ELMFS – SET09102

JUAN ALVAREZ

Alvarez, Juan
04340725 | EDINBURGH NAPIER UNIVERSITY

Table of Contents

Document Log 2
1 Introduction 3
1.1 Purpose 3
1.2 Scope 3
1.3 Definitions, acronyms and abbreviations 3
1.4 References 3
1.5 Overview 4
2 Overall description 4
2.1 Product perspective 4
2.2 Product Functions 4
2.2.1 SMS messages 4
2.2.2 Email messages 5
2.2.3 Tweet messages 5
2.2.4 File type 6
2.3 User Characteristics 6
2.4 Constrains 6
2.5 Assumptions and dependencies. 6
2.6 Apportioning of requirements 6
2.7 User documentation 6
3 Specific requirements 7
3.1 External interface requirements 7
3.1.1 User interface 7
3.1.2 Hardware interfaces 8
3.1.3 Software interfaces 8
3.2 Functional requirements 8
3.2.1 Functional requirement 1.1 8
3.2.2 Functional requirement 1.2 8
3.2.3 Functional requirement 1.3 8
3.2.4 Functional requirement 1.4 9
3.2.5 Functional requirement 1.5 9
3.2.6 Functional requirement 1.6 9
3.2.7 Functional requirement 1.7 10
3.2.8 Functional requirement 1.8 10
3.2.9 Functional requirement 1.9 10
3.3 Event validation 10
3.3.1 Process Text Boxes button 10
3.3.2 Previous button 11
3.3.3 Next button 11
3.3.4 Process File button 11
4 Non-Functional requirements 12
4.1.1 Easy to use layout 12
4.1.2 Response time 12

4.2 Design constrains **12**

4.3 Software system attributes **12**

4.3.1 Reliability 12

4.3.2 Availability 12

4.3.3 Maintainability 13

5 Diagrams ***13***

5.1 Use case diagram **13**

5.2 Class Diagram **14**

6 Test plan ***14***

6.1 Objectives **14**

6.2 Scope **14**

6.3 Testing Strategy and levels **15**

6.3.1 Automated unit testing 15

6.3.2 Integration testing 15

6.3.3 System testing 15

6.3.4 Inspection or desk checking 15

6.3.5 Acceptance testing 15

6.4 Environmental needs **15**

6.5 Staffing and training **15**

6.6 Risk **15**

6.7 Test Schedule **16**

6.8 Features to be tested **16**

6.8.1 User interface items 16

6.9 Navigation Test Log **17**

6.10 Event Validation Test Log **18**

6.11 Data Validation Test Log **22**

6.12 Automated Testing **25**

7 Version control plan ***26***

8 Evolution strategy ***27***

Bibliography ***29***

Document Log

Version	Date	Summary of Changes	Author
1	25/11/2019	Original version	Juan Alvarez
1.1	28/11/2019	Added Use and class diagrams	Juan Alvarez
1.2	01/12/2019	Added Test Plan	Juan Alvarez
1.3	02/12/2019	Updated document to include “Additional Requirements” from document SET9102 Coursework 19-20	Juan Alvarez
1.4	02/12/2019	Added Test Logs	Juan Alvarez

1 Introduction
This section will give an overview of what will be included in the document, it will describe the purpose of the document and will include abbreviations and definitions if it is necessary.

1.1 Purpose
The aim of the document is to provide a description of the requirements of the Euston Leisure Messaging (ELM) software showing the porpoise and a complete description of the system that we aim to create. It will as well show the interaction of users with the software.

1.2 Scope
Euston been a medium sized city requires a software that will be able to control all the messaging capabilities of their sport centres in the city. In the ELM software it will receive all incoming messages and will validate them and separate them by categories (SMS, Emails and Tweets). Each category needs to be handled differently and we will cover that in another section of this report.

The software will as well have to provide some statistics of the messages received.

Originally there is no need to store the data permanently, as the statistics needed in the client requirements are per session. But there is scope for future evolution that we will cover later on in this report.

1.3 Definitions, acronyms and abbreviations
Table 1 – Definitions

Term	**Definition**
User	Someone who uses the software
DESC	Description
RAT	Rational
DATAVAL	Data Validation, the type of data validation implemented
TAG	Unique identifier
PLAN	The level in which we can claim good success has been achieved
WISH	Desirable level of achievement that may not be attainable
MUST	Minimum level required to avoid failure
DEP	Dependency
GIST	Short description of the concept contained in the statement
SCALE	Scale of measures used by the requirement contained in the statement

1.4 References
This document references the project brief provided by the client (SET09102 Coursework 19-20) and the diagrams and files included on the report or mentioned at the end on the appendix.

1.5 Overview
The rest of the report will include a class diagram illustrating the classes require to perform the needed operations, a description of the user interface and requirement specifications as well as a test plan.

It will then go on to specify how the version control is intended to be done and will propose a strategy to evolve the program to be of better use in future.

2 Overall description
We will give an overview of the whole system in this section, including all functionalities and specifying the needs of each component.

2.1 Product perspective
Euston Leisure Messaging is a software that will accept input of different types of messages and will deal with them independently.

It will allow users to input messages using a user interface and after processing will show the result in a Results panel.

At the end of the session it will provide the possibility to output some statistics and as well see the messages introduced during the session.

The messages will be saved in Json file/s (Microsoft, 2019) for future reference, for that after some research I decided to use Newtonsoft json as it is widely available and very much used in industry, so it was easy to resolve my doubts regarding its use. (Newtonsoft, 2019)

2.2 Product Functions
The user will input data into the text boxes provided and the system will deal with the data differently depending what type of message has been introduced.

The message header will include a letter ("S", "E" or "T") followed by 9 numeric characters. Depending on the content of the header the system will deal with the message body differently, as specified as follows.

2.2.1 SMS messages
When the message header starts with "S" it will indicate that the message is a SMS and the system will deal with it as follows.

The message body will have a Sender that will be an international phone number, then there will be a message text with a maximum of 140 characters.

In the message text there could be some textspeak abbreviations, when that happens the program has to deal with it by expanding it to the full form. For example, if the message contains "ASAP" it will expand it to "ASAP < As Soon As Possible >". (Dasblinkenlight, 2013) The whole list of abbreviations is provided in a separate text file called "textwords.csv". (QuackWare, 2009)

2.2.2 Email messages

When the message header starts with “E” it will indicate that the message is an Email and the system will deal with it as follows.

The message body will have a Sender in the form of a standard email address.
Then a subject will follow with 20 characters.
Finally, a message text of a maximum of 1028 characters will follow.
The message text may contain embedded hyperlinks in the form of standard URLs, those URLs will have to be removed and replaced by “<URL Quarantined>” and the removed URL will be added to a list.

There can be 2 types of emails, Standard email messages and Significant Incident Reports. Significant Incident Reports emails will have the Subject in the form or “SIR dd/mm/yy” and a message body that will begin as follows:

Sport Centre Code: 66-666-99

Nature of Incident: Which will be one of the following:

Theft of Properties
Staff Attack
Device Damage
Raid
Customer Attack
Staff Abuse
Bomb Thread
Terrorism
Suspicious Incident
Sport Injury
Personal Info Leak

The system will have to create a SIR list with the Centre Code and the Nature of Incident that latter it will be able to output into the Panel provided for it and a json file.

2.2.3 Tweet messages

When the message header starts with “T” it will indicate that the message is a Tweet and the system will deal with it as follows.

The message body will include a Sender in the form of a Twitter ID starting with “@” and following with maximum 15 characters and then the Tweet text with a maximum of 140 characters.

Tweets may contain textspeak abbreviations, when that happens the program has to deal with it by expanding it to the full form. If the message contains “ASAP” it will expand it to “ASAP < As Soon As Possible >”. The whole list of abbreviations is provided in a separate text file called “textwords.csv”.

Tweets may contain hashtags, that is a group of characters following a “#”, when they appear the system will add them to a list and will count the number of uses of each to produce a trending list.

Tweets may contain embedded Twitter IDs, starting with “@” and followed with a maximum of 15 characters. When that happens, the system will add the occurrence to a Mentions list.

2.2.4 File type

The file the application can upload and work with is a txt file where messages occupy a line each.
In each line there will be a Message ID followed by a Message Body, they will be separated by an asterisk (*), this is necessary for the application to be able to process the file.

2.3 User Characteristics

The client didn’t request any specific user or any type of user validation. When a user has access to the computer where the system will be installed s/he will be able to use ELM freely.

2.4 Constrains

The limit of the system is the hardware and software where it will have to be installed. The system has to be a reasonable spec computer with Windows 10 installed and the Net framework available to be able to run the .exe file.

2.5 Assumptions and dependencies.

It is assumed that the client’s offices have computers available of enough specification to be able to run the software. If it is not the case it is recommended that a new computer system is acquired before the installation date.

2.6 Apportioning of requirements

At this moment in time it is expected that the required specifications will be delivered on time in the first release of the product. There are some future recommendations that we will cover in the section of Evolution strategy, but those are not provided at this time and would have to be costed and agreed at a future iteration once the original product specifications are covered and delivered. Those could be considered for a second release of the product since the system will be developed with scalability scope.

2.7 User documentation

The client has not requested a user manual, but if required we can agree some training sessions with the client’s employees as well as the possibility to provide user manual for ELM.

3 Specific requirements

In this section we will cover all the functional and quality requisites of the application.

3.1 External interface requirements

In this section the inputs of the user and outputs of the application will be described, a prototype of the user interface will be provided. Please note that in implementation some things may change in future iterations. As we are dealing with an Agile approach, we are open to possible changes between iterations.

3.1.1 User interface

A user will be able to input text into the "Message ID" textbox and the "Message Body" textbox. Then pressing the "Process Text Boxes" button will make the program work and the application will provide a result in the "Result" panel.

The "Explore" button allows to explore the computer and search for a file, the path of the file will be added to the textbox on its left. The user will be able to press the button "Process File" for the application to process it according to the specifications on section 2.2.
To be read the file, the file will be formatted by having each message in a single line, the Message ID will be first followed by the MessageBody separated by an "*". If the text file is not formatted that way the application will not be able to read it properly.
The Buttons "<-" and "->" are there to navigate between the messages in the file. When the user finds a message that needs to be processed, the user will have to press the "Process Text Boxes" button.

Pressing the button "Output" will output the statistics of trending list, list of Mentions and SIR list and save them to a file.
Pressing the "Clear" button will clear the Results text box.

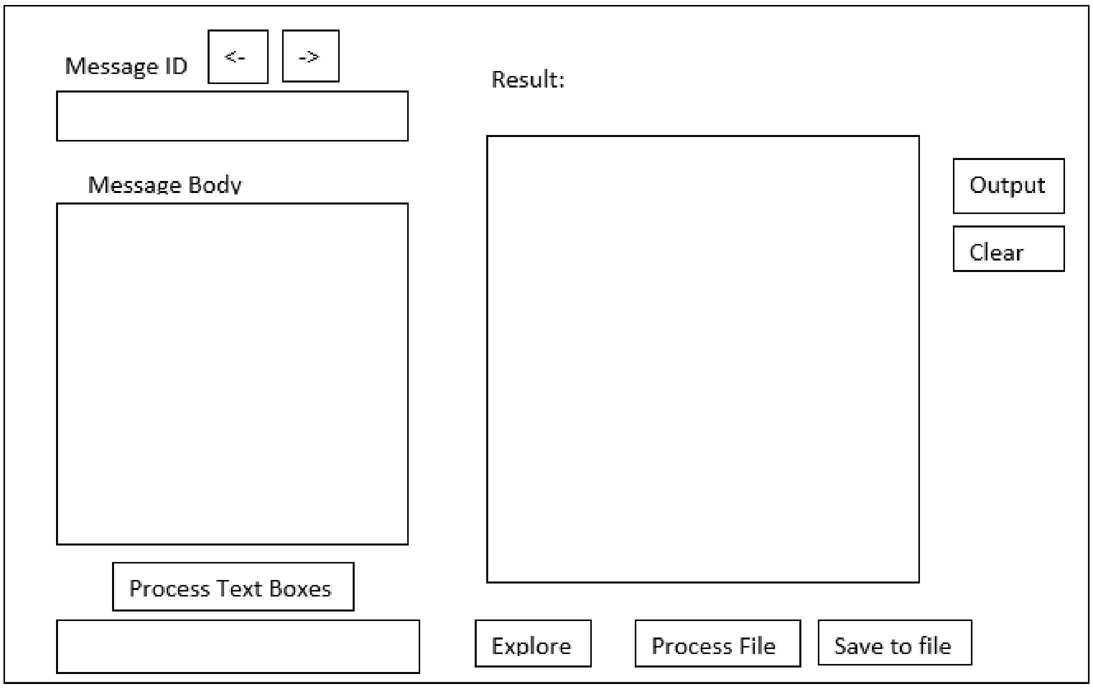

Fig. 1 User interface of ELM application.

3.1.2 Hardware interfaces

The computer where the ELM application will run doesn't need to be of high spec, but it is recommended that the highest it is the faster the program will run. It will need to have reasonable amount of hard drive space available to the application since every time the application runs it will produce and save some files into the computer's hard drive.

3.1.3 Software interfaces

The ELM application will have to communicate with the operating system in order to be able to access or explore the system folders as well as saving data in json files into the system's folders. The required operating system is Windows 10 with the Net framework installed on the computer.

3.2 Functional requirements

This area will cover the different actions the users can perform in the ELM application.

3.2.1 Functional requirement 1.1

ID: ELM1
TITLE: Adding Message ID
DESC: A user shall be able to enter a Message ID comprised of a letter and 9 numbers that will be processed when the button "Process Text Boxes" is pressed.
DATAVAL: Data has to be a letter (either a "S" an "E" or a "T") followed by 9 numbers. The message Id text box does not allow inputs over 10 digits, if the user enters incorrect data format the system will prompt the user to correct it before proceeding once the button "Process Text Boxes" is pressed. (Microsoft, 2019)
RAT: In order for a user to be able to input Message Id.
DEP: None

3.2.2 Functional requirement 1.2

ID: ELM2
TITLE: Adding Message Body
DESC: A user shall be able to enter a Message Body
DATAVAL: Depending if the message is a SMS, an Email or a Tweet the system will deal with it accordingly as per section 2.2 of this report. The processing will be executed once the "Process Text Boxes" is pressed. In the case of SMS and Tweets text speak abbreviations will be handled as specified in sections 2.2.1 and 2.2.3, In the case of Tweets a list of Mentions and a Trending list will be produced as per section 2.2.3 of this report. In the case of Emails a SIR list will be produced as per 2.2.2 of this report.
RAT: In order for a user to add a Message Body
DEP: ELM1

3.2.3 Functional requirement 1.3

ID: ELM3
TITLE: Pressing "Process Text Boxes" button
DESC: A user shall be able to press the "Process Text Boxes" button, the message could be entered manually or through a file. Each message has to be processed separately.

DATAVAL: Upon pressing the button the text boxes “Message ID” and “Message Body” will be process and validated as per ELM1 and ELM2. If either of them is empty, the user will be prompted to fill it in correctly. If the data is incorrect the user will be prompted to correct it. (Rexegg, 2015)
RAT: In order for the user to be able to process data in “Message ID” and “Message body” textboxes. SMS and Tweets will have to contain no more than 140 characters and Emails no more than 1028, a pop up message will inform when trying to process messages that are too long.
DEP: ELM1, ELM2 and ELM5

3.2.4 Functional requirement 1.4

ID: ELM4
TITLE: Pressing “Explore” button
DESC: A user shall be able to explore the computer’s hard drive using the “Explore” button in order to be able to find a specific file to process. Once it is found the path will be added to the textbox at the left of the button. (QuackWare, 2009)
DATAVAL: As the finding of the file will be done exploring the computer through and window there is no data validation needed.
RAT: In order for the user to find a file for the ELM application to process.
DEP: None

3.2.5 Functional requirement 1.5

ID: ELM5
TITLE: Pressing “Process File” button
DESC: A user shall be able to get the application to process a file by pressing the “Process File” button after finding such a file with the “Explore” button.
DATAVAL: The system will ensure the file is in the correct format to be able to process it or it will not be able to do so and would inform the user. Files must be text files divided into messages separated in lines. Each line will contain a Message ID followed by an “*” and then the Message Body.
RAT: In order to load messages in groups rather than one by one.
DEP: ELM4

3.2.6 Functional requirement 1.6

ID: ELM6
TITLE: Pressing “Output” button
DESC: A user shall be able to display the list of SIRs, Mentions and trending by pressing the button “Output”. A copy of those will be automatically saved into a text file called “MentionsHashtagsadnEmails.txt” in the computer. The file will be save on the folder “ELMLogs” of the root directory, if that folder doesn’t exist it will be created. (Microsoft, 2019)
DATAVAL: There is no need of data validation in this case as the data is already properly stored in the system and has been validated previously. Data has to be stored in the application already for this button to work or there would not be any results displayed or saved.
RAT: In order for a user to be able to view Trending list, list of Mentions and SIR list.
DEP: ELM1, ELM2, ELM3, ELM4 and ELM5

3.2.7 Functional requirement 1.7
ID: ELM7
TITLE: Pressing “Previous” (<-) button
DESC: When processing message from a file a user will be able to navigate to the previous message from the file using this button.
DATAVAL: This button will just show messages on the text boxes Message ID and Message Body, before sowing the previous content of the text boxes will be cleared. When the first message has been shown and there are no more messages a pop-up window will indicate that there are no more messages.
RAT: In order for a user to be able to view previous message from a file of messages.
DEP: ELM4, ELM5

3.2.8 Functional requirement 1.8
ID: ELM8
TITLE: Pressing “Next” (->) button
DESC: When processing messages from a file a user will be able to navigate to the next message from the file using this button.
DATAVAL: This button will just show messages on the text boxes Message ID and Message Body, before sowing the next content of the text boxes will be cleared. When the last message has been shown and there are no more messages a pop-up window will indicate that there are no more messages.
RAT: In order for a user to be able to view next message from a file of messages.
DEP: ELM4, ELM5

3.2.9 Functional requirement 1.9
ID: ELM9
TITLE: Pressing Clear button
DESC: When pressing the clear button, the field “Result:” will be cleared.
DATAVAL: There is no need for data validation as the content of the field will be cleared.
RAT: In order for a user to be able to clear the field “Result:”.
DEP: None

3.3 Event validation
3.3.1 Process Text Boxes button
ID: EV1
TITLE: Process Text Boxes button
WHERE: ELM application’s main screen.
DESC: Verifies that the data in “Message ID” and “Message Body” text boxes is of the correct format and will throw and error message if it is not. The user will be able to acknowledge the error message and then go back to the application in order to correct it.
Message ID text box cannot be longer than 10 characters, the first character must be either “S”, “E” or “T” and the rest numbers.
If the message is an SMS, the application will make sure there is an 11 digits telephone number at the beginning of the Message Body, the rest of the message could be empty as it is possible to send empty messages and the application should be able to process those.

If the message is a Tweet, the application will make sure there is a tweeter ID (@...) at the beginning of the message body or the message would not be processed. The rest of the message could be empty or have content, as the system has to accept the possibility of receiving empty messages.
If the message is an Email, the application will make sure there is an email address at the beginning of the message, or the message would not be processed, and a pop-up message will indicate the user to introduce a sender's email address. The application will ensure there is a subject, it could be an empty subject as some emails are sent without them, but in that case the subject has to include 20 spaces. If there is no subject a pop-up message will prompt the user to include one. The content of the email could be empty, as the application needs to have the ability to process empty emails if any is received.
RAT: To ensure the data is entered in the correct format.
DEP: None

3.3.2 Previous button
ID: EV2
TITLE: Previous button
WHERE: ELM application's main screen.
DESC: Verifies if there is a previous message and if there are no more messages it will indicate so in a pop-up message. Before loading any messages, it will clear the text boxes Message ID and Message Body.
RAT: To ensure messages are loaded correctly.
DEP: None

3.3.3 Next button
ID: EV3
TITLE: Next button
WHERE: ELM application's main screen.
DESC: Verifies if there is a next message on the text file and if there are no more messages it will indicate so in a pop-up message. Before loading any messages will clear the text boxes Message ID and Message Body.
RAT: To ensure messages are loaded correctly.
DEP: None

3.3.4 Process File button
ID: EV4
TITLE: Process File button
WHERE: ELM application's main screen.
DESC: Where if the file is the correct file type, if it isn't it indicates in a pop-up message the structure of file that is required for the program to work (txt file with messages separated by lines and message id and message body separated by *).
RAT: To ensure that program doesn't crush and loads the correct type o file.
DEP: None

4 Non-Functional requirements

In this chapter we cover the requirements of usage of the system, how responsive it should be and what the user can expect from the software.

4.1.1 Easy to use layout

ID: PR1
TITLE: Easy to use layout
DESC: The input and resulted information should be easy to use and read, all in the same page so the user doesn't have to suffer page's loading time and delays.
RAT: In order for the user to have an easy experience with ELM application.
DEP: None

4.1.2 Response time

ID: PR2
TITLE: Response time
DESC: The processing time of the data should be adequate for daily usage.
MUST: Each processing of data cannot take longer than 5 seconds.
WISH: No more than 1 second.

4.2 Design constrains

The only possible limitations come from the hardware/software where the ELM application is going to be installed. There is a need to ensure that enough hard drive space is available, but as the files stored are going to be json files that don't take much space a recommendation of 100 MB of initial availability is more than sufficient, this availability would make the program future proof as is a lot more than it requires originally.

4.3 Software system attributes

In this section we will cover the desired availability and maintainability of the ELM application.

4.3.1 Reliability

ID: PR3
TAG: SystemReliability
GIST: The reliability of the system
SCALE: The reliability that the system processes the text correctly each time
PLAN: In 100% of the processes
WISH: 100% of the processes

4.3.2 Availability

ID: PR4
TAG: SystemAvailability
GIST: The availability of the system when it is been used
SCALE: The average system availability
PLAN: In 99% of the time
WISH: 100% of the time

4.3.3 Maintainability

ID: PR5
TITLE: System extendibility
DESC: The application should be easy to modify and extend. The code should be easy to read and maintain and it should be easy to implement new features in future
RAT: In order for future implementations of different features
DEP: None

ID: PR6
TITLE: System testability
DESC: Testing should be easy to carry in order to be able to find bugs and fix them
RAT: To be able to test ELM application
DEP: None

5 Diagrams

5.1 Use case diagram

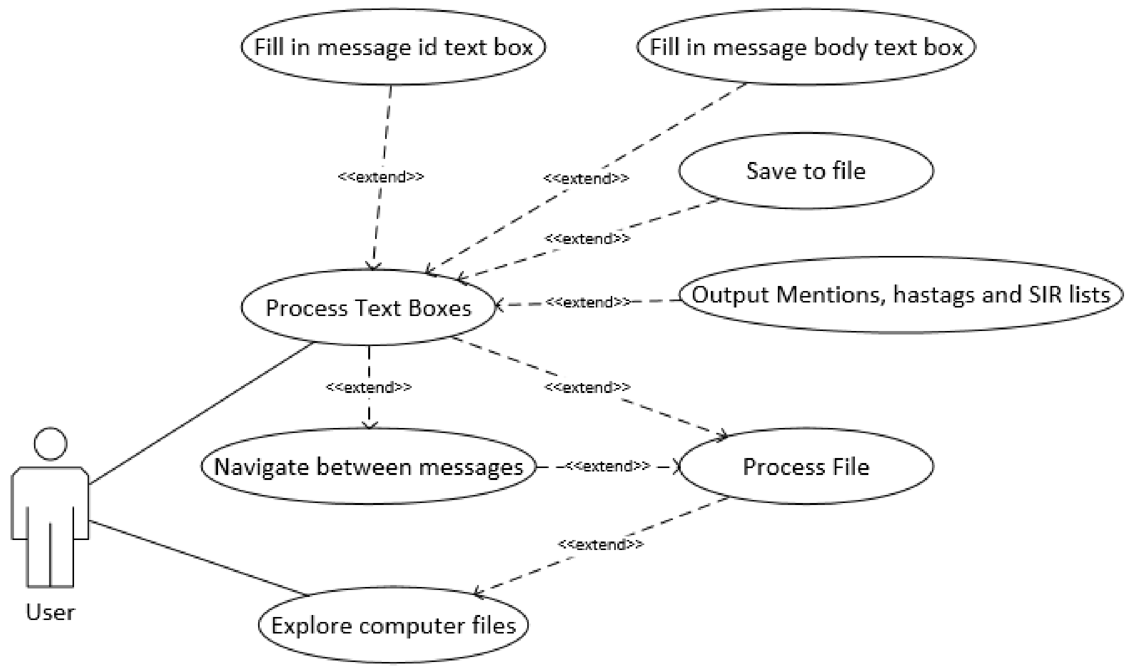
Fill in message id text box
Fill in message body text box
Save to file
Output Mentions, hastags and SIR lists
Process Text Boxes
Navigate between messages
Process File
Explore computer files
User
<<extend>>

Fig. 2 Use case diagram for EML application

5.2 Class Diagram

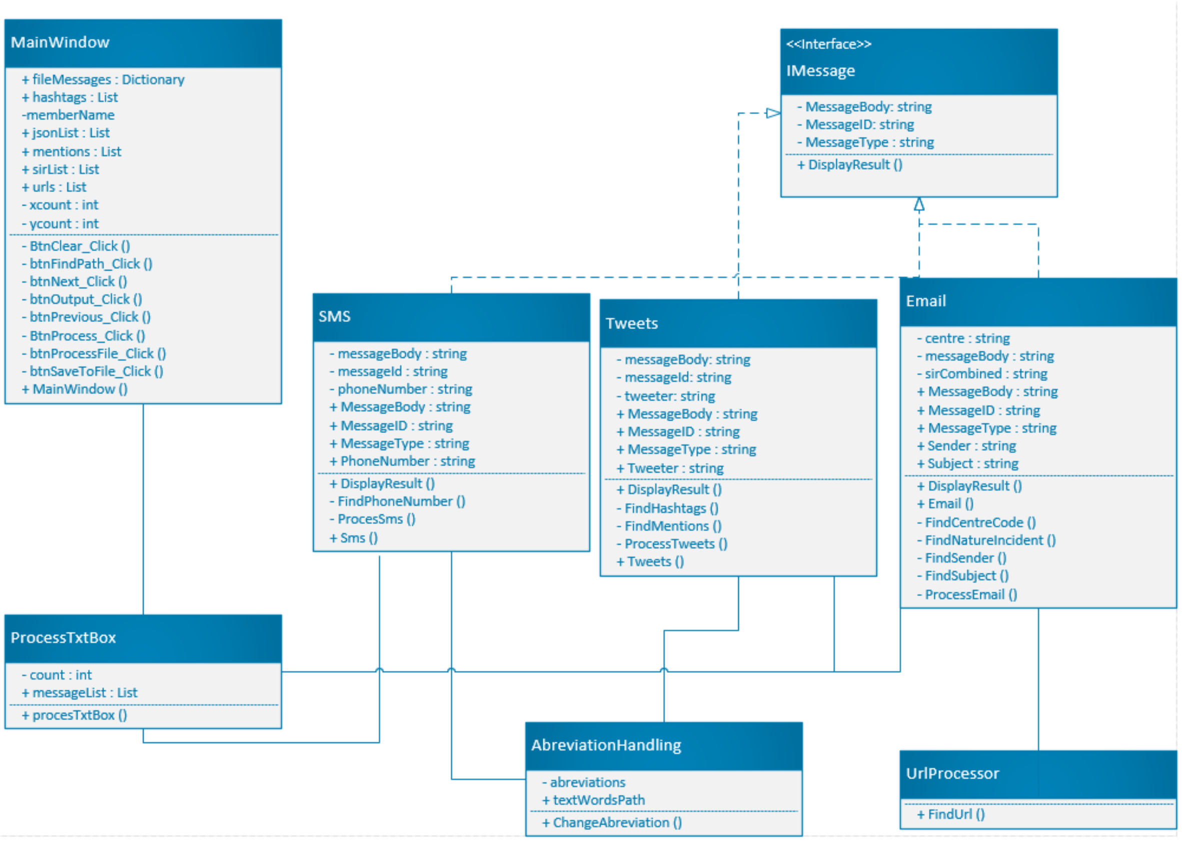

Fig. 3 Class Diagram for ELM application

6 Test plan

ELM application is conceived to deal with the messages that arrived to the Euston Leisure Centres by form of SMS, Email or Tweets. This section of the document will cover the testing strategies and methods use to ensure the software is up to standard.

6.1 Objectives

The objectives of this tests is to eradicate any bugs the software may encounter. When the tests are done and documented the errors found, if any, will be corrected before deploying the prototype in the client’s premises.

6.2 Scope

There will be an inspection of the code in a white box scenario, if any issues are found they will be fixed even before the next set of testing is done.
There will be automated unit testing to ensure that the key components of the program work as expected.
How easy is the interaction between the user and the application will be tested in a black box scenario.
By the end of the testing a fully working application will be produced.

6.3 Testing Strategy and levels

This are the types of testing that I will perform in this plan:

6.3.1 Automated unit testing

In this case a partial bottom up strategy was used, as the testing started while in the programing phase, as some unit testing was done in a white box scenario while the program was still in development.

6.3.2 Integration testing

The next step will be an integration testing to ensure all components work as expected with each other.

6.3.3 System testing

In a black box scenario, the Validation or System testing will be performed, where the function of the system as a whole is thoroughly tested.
Each interaction the user has with the application needs to be tested to ensure the program works as expected.

6.3.4 Inspection or desk checking

Finally, an inspection or desk check will be performed to ensure that all the code is correct before producing it for the client.

6.3.5 Acceptance testing

Eventually the client will be prompted to do an acceptance testing of their own to ensure they are satisfied with the product.

6.4 Environmental needs

This is the needed software to carry on the testing project:

Hardware	**Software**
PC	Visual Studio
PC	Microsoft Word
PC	Windows 10 with NET framework

6.5 Staffing and training

This is a simple application, the training for the Leisure Centre's employees should take no longer than 2 hours. There will not be a need for extra staff as I can perform the training myself. A user guide was not requested, but one could be provided if necessary if the resources are made available for it.

6.6 Risk

The only risk I foresee is not having enough time to finish on time the whole project and testing. If that is the case some over time may be required. Another risk may be lack of knowledge of some tools I need to use. For this case I will have to make do with what I have but I need to identify that lack of skills and address it for future projects.

6.7 Test Schedule

This testing will show the needed time for each testing task.

Task Name	Start	Finish	Comments
Automated unit testing	25/11/2019	05/12/2019	Approved
Integration testing	01/12/2019	05/12/2019	Approved
System testing	02/12/2019	05/12/2019	Approved
Inspection	03/12/2019	05/12/2019	Approved
Acceptance testing	05/12/2019	06/12/2019	Pending

6.8 Features to be tested

Unit testing of the text boxes Message ID and Message Body as well as the buttons “Process Text Boxes”, “Explore”, “Process File”, “Save to File”, “Previous”, “Next”, “Output” and “Clear”.

Functional testing of the 8 text buttons in the application interface.

Functional testing of the 2 text boxes (Message ID and Message Body) the user has to interact with.

6.8.1 User interface items

The user will be able to fill in a Message ID, it will have to be a capital letter (S E or T) followed by 9 numbers.
The user will be able to fill in a Message Body:

- If the Message is a SMS the message body text box has to start with a 11 digits phone number followed by the text of the message, it there are abbreviations they have to be handled as per SRS specifications. (2.2.1 of this document)
- If the Message is an Email the message body text box has to start with an email followed by a 20-character subject and followed by the content of the email. The content of the email may have urls, if there is any it will be quarantine as per the SRS specifications. There are 2 types of emails, Standard Emails and SIR emails, if the email is a SIR email the subject with have the format “SIR dd/mm/yy” and the content will include a Centre code and an incident name, the application will deal with it as per the SRS specifications. (2.2.2 of this document)
- If the Message is a Tweet, the message body text box has to start with a Tweet ID followed by the content of the tweet. If there are mentions (“@”) or hastags (“#”) they will have to be added to the list as per the SRS specifications. (2.2.3 of this document)

The user will be able to press the “Process Text Boxes” button, when it is done the content of the text boxes Message ID and Message Body will be process. If the Message ID is not correctly configured (is shorted than 10 characters and does not start with either S, E or T) a pop up message will prompt the user to correct it.
If a SMS message does not have a sender phone number, the user will be prompted to include one.
If an Email message doesn’t have a sender email or a 20-character subject, the user will be prompted to include them.

If a Tweet message doesn't have a sender tweeter id, the user will be prompted to include one.
If all data is correctly formatted when pressing the button "Process Text Boxes" the message will be displayed in json forma in the "Result:" panel.

The user will be able to press the "Explore" button, a pop-up windows explorer will open and the user will be able to choose the file to process. The path will be included in the text box provided for the path.
The user could enter a path manually in the path text box.

The user will be able to press the "Process File" button to be able to process messages in a text file. If the file is not the correct format or file type the user will be informed of what type of file and what format is required for the application to run. (Section 2.2.4 of this document)

When a file has been uploaded the user will be able to press the "Next" button to view the next message of the file. If there are no more message a pop-up message will inform the user.

When a file has been uploaded the user will be able to press the "Previous" button to view the previous message of the file. If there are no more previous messages a pop-up message will inform the user.

When messages have been processed the user will be able to press "Save to File" button, this will save all the content of the "Result:" panel into a json file called "output.json" in a folder called "ELMLogs" in the root directory of where the program is installed. The system will check if the folder exists, if it doesn't it will create it and create the file too. If the folder and file already exist it will write to the end of the file, not deleting the previous content of the file.

When the user presses the "Clear" button the content of the panel "Result:" will be cleared.

When several messages have been processed the user will be able to press the "Output" button, this will display the lists of Mentions, Hashtags and SIRs in the "Result:" panel. The list of Hashtags will not display more than once each hashtag but instead will mention the number of time each hashtag has occurred in the session of messages.
The application will as well save the content of those lists into a text file called "MentionsHastagsandEmail.txt" in a folder called "ELMLogs" in the root directory of where the program is installed. The system will check if the folder exists, if it doesn't it will create it and create the file too. If the folder and file already exist it will write to the end of the file, not deleting the previous content of the file.

6.9 Navigation Test Log

Client: Euston Leisure Tested by: Juan Alvarez
Date 02/12/2019

Window	Control	Expected Result	Actual Result	Comments
MainWindow	“Explore” button	Pop up window appears and allows user to navigate between the PC’s folders. It allows user to add a path to the path text box.	As expected	

6.10 Event Validation Test Log

Client: Euston Leisure Tested by: Juan Alvarez
Date 02/12/2019

Control	User Interaction	Expected Result	Actual Result	Comments
`BtnProcess_Click`	Clicks button with right data in Message ID and Message Body text boxes	Message is correctly processed and result in json format is outputted in “Result:” panel	As expected	
`BtnProcess_Click`	Clicks button with right Message Body text box content but short Message ID text box content although the first character is either S, T or E (capital).	Message “Message ID has to contain a letter and 9 digits” appears	As expected	
`BtnProcess_Click`	Clicks button with a Message ID for an	Message “You have to add a 11 digits Phone Number at the beginning of the Message Body” appears	As expected	

	SMS but without sender telephone number in the Message body.			
`BtnProcess_Click`	Click button with Email Message Id but without sender email in the message body	Message “You have to add a sender email address at the beginning of Message Body” appear	As expected	
`BtnProcess_Click`	Clicks button with email message id and sender email but without any more content afterwards or with a content of less than 20 characters	Message "There must be a Subject of 20 characters after the sender's email in Message Body" appears	As expected	
`BtnProcess_Click`	Clicks button with a Message ID for a Tweet but without sender details in the Message body.	Message “You have to add a sender in Message Body” appears	As expected	

btnFindPath_Click	Clicks button	Explorer window appears and allows user to choose a file to add to the path text box	As expected	
BtnProcess_Click	Clicks button with right Message Body text box content but Message ID text box content doesn’t start with either S, T or E (capital).	Message “Message ID has to contain a letter and 9 digits” appears	As expected	
btnProcessFile_Click	Clicks button after exploring the PCs folders and finding/choosing the right type of file.	File is loaded and first message of file display in Message ID and Message Body text boxes.	As expected	
btnProcessFile_Click	Clicks button after exploring the PCs folders and finding/choosing the right type of file.	Message "The file has to be a txt file. MessageId and Message Body have to be separated by * and messages have to be in different lines. File has not been uploaded" appears	As expected	
btnNext_Click	Clicks button after uploading	The next message of the file appears until there are no more messages. Then the	As expected	

	message from a file.	message “There are no messages left” appears		
`btnPrevious_Click`	Clicks button after uploading message from a file.	Message “There are no messages left” appears	As expected	
`btnPrevious_Click`	Clicks button after uploading message from a file and having navigated with the next button at least once.	The previous message on the file appears until there are no more previous messages. Then the message “There are no messages left” appears	As expected	
`BtnClear_Click`	Clicks button with content in the Result: text box.	Content of Result: panel gets deleted	As expected	
`btnOutput_Click`	Click button, with or without content in the Result: text box after processing several messages	Content List of Mentions:, List of Hashtags: and List of SIR: are saved into file “MentionsHastagsandEmail.txt” in the ELMLogs directory of the root directory (if it doesn’t exist it will be created) Does not deleted previous content of the file but adds to the actual content. Then displays in the “Result:” panel. What was saved into the file. If there are no processed messages it will display: List of Mentions: List of Hashtags:	As expected	

`btnSaveToFile_Click`	Clicks button after processing messages	List of SIR: The processed messages are save into file "output.json" in the ELMLogs directory of the root directory (if it doesn't exist it will be created). Does not deleted previous content of the file but adds to the actual content.	As expected	

6.11 Data Validation Test Log

Client: Euston Leisure Tested by: Juan Alvarez
Date 02/12/2019

Test Case	Test Item	Expected Result	Actual Result	Comments
`txtBoxMessageId`	**Normal**			
	S123456789	Input accepted and message processed	As expected	
	E2342	Error Message displayed		
	t123456789	Error Message displayed		
	Boundary			
	$%&	Error Message displayed		
	34543tr	Error Message displayed		
	S123456789012	Not permitted, field max length 10 digits		
	Extreme			
	$	Error Message displayed		
	*	Error Message displayed		
	Blank	Error Message displayed		
`txtBoxNessageBody`	**Normal SMS**			
	12345678901	Input accepted and message processed		

	12345	Input accepted and message processed		
	12345678901 This is to test ASAP	Input accepted and message processed		
	<u>Boundary SMS</u>			
	$%&	Error Message displayed		
	34543tr	Error Message displayed		
	S123456789012	Input accepted and message processed		
	<u>Extreme SMS</u>			
	$	Error Message displayed		
	12345678901 This is a really long message, longer than the 140 characters the program allows, so when used it will give an error message ok may be it was not long enough, what about now?	Error Message displayed		
	Blank	Error Message displayed		
	<u>Normal Tweet</u>			
	@Juan	Input accepted and message processed		
	@	Error Message displayed		
	Qerewerwe	Error Message displayed		
	<u>Boundary Tweet</u>			
	$%&	Error Message displayed		
	34543tr	Error Message displayed		
	S123456789012	Error Message displayed		

	Extreme Tweet			
	$	Error Message displayed		
	@Juan This is a really long message, longer than the 140 characters the program allows, so when used it will give an error message ok may be it was not long enough, what about now?	Error Message displayed		
	Blank	Error Message displayed		
	Normal Email			
	juan@juan.com This is the subject and some text	Input accepted and message processed		
	juan@juan.com	Error Message displayed		
	Just something random	Error Message displayed		
	Boundary Email			
	$%&	Error Message displayed		
	34543tr	Error Message displayed		
	S123456789012	Error Message displayed		
	Extreme Email			
	$	Error Message displayed		
	*	Error Message displayed		
	Blank	Error Message displayed		

6.12 Automated Testing

Client: Euston Leisure　　Tested by: Juan Alvarez
Date 02/12/2019　　Where: ELMFSTest.cs

Test Method	Input	Expected output	Result	Comments
AbbreviationTest()	This is to test if ASAP is handled correctly	This is to test if ASAP <As soon as possible> is handled correctly	As expected	
AbbreviationTest()	This is to test if AAS is handled correctly	This is to test if AAS <Alive and smiling> is handled correctly	As expected	
AbbreviationTest()	This is to test if ROTFL is handled correctly	This is to test if ROTFL <Rolling on the floor laughing> is handled correctly	As expected	
UrlProcessorTest()	This is to test if http://google.com is handled correctly	This is to test if <URL Quarantined > is handled correctly	As expected	
UrlProcessorTest1()	This is to test if http://facebook.com is handled correctly	This is to test if <URL Quarantined > is handled correctly	As expected	
UrlProcessorTest2()	This is to test if http://napier.com is handled correctly	This is to test if <URL Quarantined > is handled correctly	As expected	
processTxtBoxTest1()	12345678912 This is the content of message body	This is the content of message body	As expected	
processTxtBoxTest2()	@testcase This is the content of message body	This is the content of	As expected	

		message body		
processTxtBoxTest3()	noone@email.com This is the subject This is the content of message body	This is the content of message body	As expected	
processTxtBoxTest4()	noone@email.com This is the subject This is the content of message body	This is the content of message body	As expected	
processTxtBoxTest5()	"e1234"	""	As expected	The test is a success as the method found an incorrect message id
processTxtBoxTest6()	"1234"	""	As expected	The test is a success as the method found an incorrect message id
FindPhoneNumber1()	12345678901	12345678901	As expected	Please note that to be able to run this test method the sms class has to be changed to public and the FindPhoneNumber method in that class has to be changed to public.

7 Version control plan

The recommendation for version control plan is to use GitHub in other to coordinate the different engineers working on the software, because it is easily accessible by everyone and has all the needed tools for this project.

As an agile approach will be undertaken there is potential for changes to the software specifications, so we have to ensure we can access the different versions and have back-ups of the different versions that are been worked on in order to revert back to one of them in case of need.

The format of version control used will be Distributed to allow collaboration of different software engineers in case of need.

The repository will be stored centrally at the GitHub website and each engineer will have a workspace locally once the content is checked out of the repository.

To avoid conflicts Branching will be used so each engineer will be able to work on its side of the code and commit it back to the main line in order and that way not lose any of the work.

Optimistic locking will be used, so when merging the code back to the main line the engineer will have to update the local copy of the file to include the latest repository changes before checking it in.

As the work is going to be conducted in an Agile approach the sprints will be short, every week there will be a checking in of the updates to ensure all aspects of the project are tested and that the system as whole works correctly. This will be accompanied by daily meetings where everyone involved in the project will update the rest of the team in their particular progress and if there are any needs or road blocks that have to be solved before proceeding.

8 Evolution strategy

The ELM application is, so far, a low maintenance one, it is limited to one computer and to data you input in that moment, be it with a file or manually.

Eventually, with constant use, it will require further hard drive space unless there is a regular deletion of old unneeded files. The json and txt files used to store data will eventually become too big unless it is cleaned regularly.

So, the first thing to do for long term maintenance is deletion of old logs, it could actually be coded in the program itself to be done automatically every so often, or to have the possibility of doing it manually with a button choosing time frames. This would mean a step on the evolution of the program.

The next step on the evolution would be to include other platforms in the program, like Facebook and Instagram.

Another step on the evolution is to automate the processes, right now you have to enter manually, either one by one or in a file, the messages. The next evolutionary step would be to get it connected to the internet or may be even base it on a web server, and program it to receive and process the messages automatically, as they arrived in the different systems. In order to save time and cost, so no one has to introduce them manually. This will require a total re-engineering of the system and could prove costly, but in the long term it would prove worthwhile and the time saving, efficiency and usefulness of the resulting system would mean that in reality the client would be saving money with the cuts to staffing time needed to operate the system.

I would as well recommend a more permanent storage of the messages, like a SQL database implemented in a server. This will cut the time required to maintain old logs.

Plugins/modules for the different types of messages should be added to the program and connectivity to the different systems is needed.

Eventually the reporting could be improved as well, instead of publishing the results in a json file, it could be done into a web browser interface so the user can access the data easily and from anywhere, not needing to access the computer where the program sits.

This are just some ideas that can be studied, each of the changes should go through Change request. It would mean analysing the impact it would have on users and systems, and the benefits that each change could bring to the organization. That should be compared with the cost of producing it and a decision should be made.

If it is approved there would be a need to plan a new release, adapting it to the platform available or may be even changing it to a different platform, like a web server, so it is more accessible for the user.

Once the changes are implemented in the code and tested appropriately, through automated regression testing in particular, the new system release should be made available to the client and tested in house. Training to the users would be required if the new implementation of the ELM application has changed dramatically.

As the ideas and possible evolution of the software are many and very different, this could not be done in a single iteration, so several iterations and constant meetings must be had with the client. Following an agile approach would benefit the evolution of the system so the client approves each new feature before release and is happy with the results.

All these possible changes are recommended to start soon after the original version of the software is delivered, that way we will keep the team on board and avoid the phase of program understanding if we lose the original programmer and have to rely on new ones.

All this potential evolutionary changes to the ELM application have to be done side by side with the maintenance of the system. Continuous analysis has to be carried out to ensure any faults are corrected if and when they appear.

If the client changes the operating environment, we have to adapt the software it.

If the requirements of the application changes, we have to perfect the software to ensure that new functionality is achieved. Some of those possible new functionalities have been covered in earlier paragraphs.

Originally there will not be a need to refactor the program, as it is brand new and will be coded efficiently and thinking about low maintainability from the beginning, but it is something we have to keep in mind for future, since constant changes could compromise the original structure of the program over time.

Bibliography

Dasblinkenlight. (2013, 06 02). *Convert a txt file to dictionary<string, string>*. Retrieved from https://stackoverflow.com/: https://stackoverflow.com/questions/16885438/convert-a-txt-file-to-dictionarystring-string

Microsoft. (2019). *Directory.CreateDirectory Method*. Retrieved from Microsoft.com: https://docs.microsoft.com/en-us/dotnet/api/system.io.directory.createdirectory?view=netframework-4.8

Microsoft. (2019, 09 16). *How to serialize and deserialize JSON in .NET*. Retrieved from Microsoft.com/: https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-how-to

Microsoft. (2019). *String.StartsWith Method*. Retrieved from Microsoft.com: https://docs.microsoft.com/en-us/dotnet/api/system.string.startswith?view=netframework-4.8

Newtonsoft. (2019). *Json.NET Documentation*. Retrieved from https://www.newtonsoft.com/: https://www.newtonsoft.com/json/help/html/Introduction.htm

QuackWare. (2009, 08 13). *How to Read and Write to Text File - C# Tutorial Visual Studio 2008*. Retrieved from Youtube.com: https://www.youtube.com/watch?v=q9TRNbWv99M

Rexegg. (2015). *Using Regular Expressions with C#*. Retrieved from https://www.rexegg.com/: https://www.rexegg.com/regex-csharp.html