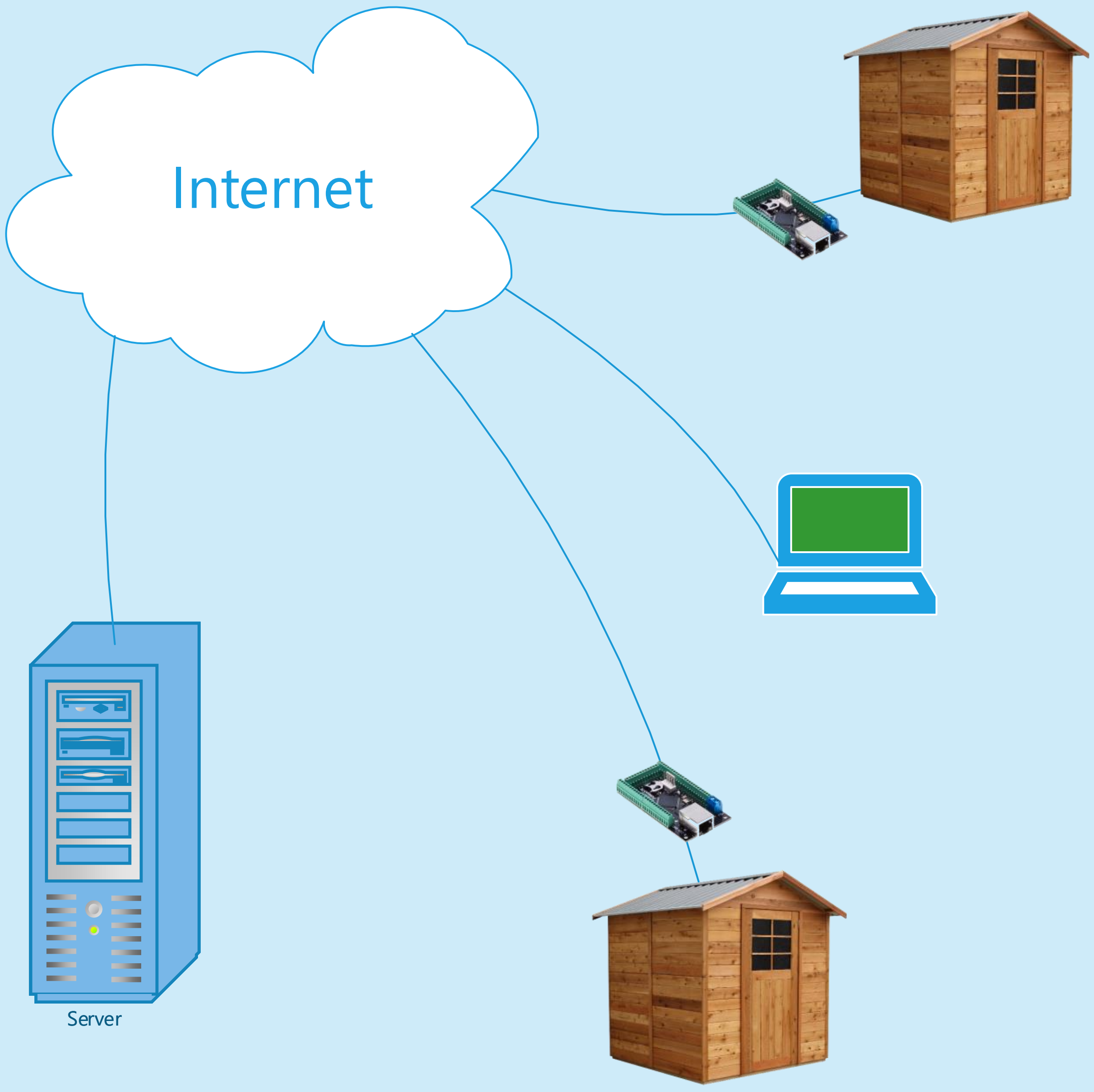
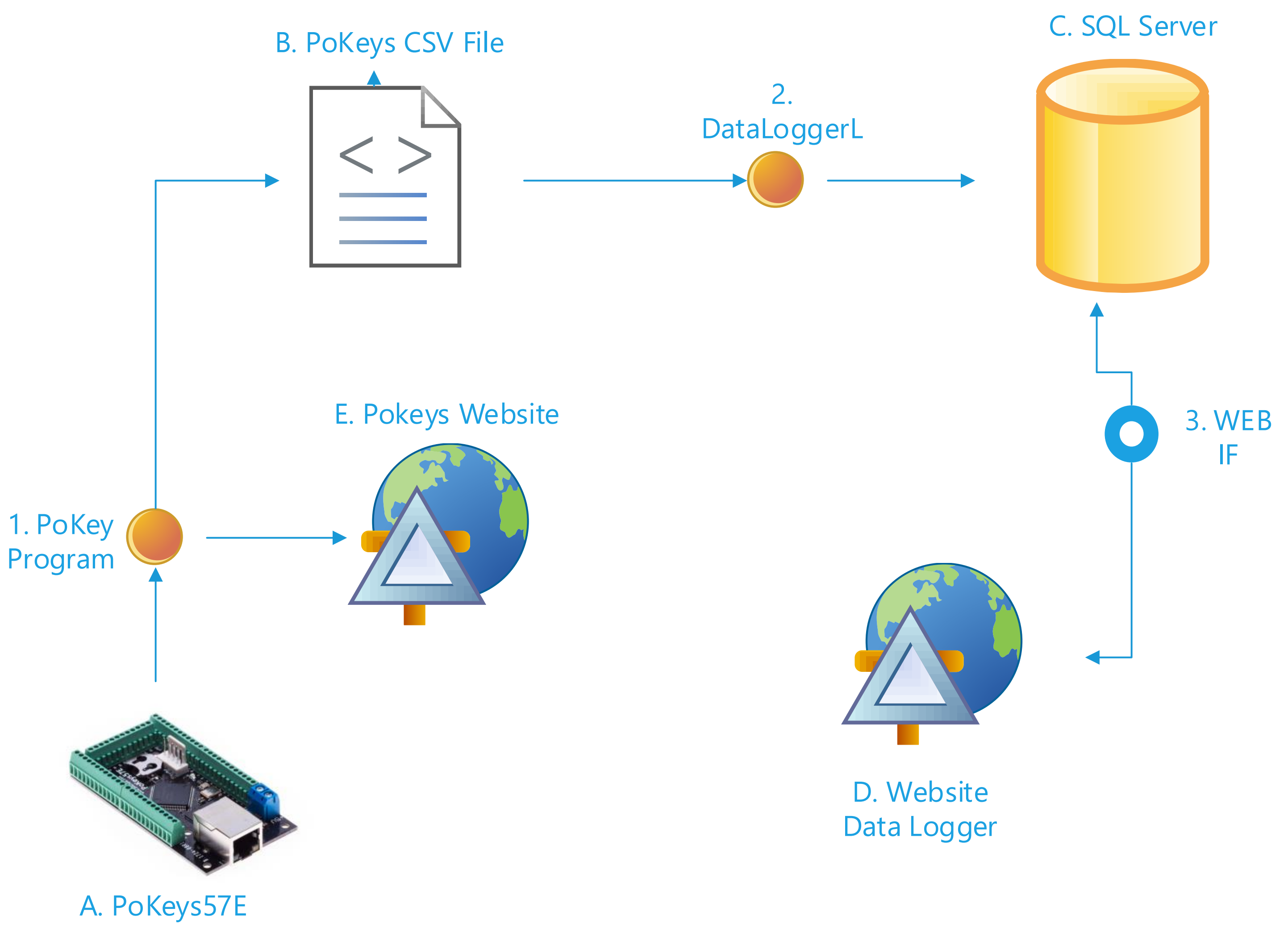


DataLoggerL

Logger For PoKeys57E
Program Documentation


B. PoKeys CSV File
C. SQL Server
2. DataLoggerL
E. Pokeys Website
3. WEB IF
1. PoKey Program
D. Website Data Logger
A. PoKeys57E

A. Pokeys57E

PoKeys57E is a Small IO card built into a little red box which you can connect sensors to and then configure via PoKeys Configuration program.

https://www.poscope.com/

B.PoKeys CSV-File

PoKeys57E sends it's values in a CSV file with a HTTP put request to DataLoggerL
Ex. Parameter,value
Place,Bromarf/n
Param,22.3/n

C. SQL Server

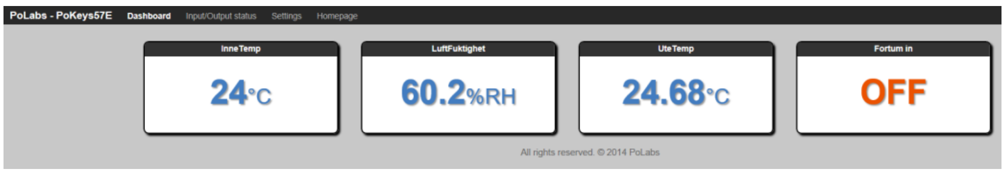
The database which all the values are stored in Can be any database, in our case it is SQL Server 2012.
The database is named DataLoggerL and has 3 separate tables, Parameters, Places and Values.
Parameter contains all the accepted parameters.
Places contains all the accepted locations
Values contains all the values which are sent from the PoKeys57E.

D. DataLogger Website

When using the website you will be able to see graphs and tables of the values.

E. PoKeys57E Website

You can view what values the sensors are sending by using the PoKeys default website.
You can also make changes in the PoKeys Configuration Program using the website.

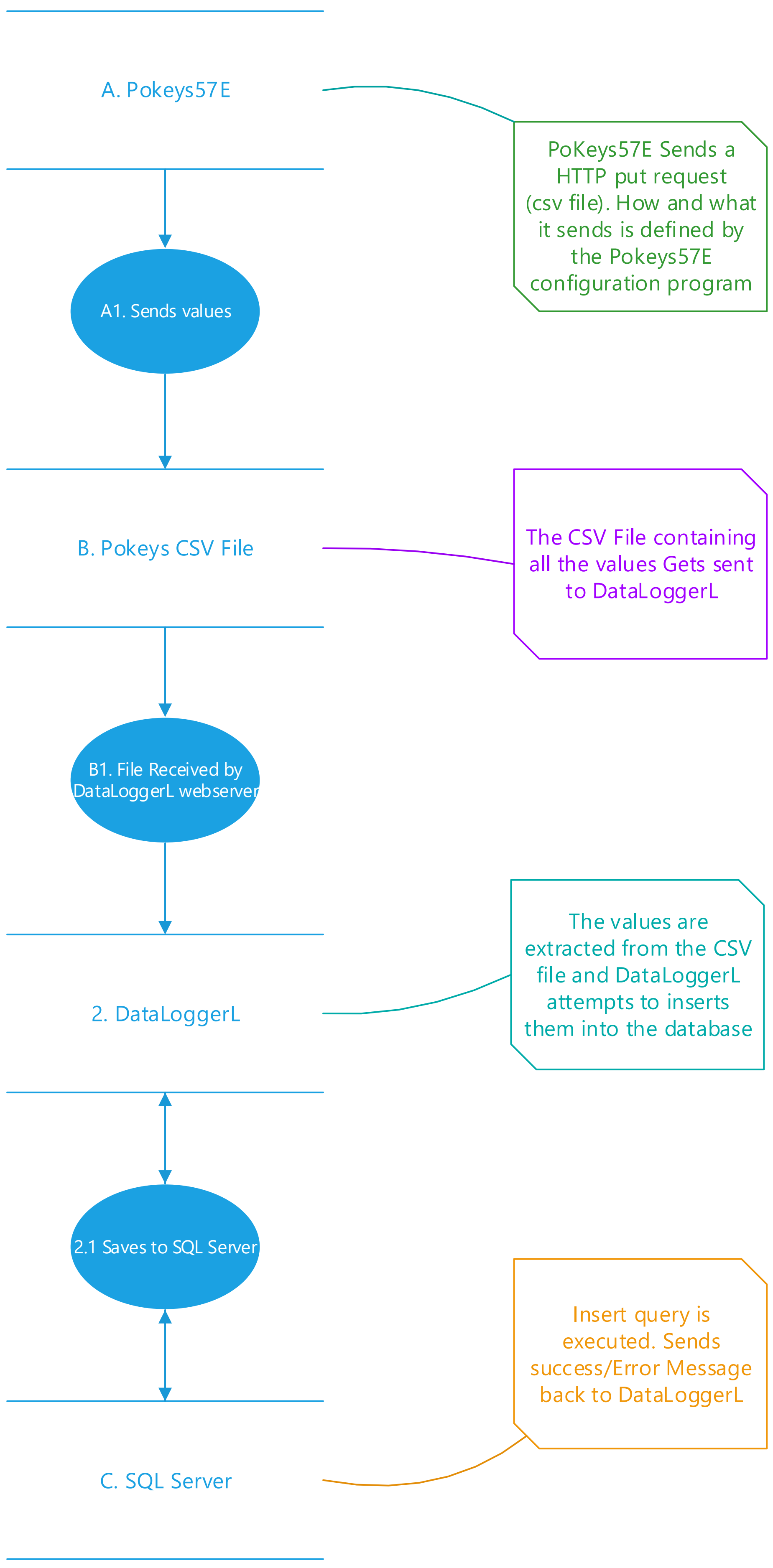


1.PoKeys Program

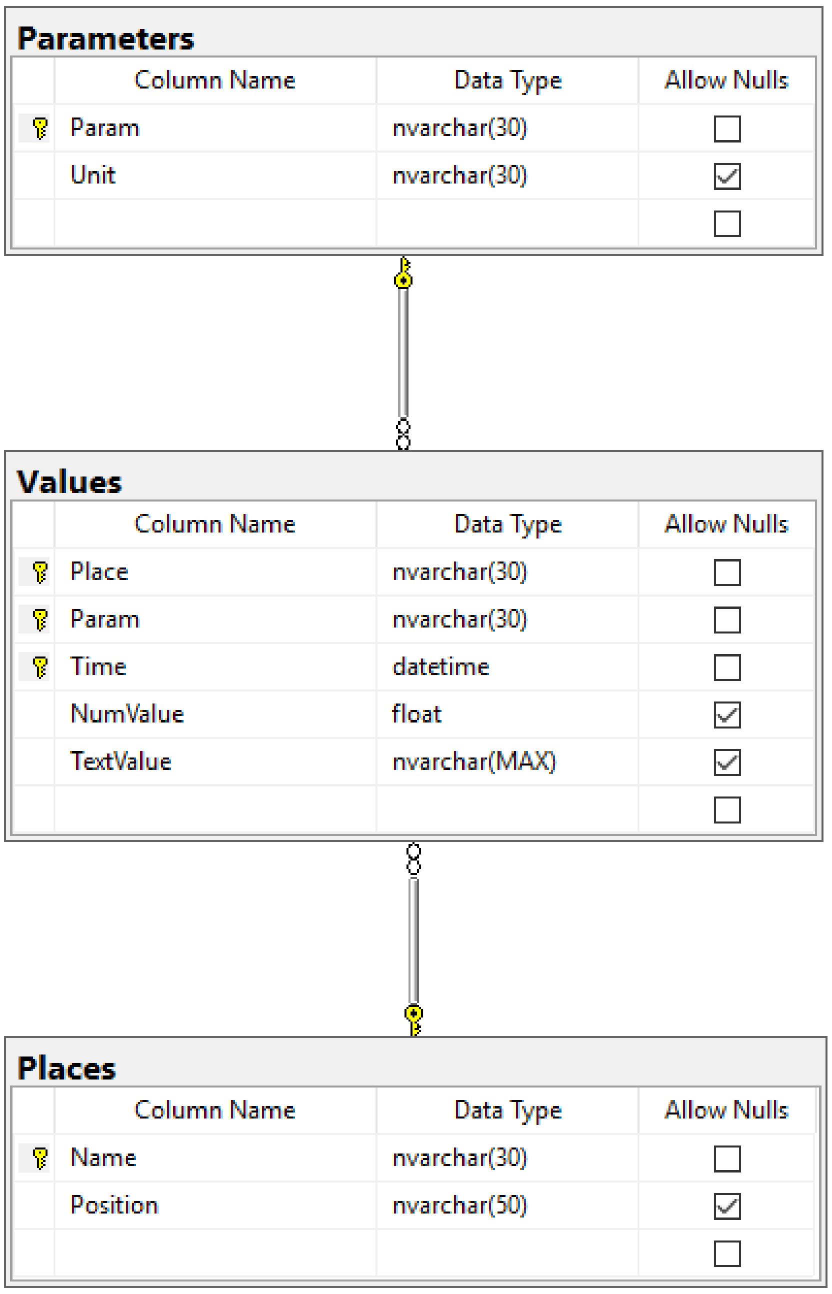
You can define how and where it sends its values.
You also configure the sensors from the program.
When hosting DataLoggerL on a webserver you need to Define the host and port before sending values.

2. DataLoggerL

DataLoggerL receives the CSV file from the PoKeys57E and then tries to insert it into the database, if something goes wrong it inserts everything including the error message as an "Error" into the database. DataLoggerL is hosted on a webserver which will need to have a reverse proxy for DataLoggerL to be able to receive the requests.


A. Pokeys57E
A1. Sends values
B. Pokeys CSV File
B1. File Received by DataLoggerL webserver
2. DataLoggerL
2.1 Saves to SQL Server
C. SQL Server
PoKeys57E Sends a HTTP put request (csv file). How and what it sends is defined by the Pokeys57E configuration program
The CSV File containing all the values Gets sent to DataLoggerL
The values are extracted from the CSV file and DataLoggerL attempts to inserts them into the database
Insert query is executed. Sends success/Error Message back to DataLoggerL

Databasen DataLoggerL



Parameters

	Column Name	Data Type	Allow Nulls
🔑	Param	nvarchar(30)	☐
	Unit	nvarchar(30)	☑
			☐

Values

	Column Name	Data Type	Allow Nulls
🔑	Place	nvarchar(30)	☐
🔑	Param	nvarchar(30)	☐
🔑	Time	datetime	☐
	NumValue	float	☑
	TextValue	nvarchar(MAX)	☑
			☐

Places

	Column Name	Data Type	Allow Nulls
🔑	Name	nvarchar(30)	☐
	Position	nvarchar(50)	☑
			☐

Program Tests

Action 1: Using DataLoggerL with faulty connection string.
Result 1: DataLoggerL tests database connection, upon noticing that the connection doesn't work it sets the property active to false. When DataLoggerL receives a request from PoKeys57E it will not try to insert it into the database.

Action 2: Inserting a parameter with a Textvalue instead of a Numbervalue.
Result 2: PoKeys57E is not able to send Textvalues, however DataLoggerL is able to identify and insert textvalues.

Action 3: Inserting with faulty parameters.
Result 3: When attempting to insert something with an faulty parameter it will "catch" the exception and then attempt again by replacing previous parameters with "error" and inserting the previous values including the exception error into "Textvalue". If it fails a second time it will write everything including the first and the second exception error into the console.

Action 4: Sending CSV file with an uneven amount of parameters/values.
Result 4: OutOfRangeException. When trying to access a paramter that doesn't exist it will throw an error.
Fix 4: DataLoggerL will not try to insert anything if it receives and HTTP request with an uneven amount of parameters/values

Action 5: Running the program on a port that's blocked/in use
Result 5: WebListenerException: Prefix is already registered. Only way to fix is to run on a port that isn't in use.

Action 6: Configuring PoKeys57E to send HTTP Request to a faulty destination
Result 6: Nothing happens, only way to prevent this is to not do it.

Action 7: Sending empty CSV file
Result 7: PoKeys57E is not able to send empty CSV files. Was not tested.

Action 8: Launching DataLoggerL without port parameter
Result 8: DataLoggerL crashes
Fix 8: DataLoggerL will now default to port 5000 when not receiving a port as a parameter.

Action 9: Running DataLoggerL without Administrator rights
Result 9: WebListenerException: access Denied. DataLoggerL must be run as an Administrator

Action 10: Sending CSV File with negative values
Result 10: DataLoggerL inserts the negative values normally into the Database

Technical Documentation

Launching DataLoggerL:

To launch DataLoggerL you need to run the .exe file in administrator mode, doing it this way will make dataLoggerL listen to the default prefix which is "http://*:5000", if you want to specify URL and port to listen to you will need to run DataLoggerL from the Command Prompt (note that you need to run the command prompt in administrator mode).
To launch DataLoggerL from the Command Prompt you need to navigate to the directory that the DataLoggerL.exe file is located in and then typing "dataloggerl.exe" (name of .exe file) into the Command Prompt, run DataLoggerL with a custom parameter by typing in "dataloggerl.exe parameter" where "parameter" is the URL and port you want to listen to
ex. "dataloggerl.exe http://logger.guje.fi:5000".

Changing Connection String:

The connection string is stored in the "appsettings.json" file. To change it simply locate the file and open it in a text editing program and change the text inside the quotation marks. Next time you launch DataLoggerL it will use the new connection string. If DataLoggerL cant connect to the database using the connection string it will write out the error message "No Database Connection", if DataLoggerL receives an request when it cant connect to a database it will not try to insert anything.

PoKeys57E Configuration:

To learn how to configure PoKeys57E to send HTTP Put requests properly to DataLoggerL read the official PoKeys57E User Manual.
Note that the PoKeys57E will need to send the request to "URL/log/LOCATION" where URL is the address and LOCATION the identifier where the Pokeys57E is, which will be inserted into the database. Ex. "example.com:5000/log/ExampleTown".

Database Tables:

The database tables needs to be set up identically as shown on the database page of this documentation or else DataLoggerL wont be able to insert any values. The database name can be anything and DataLoggerL chooses database depending on connection string.

MiddlewareDataLogger.cs 1/3

```
using DataLoggerL;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

public class MiddlewareDataLoggerL
{
    private readonly RequestDelegate _next;

    public MiddlewareDataLoggerL(RequestDelegate next)
    {
        //Constructor variables gets set once when running DataLoggerL
        active = false;
        Active = true;
        _next = next;
    }

    private bool active;
    public bool Active
    {
        //Tests Database Connection by usin TestDbConnection. Sets active depending on result
        get
        {
            return active;
        }

        set
        {
            if (value)
            {
                try
                {
                    TestDbConnection();
                    System.Console.WriteLine("Connected to Database Succesfully");
                    active = true;
                }
                catch
                {
                    System.Console.WriteLine("Could not connect to Database");
                    active = false;
                }

            }
            else
            { active = value; }

        }
    }

    private void TestDbConnection()
    {
        //Tests Database Connection. Is used in Active
        var conn = new SqlConnection(Startup.Configuration["ConnectionString"]);
        string sql = @"SELECT * FROM [dbo].[Values]";
        SqlCommand cmd = new SqlCommand(sql, conn);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
    }
```

MiddlewareDataLogger.cs 2/3

```
private bool SaveValue(string Place, string Param, dynamic NumValue, dynamic TextValue)
    {
        //Recursive Function. returns true or false depending if it succeeds
        bool retval = false;
        if (Active) //If Database is connected this will run
        {
            var conn = new SqlConnection(Startup.Configuration["ConnectionString"]);
            string sql = @"INSERT INTO [dbo].[Values] ([Place], [Param], [NumValue], [TextValue]) VALUES 
(@Place, @Param, @NumValue, @TextValue)";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add("@Place", SqlDbType.NVarChar).Value = Place;
            cmd.Parameters.Add("@Param", SqlDbType.NVarChar).Value = Param;
            cmd.Parameters.Add("@NumValue", SqlDbType.Float).Value = NumValue;
            cmd.Parameters.Add("@TextValue", SqlDbType.NVarChar).Value = TextValue;
            try
            {
                conn.Open();
                cmd.ExecuteNonQuery();
                conn.Close();
                retval = true;
            }
            catch (Exception e) //If it cant insert original values it will try to insert error values
            {
                if (Param != "Error") {
                    if(SaveValue("Error", "Error", DBNull.Value,Place + " " + Param + " " + NumValue + " 
" + TextValue + " Error: " + e))
                    {
                        retval = true;
                    }
                }
                else // If it cant insert error values it will write out Error and error text in console
                {
                    System.Console.WriteLine("ERROR~");
                    System.Console.WriteLine(e);
                    System.Console.WriteLine("ERROR");
                    System.Console.WriteLine(TextValue);
                    System.Console.WriteLine("~ERROR");
                }
            }
        }
        else
        {
            System.Console.WriteLine("No Database connection");
        }
        return retval;
    }

    public async Task Invoke(HttpContext context)
    {
        //Gets Values from Stream Sent From PoKeys57E
        var bodyReader = new StreamReader(context.Request.Body);

        var path = context.Request.Path.Value; //Gets whole link
        var location = path.Split('/').Last(); //Gets location from link. If link is properly made ex. 
logger.guje.fi/log/"location"

        if (path.Contains("log"))
        {
            string body = await bodyReader.ReadToEndAsync();
            List<string> Values = body.Split('\n', ',').ToList<string>();
            //Splits all values and makes array
```

MiddlewareDataLogger.cs 3/3

```
if (Values.Count % 2 == 0) { //Checks if array is dividable by 2
                for (int i = 0; (Values.Count) > i; i += 2) //Runs for loop once for every two values in 
array
                {
                    double NumValue;
                    bool isNumeric = double.TryParse(Values[i + 1], NumberStyles.Any, 
CultureInfo.InvariantCulture, out NumValue);
                    //Check if value is a number

                    string param = Values[i];
                    if (isNumeric)
                    {
                        var TextValue = DBNull.Value; // Sets Textvalue to null if "values[i + 2]" was a 
number
                        if(SaveValue(location, param, NumValue, TextValue))
                        {
                            context.Response.StatusCode = 201; //If SaveValue returns true respond with 
status code 201
                        }
                        else
                        {
                            context.Response.StatusCode = 400; //If SaveValue returns false respond with 
status code 400
                        }
                    }
                    else
                    {
                        var NumberValue = DBNull.Value; // Sets NumberValue to null if "values[i + 2]" 
was a string
                        string TextValue = Values[i + 1];
                        if(SaveValue(location, param, NumberValue, TextValue))
                        {
                            context.Response.StatusCode = 201; //If SaveValue returns true respond with 
status code 201
                        }
                        else
                        {
                            context.Response.StatusCode = 400; //If SaveValue returns false respond with 
status code 400
                        }
                    }
                }
            }
            else
            {
                System.Console.WriteLine("ERROR~");
                System.Console.WriteLine("Uneven amount of Values/Parameters");
                System.Console.WriteLine("~ERROR");
                context.Response.StatusCode = 400;
                //Writes Error to console and responds with status code 400
            }
        }
    }
}
```

Program.cs Code

```
using System.IO;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Builder;
using Microsoft.Net.Http.Server;
using System.Linq;

namespace DataLoggerL
{
    public class Program
    {
        public static void Main(string[] args)
        {
            //If an argument is set. Set variable url to argument
            string url = "http://*:5000";
            int arguments = args.Count();
            if(arguments > 0)
            {
                url = args[0];
            }

            var host = new WebHostBuilder()
                .UseUrls(url) //Use variable url to choose prefix to listen to
                .UseContentRoot(Directory.GetCurrentDirectory())
                .UseWebListener(options =>
                {
                    options.ListenerSettings.Authentication.Schemes = AuthenticationSchemes.None;
                    options.ListenerSettings.Authentication.AllowAnonymous = true;
                })
                .UseStartup<Startup>()
                .Build();

            host.Run();
        }
    }
}
```