Contents

[Introduction 2](#_Toc498767852)

[Requirements Analysis 3](#_Toc498767853)

[Input Messages 3](#_Toc498767854)

[Categorise Messages 3](#_Toc498767855)

[Validate Messages 3](#_Toc498767856)

[Sanitise Messages 3](#_Toc498767857)

[Save Message 4](#_Toc498767858)

[System Design 5](#_Toc498767859)

[Activity 5](#_Toc498767860)

[Sequence 5](#_Toc498767861)

[Class 5](#_Toc498767862)

[Implementation 6](#_Toc498767863)

[Testing 7](#_Toc498767864)

[Objective and Scope 7](#_Toc498767865)

[Test Item 7](#_Toc498767866)

[Test Deliverables 7](#_Toc498767867)

[Features to be tested 7](#_Toc498767868)

[Testing Methods 7](#_Toc498767869)

[Environmental Needs 8](#_Toc498767870)

[Test Schedule 8](#_Toc498767871)

[Test cases and construct test in vs (screen shots) 8](#_Toc498767872)

[Evolution 9](#_Toc498767873)

[Appendices 10](#_Toc498767874)

[Appendix A 10](#_Toc498767875)

[Appendix B 11](#_Toc498767876)

[Appendix C 12](#_Toc498767877)

# Introduction

# Requirements Analysis

From the project brief, user stories were created which helped to identify the key features that the application would have to provide. These features were then expanded upon to highlight the interactions between an actor (the user) and the system (the application) in a Use case diagram, which can be found in Appendix A. From the Use case diagram, I was able to ascertain the main functional and non-functional requirements that the application is expected to meet.

## Input Messages

Input messages is the primary use case as all the other use cases rely on there having been a message input into the system. It has a non-functional requirement of Ease of use, this meaning that the application should be intuitive, easy to follow and understand what can be done on each page. Input messages is a *generalisation* of both Input manually and Input from text file. The application is to allow for both methods of inputting messages, this allows the user the control over whether they only need to input a small amount manually or input a larger batch from file.

## Categorise Messages

Input messages *includes* the use case Categorise messages as when the user clicks the convert button to input the message into the application, the application categorises the message based on the first character in the message header.

## Validate Messages

Categorise message *includes* the use case Validate message as once the application has identified which of the 3 possible messages has been input by the user, the application validates that the message header is in the correct format then splits the main body text and then validates the individual parts.

An example of this would be a message input by the user with the first character of the message header being ‘E’ from this the application identifies the message as an email. This allows the application to break the main body text into different part from what it would have if the message was a Tweet or SMS.

## Sanitise Messages

Validate message *includes* the use case Sanitise message as once the main body of text has passed its validation there is specific information that needs to be extracted and handled appropriately. This differs between the 3 messages types in the following ways:

A SMS main body of text is checked for any text speak abbreviations (contained on a supplied .csv file) expanded to include its full form before saving.

An Email can be split into 2 types:

* Standard email comes with a subject that is 20 characters long exactly.
* Significant incident report email has a subject that is in the format ‘SIR dd/mm/yy” with the body of the text formatted with the sport centre code on the top line and the nature of incident on the second line.

Both emails are checked for any URL’s, if any are found then the application will remove the URL and replace the text with ‘<URL Quarantined>’ with the URL being added to a Quarantined list.

A Tweet’s main body of text is checked for any text speak abbreviations in the same way as the SMS in addition to that the tweets are checked for any mentions or hashtags. If either are found, they are recorded and displayed in their separate lists on the menu page, with the list ordered by the number of times each is found.

## Save Message

The use case Save message *extends* from Sanitise message as the message must have been through the process of being categorised, then validated then sanitised successfully before the user has the option to save the message to a Json file.

Save messages has the non-functional requirement of Response Time when the user saves the message, at this stage of the process we don’t want the application to feel like it is unresponsive or ‘hanging’ while the user waits therefore the response time is key to improving the user experience with the application.

# System Design

## Activity

## Sequence

## Class

# Implementation

C# wpf

# Testing

A Top-Down testing strategy is used for testing this application. This is done incrementally where the higher levels of the system are tested first, such as the navigation buttons which allow the user to either move between pages or exit the application.

To test this application, defect testing will be used to attempt to simulate as many potential outcomes as possible with the intent on finding any bug or logic errors. Both unit and integration testing will be carried throughout the development phase.

## Objective and Scope

The objective of testing this application is to ensure that the software requirements that were obtained from the Requirement Specification Use Case diagram, are all implemented and complete.

The scope of the testing is to cover all functional and non-functional requirements.

## Test Item

Euston Leisure Message Filtering Service Version 1.0 is the single test item that will be tested.

## Test Deliverables

For this application the test deliverables will be this test plan document, the test strategy, the test cases and test logs.

## Features to be tested

* Input Messages manually
* Input Messages from file
* Save Messages
* View Stored Messages
* Export Json File
* GUI buttons
* Trending List
* Mentions List
* SIR list

## Testing Methods

For the testing of this application we be using both Black box (functional) and White box (structural) approaches to testing software.

The Black box testing approach will be used when testing Event Validation and Page Navigation.

The White box testing approach will be used to test the input validation, where multiple test paths and boundaries need tested.

## Environmental Needs

To complete the testing of this application, a PC running Windows 10 and Visual Studio 2017 was used. The minimum requirements to run Windows 10 are:

* Processor: 1 gigahertz (GHz) or faster processor or SoC
* RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit
* Hard disk space: 16 GB for 32-bit OS 20 GB for 64-bit OS
* Graphics card: DirectX 9 or later with WDDM 1.0 driver

Microsoft Word was used to record all data in the Test Logs and all testing was done by myself.

## Test Schedule

## Test cases and construct test in vs (screen shots)

# Evolution

Have the app connect to a central server, means that all the sports centres can input the info from local site

# Appendices

## Appendix A



## Appendix B

## Appendix C