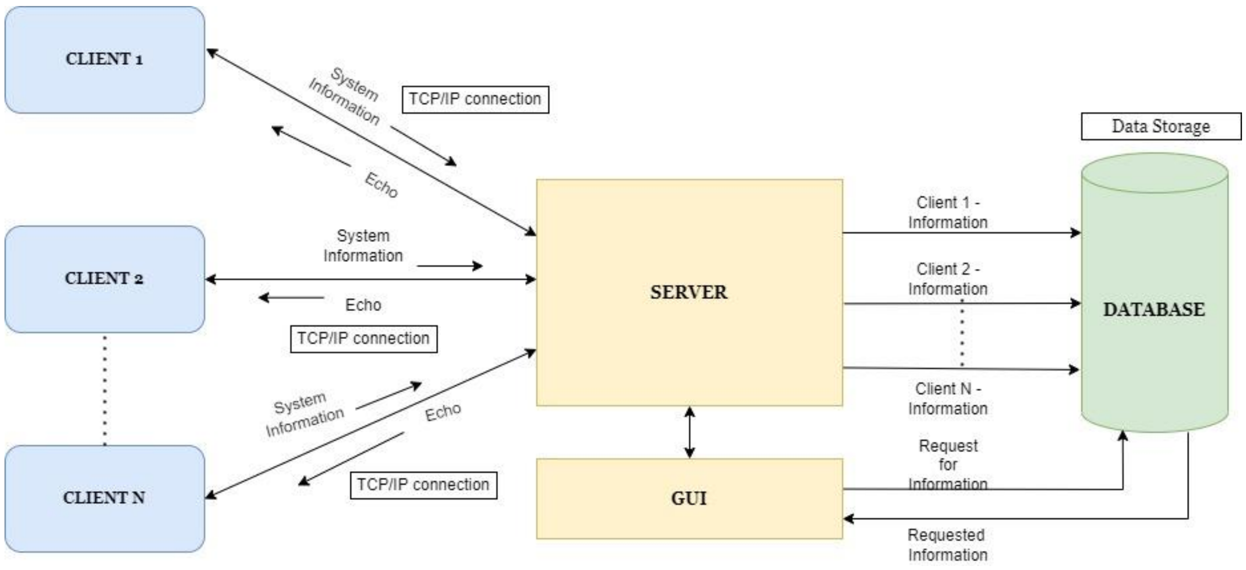
3. Implementation Details

A central monitoring system using the client-server architecture has been implemented, which gathers system data from numerous nodes throughout a network. The client periodically retrieves relevant data and transfers it across the TCP/IP network to a designated server. In order to provide a server with an update on the system's status, the following parameters must be retrieved and delivered by the client periodically:

1. System Name
2. RAM Usage
3. CPU Utilization
4. System Idle Window
5. HDD Utilization

These parameters are then sent from the server to the database on a regular basis by establishing the database connection. Furthermore, the GUI on the server side, which has been integrated with the database, assists the user in retrieving specific information about clients. This information is displayed in the form of charts, graphs and tables provided by the GUI. This helps the user gain insights and make sense of the data in the database.

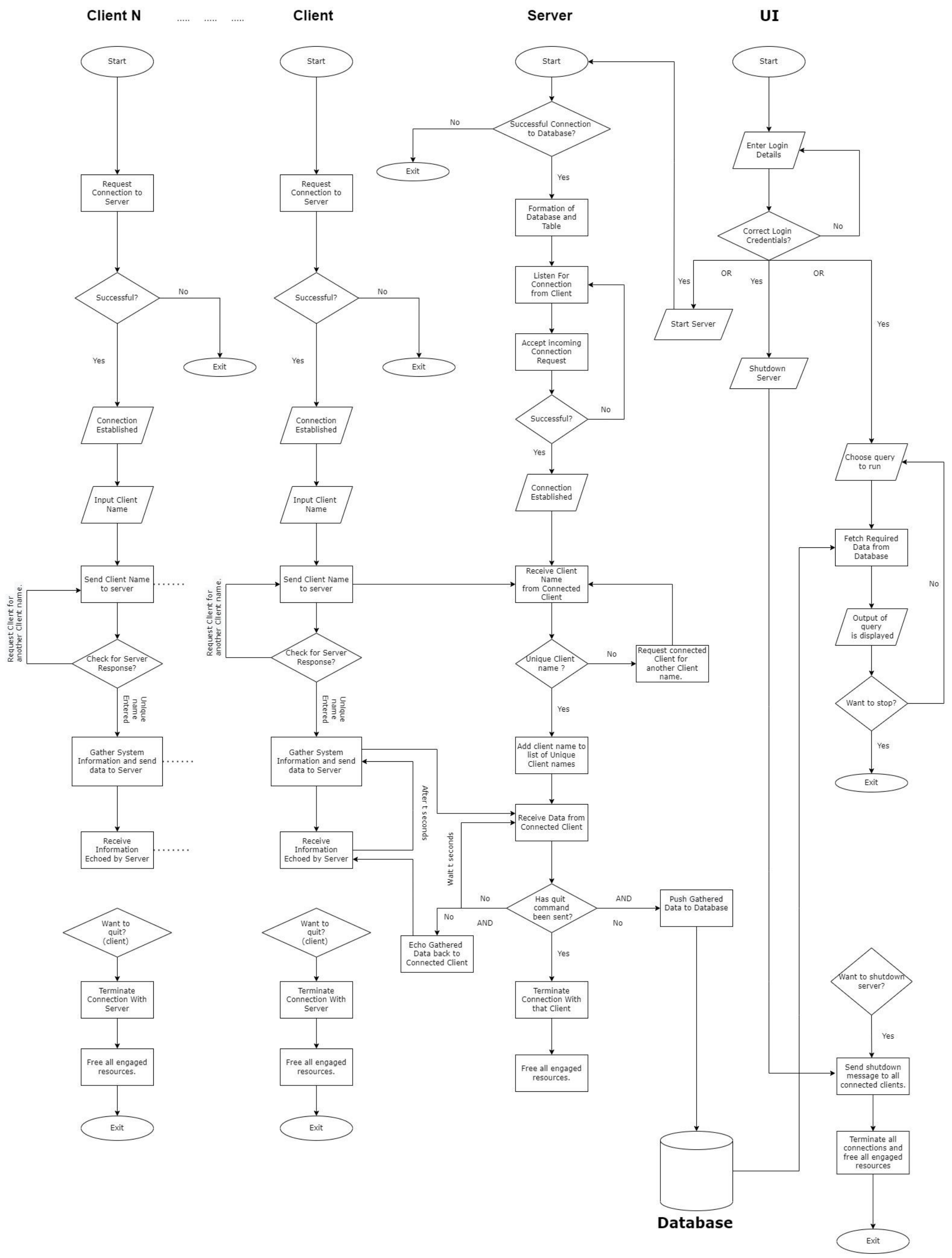
The software architecture used in the solution is as follows:



The architecture meets the basic requirements mentioned above for both the client and server.

As in the figure, the server receives data from multiple connected clients and echoes the message back to corresponding client. It also pushes client data into the database, which can then be extracted based on different criteria for analysis and other purposes using the UI.

A more detailed flow of implementation is shown in the flowchart below. It gives a step by step description of how the application works on the client and server end with UI.


Client N
....
Client
Server
UI
Start
Request Connection to Server
Successful?
No
Exit
Yes
Connection Established
Input Client Name
Send Client Name to server
Request Client for another Client name.
Check for Server Response?
Unique name Entered
Gather System Information and send data to Server
Receive Information Echoed by Server
Want to quit? (client)
Terminate Connection With Server
Free all engaged resources.
Exit
Start
Request Connection to Server
Successful?
No
Exit
Yes
Connection Established
Input Client Name
Send Client Name to server
Request Client for another Client name.
Check for Server Response?
Unique name Entered
Gather System Information and send data to Server
After t seconds
Receive Information Echoed by Server
Want to quit? (client)
Terminate Connection With Server
Free all engaged resources.
Exit
Start
Successful Connection to Database?
No
Exit
Yes
Formation of Database and Table
Listen For Connection from Client
Accept incoming Connection Request
Successful?
No
Yes
Connection Established
Receive Client Name from Connected Client
Unique Client name ?
No
Request connected Client for another Client name.
Yes
Add client name to list of Unique Client names
Receive Data from Connected Client
Wait t seconds
Has quit command been sent?
No
AND
No
AND
No
Echo Gathered Data back to Connected Client
Push Gathered Data to Database
Yes
Terminate Connection With that Client
Free all engaged resources.
Database
Start
Enter Login Details
Correct Login Credentials?
No
Yes
OR
Yes
OR
Yes
Start Server
Shutdown Server
Choose query to run
Fetch Required Data from Database
Output of query is displayed
Want to stop?
No
Yes
Exit
Want to shutdown server?
Yes
Send shutdown message to all connected clients.
Terminate all connections and free all engaged resources
Exit

This flowchart shows how the server, server side GUI, the clients, and the database interact with each other from beginning to end. A detailed flow of the way in which a connection is established between a single client and server, followed by their communication process and finally the way in which the connection termination occurs is shown, but it is to be noted that multiple clients can interact with the server in exactly the same manner. Further, the steps in implementing the storage of data and fetching it using the UI have also been displayed. This flowchart is meant to give the user a detailed understanding of how the application works.

4. Features

Basic Features:

<u>Client Features:</u>

1. On running the client.exe file, a TCP/IP connection with the server is established provided the server was listening for connections.
2. The client is capable of gathering system information including System Name, RAM usage details, CPU utilization details, HDD utilization details, and System Idle window.
3. The gathered information is structured into a json file by the client.
4. Then, the data is sent to the server with which the client is already connected.
5. Whenever data is sent to the server, the data received by the server is echoed back to the client and is displayed on the client side console.
6. The process of gathering system information and sending the data to the server is repeated periodically.

<u>Server Features:</u>

1. Once the server.exe file is run, it listens for connection requests from clients.
2. Once an incoming client connection request is detected, a connection to the client is established.
3. The server receives data from the client and echoes it back to that client.
4. The server is connected to a mySQL database to update and store information received from the clients, connection status, etc.
5. On receiving the data, the server pushes the received data to the database.
6. The server is capable of handling multiple clients at the same time.

Special Features:

<u>Client Features:</u>

1. Upon connection to server, the client must enter a ‘Client Name’. This is used to identify the particular client, and therefore must be unique.
2. This client name is sent to the server first, which checks if it is unique within the list of connected clients. The process of system information gathering and sending of data is initiated only once the server confirms that the Client name entered is unique within the list of connected clients. Otherwise, the client is requested to enter a name again. This process is repeated until the client enters a unique name.

3. The entire process of system information gathering and sending to the server runs periodically and has been automated. It starts once the client manually enters their client name and it is confirmed to be unique.
4. The client can disconnect from the server anytime by entering '\quit' into the console. This safely disconnects the client from the server and releases all resources to allocated to it. This command terminates the data sending periodic function too.
5. Exception handling has been implemented to handle abrupt/ unexpected cases such as server disconnection, client being unable to send or receive data, etc. Each exception displays an error message which is user friendly and easy to understand.

<u>Server Features:</u>

1. The server maintains a list of all connected clients, and does not allow two clients with the same client name to connect at the same time. If a new client sends in a client name that already exists in the connected-client list, the server requests the new incoming client to enter a different name.
2. The server displays messages to notify the user when something important happens. For example: Every time a new connection is added, or removed, a message is displayed. Every time data is sent by a connected client, a message confirming that the data has been received from so-and-so client is displayed on the console.
3. If a client gets disconnected, then all the resources related to that client are cleaned up.
4. The server can be shut down anytime by typing \quit into the console. This sends a shutdown message to all connected clients and then terminates the connection with them. It will then release all engaged resources.
5. Exception handling has been implemented to handle abrupt/ unexpected cases such as the server being unable to send or receive data, server being unable to connect to database, etc. Each exception displays an error message which is user friendly and easy to understand.

<u>GUI Features:</u>

A server side GUI has been implemented with the following features:

1. To get to the dashboard of UI, the user needs to authenticate themselves though a login form. Only users with correct username and password will be allowed to access the server GUI.
2. The GUI is connected directly to the database in which the server stores clients' information.
3. The dashboard has a variety of features including:
   3.1 A start server button which starts and activates the server.
   3.2 A stop server button which shuts the server down.
   3.3 A refresh button which updates the list of client names so it is dynamic and shows real time data.
4. The GUI allows the user to fetch data from the database by running any query present in the query drop-box in the dashboard. The following queries are available to run:
   4.1 Fetch all the data associated with a client based on Client Name
   4.2 Fetch the data of the 'n' latest entries of a particular client. The user can choose the value of 'n' to be anywhere between 1 and 20 here.
   4.3 Find the client which has maximum available RAM within the 'n' latest database entries. The user can choose the value of 'n' to be anywhere between 1 and 20 here.

   4.4 Find the client which has minimum available RAM within the ‘n’ latest database entries. The user can choose the value of ‘n’ to be anywhere between 1 and 20 here.
   4.5 Find the client with minimum CPU usage within the ‘n’ latest database entries. The user can choose the value of ‘n’ to be anywhere between 1 and 20 here.
   4.6 Find the client with maximum CPU usage within the ‘n’ latest database entries. The user can choose the value of ‘n’ to be anywhere between 1 and 20 here.
   4.7 Find the client which has maximum available disk space within the ‘n’ latest entries. The user can choose the value of ‘n’ to be anywhere between 1 and 20 here.
   4.8 Find the client which has minimum available disk space within the ‘n’ latest entries. The user can choose the value of ‘n’ to be anywhere between 1 and 20 here.

5. The GUI displays the results of these queries using tables, graphs, pie-charts and other such visual tools which helps the user to analyse the data fetched and draw meaningful conclusions from it.
6. Every button whether in the login form or dashboard or the client info form has an exception handling functionality to handle exceptional situations. Each exception displays an error message which is user friendly and easy to understand.
7. The GUI functionality is independent of the server. That is, the GUI can perform all above functions even when the server is not running and adding data to the database.
8. If the GUI is used when server is not currently set to running, then it fetches data from the last saved version of the database. That is, if the server pushed data into the database at some previous time and this was saved inside, this saved data will be used in answering queries on the UI until more data is added inside.
9. Proper termination of connection with database is ensured with each query so that engaged resources are released making the UI less bulky.

All of these features work together to provide a smooth user experience for both the client and server, and help the server achieve the goal of analysing the system information collected and interpreting it to make optimal decisions for resource and task allocation and other such purposes.