TagIt.FM

Software requirements Specification (SRS) Document

MM

Members: Jeremiah Tomas

Anna Malmberg

Sandhya Joshi

Zachary Wawrzaszek

Stavros Bannoura

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary**  **Author(s)** | **Description of Version** | **Date Completed** |
| **1.1** | Sandhya Joshi, et al. | Draft Release | 2020-6-21 |
| **1.2** |  |  |  |

**Review and Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approving Party** | **Version Approved** | **Signature** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Requirements Document Review History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewer** | **Version Reviewed** | **Signature** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc43478981)

[1.1 Purpose 5](#_Toc43478982)

[1.2 Scope of this Document 5](#_Toc43478983)

[1.3 Definitions, Acronyms and Abbreviations 5](#_Toc43478984)

[1.4 Overview 6](#_Toc43478985)

[1.5 Business Context 6](#_Toc43478986)

[2 General Description 6](#_Toc43478987)

[2.1 Product Features/Functions 6](#_Toc43478988)

[2.2 User Characteristics 7](#_Toc43478989)

[2.3 General Constraints 7](#_Toc43478990)

[2.4 Collaboration Tools 7](#_Toc43478991)

[2.4.1 Communication 7](#_Toc43478992)

[2.4.2 Document Storage and Control 7](#_Toc43478993)

[2.4.3 File Sharing and Management 7](#_Toc43478994)

[2.4.4 Project Management 8](#_Toc43478995)

[2.4.5 Version Control for your documents 8](#_Toc43478996)

[3 Business Requirements 8](#_Toc43478997)

[3.1 B1 8](#_Toc43478998)

[3.2 B2 8](#_Toc43478999)

[3.3 B3 8](#_Toc43479000)

[3.4 B4 8](#_Toc43479001)

[3.5 B5 8](#_Toc43479002)

[4 Functional Requirements 8](#_Toc43479003)

[4.1 F1 8](#_Toc43479004)

[4.2 F2 8](#_Toc43479005)

[4.3 F3 8](#_Toc43479006)

[4.4 F4 8](#_Toc43479007)

[4.5 F5 8](#_Toc43479008)

[4.6 F6 9](#_Toc43479009)

[4.7 F7 9](#_Toc43479010)

[4.8 F8 9](#_Toc43479011)

[5 Interface Requirements 9](#_Toc43479012)

[5.1 User Interfaces 9](#_Toc43479013)

[5.1.1 GUI 9](#_Toc43479014)

[5.1.2 CLI 9](#_Toc43479015)

[5.1.3 API 9](#_Toc43479016)

[5.2 Hardware Interfaces 9](#_Toc43479017)

[5.3 Communication Interfaces 9](#_Toc43479018)

[5.4 Software Interfaces 9](#_Toc43479019)

[6 Performance Requirements 10](#_Toc43479020)

[6.1 P1 10](#_Toc43479021)

[6.2 P2 10](#_Toc43479022)

[7 Other Non-Functional Attributes 10](#_Toc43479023)

[7.1 Security 10](#_Toc43479024)

[7.1.1 N1 10](#_Toc43479025)

[7.2 Reliability 10](#_Toc43479026)

[7.2.1 N2 10](#_Toc43479027)

[7.3 Maintainability 10](#_Toc43479028)

[7.3.1 N3 10](#_Toc43479029)

[7.3.2 N4 10](#_Toc43479030)

[7.4 Portability 10](#_Toc43479031)

[7.4.1 N5 10](#_Toc43479032)

[7.5 Resource Utilization 10](#_Toc43479033)

[7.5.1 N6 10](#_Toc43479034)

[7.6 Serviceability 10](#_Toc43479035)

[7.6.1 N7 10](#_Toc43479036)

[8 Operational Scenarios 10](#_Toc43479037)

[8.1 Scenario A 10](#_Toc43479038)

[8.2 Scenario B 11](#_Toc43479039)

[8.3 Scenario C 11](#_Toc43479040)

[9 Use Case and User Stories 11](#_Toc43479041)

[9.1 Use Case 11](#_Toc43479042)

[9.1.1 B1 UC01 11](#_Toc43479043)

[9.1.2 B2 UC01 11](#_Toc43479044)

[9.1.3 Use Case Diagram 12](#_Toc43479045)

[9.2 User Stories 12](#_Toc43479046)

[9.2.1 User Stories per Business Requirement 12](#_Toc43479047)

[9.2.2 User Stores per Functional Requirement 13](#_Toc43479048)

[10 Sequence Diagrams 13](#_Toc43479049)

[10.1 Sequence Diagram Of Requirements I1 And P6 13](#_Toc43479050)

[10.2 Sequence Diagram Of Requirement N2 14](#_Toc43479051)

[11 Appendices 14](#_Toc43479052)

[11.1 MOSCOW 14](#_Toc43479053)

[11.1.1 MUST 14](#_Toc43479054)

[11.1.2 SHOULD 15](#_Toc43479055)

[11.1.3 COULD 15](#_Toc43479056)

[11.1.4 WON’T 15](#_Toc43479057)

[11.2 Procedures for Project Requirement Changes 16](#_Toc43479058)

[11.2.1 Example Change 16](#_Toc43479059)

[11.3 Schedule 17](#_Toc43479060)

[11.4 Requirements Traceability Matrix 17](#_Toc43479061)

[11.5 Verification and Validation Testing Document 17](#_Toc43479062)

# Introduction

## Purpose

The purpose of this document is to describe what a tag-based file manager system will do. 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. This product will be used to enhance the search capabilities in the user’s database system.

## Scope of this Document

The intended readers of this document are current and future developers working on “Team Exists As Metonym”, hereafter referred to as the team, and the sponsors of the project. The plan will include, but is not restricted to, a summary of the system functionality, the scope of the project from the perspective of the team, 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

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.

Our tag-based file manager, TagIt.FM, stores the metadata in the form of custom tags in a separate database. Instead of working directly in the limited file manager of an operating system, we will offer a clean user interface that can sort based on any number of user-selected tags. The program searches and opens with the most recent search displayed to ensure continuity and clarity.

## Business Context

In today’s world, providing the best customer experience is as important as getting a new customer. To achieve this goal every business needs to maintain its customer data and to keep records of all customers who interact with the company. Our software can be used by any organization that has a database It can be used to search customer information, user information, medical history, personal data, or even recent searches the employee has performed online. TagIt.FM 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 is stored in the database. Another employee can obtain the same information if he or she wants to access the same customer's medical history using any of the tags associated with the customer's file. 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 sort based on any number of user-selected tags. The program will remember 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 aim at being "invisible", allowing businesses to instantly search, sort, and access important data. Using predictive technologies, the program will search the file based on the tag-name, providing immediate, crucial feedback.

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.

## User Characteristics

The client, MHC, is an insurance company that maintains its data for each policy member in a database. According to HIPAA, each policy holder’s data must be kept confidential. Policy holder data contains personal information like social security number and date of birth. Additionally, the database contains information about a policy holder’s medical history, hospital visits, prescription medications, surgeries, etc. This information gets updated at minimum annually, or as the policy holder medical information changes. The data is stored internally at the organization in a database and policy holders are only allowed to access their own information. As this information would be used by the employees at MHC, the business needs to keep track of all these records.

Most of the policy members call the customer care line with questions regarding their benefits, claims, and billing or with questions regarding specialists who are members of the insurance network. The employee needs 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 be able to access this 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 enquires. 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 operated on Windows or Linux operating system.

2. The maximum size of a database file is 2147483646 pages.

## Collaboration Tools

### Communication

The team has been using Slack and GitHub as the primary collaboration tools, Slack is mainly used for communication. The project manager has created channels based on the project deliverables. Discussions regarding the specific topics related to each deliverable on the respective channel. The main channel is used for discussing general topics and communicating with the team. The meeting channel is used for bi-weekly team meeting.

### Document Storage and Control

All project documents are stored in a GitHub repository. The project manager has given permission to all team members to commit changes to these documents.

### File Sharing and Management

GitHub is used for storing and sharing project related files. The team repository contains all documentation that concerns the project.

### Project Management

GitHub and Slack are for project management. The readme document on GitHub contains the to-do list with current tasks and organized by assigned team roles. All the team members are required to read the list and complete tasks by the due date. Slack is used for bi-weekly meetings, questions, and discussion.

### Version Control for your documents

As progress is made to the project, all the members keep updating their files in the GitHub repository. GitHub allows for reverting documents back to previous versions and to see changes that have been made.

# Business Requirements

## B1

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

## B2

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

## B3

Software must find files when filename, filetype, and/or location are unknown.

## B4

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.

## B5

Customer hold times during peak hours shall be 5 minutes or less.

# Functional Requirements

## F1

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

## F2

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

## F3

Managers shall be able to create additional tags that their group requires.

## F4

Tags shall not allow users to access documents that they do not have permissions to access.

## F5

Administrator logins shall be required to manage the tag database.

## F6

Relevant employee logins will be required to create, update, and delete tags.

## F7

System-assigned tags shall only be deleted by managers, while agent-assigned tags may be deleted by agent.

## F8

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

# Interface Requirements

## User Interfaces

### GUI

#### I1

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

#### I2

Main pane must be able to change target directory.

#### I3

User interface for managing personal tags shall require log in.

#### I4

GUI will include history tab for recently searched and accessed tags/files.

#### I5

Secondary pane must contain an option to allow/disallow searching subdirectories.

### CLI

### API

## Hardware Interfaces

The application will be available on x86 systems.

## Communication Interfaces

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

## Software Interfaces

The application will need to interface with the filesystem of Microsoft Windows and Linux

# Performance Requirements

## P1

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

## P2

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

# Other Non-Functional Attributes

## Security

### N1

The application shall make no changes to any file or file's metadata.

## Reliability

### N2

System should be available 24 x 7 for all users.

## Maintainability

### N3

Database system should perform database maintenance every midnight.

### N4

System should not have any external dependencies.

## Portability

### N5

Software must be able to support several hundred user’s/employee’s ability to create, access, and modify tags simultaneously.

## Resource Utilization

### N6

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

## Serviceability

### N7

Tags must be able to be removed without side effects.

# 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.

## Scenario B

Customer wants to know what his benefits are if he has another knee surgery for his torn ACL.

## Scenario C

Manager wants to pull data of people who have recently ordered prescription for Chloroquine phosphate.

# Use Case and User Stories

## Use Case

### B1 UC01

Title: Customer interaction

Primary Actor: Customer service representative

Success Scenario: Customer service representative receives call from customer. Customer is identified and poses question. Customer service representative locates applicable plan and benefits. Customer service representative provides estimate to customer.

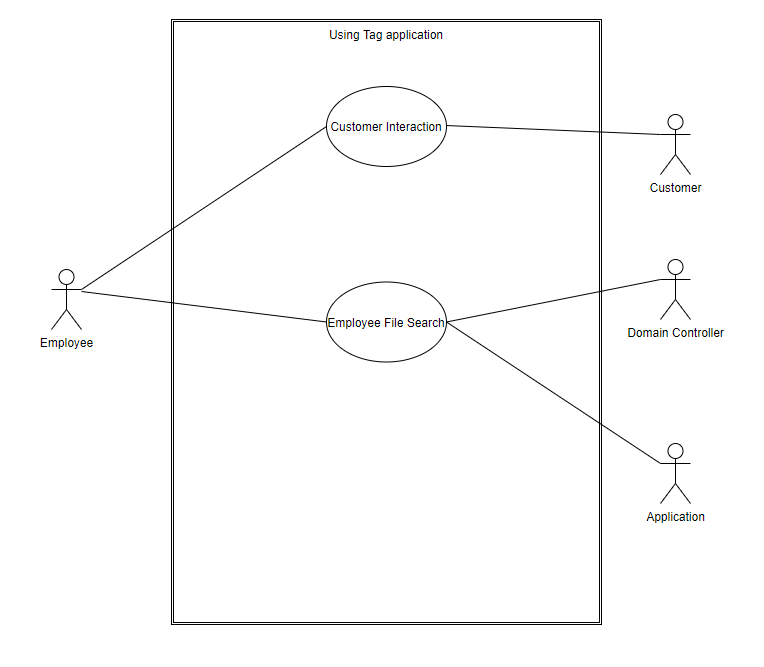
### B2 UC01

Title: Employee file search

Primary Actor: Employee

Success Scenario: Employee logs in to PC. Employee logs in to tag system. Employee starts search. Application suggests results based on history. Application creates list of results based on finalized search. Employee views found records.

### Use Case Diagram



## User Stories

### User Stories per Business Requirement

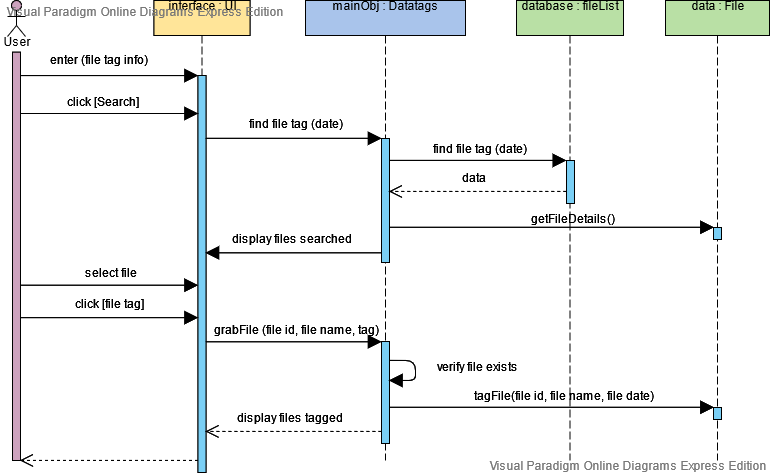
* As a customer care representative, I need quick access to policy information, benefits and estimates to share with the customer. (B1 US01)
* As a customer care representative, I want to access all case related documents while on the phone with the customer. (B2 US01)
* As a manager, I want my employees to be able to find training documents and report templates when they need them. (B3 US01)
* As legal counsel, I need to be able to audit the software to ensure confidentiality of all customer records. (B4 US01)
* As a customer care representative, I want all case related documents to be easy to organize and view. (B5 US01)

### User Stores per Functional Requirement

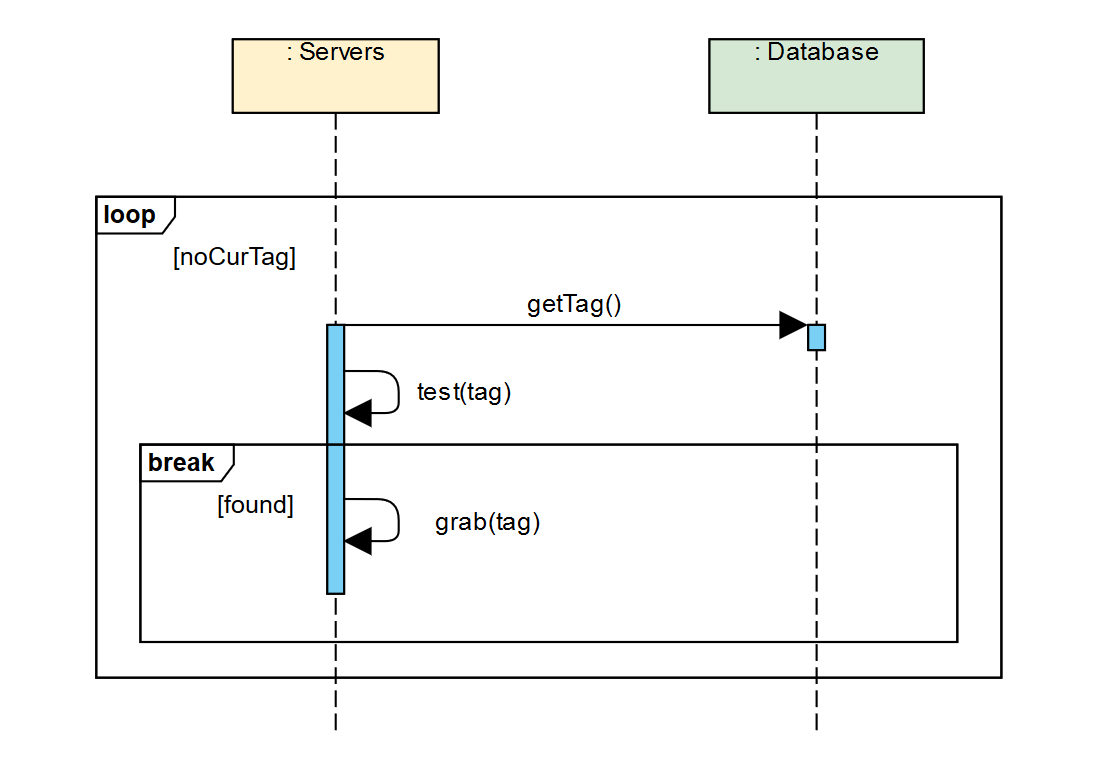
* As an administrator, I need to be able to log in to the tag database to perform routine maintenance. (F2 US01)
* As a manager, I need to be able to log in to the tag database and create new tags, delete old tags, and edit existing tags for clarity and content. (F3 US01)
* As a manager, I need to create specialized tags for my employees to ensure they can find all documents that they need. (F6 US01)
* As legal counsel, I need to be able to audit employee permissions to ensure nobody has received access past their clearance level. (F7 US01)
* As a customer care representative, I want to be able to sort found files based on date created and claim type. (F8 US01)

# Sequence Diagrams

## Sequence Diagram Of Requirements I1 And P6



## Sequence Diagram Of Requirement N2



# Appendices

## MOSCOW

### MUST

* MHC wants to maintain its quality of service by providing best customer care in a timely manner.
* MHC wants to maintain all its customer's data so that it can be searched efficiently and effectively.
* Software must find files when filename, filetype, and/or location are unknown.
* 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.
* Customer care representatives shall be able to retrieve the policy holder information using a keyword from their database.
* Insurance agents shall be able to log in to the system and find related documents by using keywords associated with the client or project.
* Managers shall be able to create additional tags that their group requires.
* Tags shall not allow users to access documents that they do not have permissions to access.
* GUI will include a centralized (easily spotted) search bar for the searching of tags.
* Main pane must be able to change target directory.
* Software will be able to pull and display files based on search criteria in a reasonable amount of time: less than 5 seconds.
* Software will be relatively lightweight and not affect other business-necessary operations that must also be performed.
* The application shall make no changes to any file or file's metadata.
* Tags must be able to be removed without side effects.

### SHOULD

* Customer hold-times during peak hours shall be 5 minutes or less.
* Administrator login shall be required to manage the tag database.
* Relevant employee login will be required to create, update, and delete tags.
* System assigned tags shall only be deleted by managers, while agent assigned tags may be deleted by agent.
* Files that have been returned from a tag search will have additional methods available to sort based on filetype.
* User interface for managing personal tags shall require log in.
* GUI will include history tab for recently searched and accessed tags/files.
* Secondary pane must contain an option to allow/disallow searching subdirectories.
* System should be available 24 x 7 for all users.
* Software must be able to support several hundred user’s/employee’s ability to create, access, and modify tags simultaneously.
* Software must keep all tags up to date in real time as they are created, updated, and/or deleted.
* Database system should perform database maintenance every midnight.
* System should not have any external dependencies.

### COULD

* System shall always cross-reference customer ID before assigning customer specific tags to any documents.
* Customers shall not be able to access the tag system.
* Windows login shall be recognized when adding tags to shared files or folders.
* Start searching within 2 seconds after a tag match is guessed but the user is still typing.
* Customer care representative shall be able to retrieve all the case history by searching caseId or tag number.
* The application shall produce regular reports on most popular tags and searches.
* Tags shall stay assigned to document if moved to a different project or directory.
* Files that have been found during search will support drag and drop.
* User interface for adding personal tags shall be on mouse-right-click drop down.
* User interface shall offer options for personalized color schemes for tags.

### WON’T

* All customer correspondence shall be compiled and mailed out the same business day as customer request was made.
* When a case is closed, all relevant documents shall be transferred to customer file within one business day.
* Software shall consider HIPAA regulations and all data and transfers of data shall be completely secure.
* System shall maintain a user portal for policy members using customer specific tags.
* System shall create client and project specific tags when a new client or project is created.
* The application must be able to support users effectively organizing files.
* To ensure confidentiality, if a customer tag is present on a document, a document move must be approved by a manager or group leader.
* Project relevant tags shall be assigned to any document that is added to a tagged project folder.
* 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

Status: Open

Requirement ID: [ID of requirement (e.g., 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

## Requirements Traceability Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement ID | Applicable Role | Testcase Description | Status | Tested By |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Verification and Validation Testing Document