Optimizing HVAC Manufacturing Process

Introduction:

The client is a U.S based HVAC manufacturer with over 30 years of experience, specializing in air movement and control products, such as dampers, louvers, roof curbs, diffusers, and ventilators for commercial and industrial markets. Known for their commitment to quality and customization, they operate four manufacturing facilities with quick turnaround capabilities and in-house testing for UL-listed products. Using two applications, the client was addressing two major workflows in the company. One was targeted at the customer orders and sales, while the other one was meant to track major stages of manufacturing leading to fulfillment of orders. With this project the client wanted to achieve a couple of goals. The first one was to get rid of several factors that were contributing to the applications running sub-optimally. And the second one was to migrate to the latest framework in order to address their current and future needs.

Client Details:

Name: Confidential | **Location:** USA | **Industry:** Manufacturing

Technologies:

Linux, Apache 2, MySQL, PHP, Laravel, React, JavaScript, jQuery, Bootstrap, AWS, Git, HTML, CSS, SCSS

Optimizing HVAC Manufacturing Process

Project Description:

The scope of the project included two applications:

1. **Pricing Program**
   This application provided a platform for customers to select products, configure options, and place orders. It includes user management features and generates a variety of sales-related reports.

2. **Plant Traveler System**
   This application tracked the various stages of manufacturing and product fulfillment for each order, ensuring efficient monitoring from production through delivery.

Situation

The client stated that the applications were developed by multiple programmers over the years without following a single coding standard. Code redundancy, hard-coded values, and deprecated code had made the codebase challenging to manage. As a result, the applications were vulnerable to security threats and adding new features or even to maintain the system was proving to be difficult.

The client requested that we address the following areas:

- Improve code quality and remove vulnerabilities
- Ability to Add or update products
- Ability to Add or update features
- Migrate the application from an outdated version of Core PHP to the latest Laravel framework

Optimizing HVAC Manufacturing Process

Solution:

Improved Code Quality

We moved hard-coded credentials to environment configuration files and improved code quality by removing hard-coded values, eliminating code redundancy, and removing dead code. We also updated deprecated functions and added form validations.

Feature Enhancements and Additions

1. **Maintenance System**
   The client did not have an application to add or update data, relying instead on their in-house developers to make direct changes to the database.

   We developed a Laravel application from scratch, focusing on ease of use. This application enables the client to manage users, customers (companies), rush orders, discounts, products, orders, and reports.

2. **Auto Scheduler**
   Previously, the client had to manually search for available manufacturing and shipping dates and select them. They wanted this process to be automated.

   We developed an application using React and PHP that automatically checks for available dates and assigns manufacturing and shipping dates. This application also allows users to overwrite pre-selected values, create shipments, add products to each shipment, and schedule both manufacturing and shipping dates.

3. **Unified Multiplier Calculation**
   Initially, multipliers were calculated in multiple places, such as the price sheet and multiplier report, leading to values getting out of sync and making maintenance a challenge. Additionally, new multipliers were not supposed to affect existing orders or price sheets

created within the last 30 days—a condition that was previously managed by hard-coding price sheet IDs and table names.

To streamline this process, we created a class with functions to handle multiplier calculations for each product type in the system. This class checks the creation date of price sheets to determine if the new multiplier can be applied, removing the need for hard-coding of values. The class is now used by the price sheet, which saves the calculated multiplier, allowing other programs to directly fetch multiplier data from the database as needed.

Additionally, we introduced company-specific multipliers, which are managed in the maintenance system.

4. **Rush Control**

   Previously, enabling or disabling rush orders for a product type required manual database updates, and the client consistently relied on a developer for this task. Additionally, rush availability depended on the estimated shipment date, as there is a limit on the number of items that can be shipped per day per product. This setup often led to rush options being incorrectly made available to customers when that shouldn't have been the case.

   To address this, we grouped similar products into rush categories and developed a Rush Control system with a user-friendly interface. This system allows the client to enable or disable rush orders for a product type with a single click. We also created an additional page that displays the number of items scheduled for shipment over the next 10 days, which the pricing program now references to determine rush availability.

5. **Price sheet Duplication**

   An existing feature in the system allowed customers to duplicate price sheets, enabling quick reordering of items without manually adding and configuring each one. However, this feature previously copied both configurations and prices directly from the database, resulting in outdated prices in the duplicate sheet. Customers then had to manually update each item to apply current pricing.

Optimizing HVAC Manufacturing Process

To address this, we updated the feature by organizing product-specific pricing logic into dedicated functions. Now, when items are duplicated from the parent price sheet, the latest prices are calculated and applied to the new sheet.

Additionally, large price sheets with over 200 items would often cause the copy operation to time out and fail. To resolve this, we implemented batch processing, duplicating items in groups of 10, and enhanced the UI to display progress at each stage—creating the price sheet, fetching items, and copying items—so users are informed throughout the process.

6. **New Products and updates**
   We have added several new products. And we continue to push updates to the system for feature improvements.

Migration

We moved features such as reporting, user management, and multiplier management to the maintenance system. One by one, we are migrating features from Core PHP to Laravel, ensuring data integrity throughout the process.

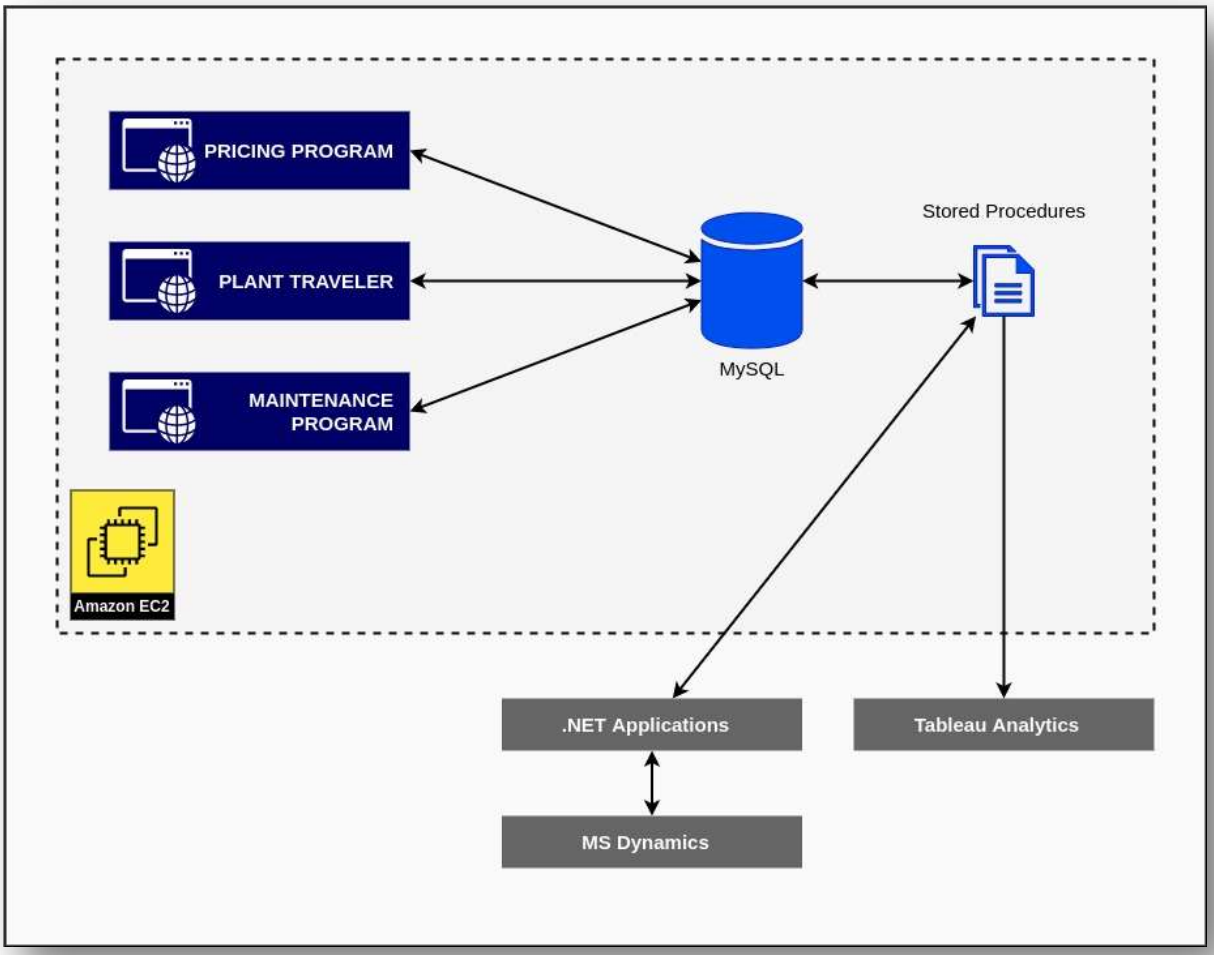
Upcoming Features

Branding

The client has three subsidiary companies and aims to enable users from each company to access the pricing program and create orders. They envision a dynamic UI where the logo and overall styling adjust automatically to reflect the specific company's branding whenever a user from that company logs in.
This feature will allow each brand to showcase and sell products through the client's application while maintaining its unique styling and identity. To achieve this, we plan to implement a common structure and leverage CSS custom properties for flexible, brand-specific styling adjustments.

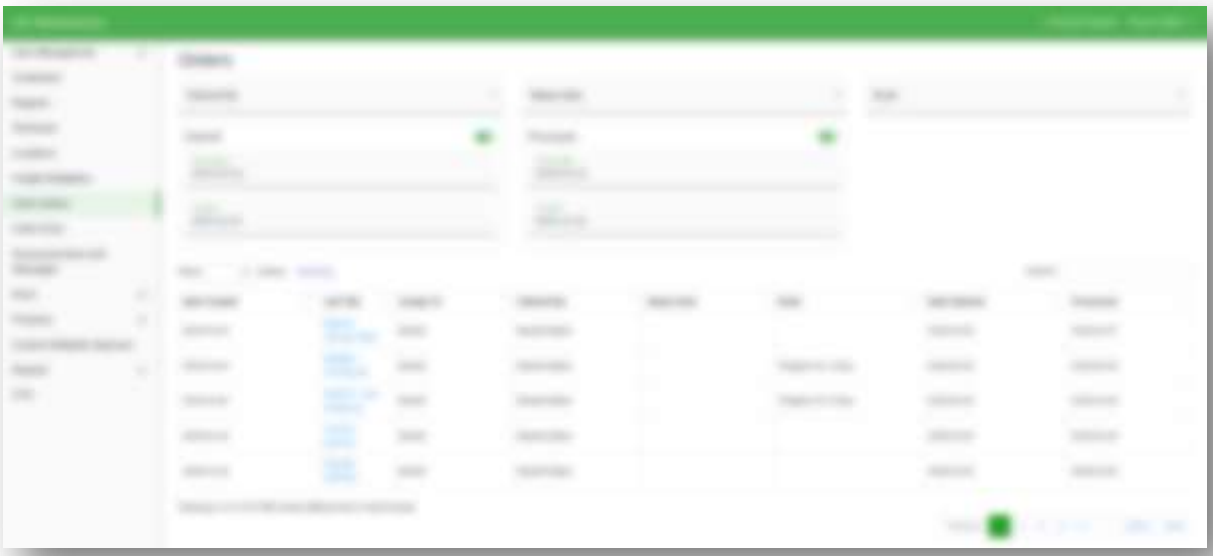
Optimizing HVAC Manufacturing Process

Architecture Diagram:

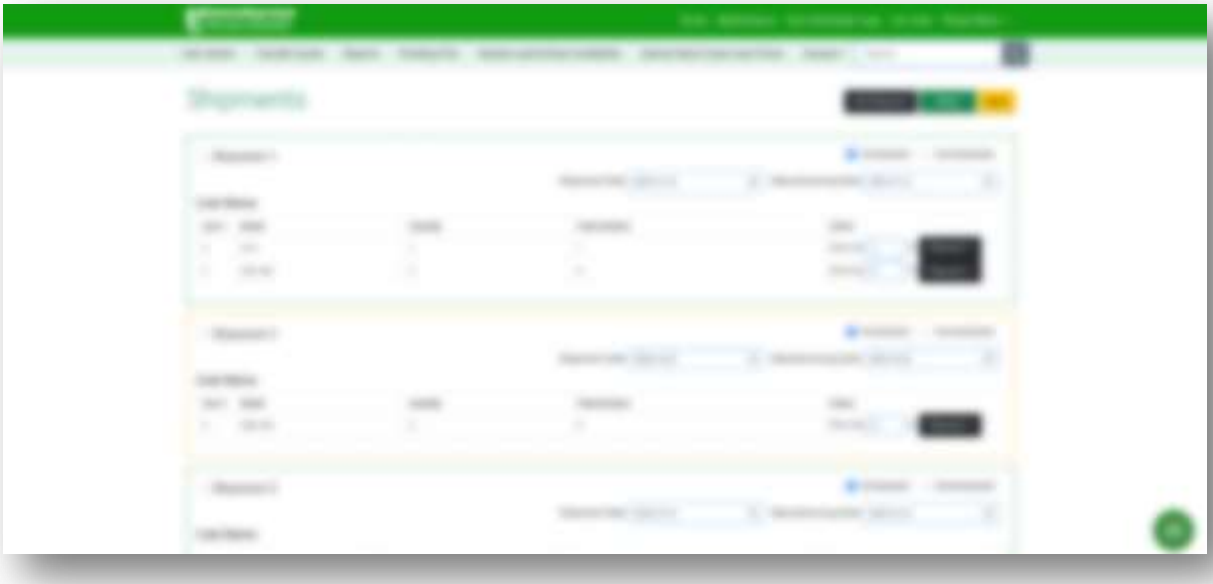


Optimizing HVAC Manufacturing Process

Screenshots:

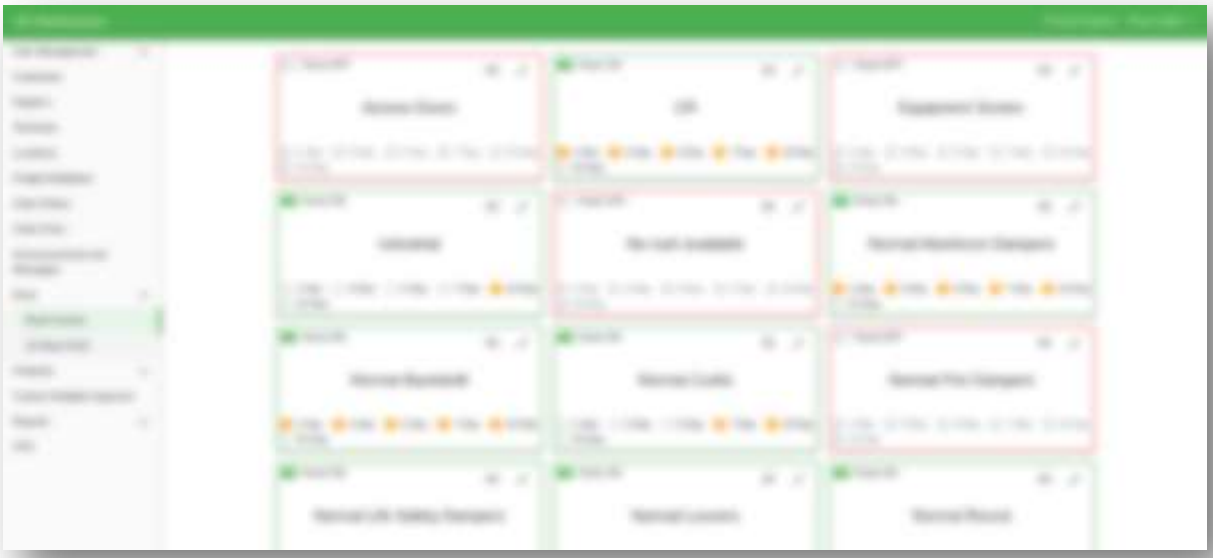


Maintenance system



Auto Scheduler

Optimizing HVAC Manufacturing Process



Rush Control

Ten Days Rush

Optimizing HVAC Manufacturing Process

Duplicate Price Sheet