**CRM Design**

|  |
| --- |
| MJ Logistics Gaming Company |
| Software Engineering |
| CRM Development Project |

|  |
| --- |
| Desmond Ford  5-2-2023  [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

The system proposed would be a web-based CRM that manages user, customer, and business data to manage sales, internal communications, analytics, and process automation. This software would be hosted on internal servers to allow for the use of existing resources. Edge servers can be used in the future for expansion. Edge servers increase the scalability and ease of use for remote users.

# A2. Overview of the Problems

The MJ Logistics Gaming Company is currently growing faster than its internal systems can keep up with and their CRM solution is composed of a cluster of many manual and automatic processes. These processes also are part of user-friendly but slow general-purpose technologies.  
  
The solution proposed will allow for the consolidation of these processes into a user-friendly purpose-built design that allows for fast and easy internal applications and communications all in one place. This includes automating manually performed processes for a greater increase in efficiency.

# A3. Goals and Objectives

The system must incorporate all manual and automated processes currently used by MJ Logistics Gaming Company.

The system must be modular and flexible to allow for integration with other applications currently used by the company and to be integrated with applications in the future.

The system must be lightweight and easy to use for any user.

The system must be secure and must comply with all laws for data management.

# A4. Prerequisites

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Prerequisite | Description | Completion Date |
| 1 | Process Evaluation | The company needs to provide detailed information about all currently used internal applications and processes so they can be properly implemented into the new system. | 5/15/23 |
| 2 | Data Consolidation | The data management within the company is currently spread out into many systems. This data needs to be consolidated into a well-managed database. | 6/5/23 |
| 3 | Hardware Evaluation | The resources the company currently possesses to operate this software need to be evaluated. A decision on if the solution should use the existing internal hardware or if upgrades to the existing systems will be needed. | 5/12/23 |
| 4 | Software Demonstration | A simplistic version of the software needs to be created to demonstrate the potential of the new system and gather feedback for user needs and design. | 6/5/23 |

# A5. Scope

WILL:

* Provide a consolidated location for all user, customer, and business data
* Provide data analytics tools as described in the CRM requirements
* Be secure and data will be managed by all laws pertaining to the information gathered
* Be able to connect to other internally used software such as MS Exchange or Outlook to allow for communications
* Allow for the management of contracts and sales requirements
* Allow for part ordering both internally and from customers
* Allow for the management of internal resources like ticketing, sales processes, and pipeline management
* Have basic user management and user security

WILL NOT:

* Automatically generate datasheets or presentable information such as Excel documents or PowerPoint slides
* Have an internal communications system, such as calls, mail, or messaging. Communications will be handled by external applications

# A6. Environment

The front-end environment will be a web-based application that will be built using Electron to allow for the application to be independent of any browser.

The back end of the application will be hosted on the internal hardware available. There will be multiple SQL databases that handle the large amounts of relational data that the company must manage. Since the front end of the application will be an Electron Application the data analysis tools and any other major calculations will need to be handled server-side. Requiring robust hardware to handle the large number of users performing calculations all at once.

This solution does allow for lower resource requirements on the user side and allows the application to be used by anyone anywhere.

# Requirements

## Business Requirements

The system will be able to consolidate information to allow it to be easily accessible to the user. It will save and load data as requested and have the ability to provide data reporting functions. The data displayed will be able to be managed by a secure user management system to keep unnecessary data from being displayed.

Additionally, the system will be able to manage tickets internally, exchange information between other applications for communication purposes, and will have the ability to import and export data in a format that is standardized.

## User Requirements

The system will be lightweight, modular, and easily scalable to allow for many users to be able to access the system at once from anywhere that it is needed. It will be built as an Electron application to allow it to be used by any operating system, including as an application on mobile.

The system will be able to manage users' contacts and the users will be able to manage their personal preferences to allow for a customizable workflow for a default setup and on a contact-to-contact basis.

## Functional Requirements

The system will have a ticketing system that can manage and report calls and attach them to existing email solutions. It will manage orders for both users internally and externally, and manage contracts with proper authorization and signing required by law. The system will be able to provide analytics of the sales pipeline and will have the tools needed to manage and implement workflows and activities into the sales process.

## Non-Functional Requirements

The system will have basic data management functions that integrate with existing databases and consolidates information into a more robust database. The application and database will be hosted on the local hardware provided.

The system will be able to take information from the database and collect it into various analytical applications such as using internal information to provide quotes or using sales information to provide forecasting information for future sales.

# Software Development Methodology

Examine the Agile methodology and compare it to other software development methodologies.

# C1. Advantages and Disadvantages

## Advantages of the Agile Method

Customer Satisfaction – The system will be developed in a modular way, which will allow the application to be made in sections that can be given to the product owner and integrated slowly with the overall users. Allowing for a smooth transition between existing processes and the new process.

Adaptability – The system has a large number of different moving pieces. At any time, the product owner may decide to make changes to the requirements based on the results of the application that has been produced.

Continuous Improvement/Development – As the software is being developed and designed to be modular, flexible, and expandable. The system will likely need constant maintenance and will be updated continuously throughout the lifecycle of the application. Agile is perfectly designed for this type of application.

## Disadvantages of the Agile Method

Many of the pros of Agile development may also be considered cons in the long run depending on perspective.

Continuous improvement – While it is good to be designed with infinite flexibility in mind, the software may have no definite end and the company may need to hire full-time staff to manage and upkeep the software for the foreseeable future.

Little Documentation – Agile is extremely fast-paced and prioritizes a working product more than a well-documented one. Given the long-term nature of the project, maintenance on the application may end up being time and resource-consuming as a result.

Scope Creep – While it is good that the product owner can request changes as new features are released. The scope of the project may get out of hand. Given the large variety of requirements for this project, scope creep is likely.

## Advantages of {Waterfall}

Clear structure – Waterfall has a clearly defined structure to the process. This allows for on-time development that is well planned out in the beginning with no surprises in costs or time sinks.

Clear Communication – For long-term development, it is good to have a well-documented piece of software to work on. Waterfall would produce a product that would be well-documented and easy to modify in the future.

As Expected – The end product will be exactly as the user expected to receive based on information and requirement gathering performed at the beginning of the process.

## Disadvantages of {waterfall}

Little Flexibility – The project is clearly laid out at the beginning of the process and not many changes can be made in the middle of development.

Non-modular delivery – Waterfall does not allow for the application to be given to the product owner in pieces, instead the project is completed and given to the owner as one big piece. Given the requirement for this project, this does not seem like a good fit.

Slow – Waterfall doesn’t do testing until the latter half of development which means that the end product will be produced more slowly than if another method was used. The project seems marginally time sensitive given the growing pace of the company, making this a con.

# C2. Best suited

Agile is the best development methodology for this project as it provides a modular, fast-paced, and flexible environment for the project. In addition, the product owner can make alterations or give user feedback mid-development that the team can use to increase the quality of the product.

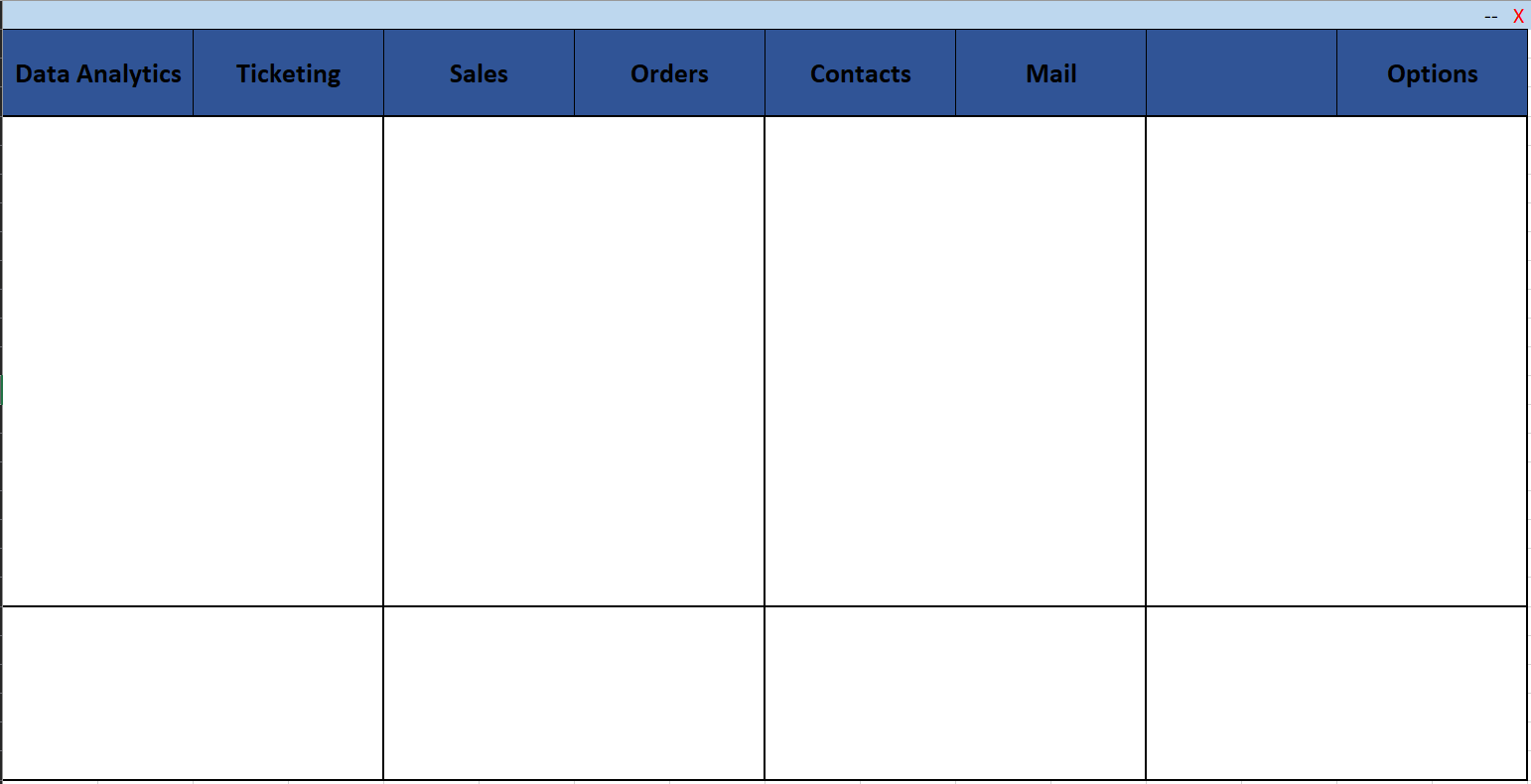
# Create Two Representations of the Software Solution

## Representation 1

This is a GUI diagram of the main page that the user will be able to customize with various UI panels of their choice. The top dark blue row is a collection of buttons the user can interact with to switch to other pages in the application.

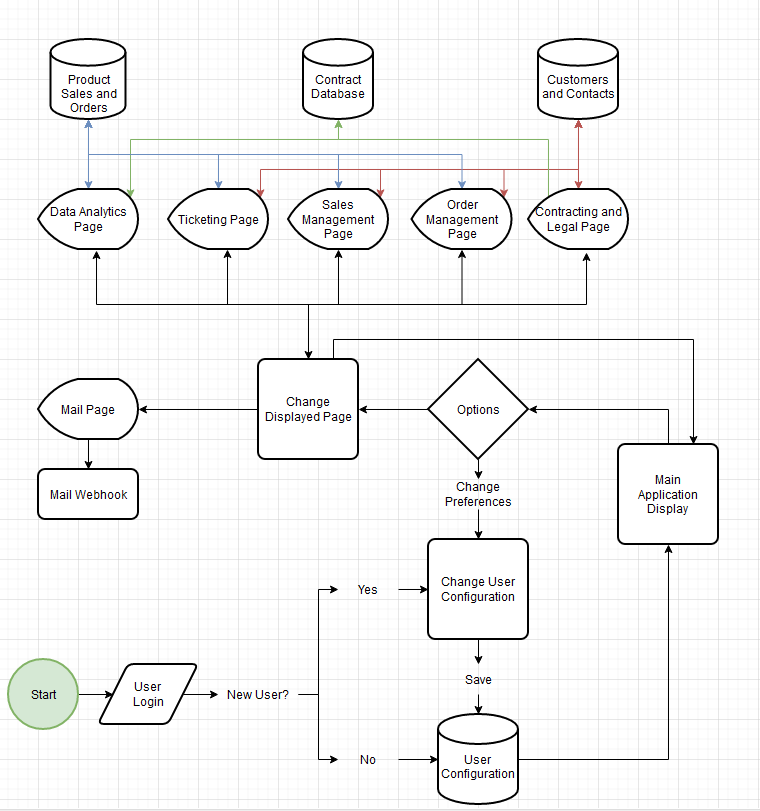
The large panel boxes will show verbose data from a panel while the smaller panel boxes at the bottom will show condensed at-a-glance information. The user will be able to choose what information is placed into which panels and determine how that information is displayed as a result. These panels will be filled with some default configuration once those assets are complete.

The top light blue row is simply the window border to show that this application is independent and running as an Electron application.



## Representation 2

This flow chart represents the user configuration and main UI workflow. It shows the user signing in then the server loading their configuration or requesting a configuration if they are a new user.  
  
There are various options for the user to choose from that change the page to different services available to the user. These services interact and link with various databases needed for their operation.



# Testing

# User Config

|  |
| --- |
| Requirement to be tested:  If the user can change the configuration of the main page |
| Preconditions: Conditions that must be present before the test case can successfully run.  Default Main Page UI elements to change An options/configuration Page |
| Steps: The steps the tester must execute to test the feature.   1. Start the application 2. Click on the options tab / navigate to the options page 3. Change the configuration of the main page 4. Click Apply 5. Navigate back to the main page 6. Check that changes have been made and are not the default configuration |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  Changes the user makes should happen immediately. UI elements on the page should automatically scale to where they are placed on the main page. The UI elements should display the correct information as configured. |
| 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.  Pass: The configuration is changed successfully Pass: The UI elements display the correct information and scale correctly  Fail: No configuration is changed Fail: There is no longer any configuration Fail: UI Errors |

# User Pref Save/Load

|  |
| --- |
| Requirement to be tested:  If the configurations made by the user can be saved and loaded |
| Preconditions: Conditions that must be present before the test case can successfully run.  Options page Settings and configurations to change Database for user data Account management system |
| Steps: The steps the tester must execute to test the feature.   1. Log into the application as a test user 2. Navigate to the options menu 3. Make changes to the configuration and settings 4. Apply the settings 5. Check that the settings were successfully changed 6. Log out of the application 7. Log back into the application and verify that the configurations saved between sessions |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  The user’s configuration changes should save between logging in and out of the application |
| 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.  Pass: The configurations should be saved to the database Pass: The configurations successfully load into the application when the user logs in  Fail: No configurations are saved Fail: The configurations are not loaded or are loaded incorrectly |

# Ticket Creation

|  |
| --- |
| Requirement to be tested:  Test that the user can create a ticket inside the Ticketing Page and fill out information about the contact. |
| Preconditions: Conditions that must be present before the test case can successfully run.  Ticketing page Sales Database Contact Database User accounts |
| Steps: The steps the tester must execute to test the feature.   1. Log into the application 2. Navigate to the ticketing page 3. Click on the create new ticket button 4. Add contact information manually 5. Add contact information for the database 6. Add information in the ticket description 7. Save the ticket to the database 8. Search for the ticket in the search bar 9. Click on the ticket and verify the information is saved correctly |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  The ticket should be created and information should be called from various databases as needed. This data can be used to fill out the information in the ticket automatically.  The ticket is then saved to a ticketing database that can be searched later. The tickets should be viewable and information should be inside the ticket. |
| 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.  Pass: Ticket is created Pass: Information is entered into the ticket Pass: Information is called and entered into the ticket from a database Pass: The Ticket is saved to the ticket database and can be searched  Fail: The ticket cannot be created Fail: UI errors in the ticket Fail: Information cannot be called from the databases (given the databases are functional) Fail: Ticket saving/loading database errors |

# Sources

Rod Stephens. (2015). *Beginning Software Engineering*. John Wiley & Sons, Inc

*Note: See the sources section in the requirements and rubric. If you did not use any outside sources, you may delete this section.*