## Nayax / VendSys .NET Technical Project

**Contact**: Jon Hunt or John Davies ([jon.hunt@vendsys.com](mailto:jon.hunt@vendsys.com) or [john.davies@vendsys.com](mailto:john.davies@vendsys.com))

**Goal**: To demonstrate proficiency in C# .NET, ASP.NET Core Minimal APIs and Microsoft SQL.

**Background**: In our industry, unattended retail devices like vending machines output the machine data in a format called DEX. The NAMA DEX specification is attached to this project (as a PDF) and contains the Appendix references for fields in the DEX report. Refer to the specification for field-level information that we will be extracting.

Also attached are two text files which are DEX reports.

You will note that the files each have a single set of Header fields (”Dex Meters”) \_eg the “ID “and “VA” segments. These are followed by a variable length list of “DEX Lane Meter” fields – the “PA” segments. There are many more segments, but you can ignore them.

**Requirements**:

1. Build a simple Web page with 2 buttons that trigger the sending of a DEX file to a web service. You can use any stack for the UI you wish. Hard code 2 string fields to hold the text of the two supplied DEX reports:
   1. Button A to send one report (with parameter Machine A)
   2. Button B to send the other report (with parameter Machine B)
2. Build an ASP.NET Core 9 Minimal API with a single endpoint: POST /vdi-dex. This endpoint should accept a DEX file in the HTTP request body.
   1. The API must implement HTTP Basic Authorization (using the HTTP header Authorization) <https://en.wikipedia.org/wiki/Basic_access_authentication#:~:text=In%20basic%20HTTP%20authentication%2C%20a,HTTP%201.0%20specification%20in%201996>.
   2. For this API, the only credentials that should be accepted are username vendsys, password NFsZGmHAGWJSZ#RuvdiV
   3. You do not need to implement a Users table, for simplicity the credentials can be stored in the appsettings.json file.
3. Create a SQL Server LocalDb (or SQL Express Db) containing 2 tables to store this data from the DEX file. When submitting your project solution, include a .bak backup of the SQL database. The only fields we want to store are:
   1. **DEXMeter** 
      1. Machine (either “A” or “B” from the API request)
      2. DEXDateTime (will be unique for each machine)
      3. MachineSerialNumber (from ID101 field, Appendix A page 30)
      4. ValueOfPaidVends (from VA101 field, Appendix A page 45)
   2. **DEXLaneMeter**
      1. ProductIdentifier (from PA101 field, Appendex A page 36)
      2. Price (from PA102 field)
      3. NumberOfVends (from PA201 field)
      4. ValueOfPaidSales (from PA202 field)
   3. Construct the necessary primary keys and necessary foreign key to link the tables. Use appropriate data types based on the DEX specification
4. Use one or two stored procedures to save the data to the tables. Either:
   1. A single stored procedure called SaveDex which takes the entire DEX file as a varchar and is parsed within T-SQL, and then updates,

Or

* 1. parse the DEX file in the API in C# and call 2 stored procedures for simplicity ie SaveDEXMeter and SaveDEXLaneMeter with appropriate parameters.

When it runs you should be able create successive entries into the tables.

You can elaborate as you wish e.g. another button with a procedure to clear the tables, but is not required.

Code should be clean, easy to read and well commented. The API should be fast and support concurrent requests.

Estimated time to complete this project is 2-4 hours.

Feel free to reach out to Jon Hunt if you have any questions at all, or would like clarification on any part of the assignment.