Updated Software Solution

MJ Logistics

A New Software Solution for MJ Logistics

D284: Software Engineering

Kiem Bui

5-9-2024

Version 1.0

WESTERN GOVERNORS UNIVERSITY

CONTENTS

A. INTRODUCTION

We are proposing the following solution for MJ Logistics. In the following sections requirements, methodologies, design, and testing will be discussed.

A1. INTRODUCTION AND PURPOSE STATEMENT

The purpose of this document is to outline the details of a new CRM solution for MJ Logistics. This proposed solution is a vertical slice demonstration of a fully functioning technology stack that will unify, replace, and update several disjointed processes and technologies that currently attempt but fail to completely address the full requirements outlined in the CRM Requirements document.

The proposed solution meets only 4 of the requirements outlined in the requirements doc, which are detailed in section B. However, it is designed to be flexible and easily extendable to rapidly include additional features from the requirements document over subsequent iterations, until eventually covering all requirements outlined in the document, as well as additional requirements that may arise moving into the future.

This new system is being proposed to replace these outdated and disjointed methodologies with a unified interface that meets a small subset of the CRM Requirements as a proof of concept, and to generate feedback. If accepted, the back-end technologies will also serve as the foundation for the system to eventually expand to cover all current CRM processes outlined in the documentation.

A2. OVERVIEW OF THE PROBLEMS

Over the years, the current CRM solution has naturally evolved into an environment of many custom-built tools via spreadsheets and database management systems, which are disconnected from each other. In addition, these tools have spread across multiple offices and amongst members who work remotely from home. This has resulted in different team members following different processes to accomplish the same tasks, and manual workarounds to accomplish tasks.

While this was somewhat manageable in previous years, the 42% increase in sales over the past 2 years has led the company to outgrow its current solution. A lack of cohesion and a non-unified infrastructure have made CRM processes difficult to track and manage, and can also leave the company vulnerable to security and data breaches. The disjointed and improvised nature of these tools and processes allows us to unify processes into a well-thought-out and up-to-date system, giving us centrally controlled oversight and a unified process structure that can be documented and distributed organization-wide. Unifying these processes will also allow us to streamline them and employ automation to maximize efficiency.

The proposed solution aims to solve these problems with a series of end user facing web applications that interact with a centralized database (end users being both customers and internal org members). The web applications provide an easy-to-use interface that will replace the many improvised and disjointed processes members of our organization are currently using to handle CRM tasks.

The solution is built on an up-to-date technology stack using modern software architecture paradigms. It is structured to run using dynamic cloud hosting technologies, which will allow it to scale to MJ's current peak concurrent user requirements and well beyond while optimizing cost. It is also secured via authentication and authorization technologies which integrate into our organization’s current OIDC user

structure and policy system. By unifying all CRM processes through this system, we can provide a holistic experience that provides coherence, simplicity, performance, and security for the processes it is replacing.

A3. GOALS AND OBJECTIVES

The goal of the proposed solution is to provide a centralized organization-wide system that addresses the scalability and order management requirements outlined in the CRM Requirements document while providing an easy-to-use user experience that is compatible across all modern major browsers and platforms. The CRM outlined in this proposal provides a cohesive, scalable, solution that allows users to browse available products, get quotes, place orders, and manage their orders. A second admin portal provides a place for team members to handle administrative tasks such as database purging, and customer reps to handle customer support. This portal is locked behind modern authentication and authorization protocols.

Our peak objective is to deploy this CRM system and expand it to eventually address all functional requirements while maintaining expectations and key technical objectives outlined in the CRM Requirements document.

A4. PREREQUISITES

Number	Prerequisite	Description	Completion Date
1	SQL Server	Setup Azure SQL Server	Week 1
2	Collect Data	Collect and consolidate data necessary for the current feature set across the organization and load it into the centralized database	Week 1-2
3	Setup Cloud Services	Design and deploy Azure Load Balancer and ASP.NET Core Web APIs into Azure hosting	Week 2-4
4	Client Devices	All client devices for this system must support the latest OS for their respective platform (Windows 11, Android 14, etc). The solution has also only been tested on the latest updated version of web browsers, so client devices must also update their browser software.	Week 4

A5. SCOPE

The proposed solution is a vertical slice demonstration handling order management and user scaling. Ticketing, reporting, sales tracking, and forecasting are not yet implemented. However, the system is designed to be flexible, modular, and rapidly extendable to steadily integrate these functional requirements. As such, any requirements detailed in the CRM Requirements document not explicitly addressed in section B are to be considered out of scope for the current solution, as this system is a minimal viable product, rather than complete solution for all requirements.

This solution is designed to be an in-place replacement for the existing tooling and processes it covers. As such, updates and maintenance for the current custom tools and processes employed by team members are out of scope. However, training for employees to transition from their current workflows to using the new CRM is in scope and will be provided. Furthermore, migration of data from the current

custom tooling (spreadsheets, improvised databases, etc.) into the new centralized database is also in scope and will be essential to the full adaptation of the new system.

The current solution's front end consists of 2 web portals that are compatible across all devices. Native apps are currently out of scope for this solution, but can rapidly be developed and integrated into the solution's back-end services in the future.

A6. ENVIRONMENT

The data layer for this solution is an MS SQL Server containing databases for the organization. The data is regularly synced across multiple servers to ensure data integrity, and to minimize data loss. All databases are deployed in the Microsoft Azure ecosystem and are housed in data centers located in the US. This solution ensures our data is stored in the US, and that we comply with current laws and regulations. Finally, the data is also downloaded and stored on-premises via tape drives to provide an additional fail-safe.

Under normal operating conditions, the databases exclusively communicate directly with a pool of ASP.NET Core Web APIs hosted on Azure virtual machines. These Web APIs act as a guard between the database and the rest of the internet and control access via authentication tokens provided by our organizational OIDC provider. To be able to scale with concurrent user count and optimize cost, the Web APIs are managed via an Azure Load Balancer, also hosted in our Microsoft Azure tenant. All requests from the front end first go to this load balancer, which actively monitors the pool of virtual machines and concurrent user count. The load balancer then selects a virtual machine to forward the request to or spins up a new VM to help manage the load. Under periods of low load, the balancer will also sleep VMs to save on operating costs.

The front end consists of two web apps built on ASP.NET Core Blazor Server, hosted in Microsoft Azure. The first web app is a marketing and shopping page for users to purchase products, view order history, and submit questions to customer support. The second web app is an admin portal for organization members to handle technical administrative tasks (access to database operations, customer management, customer support, etc.). This admin portal is secured using OIDC, which is linked with the current organization's user database. Access to the various admin pages and the portal itself is managed via the organization's current user policies. Both of these web portals have been tested on the latest Android, iOS, and Windows operating systems, on the following devices using the latest major browsers:

- Google Pixel 8 (Android)
- iPhone 15 (iOS)
- iPad 10th Gen (iOS)
- Lenovo Thinkpad (Windows)

B. REQUIREMENTS

The three subsections below will outline the following 4 requirements defined in the CRM Requirements:

- Scalability: Ability to scale with growing peak demand and total user count
- Browser and OS Compatibility: Compatibility with the latest browsers and platforms
- Order Management: Ability to efficiently manage an order from quote to pending order to completed sale

- Hard and Soft Deletes: Ability to 'Soft Delete' and 'Hard Delete' data

USER REQUIREMENTS

Scalability

The new CRM system must be able to store a minimum of 2000 total users, and 500 concurrent users at any time. The proposed solution achieves this by using virtual machines running ASP.NET Core Web APIs that can dynamically be spun up or slept depending on load. A Microsoft Azure Load Balancer is also employed to manage these backing Web APIs, which enables the system to save money on off-peak hours, and automatically scale up to handle load during peak hours. This will also allow the system to easily scale into the future in case of further expansion and increases in peak load. As we scale up, we can easily increase the number of virtual machines on standby to scale with the growing peak user count. Our stress test located in section E demonstrates that we were able to meet the stated requirements.

Browser and OS Compatibility

Our new CRM system must also support the following OS's and browsers:

- latest Chrome and Chromium
- latest Firefox
- latest Microsoft Edge
- latest Safari
- mobile and tablet devices' application support systems
- latest iOS systems
- latest Microsoft operating system
- latest Android systems

This proposed solution addresses this using Blazor ASP.NET Core server-side web applications hosted with Microsoft Azure. The site itself has been designed with compatibility with all of the above-listed browsers in mind. Since the browsers listed above actively support the latest OS platforms, the solution will meet the full requirements for OSs and browsers. Our usability test in section E demonstrates users were able to efficiently and functionally use the web portals for the proposed CRM on all of the above-listed platforms, and that user experience was not degraded between devices.

FUNCTIONAL REQUIREMENTS

Order Management

The new CRM system needs to efficiently be able to manage the entire ordering process, from a quote to an order to a completed sale. The following features regarding order management need to be implemented in whatever CRM system we move to:

- order tracking
- taking orders
- converting quotes to orders
- reordering
- part ordering
- customer self-serve (i.e., portal)

The proposed system handles this with 2 elements of our tech stack. First, the backing database (MS SQL Server, hosted on Azure), has a schema with tables for quotes and orders. Second, the Blazor web portal allows customers to see their request quotes, complete orders, and view their order history to reorder previous orders. Our Usability Test in section E demonstrates that users can accomplish tasks related to all the required features listed above.

NON-FUNCTIONAL REQUIREMENTS

Hard and Soft Deletes

The new CRM system needs to be able to both 'soft delete' (remove data from view while keeping it in the database) and 'hard delete' (completely remove data from the database). Hard deletes need to be restricted to specific roles and permissions. Our system accomplishes this by adding a delete column of type Boolean to every table in the database. When we want to soft delete, we simply mark the deleted column to true, where it will then be filtered out by our UI. Hard deleting is accomplished via the admin portal, with operations gated behind the current OIDC provider and restricted only to users with the proper permissions. The deletes test in Section E demonstrates this functionality.

C. SOFTWARE DEVELOPMENT METHODOLOGY

In the subsections below this proposal will discuss the advantages and disadvantages between the Agile and Waterfall software development methodologies. In C2 we will go over our recommendation for development methodology and justify it.

C1. ADVANTAGES AND DISADVANTAGES

ADVANTAGES OF THE AGILE METHOD

Rapid Development

Agile focuses on short development cycles to deliver a minimal viable product to get it in front of a customer for bug testing and feedback ASAP, and then fixing and improving it over subsequent iterations

Well Tested Code

Due to the iterative nature of agile, the code is frequently end-to-end tested by customers (or beta testers), quickly revealing bugs, and yielding relevant feedback at a high velocity.

Flexibility

The iterative nature and quick feedback of the agile methodology also means less development time is wasted on features that are no longer relevant due to changing requirements. The shorter development lifecycle minimizes divergence between the product being developed and consumer needs and expectations.

DISADVANTAGES OF THE AGILE METHOD

Potential Code Instability

The rapid pace of software releases necessitates disciplined coding practices. Developers are pulled in two directions at once: coding for speed to meet deadlines, and coding for quality to

minimize bugs. Both the developers and the managers need to be extremely disciplined: managers need to set realistic feature sets, goals, and deadlines for each development cycle to minimize pressure, and developers need to be able to produce high-quality code at a high rate to minimize time wasted on bug fixes. Furthermore, the quick release of new features to the public adds the risk of catastrophic bugs, which could damage consumer data.

Communication Requirements

Agile projects rely on frequent, almost continuous communication with consumers. This is a heavy time requirement, which can be burdensome to teams wanting to focus on development. Though this can be mitigated with a representative to act as a go-between from the consumers to the development team, and it is recommended, that is either one extra person to hire or one person who is not developing the actual product.

Feature Creep

Due to the high level of interaction with end users, it is possible for the development team to become overwhelmed with feature requests. Undisciplined project managers and unfocused teams can quickly find that the feature set for their application has ballooned way beyond the original project scope and organizational budget. Project managers need to be able to manage demanding customers, and developers need to stay focused on the task at hand.

ADVANTAGES OF WATERFALL

Predictability

Waterfall is the staple of predictive modeling for software development. By ensuring that we finish each step of the development process before moving on, we can count on a clear and coherent plan for the next stage. The requirements and design of the product are fully defined and unchangeable once development starts, and in ideal conditions, the organization will know exactly what the end product will be.

Cost Savings

By spending the time upfront on having well-defined requirements and a thorough design for the end product, developers have a clear picture of what technologies and processes the end product will employ. This speeds up the actual implementation stage and cuts down on developers wasting time exploring other solutions that lead to dead ends.

Easier Maintenance

Since the usability flow and feature set are well-defined, developers have a broad overview of the full product from the beginning of the implementation stage. Furthermore, since the feature set is also *strictly* defined and unchanging, developers can design a more holistic self-contained product. This allows the product to be more elegant, rather than with more iterative solutions, where developers either need to try to predict and plan for features and extensibility that may come down the line in the future, or refactor or shoehorn new code in as new feature requests come up.

DISADVANTAGES OF WATERFALL

Slower Deployment

Since emphasis is placed on strict definitions of requirements, design, and documentation, the product is developed as a single holistic unit. While this has its advantages, it also means that the product can't be tested and deployed to end users until all aspects of the product are complete. The minimum viable product in this scenario is *the entire product*, and a currently workable prototype containing a working feature set that users would gladly pay for right then could be sitting waiting on other features to be finished.

Inflexibility

No matter how well things are planned, no design is perfect. Strict adherence to Waterfall makes it difficult to address issues that were overlooked during the design process. Because the development cycle is also slower, this increases the risk of the product being developed and the needs of an evolving marketplace diverging. This inflexibility and longer release cycle also means that the product is locked into the prescribed tech stack. The product will not be able to take advantage of newer more attractive technologies and paradigms.

Wasted Time

Because development can't start until the design is completely thought out and defined, developers can find themselves sitting idle while the minutiae of the product design is worked out. Gains made in predictability and stability must be weighed against the trade-off of a slower release schedule, and wasted developer time.

C2. BEST SUITED

Considering the advantages and disadvantages discussed in C1, this proposal recommends sticking with the agile methodology for the reasons below.

Proposal Scope

Consider the scope of this proposal. It only covers 4 of the various requirements outlined in the desired new CRM spec. using the agile method means we can quickly deploy the new CRM that handles these requirements, and iterate and expand it to cover the remaining features, rather than waiting a manyfold longer time for a fully finished product that covers all of the requirements outlined in the CRM.

Familiar Development Processes

Our management and development teams are currently structured around the agile method, and the 42% growth we've seen in the past 2 years indicates our success with it. Adapting to a different software methodology would require time, organizational restructuring, and possibly limit the performance of team members who have developed skills that synergize implementation, design, and customer relations

Flexibility

Considering our rapid growth in the past 2 years, our CRM requirements and internal tooling are hard to predict. Moving to a waterfall method will make MJ vulnerable to being locked into developing a product that diverges from unforeseen requirements and feature changes that will come up during its long development lifecycle. Remaining with the agile methodology means we can adapt our deployed solution as quickly as our company grows and requirements change.

Feature Creep

We have the benefit of having our CRM requirements already well-defined and laid out. This clear definition of requirements gives us a stable set of goals and targets for the first few iterative cycles while staying with the agile methodology keeps us flexible enough to adjust the specs as needs arise. Furthermore, most of our CRM consists of internal tooling, which will also help to control the amount and types of feature requests for our solution.

D. DESIGN

Below you will find 2 sample diagrams demonstrating design aspects of the proposed solution. Figure 1 is a flow chart of the interaction between a client machine using the Blazor Site to make a request, the load balancer dynamically scaling the available ASP.NET Web APIs, and processing the request. Figure 2 is an ERD diagram showing how customers, sales quotes, and orders will be stored in the database

FIGURE 1

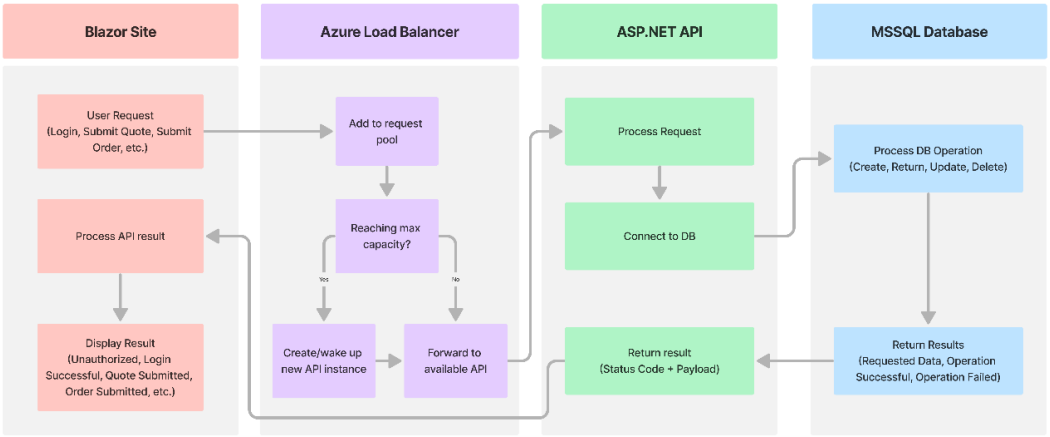
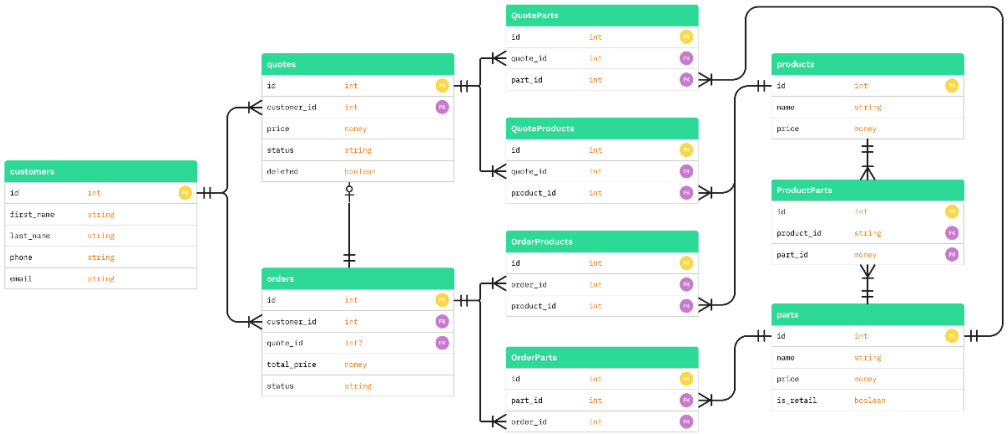


FIGURE 2



E. TESTING

STRESS TEST

Requirement to be tested:

Whether or not the performance of the software solution is diminished under a load of 500 concurrent users

Preconditions:

- The Testing server will be deployed and running prior to testing
- 10 Testing user accounts will be created and loaded into the user database prior to testing
- Because it is not feasible to have 500 users scheduled and ready to perform the necessary tasks to test the solution under peak load, a looping script that simulates a singular end-user interacting with the system must be prepared. It will run indefinitely until terminated and must be designed to be lightweight enough to be able to have hundreds of instances of itself running on a single machine. The script itself will perform the following:
    1. Randomly select 1 of the 10 testing user accounts and log in using the same endpoint as the login request the web application uses
    2. Sleep for 10 seconds
    3. Create an order for a randomly selected product using the same endpoint for placing an order on the web application
    4. Sleep for 10 seconds
    5. Cancel and delete the order that was just created using the same endpoint for canceling and deleting orders on the web application
    6. Check for a manual termination request
    7. If no termination request is present, loop back to step 2 and repeat the order create and delete process
    8. If the termination request is present, the script ends
- 3 high-powered machines that can easily handle 200 instances of the testing script running simultaneously must be set up and prepared with the script.

Steps:

1. Log in to the application manually with an observing device (not a testing script running device)
2. Click around the site, and observe the overall responsiveness and performance of the site.
3. Have the first machine start 200 instances of the testing script to simulate 200 concurrent users
4. Observe the performance of the site again, check the status of the back-end web APIs
5. Have the second test machine start 150 more instances, upping the simulated count to 350 users
6. Observe the performance of the site again, check the status of the back-end web APIs
7. Have the final test machine start 150 more instances, upping the simulated count to 500 users
8. Observe the performance of the site again, check the status of the back-end web APIs
9. Have all 3 machines signal termination for the running test scripts, concluding the test

Expected results:

- The performance of the site is consistent and stable at all 3 concurrent user loads (200, 350, 500)
- A 2nd back-end web API comes online and begins serving users in parallel to the first one once the simulated concurrent user load surpasses 400

Pass/Fail: Pass

We observed the site to remain performant and responsive across all 3 concurrent user loads.

Furthermore, as the user load began to hit the 350 mark, we saw the load balancer start a second instance of the web API and began evenly distributing the load amongst both web APIs.

USABILITY TEST

Requirement to be tested:

Whether or not the site provides a pleasant, and easy-to-navigate experience on the OSs and browsers listed in the OS and Browser Compatibility requirements in section B.

Whether or not users can complete all of the usability tasks outlined in the Order Management requirements in section B

Preconditions:

- The following devices will be acquired and set up to spec before running the Usability Test:
  - Google Pixel 8 with Android 14 and Firefox and Chrome installed
  - iPhone 15 with iOS 17 and Safari, Firefox, and Chrome installed
  - iPad 10th Gen with iOS 17 and Safari, Firefox, and Chrome installed
  - Lenovo Thinkpad with Microsoft Windows 11, Edge, Firefox, and Chrome installed
- 4 users to test on each device, each with a testing account already set up

Steps: The 4 devices above will be distributed amongst the 4 users, and each will complete the below usability tasks on all browser apps installed on the device. Users with handheld devices will also rotate the device into landscape and back during each usability task to test the website layout.

1. Login to the site
2. Select a product and get a quote from the Products page
3. Navigate to the quote in the ‘My Quotes’ section under ‘My Account’
4. Select ‘Order Now’ to turn the quote into an order and checkout
5. Check the status of the order in the ‘My Orders’ section under ‘My Account’
6. At this point, a developer will manually mark the order completed to simulate delivery
7. Check the status of the order in the ‘My Orders’ section again
8. Select the previous test order made in the Order History page
9. View available replacement parts for the product purchased in the test order, order one, and complete checkout
10. Navigate back to the Order History page
11. Place another order by clicking the Reorder button and checkout
12. View the order status page again

Expected results:

- The page displays properly and is responsive for all screen sizes and browsers for each of the 4 devices
- All 4 users can easily navigate and complete the usability tasks throughout the test process
- The order status page properly updates and reflects the status of the order as shipped, and then delivered during the appropriate steps

Pass/Fail: Pass

All users reported the site layout remained responsive to device orientation and window size throughout their testing. Their experience was also consistent across all browsers tested.

The ordering process behaved as expected, and the database was properly updated throughout the testing process. Users found the usability tasks easy to accomplish and reported that they found the website to be intuitive to use.

DELETION TEST

Requirement to be tested:

- Whether or not soft and deletes work

Preconditions: Conditions that must be present before the test case can successfully run.

- Prepare a user account with a sales quote in the database

Steps: The steps the tester must execute to test the feature.

1. Login with the testing user account
2. View the quotes page and delete the quote
3. Delete a quote
4. Refresh the quotes page
5. Login with a testing admin account
6. Navigate to the test user and purge the quote from the database

Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.

- After step 4 the quote should no longer display to the user, and the quote should still be in the database and flagged as deleted
- After step 6 the quote should be completely removed from the database

Pass/Fail: Pass

Deleting the quote from the user portal successfully removed it from display in the user quotes, but the entry remained in the database and was flagged deleted. After purging the quote using the admin portal, the quote was completely removed from the database.