Purchase Order System

Purpose

The Purchase Order System was initially developed to replace our outsourced PO software and reduce reliance on paper-based processes. After successfully implementing signature pad integration in our Final Inspection program, we began exploring other areas where we could reduce waste and cost. Through conversations with multiple departments, it became clear that purchase orders were one of the most frequently used and paper dependent documents — making them an ideal candidate for automation and internal development.

Key Tech Stack

- Frontend: Angular, TypeScript, HTML, CSS, Material UI
- Backend: ASP.NET Core Web API (C#)
- Database: SQL Server, Entity Framework Core
- Other Tools: IIS, Active Directory (auth), Git, SSMS

Key Features

- Dynamic PO Drafting with auto-save (files, line items, contact info, etc.)
- Allowed duplication, edits, approvals, and title/category updates based on AD grouping.
- Approval Workflow based on amount, department, urgency
- Email Routing for urgent POs and approvals
- Multi-Plant Support with logo/styling/user group differentiation
- Digital File Attachments tied to each PO, carried through versions
- PO Number Tracker with auto-generated, shared tracking across servers
- Versioning System that keeps revision history with date/user stamps
- Draft Editor that mirrors live PO form but loads saved drafts

Problems and Solutions

Problem:

Users would lose data during long form submissions. Whether it was getting called away before finishing or inadvertently closing the window.

Solution:

To combat this and make sure that users didn’t have to start each PO from scratch each time, I implemented an auto save feature that triggered after 30% of the form was filled out. After that the autosave triggered every 45 seconds (if changes were made) and any time a file was uploaded. We chose these as benchmarks that we could adjust as needed. As a result, we eliminated a lot of frustration and duplicate entries of information.

Problem:

Multi-plant operations needed separate PO systems — but corporate wanted oversight and number synchronization.

Solution:

Integrated cross-server logic to allow PO numbers, suppliers, contacts, and line items to sync across servers. Fallback logic ensures the draft is saved locally if a cross-server call fails. This reserves the PO number, so we don’t accidentally double tap it when connectivity returns and ensures that we don’t create PO numbers out of order.

Stumbles and Learned Skills

Upon our initial development stage, I was given a paper form to base our layout and design on. After 5 months of adding everything on the form, I flew down to Mexico to roll out our first test site to power users at that plant. Not only did they have parts of the form that they didn’t use, but they also had multiple fields that they had no use for. As it turns out, the form was a previous revision of the one I based my design on. Getting the program in front of people brought to light the areas that weren’t needed as well as additional areas that were. After going to Mexico thinking we would deploy the program within a month, we had to alter our timeline for the entire project by an additional 3 months. This taught me the importance of prioritizing frequent

communication with users during the building process as well as getting an MVP in front of them as early as possible.

Program Impact

• Eliminated outsourcing of the PO system, saving approximately $60,000 per year across four plants.

• Allowed more users access to the PO system by removing per-user license restrictions.

• Streamlined PO creation with predictive text for suppliers and part info, reducing duplicate entries.

• Created a digital paper trail from creation to closure with timestamps and logged users.

• Cut paper consumption by an estimated 10% across all plant locations.

• Received positive feedback and has been actively maintained for over 2 years.