**Updated Software Solution**

|  |
| --- |
| MJ Logistics |
| A New Software Solution for MJ Logistics |
| D284: Software Engineering |

|  |
| --- |
| Kiem Bui  5-5-2024  Version 1.0 |

Contents

[A. Introduction 3](#_Toc117147385)

[A1. Introduction and Purpose Statement 3](#_Toc117147386)

[A2. Overview of the Problems 3](#_Toc117147387)

[A3. Goals and Objectives 3](#_Toc117147388)

[A4. Prerequisites 3](#_Toc117147389)

[A5. Scope 3](#_Toc117147390)

[A6. Environment 3](#_Toc117147391)

[B. Requirements 4](#_Toc117147392)

[Business Requirements 4](#_Toc117147393)

[User Requirements 4](#_Toc117147394)

[Functional Requirements 4](#_Toc117147395)

[Non-Functional Requirements 4](#_Toc117147396)

[C. Software Development Methodology 4](#_Toc117147397)

[C1. Advantages and Disadvantages 4](#_Toc117147398)

[Advantages of the Agile Method 4](#_Toc117147399)

[Disadvantages of the Agile Method 4](#_Toc117147400)

[Advantages of {A Different Method} 4](#_Toc117147401)

[Disadvantages of {A Different Method} 4](#_Toc117147402)

[C2. Best suited 5](#_Toc117147403)

[D. Create Two Representations of the Software Solution 5](#_Toc117147404)

[Representation 1 5](#_Toc117147405)

[Representation 2 5](#_Toc117147406)

[E. Testing 6](#_Toc117147407)

[Test Name 1 6](#_Toc117147408)

[Test Name 2 7](#_Toc117147409)

[Test Name 3 8](#_Toc117147410)

[F. Sources 8](#_Toc117147411)

# Introduction

# A1. Introduction and Purpose Statement

Provide a brief introduction and explain the purpose of the proposed system.

# A2. Overview of the Problems

Describe the problems the company is currently facing and how the proposed solution will solve those problems.

# A3. Goals and Objectives

Provide the goals and objectives for the project and solution.

# A4. Prerequisites

List the prerequisites, their descriptions, and future completion dates. Be sure to be clear and concise for all listed prerequisites. (You may add lines for additional prerequisites if needed.)

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Prerequisite | Description | Completion Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# A5. Scope

Provide a brief overview of what the proposed solution will cover and what the proposed solution will not cover. It is important to set clear boundaries for the project.

# A6. Environment

Describe the front-end and back-end environments that the solution will be deployed in.

# Requirements

The three subsections below will outline the 4 following 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. This proposed solution will achieve this by using ASP.NET Core Web APIs that can dynamically be spun up or sleep depending on load. A Microsoft Azure Load Balancer will also be employed to manage these backing Web APIs, which will enable 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 increase in peak load. As we scale up, we can easily add more virtual machines on standby to handle scale with the peak user count. You will see in our stress test located in section E that we were able to achieve this in testing with our proposed solution

### 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 a Blazor ASP.NET Core server-side web application 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 are all actively supported in the latest OS platforms, the solution will meet the full requirements for OS’s and browsers. You will see in section E that in testing, users were able to efficiently and functionally use the web portal for the proposed CRM on all of the above listed platforms.

## 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. The backing database (MSSQL Server, hosted on Azure), has a schema with tables for quotes, and orders. 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 are able to 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 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 using traditional SQL database delete calls, gated behind the Web API and restricted only to users with the proper permissions. Section E demonstrates this functionality with our Deletes Test.

# Software Development Methodology

In the subsections below this document 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 modelling for software development. By ensuring that we completely 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 up front on having well defined requirements and a thorough design for the end product, developers have a clear picture of what the 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 is 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 are able to 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 risks 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 minutia of the product design are 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 requirement 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.

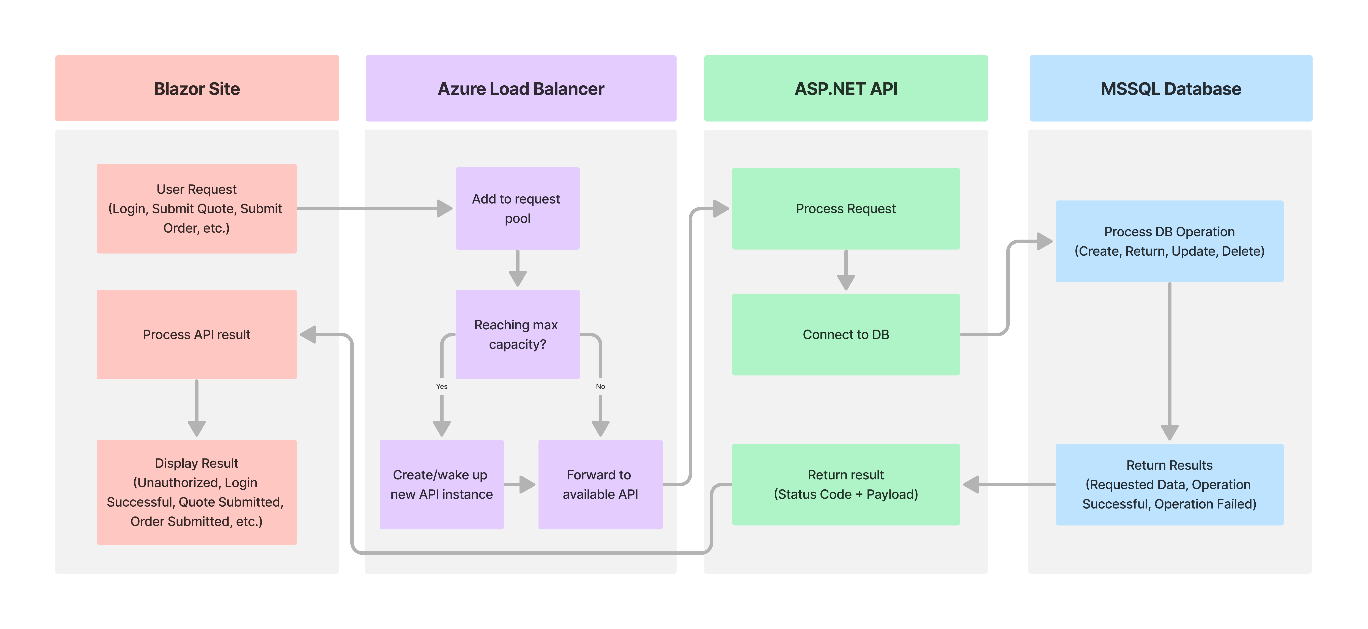
### Feature Creep

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 remaining 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.

# Create Two Representations of the Software Solution

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

Provide the second representation of the application (e.g., storyboard, flowchart, UML diagram, ERD, etc.) and a brief description of the representation.

# Testing

# Test Name 1

|  |
| --- |
| Requirement to be tested: |
| Preconditions: Conditions that must be present before the test case can successfully run. |
| Steps: The steps the tester must execute to test the feature. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc. |
| Pass/Fail: Explain why the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery or release. |

# Test Name 2

|  |
| --- |
| Requirement to be tested: |
| Preconditions: Conditions that must be present before the test case can successfully run. |
| Steps: The steps the tester must execute to test the feature. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc. |
| Pass/Fail: Explain why the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery or release. |

# Test Name 3

|  |
| --- |
| Requirement to be tested: |
| Preconditions: Conditions that must be present before the test case can successfully run. |
| Steps: The steps the tester must execute to test the feature. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc. |
| Pass/Fail: Explain why the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery or release. |