Modern Delivery Application

# Stewart Hyde

### Introduction

This is modern .Net Core 3.1 application that uses same routines as Legacy application but been enhanced for more modern fluent user interface.

### Lessons Learned

This is likely one of most of important things that I learn during creation of the project. Coming from technology even in my previous job, which is for most part more than decade old, technology has significantly change.

Previously, even at last job I believe I was in the forefront of technology with integrating C++ code in .net and using WCF Services and WPF. The following is some of things which I initially had planned for this project but switch away from

1. Use of Log4Net in .net projects
2. Use of WCF to provide network interface to service
3. Use of WPF as front end of application
4. Old style database design

But working with new technology especially .Net Core 3.1, I found that there is better approach in designed and designed is important in the project. The process of going though my learning process with new technologies have even to refactor my project to make the solution better. Some of these including the following

1. Most importantly is separation of user interface from database. I had some of this in my plans for new modern approach for working with it and learning especially from Videos from Tim Covey, that I must spend the time and refactor the entire database logic and switch from what I originally thought was a good modern database approach with Entity Framework from Microsoft but instead using stored procedures approach with Dapper. Better yet with dependency injection, this could be switch out to completely different implementation
2. The initial plan for logging was use to Log4Net like in older logger, but Microsoft has made it own but very similar technology with .Net Core 3.1 and best of all this can be switch out using Dependency Injection with 3rd party loggers like SeriLog which can communicated with Seq for a more structure logging with better reporting
3. The Initial plan for networking was to use WCF. But it appears everything is going to web interface. This may be still look at if time permits.
4. The initial plan for user interface with WPF using the Blend took for Windows application, but technology for web has involved so much and finding a new Microsoft Blazor platform really a benefit and further more allow reusable components in Blazor pages which makes it even better. One draw back for me is lack of visual design tool currently for better appearance. But my hope with time permitting is that since Blazor is based on Bootstrap for html appearance, that I could use a tool like Bootstrap studio as a front end to at least help in appearance.
5. A big part of this design is normalization of data at same time of making it compatible with the Legacy C++ system. This connection is done with XML and process that updates and converts between the two systems. Even though I designed the two systems from scratch, I did not spend as much effort on the polishing the legacy user interface with new features. But data wise, the more modern system can consume and even cross communicate with legacy system via a database.
6. I was able to get the website to work with Microsoft Azure using the free account. These accounts have a drawback because they are shared environment and slow to come up. I had a lot of trouble getting SQL database to work with it and once I did get to work, the next time I came on it had problem. For the sake of this project, I came up with an interesting solution that allowed me to overcome the SQL problem and test out replacement of database layer. So, I implemented in-memory replacement for interfaces that talk to table and kept the objects as singletons which if the service up, the contents of the in-memory database are same. Future more two of tables were generated with seed information. ItemCategories is a fix table of information and Item was generated from xml from Legacy application. For the items, I added a method TestSeedItemsSimData() inside XMLDataTest.exe which basically writes code into text file – that can be copy into DeliverySupport.Data.Sim.InitializeItems CreateDefaultItems() method code at indicated area for initializing of the data.

### Example screen shots

This is screen shot of Blazor web application home screen

A screenshot of a cell phone

Description automatically generated

This is example of Edit Items screen. The plan is adding functionality to delete and edit base on security. Delete is code but the flow needs some improvement.A screenshot of a cell phone

Description automatically generated

This same edit items with list option selected

A screenshot of a computer screen

Description automatically generated

This is example of item items by category screen. Sort by category could be nice enhancement

A screenshot of a cell phone

Description automatically generated

This is example of Edit List by Category using the list option

A screenshot of a computer screen

Description automatically generated

This is example of the New Order screen

A screenshot of a social media post

Description automatically generated

This is example with New Order screen with list selected

A screenshot of a computer

Description automatically generated

This is example when you total a New Order and entering Customer Information

A screenshot of a computer

Description automatically generated

This is example of listing the orders currently in the system

A screenshot of a computer screen

Description automatically generated

This example of sending a message to user from website

A screenshot of a cell phone

Description automatically generated

This is example of listing the messages on the system

A screenshot of a computer

Description automatically generated

This is example of About screen which can show the legacy application sample screens

A screenshot of a computer

Description automatically generated

This example of one additional information showing xml produce from when saving order on Legacy Delivery system. This xml is sent to modern database via the OrderRouterService.

A screenshot of a social media post

Description automatically generated

This example of about screen, showing the database schema for Order store in SQL DatabaseA screenshot of a social media post

Description automatically generated

This is sequence diagram showing the comparison and similarities between Legacy and Modern system. Each system uses the same database service just Legacy sends xml information to the router which then saves the contents on SQL database. Note the use of In-memory database is only appliable to Modern web only system.

A screenshot of a cell phone

Description automatically generated

This is option security, which currently only use to protect deleting items from the Items database. This is only available with actual SQL Server version. This uses Entity Framework which web application originally used but later was change use interface based Dapper logic which could also be simple switch out for the in-memory version which was helpful running on Azure in my current setup.

With the Administrator active, the ability to delete an item is present – this can be done using in-memory but it available to all user’s vis a change in settings file.A screenshot of a computer

Description automatically generated

Once the user indicates Delete, the user must confirm the action before it come into effect.

A screenshot of a computer

Description automatically generated