

SET09102 Coursework

Report



October 11, 2017

Jonathan Mitchell

40311730

Contents

[Analysis 2](#_Toc497145038)

[Requirements Specification 3](#_Toc497145039)

[Functional Requirements 3](#_Toc497145040)

[Non-functional Requirements 5](#_Toc497145041)

[User Stories 7](#_Toc497145042)

[Use-Case Model 8](#_Toc497145043)

[Design 9](#_Toc497145044)

[Class Diagram 9](#_Toc497145045)

[Implementation 10](#_Toc497145046)

[MainWindow xaml: 10](#_Toc497145047)

[AddMessage xaml’s: 10](#_Toc497145048)

[MessageAdd & ListAdd classes: 11](#_Toc497145049)

[SaveToFile & SaveToList Classes: 11](#_Toc497145050)

[SelectList xaml: 11](#_Toc497145051)

[DisplayMessage xaml 11](#_Toc497145052)

[DisplayList xaml 11](#_Toc497145053)

[Testing 12](#_Toc497145054)

[Test Plan 12](#_Toc497145055)

[Test Strategy 12](#_Toc497145056)

[Test Cases 12](#_Toc497145057)

[Version Control 13](#_Toc497145058)

[Evolution Strategy 14](#_Toc497145059)

# Analysis

Euston Leisure, an association of sport centres in the city of Euston, are looking for a software system, Euston Leisure Messaging (ELM), to be developed.

This system will provide the following service: validate, sanitize and categorise incoming messages to Euston Leisure in the form of:

* SMS text messages
* Emails
* Tweets

## Requirements Specification

### Functional Requirements

The computer system will provide the following:

* Message select window
* SMS input page
* Email input page
* Tweet input page
* Display lists Page
* Must be able to automatically identify the message type & process accordingly
* Ability to write to text files
* Ability to read from text files.

Message Select Window will:

* Provide the user with buttons to navigate to one of the following pages:
  + SMS input page
  + Email input page
  + Tweet input page
* Provide a button to navigate to display Lists page
* Provide a button to exit the application

SMS Input Page

* All input is to be verified
* Display a message if any input is incorrect/missed out
* Process all messages appropriate to its type
* A navigate button to go to message display Page
* An insert button to store message

Email Input Page

* All input is to be verified
* Display a message if any input is incorrect/missed out
* Process all messages appropriate to its type
* A navigate button to go to message display Page
* An insert button to store message

Tweet Input page

* All input is to be verified
* Display a message if any input is incorrect/missed out
* Process all messages appropriate to its type
* A navigate button to go to message display Page
* An insert button to store message

Message Display Page will:

* Provide buttons to display one of the following lists:
  + Trending list
  + Mentions list
  + SIR list
  + Email list
  + Tweet list
  + SMS list
* Each list page will have a button allowing the user to go back a page

### Non-functional Requirements

* All messages must be strings composed of ASCII characters
* Each page will provide clear details of which box(es) information is to be inserted into
* Each message will have a Message Header comprising:
  + Message ID containing:
    - Select Message type (“S”,”E”,”T”) followed by 9 numeric characters
  + Body Will Comprise:
    - SMS Messages Will contain:
      * A SENDER
        + In the form of an international telephone number
      * The MESSAGE TEXT
        + Must be no more than 140 characters
        + It may contain embedded text speak abbreviations
    - Email Messages Will contain:
      * A SENDER
        + In the form of a standard email address
      * A SUBJECT
        + Must be no more than 20 characters
      * The MESSAGE TEXT
        + Must be no more than 1028 characters long
        + It may contain embedded hyperlinks. N the form of standard URLs
    - Tweets will contain:
      * A SENDER
        + Must consist of a Twitter ID: “@”
        + Followed by a maximum of 15 characters
      * MESSAGE TEXT
        + Must not been more than 140 characters
        + Can contain the following:

Textspeak

Same as SMS above

Hashtags

Must be a string of characters preceded by a “#” sign

Twitter ID’s

As above

## User Stories

As the System

I want to be able to deal with SMS Messages

As the System

I want to be able to deal with Email Messages

As the System

I want to be able to deal with Twitter Messages

As the System

I want to be able to process SMS messages by checking if they contain textspeak abbreviations and expand them to their full form

As the System

I want to be able to process Emails to determine if they are standard Emails or Significant Incident Reports.

As the System

I want to be able to Process Emails to see if they contain Hyperlinks and quarantine them

As the System

I want to be able to process tweets by checking if they contain textspeak abbreviations and expand them to their full form

As the System

I want to be able to process hashtags in tweets and add them to a list to display how many times they are used

As the System

I want to detect the three types of messages (SMS, Email & Tweet) and write them to a file(s) in JSON format

As the System

I want to be able to display the trending list, list of “@” mentions & the SIR list

As the System

I want to be able to process the above message types from an input file

## Use-Case Model

# Design

## Class Diagram

# Implementation

A modular approach was taken. This was achieved by splitting up the prototype by adding the following folders:

* Views Folder – Contains the User-controls for the prototype. These are the basic “windows” that are displayed to the user, depending on which button they have pressed – all contain a different background colour to show where they appear when called in each user control content control.
* ViewModels Folder – contains the classes that display text/ button commands for the corresponding User-Control
* Models Folder – contains the classes were input entered is stored as an object. Input is either via user or read from a file
* Database Folder – contains the classes that read/write to txt files. This is where the JSON Serialize/de-serialize occurs
* Commands folder – contains the Relay Command class that allows pages to be loaded (as an event) into the relevant User-control content control. Tis class inherits the ICommand class

MainWindow.xaml was not added to a folder it was left out with them.

## MainWindow xaml:

## AddMessage xaml’s:

The first User Controls to be implemented where for user input. This involved creating 3:

* AddSMSView
* AddEmailView
* AddTweetView

Textblocks were created that detail what needs to be inputted into the Textboxes provided Insert button to add the message type to the relevant txt file. All text/content/command has been bound to the relevant property in the corresponding class.

And their corresponding Classes:

* AddSMSViewModel
* AddEmailViewModel
* AddTweetViewModel

These classes are where the content for each textblock/ button is set. Also input from the textboxes is stored here, verified, and if all this passes the input is to be inserted into an object of the relevant class. The class chosen is dependent on wat is to be stored. The full message is stored in a MessageAdd class object. Tweet hashtags & the @ mention are stored in a ListAdd class object.

Once this is done, the object data is then saved into the relevant file. A message then displays if this process was successful.

## MessageAdd & ListAdd classes:

This contains property references for storing the objects contents in the constructor. The EmailAdd class inherits from this class. As the Email input has an additional textbox data to store(subject). Like MessageAdd class, but stores less information.

## SaveToFile & SaveToList Classes:

These classes take to the object of MessageAdd/EmailAdd/ListAdd classes, along with a string for message type (S, T, E, I for SaveToFile & M, H for SaveToList) JSON serialises the object then stores into the relevant file. If this is not possible displays message stating so.

## SelectList xaml:

This user control is used to select which type of message/list the user wants to view. Button for each option: SMS, Email, Incident report, Tweet message views & trending, mention & SIR lists.

The content/command for these are implemented by the SelectListViewModel

## DisplayMessage xaml

This user control is not implemented now. Will return later

## DisplayList xaml

This user control displays the list for the selected option. Not implemented now.

# Testing

## Test Plan

The following pages will be tested:

* Message select window:
  + All buttons will be tested to ensure they display the correct page or exit application successfully
* SMS input page:
  + Validation of each input will be tested
* Email input page:
  + Validation of each input will be tested
* Tweet input page:
  + Validation of each input will be tested
* Display lists Page:
  + All buttons will be tested to ensure they display the correct page
  + All inserted records/lists etc will be checked to ensure they have all been processed correctly
* Text files:
  + Will be checked to ensure the written records are in JSON format and display the correct information in each column/row

## Test Strategy

## Test Cases

# Version Control

GitHub will be used to create the version control. Each iteration/addition to any representation of the software system will be added to the repository with a note detailing what was done.

Link here <https://github.com/Jalektor/Mitchell\_Jonathan\_set09102.git>

# Evolution Strategy

1. Instead of writing to a file in JSON format. Storing in a database, with encryption, would be a more suitable option. Stored internally within the system itself.
2. The system should only require minimal maintenance. As a result, maintenance costs would be kept low.
3. Adding additional security in the form of a login page? Stored within database
4. Potentially move the system to be web-based.
5. Increase input types to include Facebook?
6. Integrating anti-virus/spam filtering software to check if URL’s in websites are genuine or not – reduces number of quarantined items.