➢ 1 Cloud-Connected PLC Data Pipeline

Purpose: Real-time plant-to-cloud data transfer for smart manufacturing monitoring.

- **Tools Used & Why:**
  - **Node-RED** – Low-code flow-based programming tool to connect PLC tags, transform payloads, and push them to MQTT. Perfect for quick industrial integrations without heavy coding.
  - **HiveMQ Cloud (MQTT Broker)** – Cloud-based MQTT service ensuring reliable publish–subscribe communication between plant and cloud dashboards.
  - **Microsoft SQL Server** – Used for persistent historical logging and reporting. Chosen for strong querying capabilities and integration with BI tools.

Role in Project:

1. PLC sends tag data to Node-RED.
2. Node-RED formats and publishes it to HiveMQ Cloud.
3. Subscribers (dashboards, analytics) receive real-time data.
4. Node-RED also logs data into SQL Server for trend analysis and reports.

➢ 2 Kafka-Based IIoT Streaming with Secure Access

Purpose: Handle high-throughput industrial data streams with secure remote access.

- **Tools Used & Why:**
  - **Apache Kafka** – Distributed event-streaming platform to manage large amounts of plant data in real-time. Offers buffering and replay capability (unlike MQTT's limited queue).
  - **Node-RED** – Used as a Kafka producer and consumer to bridge industrial protocols with Kafka topics.
  - **ZeroTier VPN** – Creates a virtual static IP network so remote engineers can securely connect to plant systems without exposing ports.
  - **SQL Server** – Central storage for historical trends, eliminating duplicate storage across multiple locations.

Role in Project:

1. Plant equipment → Node-RED → Kafka producer → Kafka broker topics.
2. Kafka consumer streams data to SQL Server & dashboards.
3. Remote engineers securely view the same central data via ZeroTier network.

➢ 3 Modbus RTU/TCP Data Collection Without PLC

Purpose: Reduce costs by directly reading sensors/meters without a PLC.

- **Tools Used & Why:**
  - **pymodbus (Python library)** – Python Modbus protocol implementation to poll data directly from field devices.
  - **USB-to-RS485 Converter** – Hardware bridge for Modbus RTU over serial.
  - **Lightweight Flask/Dash dashboard** – Displays live readings on local PC/LAN without heavy SCADA.

Role in Project:

1. Modbus devices → RS485 USB adapter → Python app (pymodbus).
2. Data parsed and displayed on dashboard.
3. Historical logging possible with CSV/SQLite for low-cost setups.

➢ 4 Python-Based User Management with 21 CFR Part 11 Compliance

Purpose: Replace old VBA-based SCADA login with a modern, regulatory-compliant solution.

- **Tools Used & Why:**
  - **PySide6 (Python GUI framework)** – Builds modern desktop UI for login, role management, and audit logs.
  - **SQLite / SQL Server** – Stores user credentials, access roles, and audit logs.
  - **RBAC (Role-Based Access Control)** – Enforces permissions for operators, engineers, and admins.

Role in Project:

1. User logs in via PySide6 application.
2. System checks SQL for role and permissions.
3. All actions (logins, tag changes, alarms) recorded for audit.
4. Fully meets **21 CFR Part 11** — essential for pharma & regulated manufacturing.

---

5 Custom Python Batch Manager (Inspired by Rockwell LBSM)

Purpose: Provide PLC-like batch control without PLC memory limits.

- **Tools Used & Why:**
  - **Python (server-side)** – Executes batch logic for Units → Phases → Steps.
  - **PySide6 UI** – Modern graphical interface for recipe creation & tracking.
  - **SQL Server** – Stores all recipes, eliminating PLC memory dependency.

Role in Project:

1. Operator selects recipe → system loads Unit/Phase/Step structure from SQL.
2. Logic executed in Python server — commands sent to PLC/field devices.
3. Batch logs stored in SQL and displayed on UI & Grafana.

➢ 6 Centralized Grafana Dashboard for Remote Monitoring

Purpose: Provide global visibility of plant operations without installing dashboards locally.

- **Tools Used & Why:**
  - **Grafana** – Web-based visualization platform, perfect for time-series & SQL data.
  - **SQL Server / InfluxDB** – Stores time-series process data.
  - **Nginx Reverse Proxy (optional)** – For secure external web access.

Role in Project:

1. All plant data sent to central SQL/InfluxDB.
2. Grafana connects to database and generates live dashboards.
3. Remote users (India, overseas) log in via browser to see the same live and historical data.

➢ 7 Smart IIoT Operations Dashboard with Streamlit

Purpose: Unify plant operations, communication, and reporting into one web-based tool.

- **Tools Used & Why:**
  - **Streamlit** – Python-based app framework for interactive dashboards.
  - **Parquet file format** – Efficient columnar storage for large datasets.
  - **LAN-based chat system** – Built inside Streamlit for operator communication.
  - **PDF Export Libraries (ReportLab / WeasyPrint)** – For generating and printing reports directly.

Role in Project:

1. Operators log events, see live equipment data, and communicate via built-in chat.
2. Maintenance logs and equipment health stored in Parquet for fast queries.
3. Supervisors export daily/shift reports directly from dashboard without extra software.