**📌 Initiative 1: DLL Development for Nitric Acid Plant Simulation (Proof of Concept)**

This PFD shows a typical medium-pressure nitric acid process. Ammonia is mixed with air and oxidized over a platinum catalyst to form NO, which is further oxidized to NO₂ and then absorbed in water to form nitric acid. Heat recovery and tail gas treatment are integrated for energy efficiency and environmental control.  
 **Objective**:  
To enhance the simulation efficiency and accessibility of a Fortran-based code (developed by Weatherly) used in the Nitric Acid plant model by converting it into a DLL (Dynamic Link Library) for integration with Aspen HYSYS.

**Team Involved**:

* **Shubham** – Code replication and DLL development
* **Sunil Sir** – Integration with Aspen HYSYS via VBA scripting
* **Digpal Sir** – Validation using MATLAB

**Problem Statement**:  
The existing Fortran-based model was:

* Difficult to interpret
* Time-consuming to execute
* Not user-friendly for integration with other tools

**Solution Approach**:

1. **Initial Exploration**:  
   Evaluated using Python to convert the Fortran code into a DLL but encountered compatibility issues that halted progress.
2. **Technology Shift**:  
   Chose **C#** for its robust support for DLL creation and .NET compatibility. I upskilled by completing a **Microsoft C# certification**, enabling me to effectively replicate the original Fortran code in C#.
3. **Validation**:  
   Digpal Sir cross-verified the output by running equivalent simulations in MATLAB to ensure the logic and results were accurate.
4. **Integration with HYSYS**:  
   Sunil Sir utilized his expertise in VBA to successfully integrate the generated DLL as a **User-Defined Function (UDF)** in Aspen HYSYS.

**⚗️ Reactions Involved in Nitric Acid Production**

**🧪 1. Reactor**

**Reaction**:  
**4 NH₃ (g) + 5 O₂ (g) → 4 NO (g) + 6 H₂O (g)**  
**Conditions**:

* Catalyst: *Platinum-Rhodium (Pt/Rh)*
* Temperature: *≈ 910°C*
* Reaction Type: *Exothermic (−ΔH)*

**🔁 2. Pipe**

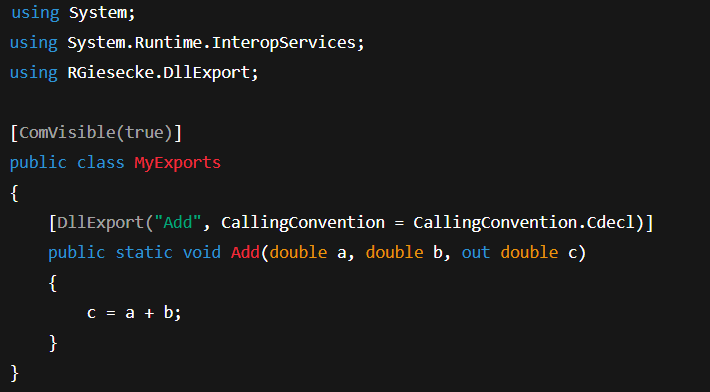
**Reaction**:  
**2 NO (g) + O₂ (g) ⇌ 2 NO₂ (g)**  
**Followed by Equilibrium**:  
**2 NO₂ (g) ⇌ N₂O₄ (g)**  
**Reaction Type**: *Exothermic (−ΔH)*

**💧 3. Absorber Column**

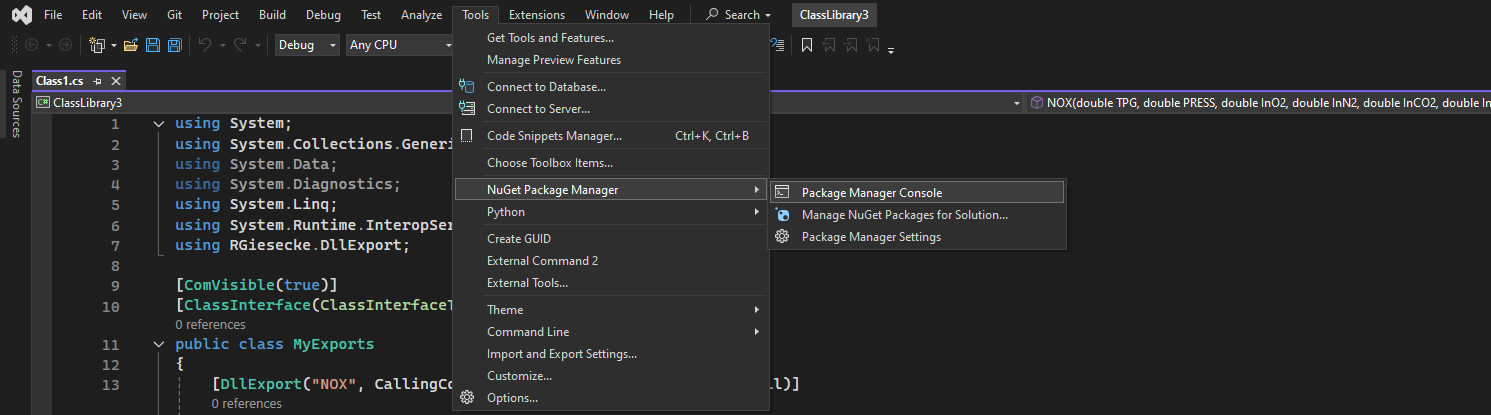
**Reaction**:  
**3 NO₂ (g) + H₂O (l) → 2 HNO₃ (aq) + NO (g)**

Alternate: **NO₂ + H₂O → HNO₃ + NO**  
**Conditions**:

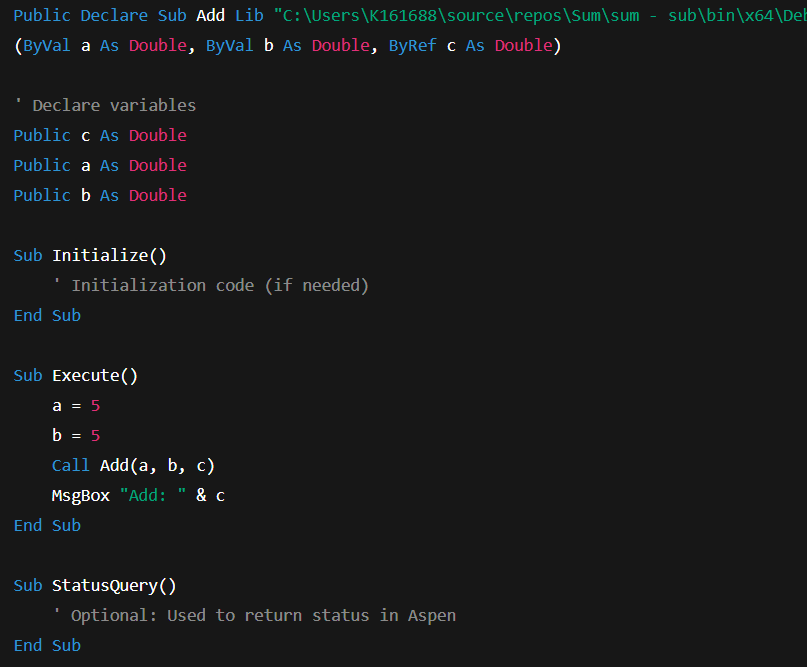
* Temperature: *≈ 50°C*
* Reaction Type: *Exothermic (−ΔH)*

To start how we can build an DLL file let’s take a small example for addition of two number where we will build an dll file and then integrate it with Aspen Hysys and see result in aspen  
  
🧩 **Part 1: C# Code – Creating the DLL**  


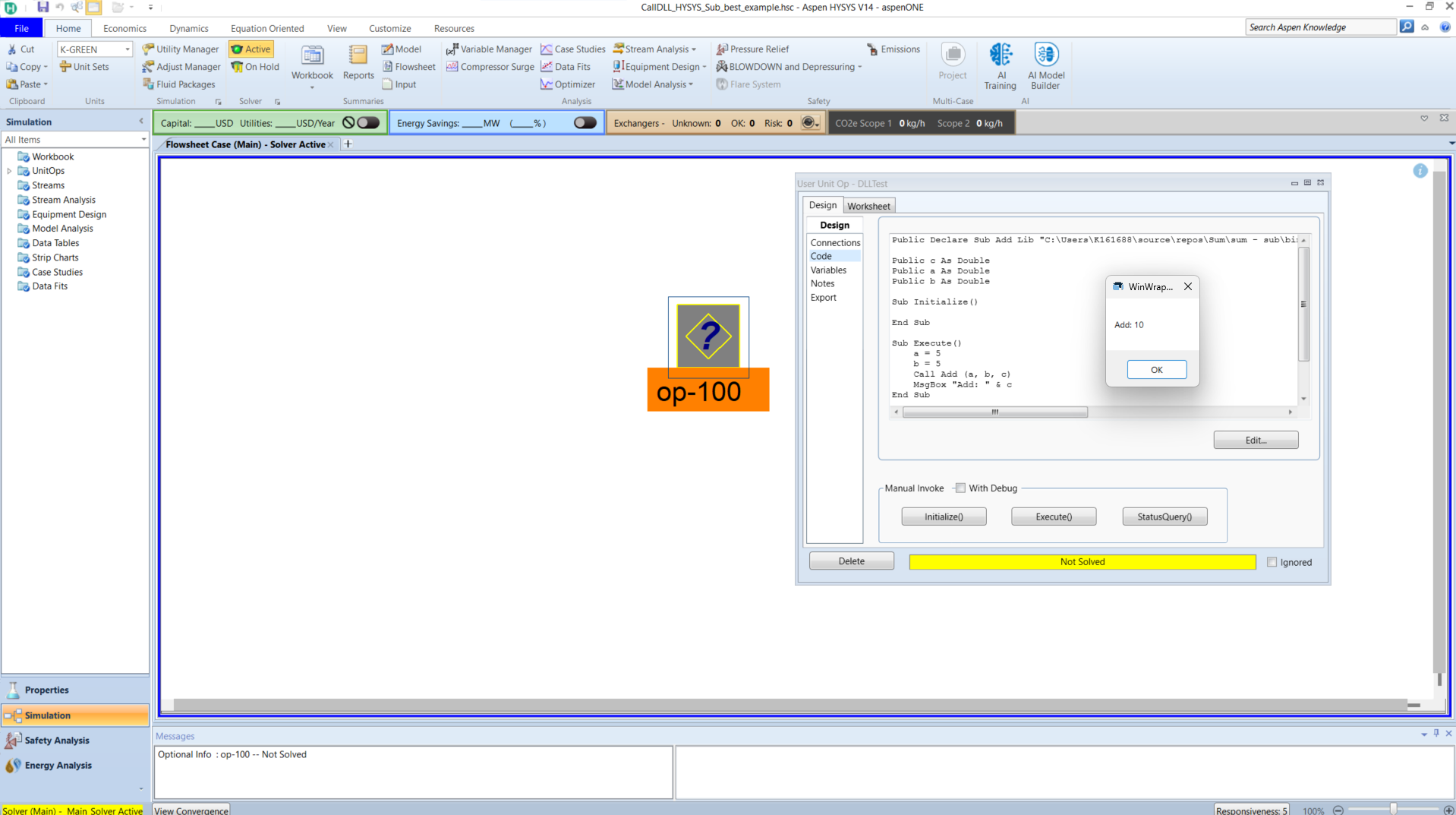
* **Install this before execution - Install-Package UnmanagedExports -Version 1.2.7**

****

| **Line** | **Meaning** |
| --- | --- |
| using System; | Imports basic C# functionality. |
| using System.Runtime.InteropServices; | Needed to expose methods to unmanaged code like VBA. |
| using RGiesecke.DllExport; | This library allows C# functions to be exported DLLs. |
| [ComVisible(true)] | Makes the class visible to COM (so VBA and HYSYS can access it). |
| public class MyExports | Defines a class where exported methods are written. |
| [DllExport("Add", CallingConvention = CallingConvention.Cdecl)] | Exports the function Add so it can be used outside the .NET runtime. |
| public static void Add(...) | A method that takes two doubles (a and b) and returns their sum in c. |

🧩 **Part 2: VBA Code – Integrating with Aspen HYSYS**

| **Section** | **Function** |
| --- | --- |
| Declare Sub Add ... | Declares the DLL function with full path so VBA can call it. |
| Public ... | Declares variables used across subroutines. |
| Sub Initialize() | Typically used by Aspen to initialize values – not needed here. |
| Sub Execute() | Main routine Aspen will call when the UDF block runs. It sends two values, calls the DLL function, and shows the result. |
| Sub StatusQuery() | Optional – used for error/status reporting in Aspen HYSYS. |

Here is the result..  


**Outcome**:  
The PoC was successfully demonstrated to **Eric, Brian** and **Satish Sir**, who appreciated the innovation. However, full-scale implementation across the Nitric Acid plant is pending due to funding constraints.  
  
I have installed Visual Studio to perform DLL using C# language and Aspen Hysys for user defined model.

**📌 Initiative 2: Habilis Project – LLM-Based Serverless Chatbot Development**

**Objective**:  
To develop a scalable, serverless chatbot solution powered by Large Language Models (LLMs) for document querying and contextual Q&A, under the *Habilis* initiative.

**Team Members**:

* Gianluigi (Lead)
* Anish (Cloud Integration)
* Sri Vidya
* Prasanth
* Shubham

**My Key Responsibilities**:

**1️⃣ Serverless Architecture Implementation**

* Designed and implemented a **serverless framework** using **Amazon Web Services (AWS)**.
* Utilized **Amazon Bedrock** to access foundation models like **Anthropic Claude, Amazon Titan, and others**.
* Learned Bedrock integration by referring to **official AWS documentation** due to limited external resources.
* Developed code to:
  + Integrate and invoke foundation models
  + Allow custom **document uploads**
  + Enable **intelligent querying** using the LLMs

⚙️ *Support from Anish was instrumental during the integration phase.*

**2️⃣ Testing & Evaluation of Chatbot**

* Conducted **end-to-end testing** of the developed solution.
* Uploaded multiple internal documents to evaluate performance, accuracy, and model understanding.
* Benchmarked the outputs against existing chatbot solutions.
* Prepared and submitted a **detailed test report** highlighting improvements, gaps, and future enhancements.

**Outcome**:  
The project progressed well in the development phase. The serverless infrastructure and Bedrock integration laid a strong foundation for enterprise-wide deployment of Habilis.  
  
**🔹 Introduction**

**I’ll give you a quick overview of the Habilis project — a digital tool we built in the INSITE team to make our work with documents faster and easier.**

**🔹 Background of the Project**

**Habilis was created to help us deal with large amounts of technical documents, reports, and standards more efficiently.**

**The idea was simple — build an assistant that can understand our questions and quickly give us the right answers from these documents using AI.**

**As part of the digital transformation journey, we saw the need for one platform where engineers can chat with documents like PDFs, ask technical questions, and get accurate results — without spending time searching manually.**

**✅ Steps to Use the Habilis Platform**

**Login Credentials**

* **Username**: shubham.kumar@kbr.com
* **Password**: shubham24

📌 For any login or credential-related issues, please contact **Gianluigi**.

**🔐 1. Login to the Platform**

* Visit the **Habilis platform URL** [HABILIS](http://54.147.79.5:3001/)
* Enter your credentials and log in.

**🗂️ 2. Access or Create a Project**

* After logging in, you will land on the **dashboard**.
* You have two options:
  + **Use an existing project**: Click on the **“Action”** button next to the desired project.
  + **Create a new project**: Click the **“Create Project”** button.

**📝 3. Creating a New Project**

* Enter a **Project Name**.
* **Upload** the document(s) you want the LLM to process.
* Click **“Create”** to initialize the project.

**⚙️ 4. Process the Document**

* After project creation, navigate to the processing settings.
* Adjust the following parameters as needed:
  + **Segment Size** – defines how much text each chunk contains.
  + **Segment Overlap** – defines how much overlap exists between segments (for better context retention).
* Click on **“Process”** to begin document chunking and embedding.

**💬 5. Query the Document**

* Once processing is complete:
  + Click the **“Action”** button again.
  + A chat environment will open where you can **ask queries** based on your uploaded documents using the embedded LLM.

**✅ Outcome**

The Habilis platform was successfully integrated with **AVEVA PI System** and implemented in the **HydroPRT plant** as a **Smart Advisory** tool. This integration enables context-aware document querying and real-time insights, enhancing operator support and decision-making on the plant floor.

**📌 Initiative 3: Heat Exchanger Fouling Evaluation Using HTRI**

**Introduction to HTRI:**HTRI, or Heat Transfer Research, Inc., provides one of the most reliable tools for thermal design and performance analysis of heat exchangers. It is widely used in industry for tasks such as design validation, rating, and troubleshooting of exchanger operations.

**Objective of the Session:**I’ll walk you through how we use HTRI to estimate fouling in heat exchangers. We’ll also look at a 114-C exchanger file where we change certain parameters and observe how fouling behavior is estimated.

**🔧 Approach & Methodology:**

**✅ Step 1: Input Design File**

* + Started with the **existing HTRI file** containing the heat exchanger’s **design parameters** (such as duty, flowrates, temperatures, pressure drops, and U-values).

**✅ Step 2: Gather Current Operating Data**

* + Collected **real-time operating values** from PI (or DCS), such as:
    - Inlet/Outlet Temperature
    - Flow rate
    - Pressure drop
    - Operating pressure

**✅ Step 3: Update HTRI File for Current Conditions**

* + Replaced design inputs with **actual plant data** to reflect current performance.

**✅ Step 4: Run the Simulation**

* **Click “Run” or “Calculate”**
* **Observe results such as:**
  + **Heat duty**
  + **Pressure drops**
  + **U-value (Overall heat transfer coefficient)**
  + **Fouling resistance calculated by HTRI**

**✅ Step 5: Perform Fouling Analysis**

* **Actual heat duty vs expected duty**
* **HTRI will provide a calculated fouling resistance (Rf\_actual)**

**✅ Outcome:**

A quick and effective method to **quantify fouling** in heat exchangers using existing HTRI design files and current field data. This reduced the need for manual calculations and improved the plant’s **predictive maintenance capability**.

**🔹 Future Work: Automation Using VBA and Dashboard Integration**

As part of our future scope, we aim to automate the fouling estimation process using **VBA scripting**. The idea is to build a **bridge between live plant data and HTRI software**, enabling automatic updates and analysis without manual intervention.

**✅ What We Plan to Achieve:**

* **Live data acquisition** from the PI system.
* **Automated input** of this data into HTRI using VBA macros.
* **Execution of HTRI simulations** in the background.
* **Extraction of key outputs** like U-value, fouling resistance.
* **Display of results on a dashboard** for real-time monitoring and decision-making.

How to create a virtual environment for Python in Visual Studio

[Choose a Python environment - Visual Studio (Windows) | Microsoft Learn](https://learn.microsoft.com/en-us/visualstudio/python/selecting-a-python-environment-for-a-project?view=vs-2022)

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

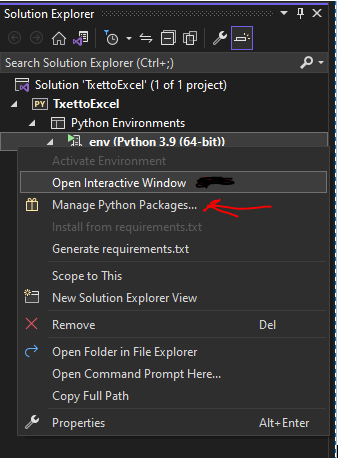
Description automatically generated

To start installing packages in virtual environment click on interactive window by Right Click on newly created virtual environment “env”

A screenshot of a computer

Description automatically generated

Then click on Manage Python Packages



Type the package you need to install e.g. pandas

A screenshot of a computer

Description automatically generated

Note that the status in interactive window will update after successful installation of Panadas package:

A screenshot of a computer

Description automatically generated