TagIt.FM

Software requirements Specification (SRS) Document



MM

Members: Jeremiah Thomas

Anna Malmberg

Sandhya Joshi

Zachary Wawrzaszek

Stavros Bannoura

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary**  **Author(s)** | **Description of Version** | **Date Completed** |
| **1** | Sandhya Joshi, et al | Draft Release | 2020-06-21 |
| **2** | Sandhya Joshi, et al | Final Release | 2020-07-20 |

**Review And Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approving Party** | **Version Approved** | **Signature** | **Date** |
| Document Specialist | 1 | Anna Malmberg | 2020-06-20 |
| Customer Advocate | 1 | Sandhya Joshi | 2020-06-21 |
| Project Manager | 1 | Jeremiah Thomas | 2020-06-21 |
| Document Specialist | 2 | Anna Malmberg | 2020-07-18 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Requirements Document Review History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewer** | **Version Reviewed** | **Signature** | **Date** |
| Document Specialist | 2 | Anna Malmberg | 2020-07-18 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table Of Contents

[Table Of Contents 3](#_Toc45955125)

[1. Introduction 6](#_Toc45955126)

[1.1. Purpose 6](#_Toc45955127)

[1.2. Scope Of This Document 6](#_Toc45955128)

[1.3. Definitions, Acronyms, And Abbreviations 6](#_Toc45955129)

[1.4. Overview 7](#_Toc45955130)

[1.5. Business Context 7](#_Toc45955131)

[2. General Description 7](#_Toc45955132)

[2.1. Product Features/Functions 7](#_Toc45955133)

[2.2. User Characteristics 8](#_Toc45955134)

[2.3. General Constraints 8](#_Toc45955135)

[2.4. Collaboration Tools 8](#_Toc45955136)

[2.4.1. Communication 8](#_Toc45955137)

[2.4.2. Document Storage And Control 9](#_Toc45955138)

[2.4.3. File Sharing And Management 9](#_Toc45955139)

[2.4.4. Project Management 9](#_Toc45955140)

[2.4.5. Version Control Of Documents 10](#_Toc45955141)

[3. Business Requirements 10](#_Toc45955142)

[3.1. B1 10](#_Toc45955143)

[3.2. B2 10](#_Toc45955144)

[3.3. B3 10](#_Toc45955145)

[4. Functional Requirements 10](#_Toc45955146)

[4.1. F1 10](#_Toc45955147)

[4.2. F2 10](#_Toc45955148)

[4.3. F3 10](#_Toc45955149)

[5. Interface Requirements 11](#_Toc45955150)

[5.1 User Interfaces 11](#_Toc45955151)

[5.1.1. GUI 11](#_Toc45955152)

[5.1.1.1. I1 11](#_Toc45955153)

[5.1.1.2. I2 11](#_Toc45955154)

[5.1.1.3. I3 11](#_Toc45955155)

[5.1.1.4. I4 11](#_Toc45955156)

[5.1.1.5. I5 11](#_Toc45955157)

[5.1.2. CLI – N/A 11](#_Toc45955158)

[5.1.3. API – N/A 11](#_Toc45955159)

[5.2. Hardware Interfaces 11](#_Toc45955160)

[5.2.1. I6 11](#_Toc45955161)

[5.3. Communication Interfaces 11](#_Toc45955162)

[5.3.1. I7 11](#_Toc45955163)

[5.4. Software Interfaces 11](#_Toc45955164)

[5.4.1. I8 11](#_Toc45955165)

[6. Performance Requirements 12](#_Toc45955166)

[6.1. P1 12](#_Toc45955167)

[6.2. P2 12](#_Toc45955168)

[7. Other Non-Functional Attributes 12](#_Toc45955169)

[7.1. Security 12](#_Toc45955170)

[7.1.1. N1 12](#_Toc45955171)

[7.2. Reliability 12](#_Toc45955172)

[7.2.1. N2 12](#_Toc45955173)

[7.3. Maintainability 12](#_Toc45955174)

[7.3.1. N3 12](#_Toc45955175)

[7.4. Portability 12](#_Toc45955176)

[7.4.1. N4 12](#_Toc45955177)

[7.5. Resource Utilization 12](#_Toc45955178)

[7.5.1. N6 12](#_Toc45955179)

[8. Operational Scenarios 12](#_Toc45955180)

[8.1. Scenario A 12](#_Toc45955181)

[8.2. Scenario B 12](#_Toc45955182)

[8.3. Scenario C 13](#_Toc45955183)

[9. Use Case And User Stories 13](#_Toc45955184)

[9.1. Use Case 13](#_Toc45955185)

[9.1.1. B1 UC01 13](#_Toc45955186)

[9.1.2. B2 UC01 13](#_Toc45955187)

[9.1.3. Use Case Diagram 13](#_Toc45955188)

[9.2. User Stories 14](#_Toc45955189)

[9.2.1. User Stories Per Business Requirement 14](#_Toc45955190)

[9.2.2. User Stories Per Functional Requirement 14](#_Toc45955191)

[10. Sequence Diagrams 15](#_Toc45955192)

[10.1. Sequence Diagram Of Requirements I2 And N5 15](#_Toc45955193)

[10.2. Sequence Diagram Of Requirement N2 15](#_Toc45955194)

[11. Appendices 16](#_Toc45955195)

[11.1. MOSCOW 16](#_Toc45955196)

[11.1.1. Must 16](#_Toc45955197)

[11.1.2. Should 17](#_Toc45955198)

[11.1.3. Could 18](#_Toc45955199)

[11.1.4. Won’t 18](#_Toc45955200)

[11.2. Procedures For Project Requirement Changes 19](#_Toc45955201)

[11.2.1. Example Change 19](#_Toc45955202)

[11.3. Schedule 19](#_Toc45955203)

[11.3.1. Planning 19](#_Toc45955204)

[11.3.2. Analysis 19](#_Toc45955205)

[11.3.3. Design 20](#_Toc45955206)

[11.3.4. Building 20](#_Toc45955207)

[11.3.5. Testing 20](#_Toc45955208)

[11.3.6. Installation 20](#_Toc45955209)

[11.3.7. Maintenance 21](#_Toc45955210)

[11.3.8. Grand Total 21](#_Toc45955211)

[11.3.9. Gantt Chart 21](#_Toc45955212)

[11.4. Requirements Traceability Matrix 22](#_Toc45955213)

[11.5. Verification And Validation Testing Document 22](#_Toc45955214)

# Introduction

## Purpose

The purpose of this document is to build an application system, called Tagit.FM, that will enhance the search capabilities in MHC's database system. The tag-based file manager system stores metadata in the form of custom tags in a separate database instead of working directly in an operating system's limited file manager. The program searches tags and displays the most recent searches to ensure continuity and clarity for the user.

## Scope Of This Document

The intended readers of this document are current and future developers working on "Tagit.FM", hereafter referred to as TEAM, and the developers at MHC. The plan will include, but is not restricted to, a summary of the system functionality and the scope of the project from the perspective of TEAM. This document specifies the business requirements, functional and non-functional requirements, interface requirements, operational scenarios, user stories, sequence diagrams, scheduling and delivery estimates, project risks and how those risks will be mitigated, the processes by which the project will be developed, and metrics and measurements that will be recorded throughout the project.

## Definitions, Acronyms, And Abbreviations

\*\*ACID\*\*: Atomicity, consistency, isolation, durability

\*\*API\*\*: Application programming interface

\*\*Customer\*\*: A person who uses services provided by the insurance company

\*\*CLI\*\*: Command line interface; a text-based interface used for entering commands

\*\*DBMS\*\*: Database management system

\*\*Git\*\*: Distributed version-control system used for tracking source code during software development

\*\*GitHub\*\*: Company that provides hosting for software development version control using Git

\*\*GUI\*\*: Graphical user interface

\*\*HIPAA\*\*: Health Insurance Portability And Accountability Act Of 1996

\*\*Metadata\*\*: Data about data, or data which further describes the properties, qualities, and/or quantities of other data

\*\*MHC\*\*: My Healthcare, fictional insurance company used for this project

**\*\*SDLC\*\***: Software development life cycle

\*\*Slack\*\*: Proprietary business communication platform that features chat and file sharing

\*\*SQLite\*\*: A database that operates directly from library function calls, without a separate process or server

\*\*Tag\*\*: A string without spaces which provides categorical information about its object

\*\*UML\*\*: Unified modeling language

## Overview

Contemporary businesses generate vast amounts of digital information, including documentation for business processes, customer lists, accounting spreadsheets, and more. Efficiently accessing this data, however, can be a problem: only filenames and some filesystem-specific metadata are easily searchable and systems that add information or tags directly to files can cause integrity problems. Additionally, modifications to the files may not be acceptable due to regulatory or other requirements. Clearly, a different solution is required.

The tag-based file manager, TagIt.FM, that will be developed by TEAM will store metadata in the form of custom tags in a separate database. Instead of working directly in the limited file manager of an operating system, it will offer a clean user interface that can sort based on any number of user-selected tags. The program will search and open with the most recent search displayed to ensure continuity and clarity for the user.

## Business Context

In today's world, providing the best customer experience is as important as getting a new customer. MHC is serving millions of people from infancy, through their working lives, and into retirement. Their mission is to help people live healthier lives and make the healthcare system work better for everyone. They follow HIPAA regulations to protect customer's data and privacy and they value customer satisfaction and the quality of customer experience.

MHC maintains policyholder data on all their customers in a database, including personal information like social security numbers and dates of birth. Records are kept by MHC of their customers' medical history, doctor and hospital visits, prescription medication, surgeries, etc. This information gets updated as the policyholder/patient medical information changes or at minimum on an annual basis.

The data is stored internally at the MHC in their central database, and policy holders are only allowed to access their own information. As this information will be used by the employees at MHC, the business needs to keep track of all these records in an organized manner.

TagIt.FM can be used by any organization that has a database. It can be used to search for any type of customer information, such as, policy information, medical history, or personal data. The software can be used to retrieve any information quickly and effectively.

The tag-based file manager will address issues of misplaced documents or forgotten file names. For instance, when an employee searches the database for changes in the recent medical history of a customer, they can create a file tag based on the customer's name or medications that are stored in the database. Another employee can obtain the same customer information by searching for those same tags that the first employee added to customer's file, making sharing and collaboration easy. TagIt.FM will provide solutions to make everyday access to business information simpler and more intuitive.

# General Description

## Product Features/Functions

TagIt.FM creates a tag-based file manager that stores metadata, in the form of custom tags in a separate database. Instead of working directly in an operating system's limited file manager, the product will offer a clean user interface that can find documents based on any number of user-selected tags. The program will remember previous searches and open with the most recent search displayed to ensure continuity and clarity. It will provide some basic tags and instructions to get users started but otherwise aims at being "invisible", allowing businesses to instantly search, sort, and access important data.

The database will be loaded with business-critical features including management controls that restrict tag creation and file tagging to specific groups or users to prevent confusion or abuse. Each tag will include an optional description, also protected within the database. Because tags are stored inside the database, they automatically come with the key benefits of ACID. Every operation is guaranteed not to harm the integrity of either the main database or the filesystem.

TagIt.FM will address the shortcomings of other solutions. Business-critical files will never be modified in any way nor will any filenames change. Users will always be able to make use of this system side-by-side with legacy file managing systems, which simplifies administration. In short, the tag-based file manager will create new opportunities for organization and collaboration by making everyday access to business information simpler, easier, and more intuitive. This product will help employees at MHC search for any customer records more accurately. TagIt.FM will be extremely beneficial for any customer care representative, who needs to find information quickly so that they can serve the customer efficiently.

## User Characteristics

Our client, MHC serves millions of people from infancy to into retirement. They store important customer information for internal use in their central database. Most of the policy members who call the customer care line have questions regarding their benefits, claims, billing, or questions regarding specialists who are members of the insurance network.

Employees, especially the customer care representatives, need to provide this information within a certain time frame. According to company customer service policy, they cannot keep the customer on hold for more than 3 min. The employee must therefore be able to access all needed information in a fast and accurate manner. They need technology that can pull information based on a person's recent medical history changes or on other recent inquiries. MHC will benefit from an application that creates file tags based on custom criteria and search history so that the information can be accessed quickly.

## General Constraints

1. The software must be running on Windows or Linux operating systems.

2. The maximum size of a database file is 2 147 483 646 pages.

## Collaboration Tools

### Communication

TEAM has been using Slack and GitHub as the primary collaboration tools; Slack is mainly used for communication. The project manager created channels based on the project deliverables and discussions regarding topics specific to each deliverable take place on the respective channel. The main channel is used for discussing general topics and communicating with the other group members. The meeting channel is used for bi-weekly meetings.

Even though Slack is an asynchronous form of communication, one of its biggest strengths of our group is constant communication and updates. Our Slack communication channel is open 24 × 7 for continuous discussion and exchange of thoughts, which assists in maintaining an active, collaborative environment. Conversations can span over multiple days, picking up instantly where they left off. During the middle of the project, the project manager held regular "office hours" throughout the week where he would be on Slack exclusively; this helped everyone feel more comfortable talking in chat and helped ease the group into the dynamic, continuous dialogs that are possible with in the medium.

As far as the weakness are concerned, our team does not have a synchronous communication channel established, like video chat or conference calls. We would get larger amounts of instant feedback while using one of these options. Sometimes, we may not be able to correctly analyze what another person's reaction to our thought is when using asynchronous communication. Although each team member conveys their feelings through emojis and offers responses, it is not as effective at communicating emotion as watching the person's reaction. This can leave some emotional distance between the team members. This is might be the only weakness we could find in collaboration within with our team

### Document Storage And Control

All project documents are stored with Git and mirrored in a GitHub repository, which provides a neutral, shared location where everyone can upload files. Since the repository is public, everyone can see what is in it at any time. The project manager has given permission to all team members to commit changes to these documents, but nobody else has permission to make changes so that the documents remain secure. The advantage of this is that everyone retains a local copy of the team's material and can work offline, with Git handling the complexities of merging changes from different members later. The disadvantage is that incorrect changes sometimes get put into the repository; with everyone able to make commits, the process is fast, but the error-checking is nearly absent, at least up front.

### File Sharing And Management

GitHub is used for storing and sharing project related files. As noted above, the team repository contains all documentation that concerns the project, and anyone can pull a fresh copy at any time. Similarly, anyone can push their local Git commits to the team repository at any time, effectively sharing them with everyone else. The advantages of this system are that all shared documents are tracked, and the team has multiple ways to access the shared content, including locally and online at the GitHub website. The disadvantage is that this method is slower and less direct for rapid prototyping using certain tools that not every member has installed on their computer; in these cases, it has been simpler to send screenshots or comparable images directly over Slack and wait until the work has slowed down to commit to Git.

### Project Management

GitHub and Slack are both used for project management. The readme document on GitHub contains the to-do list with current tasks and is organized by assigned team roles; it is updated at least weekly, and generally a week ahead of time, so that members can plan their own schedules and work at their own pace without the pressure of surprise assignments. All team members are required to read the checklist and complete tasks by the due date. To compliment the list, Slack is used for bi-weekly meetings, questions, and discussion: assignments are frequently elaborated and explained in Slack and team progress is tracked at least at every meeting. The advantage of this combined method is low overhead. It utilizes the tools already in use by the group and can be used to both provide and capture feedback continuously. The disadvantage is that the checklist can sometimes seem impersonal since it does not have any direct collaborative features. Still, on rare occasions, schedule adjustments were made after the project manager received feedback about delays; thus, the schedule is still collaborative in a way, if not fully shared or gamified.

### Version Control Of Documents

As progress is made to the project, all the members keep updating their files locally and in the GitHub repository. Even though everyone in the team is extremely responsible for their work, Git allows for reverting documents back to previous versions, something that has been used on occasion. It also allows everyone to see the changes that have been made, not only contributing statistics to the PM but allowing the team to immediately know who to talk to about a given document. The advantage of Git is that it is a well-tested suite of tools that allows everything from tracking individual lines to managing different branches of work on the team's deliverables. The disadvantage is the learning curve: while Git is simple enough for basic file sharing and storage, many of the version-control operations—branching, merging, reverting, tracking, and even commit messages—have been sources of confusion and even frustration for team members at various points during the course of the project.

# Business Requirements

## B1

MHC wants to maintain its quality of service by providing the best customer care in a timely manner.

## B2

Employees at MHC want to search for files by filename, filetype, and/or location in an efficient manner so that they can answer the customers effectively.

## B3

The application must not have direct access to any confidential data or documents; rather, tags must be attached to files in a way that does not require read permissions to access the file tags.

# Functional Requirements

## F1

Customer care representatives must be able to retrieve the policy holder information using a keyword from their tag database.

## F2

The file tags must not allow employees to access documents that they do not have permissions to access.

## F3

The application must require an appropriate employee login to create, update, and delete tags.

# Interface Requirements

## User Interfaces

### GUI

#### I1

The employees at MHC must log to TagIt.FM in via it's GUI in order to manage tags.

#### I2

The GUI must have a conspicuous search feature that employees can use when searching for tags.

#### I3

The GUI must provide an option to allow/disallow employees from searching within subdirectories.

#### I4

Employees must be able to change the target search directory in the GUI before any search so they can expand or narrow the results.

#### I5

The GUI must include a history option that lists recently searched tags so that employees do not need to remember their past searches.

### CLI – N/A

Description: In order to make the application user-friendly, TagIt.FM will be using a GUI as the primary interface. The software will not be using a CLI at this time.

### API – N/A

Description: TagIt.FM will not use an API because all the features are exposed in the GUI and support for third-party add-ons will not be available at this time.

## Hardware Interfaces

### I6

The application must be available on x86 systems because most devices at MHC use these types of operating systems.

## Communication Interfaces

### I7

The application must use Local Area Network access to facilitate communication between the clients and the server.

## Software Interfaces

### I8

The application must interface with the filesystems of Microsoft Windows and Linux because they are commonly used by employees at MHC.

# Performance Requirements

## P1

The application must be able to pull and display files based on search criteria—with a base time of no more than 2 seconds and 0.5 seconds extra allowed for each tag used in the search—in order to provide information to the customers efficiently and effectively.

## P2

The application must not lock directories while employees are performing searches because it would reduce their ability to provide quality service.

# Other Non-Functional Attributes

## Security

### N1

The application must not make changes to any file or file's metadata because it violates data confidentiality and integrity.

## Reliability

### N2

The employees at MHC must be able to access the application 24 x 7 because the company provides service to customers at all hours.

## Maintainability

### N3

The application must have no external dependencies because the company wants to maintain the security of customer data and the ability to safely upgrade the application in the future.

## Portability

### N4

In order to increase efficiency, the application must be able to support several hundred employees' abilities to create, access, and modify file tags simultaneously.

## Resource Utilization

### N6

The application must keep all tag data updated in real time as tags are created, read, and/or deleted so that all employees can retrieve accurate search results.

# Operational Scenarios

## Scenario A

Manager wants to know, due to COVID-19, how many customers with age of 60 or above have visited the hospital (B2, F1, N5).

## Scenario B

Customer wants to know what his benefits are if he has another knee surgery for his torn ACL (B1, B2, F1, N5).

## Scenario C

Manager wants to pull data of people who have recently ordered prescription for chloroquine phosphate (B2, F2, P2).

# Use Case And User Stories

## Use Case

### B1 UC01

**\*\*Title\*\***: Customer interaction

**\*\*Primary actor\*\***: Employee

**\*\*Success scenario\*\***: An employee, in this case, a customer service representative receives call from customer. The customer is identified and poses a question to the representative. The customer service representative locates applicable plans and benefits. Customer service representative provides estimate to customer.

### B2 UC01

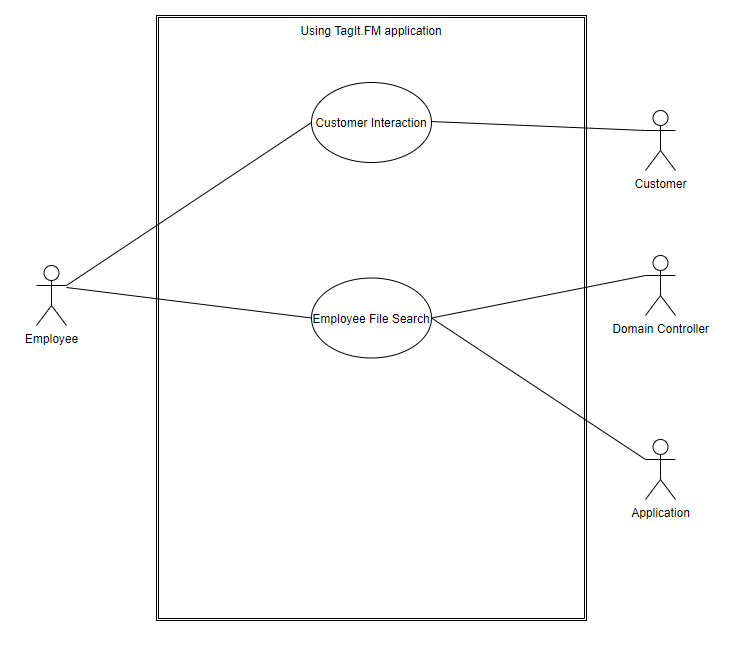
**\*\*Title\*\***: Employee file search

**\*\*Primary actor\*\***: Employee

**\*\*Success scenario\*\***: An employee logs in to a PC, and subsequently logs in to TagIt.FM. The employee starts a tag search. The application lists recent tags based on search history. The application creates a list of results based on finalized search. Employee views found records.

### Use Case Diagram

The following diagram illustrates the use cases described in the previous two sections. The first use case shows a customer service representative interacting with a customer, and the second shows an employee searching TagIt.FM for specific tags.



## User Stories

### User Stories Per Business Requirement

**\*\*B1US01\*\***: As a customer care representative, I need quick access to policy information, benefits and estimates to share with the customer.

**\*\*B2US01\*\***: As a customer care representative, I want to access all case related documents while on the phone with the customer.

**\*\*B3US01\*\***: As legal counsel, I need to be able to audit the software to ensure confidentiality of all customer records.

### User Stories Per Functional Requirement

**\*\*F1US01\*\***: As a customer care representative, I need to find information using a simple search, so I can quickly relay the information to the customer.

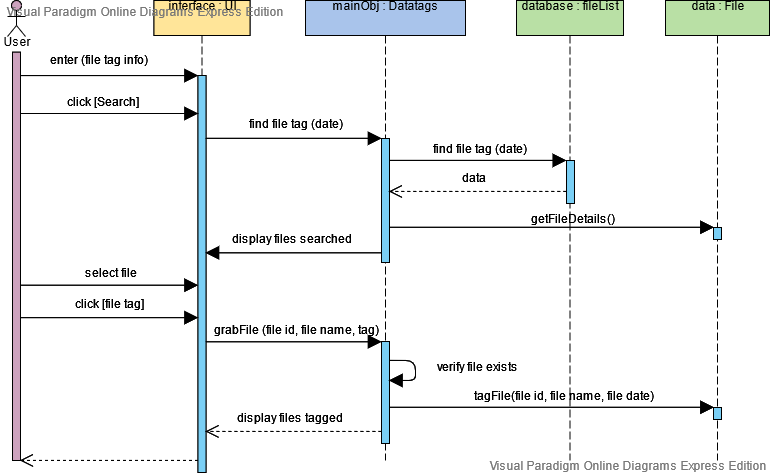
**\*\*F2US01\*\***: As legal counsel, I need to be able to confirm that tags do not allow employees access to documents past their clearance level.

**\*\*F3US01\*\***: As a manager, I need to create specialized tags for my employees to ensure they can find all documents that they need.

# Sequence Diagrams

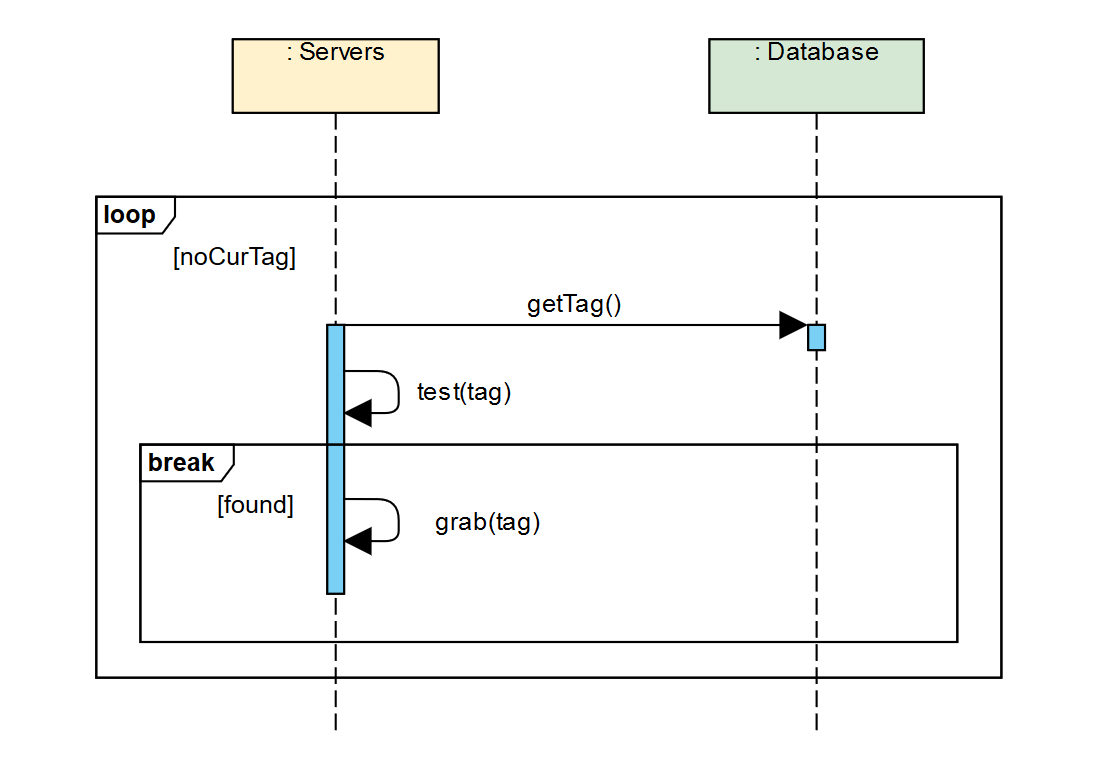
## Sequence Diagram Of Requirements I2 And N5

The following sequence diagram describes an employee using the TagIt.FM application search feature to find files that contain associated with specific tags. The employee enters the desired tag(s) into the GUI search feature and the information is relayed to the database, which locates the associated files and returns them to the GUI to be displayed as a list to the employee. The employee can then view any of the files that matched the entered tag criteria.



## Sequence Diagram Of Requirement N2

The N2 requirement states that the application must be available 24 x 7. The following sequence diagram is an excerpt of the search process showing how the TagIt.FM server will continuously communicate with the database containing the tag information, to satisfy the N2 requirement.



# Appendices

## MOSCOW

The MOSCOW method is a system commonly used to prioritize requirements for features in a software application. The requirements listed in the following sections are a historical reference of requirements and features suggested during an initial brainstorming session of TEAM. Most of the following requirements were not chosen to be included in the finalized requirements document for TagIt.FM as they would not meet the needs of the client.

### Must

**\*\*M1\*\***: MHC wants to maintain its quality of service by providing best customer care in a timely manner.

**\*\*M2\*\***: MHC wants to maintain all its customer data so that it can be searched efficiently and effectively.

**\*\*M3\*\***: Software must find files when filename, filetype, and/or location are unknown.

**\*\*M4\*\***: Software shall not have direct access to any confidential data or documents; rather, tags shall be attached to files in a way that does not require read permissions.

**\*\*M5\*\***: Customer care representatives shall be able to retrieve the policy holder information using a keyword from their database.

**\*\*M6\*\***: Insurance agents shall be able to log in to the system and find related documents by using keywords associated with the client or project.

**\*\*M7\*\***: Managers shall be able to create additional tags that their group requires.

**\*\*M8\*\***: Tags shall not allow users to access documents that they do not have permissions to access.

**\*\*M9\*\***: Software will be able to pull and display files based on search criteria in a reasonable amount of time: less than 5 seconds.

**\*\*M10\*\***: The application shall make no changes to any file or file's metadata.

**\*\*M11\*\***: GUI will include a centralized (easily spotted) search bar for the searching of tags.

**\*\*M12\*\***: Main pane must be able to change target directory.

**\*\*M13\*\***: The application will be available on x86 systems.

**\*\*M14\*\***: The application will use local area network access to facilitate communication between the clients and server.

**\*\*M15\*\***: The application will need to interface with the filesystems of Microsoft Windows and Linux.

**\*\*M16\*\***: Software will be relatively lightweight and will not affect other business-necessary operations that must also be performed.

**\*\*M17\***\*: Tags must be able to be removed without unintended side effects.

### Should

**\*\*S1\*\***: Administrator login shall be required to manage the tag database.

**\*\*S2\*\***: Relevant employee login will be required to create, update, and delete tags.

**\*\*S3\*\***: System-assigned tags shall only be deleted by managers, while agent-assigned tags may be deleted by agent.

**\*\*S4\*\***: Access to user interface for managing personal tags shall require log in.

**\*\*S5\*\***: GUI will include history tab for recently searched and accessed tags/files.

**\*\*S6\*\***: Secondary pane must contain an option to allow/disallow searching subdirectories.

**\*\*S7\*\***: System should be available 24 x 7 for all users.

**\*\*S8\*\***: Software must be able to support several hundred user's/employee's ability to create, access, and modify tags simultaneously.

**\*\*S9\*\***: Software must keep all tags up to date in real time as they are created, updated, and/or deleted.

**\*\*S10\*\***: System should not have any external dependencies.

**\*\*S11\*\***: Database system should perform database maintenance every midnight.

**\*\*S12\*\***: Files that have been returned from a tag search will have additional methods available to sort based on filetype.

**\*\*S13\*\***: Customer hold-times during peak hours shall be 5 minutes or less.

### Could

**\*\*C1\*\***: System shall always cross-reference customer ID before assigning customer specific tags to any documents.

**\*\*C2\*\***: The system shall start searching within 2 seconds after a tag match is guessed but the user is still typing.

**\*\*C3\*\***: Customers shall not be able to access the tag system.

**\*\*C4\*\***: Windows login shall be recognized when adding tags to shared files or folders.

**\*\*C5\*\***: Customer care representative shall be able to retrieve all the case history by searching caseId or tag number.

**\*\*C6\*\***: The application shall produce regular reports on most popular tags and searches.

**\*\*C7\*\***: Tags shall stay assigned to document if moved to a different project or directory.

**\*\*C8\*\***: Files that have been found during search will support drag and drop.

**\*\*C9\*\***: User interface for adding personal tags shall be on mouse-right-click drop down.

**\*\*C10\*\***: User interface shall offer options for personalized color schemes for tags.

### Won’t

**\*\*W1\*\***: Software shall consider HIPAA regulations and all data and transfers of data shall be completely secure.

**\*\*W2\*\***: The application must be able to support users effectively organizing files.

**\*\*W3\*\***: System shall maintain a user portal for policy members using customer specific tags.

**\*\*W4\*\***: System shall create client and project specific tags when a new client or project is created.

**\*\*W5\*\***: To ensure confidentiality, if a customer tag is present on a document, a document move must be approved by a manager or group leader.

**\*\*W6\*\***: Project relevant tags shall be assigned to any document that is added to a tagged project folder.

**\*\*W7\*\***: All customer correspondence shall be compiled and mailed out the same business day as customer request was made.

**\*\*W8\*\***: When a case is closed, all relevant documents shall be transferred to customer file within one business day.

**\*\*W9\*\***: System shall assign reward points to policy holder based on their medical history and lifestyle.

## Procedures For Project Requirement Changes

To submit a change request, a team member must create a numbered section in the change\_requests.txt file. This section must contain, in description list items, the status (open), the ID of the requirement, the updated requirement text, the rationale, the requester's name, and the request's open date as follows:

### Example Change

0. Example Change

**\*\*Status\*\***: Open

**\*\*Requirement ID\*\***: [ID of requirement (eg, B1); use the official req code, not the M[o]SC[o]W numbering; make a new ID if the requirement is new]

**\*\*Requirement text\*\***: [Full text of the proposed change; for new requirements, simply make a new text]

**\*\*Rationale\*\***: [\*Briefly\* describe why the change should be made]

**\*\*Requester\*\***: [Name]

**\*\*Open date\*\***: [ISO date]

*During the next team meeting, the change control board must vote on all active items. The status must be changed to "Closed" by the CA (customer advocate). The CA must add in the rest of the fields—namely the close date (the date of the vote) and the result (with description according their judgment, if it was not discussed at the time of the vote)—and then update the SRS itself with the new requirement, if approved. The format is:*

**\*\*Status\*\***: Closed

**\*\*Close date\*\***: [ISO date]

**\*\*Result\*\***: Accepted|Rejected: [brief description of why the change was accepted or rejected, including if the requirement text in the change request was modified]

## Schedule

An example project calendar for the full SDLC, detailing the time each major project task is expected to take. Times are rounded into 4-hour (half-day) blocks. Subtasks are not broken out in this version, however, and ongoing overhead for larger teams and management is not included.

### Planning

**\*\*Identification\*\***: 4 h (does not include what precedes the "Eureka!")

**\*\*Pitch\*\***: 4 h

**\*\*Formation\*\***: 8 h (includes initial management overhead, tool setup, and standardization)

**\*\*Total\*\***: 16 h

### Analysis

**\*\*Gathering\*\***: 8 h (includes brainstorming and client meetings)

**\*\*Sorting\*\***: 8 h (includes later changes and votes)

**\*\*Drafting\*\***: 4 h (includes diagrams)

**\*\*Finalizing\*\***: 4 h

**\*\*Total\*\***: 24 h

### Design

**\*\*Overview\*\***: 4 h (every step doubles in complexity)

**\*\*Details\*\***: 8 h (includes diagrams)

**\*\*Modules\*\***: 16 h (including DB design)

**\*\*Pseudocode\*\***: 32 h (template, comments, and basic logic only)

**\*\*Total\*\***: 60 h

### Building

**\*\*Graphics**\*\*: 8 h (timeboxed else they take forever!)

**\*\*DB\*\***: 4 h

**\*\*Exceptions\*\***: 4 h

**\*\*Classes\*\***: 48 h (12 × 4 h/class average from prior experience)

**\*\*Integration\*\***: 8 h

**\*\*Total\*\***: 72 h

### Testing

**\*\*Interface\*\***: 4 h (overlaps Building > Integration)

**\*\*Performance\*\***: 8 h (unattended)

**\*\*Recovery\*\***: 4 h

**\*\*Usability\*\***: 8 h (3 people minimum)

**\*\*Alpha\*\***: 4 h

**\*\*Total\*\***: 28 h

### Installation

**\*\*Marketing\*\***: 4 h (mirrors alpha)

**\*\*Manuals\*\***: 8 h (mirrors usability)

**\*\*Deployment\*\***: 8 h ("Deployment Day", just in case)

**\*\*Total\*\***: 20 h

### Maintenance

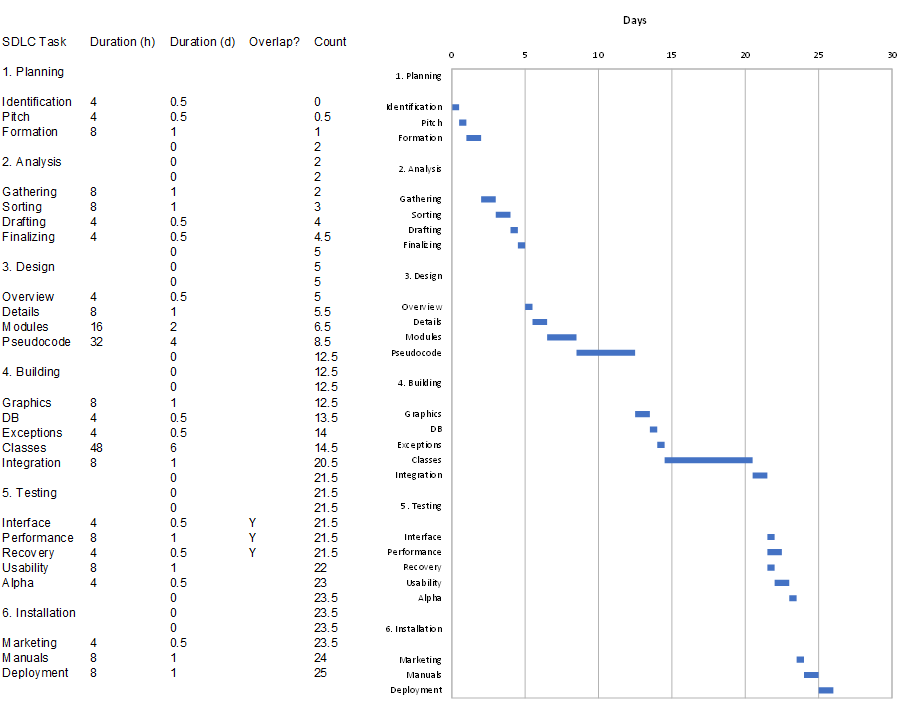
[Ongoing]

### Grand Total

**\*\*Total\*\***: 220 h (27.5 d)

### Gantt Chart

The following Gantt chart illustrates the schedule outlined in the previous sections.



## Requirements Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Parent ID | MoSCoW | TestCase ID(s) |
| B1 | None | Must | TC0001 |
| B2 | None | Must | TC0002 |
| B3 | None | Must | TC0003 |
| F1 | B2 | Must | TC0004 |
| F2 | B3 | Must | TC0005 |
| F3 | B3 | Should | TC0006 |
| I1 | F1 | Must | TC0007 |
| I2 | F1 | Must | TC0008 |
| I3 | F2 | Should | TC0009 |
| I4 | F1 | Must | TC0010 |
| I5 | F2 | Should | TC0011 |
| I6 | B2 | Must | TC0012 |
| I7 | B1 | Must | TC0013 |
| I8 | B2 | Must | TC0014 |
| P1 | B1 | Must | TC0015 |
| P2 | F1 | Must | TC0016 |
| N1 | F1 | Must | TC0017 |
| N2 | F1 | Should | TC0018 |
| N3 | F2 | Should | TC0019 |
| N4 | F1 | Should | TC0020 |
| N5 | F1 | Should | TC0021 |

## Verification And Validation Testing Document

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Role | Status | Tester | Description |
| TC0001 | Client | Null | SJ (CA) | Enter the valid tag and check to see if the results returned within expected time. |
| TC0002 | Client | Null | SJ (CA) | Validate if all file tags are searched within expected time. |
| TC0003 | Client | Null | AM (DS) | Enter valid tag and verify if all returned results are within the scope of the current user's permissions. |
| TC0004 | Client | Null | JT (PM) | Enter valid tag and check if results are returned. |
| TC0005 | Client | Null | JT (PM) | Enter valid tag in restricted directory and check if no results are returned. |
| TC0006 | Client | Null | JT (PM) | Attempt to create |
| TC0007 | Client | Null | SB (LT) | Validate if the user type account has been created to manage tags. |
| TC0008 | Client | Null | SB (LT) | Validate if the search bar can be seen and allow the user to search. |
| TC0009 | Client | Null | SB (LT) | Validate if the user can process the option in the pane. |
| TC0010 | Server | Null | SB (LT) | Validate if the change target control accepts the request of the user and switches the directory. |
| TC0011 | Server | Null | SB (LT) | Validate if the user can access the history data. |
| TC0012 | Client | Null | SB (LT) | Validate whether user type has the required computer system. |
| TC0013 | Server | Null | SB (LT) | Validate if any requests to the server did not successfully pass. |
| TC0014 | Client | Null | SB (LT) | Validate whether user type has the required filesystem. |
| TC0015 | Client | Null | ZW (LSD) | Validate that average search times remain below 5 seconds. |
| TC0016 | Client | Null | ZW (LSD) | Validate that directories can still be accessed during searches. |
| TC0017 | Client | Null | JT (PM) | Compare file pre and post-tagging; are hashes the same? |
| TC0018 | Server | Null | JT (PM) | Record downtime over an arbitrary period; was it 0 s? |
| TC0019 | Client/Server | Null | JT (PM) | Regex all imported modules; is the set of modules a subset of Python standard library modules? |
| TC0020 | Server | Null | JT (PM) | Does server pass I/O stress test simulating 200 users? |
| TC0021 | Client | Null | JT (PM) | Does the client have a method to fetch current tag data before every search executes? |
| TC0001 | Client | Null | SJ (CA) | Enter the valid tag and check to see if the results returned within expected time. |