Napier Bank Message Filtering Service Report

Napier University

Software Engineering

Contents

[Introduction 3](#_Toc467835913)

[Purpose 3](#_Toc467835914)

[Objective 3](#_Toc467835915)

[Scope 3](#_Toc467835916)

[Task 3](#_Toc467835917)

[Planning 4](#_Toc467835918)

[Feasibility Study 4](#_Toc467835919)

[Operational Feasibility 4](#_Toc467835920)

[Technical Feasibility 5](#_Toc467835921)

[Schedule Feasibility 6](#_Toc467835922)

[Requirements Specification 7](#_Toc467835923)

[Purpose 7](#_Toc467835924)

[Requirements Glossary 7](#_Toc467835925)

[User Case Diagram 7](#_Toc467835926)

[Use Case Index 8](#_Toc467835927)

[Use Case Components 8](#_Toc467835928)

[Functional Requirements 13](#_Toc467835929)

[Non-Functional Requirements 15](#_Toc467835930)

[User Interface Requirements 15](#_Toc467835931)

[Class Diagram 18](#_Toc467835932)

[Logo 19](#_Toc467835933)

[Test Plan 20](#_Toc467835934)

[Version History 20](#_Toc467835935)

[Purpose 20](#_Toc467835936)

[Objectives 20](#_Toc467835937)

[Scope 20](#_Toc467835938)

[Environmental Needs & Tools 20](#_Toc467835939)

[Testing Strategy 21](#_Toc467835940)

[Unit Testing 21](#_Toc467835941)

[Functional Testing 21](#_Toc467835942)

[Acceptance Testing 21](#_Toc467835943)

[Test Cases 21](#_Toc467835944)

[Unit Test Cases and Logs 21](#_Toc467835945)

[Functional Test and Logs 23](#_Toc467835946)

[Acceptance Test and Logs 26](#_Toc467835947)

[Test Schedule 27](#_Toc467835948)

[Risks and Solutions 27](#_Toc467835949)

[Version Control Plan 28](#_Toc467835950)

[Introduction 28](#_Toc467835951)

[Purpose 28](#_Toc467835952)

[Version Control System 28](#_Toc467835953)

[Connect to Version Control Server 28](#_Toc467835954)

[Primary Tasks 28](#_Toc467835955)

[Secondary Tasks 29](#_Toc467835956)

[Useful Features 29](#_Toc467835957)

[Branching 29](#_Toc467835958)

[Merging 29](#_Toc467835959)

[How it will be used 30](#_Toc467835960)

[Software Evolution Strategy 30](#_Toc467835961)

[Introduction 30](#_Toc467835962)

[Purpose 30](#_Toc467835963)

[Software Evolution 30](#_Toc467835964)

[Types of Maintenance 31](#_Toc467835965)

[Cost Analysis 31](#_Toc467835966)

[Factors That May Affect Cost 31](#_Toc467835967)

[Software Maintenance Activities 32](#_Toc467835968)

[Research 33](#_Toc467835969)

[Structure 33](#_Toc467835970)

[Appropriate API 33](#_Toc467835971)

[Conclusion 33](#_Toc467835972)

[Bibliography 34](#_Toc467835973)

# Introduction

Napier Bank is requesting a new Message Filtering service that will validate, sanitize and categorise incoming messages using a computerised system that will deal with the following functions:

1. Allow messages to be created
2. Allow messages to be saved onto a JSON file
3. Allow messages to be retrieved and viewed from the JSON file

The full requirements specification document can be found here: .

The Napier Bank Messages Filtering Service will benefit Napier Bank in many ways; it will help control the way in which way the messages are validated and categorised.

## Purpose

The main purpose of the system is to help bank employees filter messages more easily and gives them more control on how they’re displayed.

## Objective

The main objective of this program is to have SMS messages, Email messages and Tweets stored onto a JSON file and retrieved. Bank employees will be able to filter the messages.

The program is a package to be used to allow the bank to improve the efficiency of their business. The system that is to be developed will benefit the bank’s employees excellently.

## Scope

The Scope of the message filtering service will be:

* Creating messages such as: SMS, Email and Tweets
* Filtering messages such as: SMS, Email and Tweets

## Task

The program will be able to fulfil the following the task:

The user will be presented with a screen with three commands on it. The user will select the create message button that will allow them to create either a SMS, Email or Tweet message and will be saved in a JSON file. The filter message button will display a screen with a table that will show all the messages that are present in the system (Each message will have their own Message ID displayed in a column on the table. The help button will display a message dialog with instructions on how to use the program.

# Planning

## Feasibility Study

The objective of the feasibility study is to find out if the proposed message filtering service will benefit Napier Bank.

A feasibility study should provide Napier Bank with enough information to decide:

* Whether the project can be done?
* Can the final product benefit the intended users and organisation?
* What the alternative solution paths may be?
* Is there a worthy alternative?
* How beneficial or practical can the development of the message filtering service be to the organisation?

The feasibility will undergo three major stages of analysis such as: Operation Feasibility, Technical Feasibility and Schedule Feasibility.

### Operational Feasibility

The operational feasibility determines how well the message filtering service solves the problem and takes advantage on the tasks identified during the Scope definition. The operational feasibility will also determine the willingness of management to support the message filtering service.

**Performance** – Does the current mode of operation have an acceptably response time and offer adequate throughput?

Napier Bank does not have a message filtering that offers adequate throughput. This service will play a very important role on how the incoming messages are dealt with and makes the process of filtering the messages easier and user friendly.

**Information** – Does the current mode provide Napier Bank with timely, accurate and relevant information?

This system will provide Napier Bank with timely, relevant and accurate information. Since all the messages are being given a unique message ID, therefore it will provide meaningful information to Napier Bank and the users of the service. The chances of an incorrect message being filtered will be minimal.

**Economy** – Does the current mode of operation provide effective information services to Napier Bank regarding the overall cost? Could the costs be cut and / or a chance of increased benefits?

The initiation of the message service will reduce manual work. Napier Bank will not need to manually validate every message that a user sends. The message service will be cost effective because the less manual work needing done the less employees are needed to perform this task, thus increasing Napier Bank’s profits.

**Control** – Does the current mode of operation offer effective solutions to help protect against fraud and to guarantee the security and accuracy of information?

Napier Bank will be required by law due to the Data Protection Act 1998 to keep user information safe. The new system however will allow Napier Bank to file serious incident reports and have all URL’s quarantined to prevent the chances of a virus making its way onto the Napier Bank’s mainframe.

**Efficiency** – Does the current mode of operation use all resources to their maximum potential, including staff, work hours and number of messages validated and categorised?

A lot of time is wasted on employees having to manually validate and categorise incoming messages. The message filtering service will make the process of fetching out requested data.

**Services** – Does the current mode of operation provide an effective reliable service? Is the current system flexible and expandable?

It will take longer for people to manually filter the message. More people will need to be hired to ensure the Napier Bank filters the messages correctly, however the message filtering service will that task automatically.

### Technical Feasibility

The technical feasibility will determine if the technical requirements of the message filtering service can be achieved and if the technology to support this new system exists.

The questions that will help to determine this are?

* Is the project achievable within the limits of existing technology?
* Is it able to cope within given resource constraints?

The list provided is a list of recommended hardware that will run this message filtering service

**Recommended Hardware: -**

|  |  |
| --- | --- |
| Specification Number: | Hardware Used |
| 1 | Desktop PC / Laptop |
| 2 | Basic Mouse |
| 3 | Wired Keyboard |
| 4 | Standard Monitor |

**Recommended Software: -**

Software plays a very important role in the development of this message filtering service. Understanding what software should be used is very important to develop the message filtering service.

Windows 7 will serve as the main operating system for developing and running the message filtering service.

The software that will be used to plan and develop this message filtering service are:

1. Visual Studio 2012 /2013
2. Microsoft Word
3. Star UML with RE-Tools
4. Visio 2013

### Schedule Feasibility

The schedule feasibility allows Napier Bank plan the required tasks that are needed to complete this message filtering service.

**Deliverables**

Here is a list of deliverables that will be needed to complete this application:

|  |  |
| --- | --- |
| d.no | Deliverable |
| 1 | Use Case Diagram |
| 2 | Class Diagram |
| 3 | Prototype |
| 4 | Testing Plan |
| 5 | Test Logs |
| 6 | Version Control Plan |
| 7 | Evolution Strategy |
| 8 | Report |

# Requirements Specification

## Purpose

The purpose of the requirements specification is to analyse all collected ideas that have been provided by Napier Bank to define the system in more depth, so the consumers and the development team can understand the system with more clarity. These requirements are for Version 1.0.

## Requirements Glossary

Here is a list of the technical terms / anagrams that may be used in this report

UI – User Interface  
GUI – Graphical User Interface

## User Case Diagram

This use case diagram shows the sequence of actions needed to fulfil this requirement specification. Star UML (Sourceforge, n.d.)was used to create the following diagram with RE-Tools (utdallas, n.d.).



### Use Case Index

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Use Case ID | Use Case Name | Primary Actor | Scope | Complexity | Priority |
| 1 | Create Message | User | In | Med | 1 |
| 2 | Filter | User | In | Med | 1 |
| 3 | Choose Message Type | User | In | Low | 1 |
| 4 | Save Message | User | In | High | 1 |
| 5 | Choose Load Option | User | In | High | 1 |
| 6 | Load All Messages | User | In | High | 1 |
| 7 | Load Next Message | User | In | Med | 2 |
| 8 | Load Previous Message | User | In | Med | 2 |

### Use Case Components

#### Use Case 1

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 1 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Create Message |
| Use Case Description | Allows the user to create a message |
| Primary Actor | User |
| Precondition | The application must be open |
| Trigger | When the user clicks the Create Message button on the application’s GUI |
| Basic Flow | The user is presented with a screen that will allow them to create select their message type |
| Alternative Flow | The user may close the application |

**Brief Description**

Use Case Number: 1  
Use Case Name: Create Message  
Description: A user wants to create a message, so they’ll click the Create Message button with the intent of creating a message to be send to Napier Bank.

#### Use Case 2

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 2 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Filter |
| Use Case Description | Allows the sender to filter messages |
| Primary Actor | User |
| Precondition | The application must be open |
| Trigger | When the user clicks the Filter button |
| Basic Flow | The user is presented with a screen that will allow them to filter messages |
| Alternative Flow | The user may close the application |

**Brief Description**

Use Case Number: 2  
Use Case Name: Filter  
Description: A user wants to filter messages with the intent of selecting a variety of options to do so.

#### Use Case 3

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 3 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Choose Message Type |
| Use Case Description | Allows the user to select what type of message they’re wanting to create |
| Primary Actor | User |
| Precondition | The user must have clicked the Create Message button |
| Trigger | When the user selects an option from the combo box |
| Basic Flow | The user is presented with a screen that will allow them to create the message they’ve selected |
| Alternative Flow | The user may close the application |

**Brief Description**

Use Case Number: 3  
Use Case Name: Choose Message Type  
Description: A user wants to select the type of message they’re wanting to create with the intent of creating a SMS, Email or Tweet.

#### Use Case 4

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 4 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Save Message |
| Use Case Description | Allows the user to save the message they’ve just created |
| Primary Actor | User |
| Precondition | The user must be on the create message screen |
| Trigger | When user the clicks the Save Message Button |
| Basic Flow | The message is saved onto a JSON file |
| Alternative Flow | 1. The user may close the application 2. There may be insufficient storage |

**Brief Description**

Use Case Number: 4  
Use Case Name: Save Message  
Description: A user wants to save the message they’ve just created with the intent of having it saved to a JSON file so they can use it for filtering.

#### Use Case 5

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 5 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Choose Load Option |
| Use Case Description | Allows the sender to select how the messages are loaded |
| Primary Actor | User |
| Precondition | The user must be on the filter message screen |
| Trigger | When user looks at the options on the screen |
| Basic Flow | The user’s load selection is made |
| Alternative Flow | The user may close the application |

**Brief Description**

Use Case Number: 5  
Use Case Name: Choose Load Option  
Description: A user wants to select what load option they want with the intent of having their option show some results on the GUI.

#### Use Case 6

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 6 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Load All Messages |
| Use Case Description | The messages are loaded and are shown on the text wall |
| Primary Actor | User |
| Precondition | The user must be on the filter message screen |
| Trigger | When clicks the Load All Messages button on the GUI |
| Basic Flow | The messages are filtered base on the user’s choice |
| Alternative Flow | The sender may close the application |

**Brief Description**

Use Case Number: 6  
Use Case Name: Load All Messages  
Description: A user wants to load all messages with intent of having all the messages that are loaded displayed on the GUI.

#### Use Case 7

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 7 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Load Next Message |
| Use Case Description | Allows a message to be loaded in |
| Primary Actor | Napier Bank Mainframe |
| Precondition | The user must be on the filter message screen |
| Trigger | When user clicks the Load Next Message on the GUI |
| Basic Flow | The message is loaded and is shown on the text wall |
| Alternative Flow | The user may close the application |

**Brief Description**

Use Case Number: 7  
Use Case Name: Load Next Message  
Description: A user wants to load the next message with the intent of having one message load and be displayed on the GUI.

#### Use Case 8

|  |  |
| --- | --- |
| Use Case Element | Description |
| Use Case Number | 8 |
| Application | Napier Bank Message Filtering Service |
| Use Case Name | Load Previous Message |
| Use Case Description | Allows the previous message to be loaded |
| Primary Actor | Napier Bank Mainframe |
| Precondition | The user must be on the filter message screen |
| Trigger | When user clicks the Load Previous Message button on the GUI |
| Basic Flow | The message is loaded and is shown on the text wall |
| Alternative Flow | The user may close the application |

**Brief Description**

Use Case Number: 8  
Use Case Name: Load Previous Message  
Description: A user wants the message that was displayed last displayed on the GUI.

## Functional Requirements

This section includes the requirements that specify all the fundamental actions of the Napier Bank Messages Filtering Service.

**SMS Messages:**

* The system must provide each SMS with a Unique ID in the form of “S123456789”.
* Each message must have sender in the form of an international telephone number.
* The message text must be no more 140 characters long and provide validation to ensure that it remains within the requested range.
* Each message must also have a mechanism in place to ensure that text speak abbreviations are expanded to their full form and enclosed in “<>”. For example, “LOL” becomes <Laughing out loud>.
* The SMS message must be save in a JSON file.

**Email Messages:**

* The system must provide each Email with a Unique ID in the form of “E123456789”.
* Each message must have a sender in the form of a standard email address.
* The subject must be no longer than 20 characters long and provide validation to ensure that it remains within the requested range.
* The message text must be no longer than 1028 characters and provide validation to ensure that it remains within the requested range.
* The system must be able to automatically identify if the email is a Standard Email or Significant Incident Report
* The Email must be saved in a JSON file.

**Standard Emails**

* Any URLS found within the email must be removed and replaced with “<URL Quarantined>” in the body of the message.

**Significant Incident Reports (SIR)**

* The subject field must be filled in the form of “SIR dd/mm/yy”. For example, “Theft 20/12/2016”
* The body of the message must begin with the following standard texts on the first two lines:

Sort Code: 99-99-99

Nature of Incident:

* Sort code and SIR must be written to a SIR list.
* Any URLS must be removed and replaced with “<URL Quarantined>” in the body of the message.

**Tweets**

* The system must provide each Tweet with a Unique ID in the form of “T123456789”.
* Each tweet must have a sender in the form of Twitter ID: “@” followed by a maximum of 15 characters, for example, “@LiamSmith”.
* The Tweet messages must be no longer than 140 characters long and provide validation to ensure it stays within range.
* The tweet must contain ordinary text and text speak must be expanded like the SMS Messages.
* The system must also group any text that is preceded with a “#” must be added to a hashtag list that will count the number of uses and add them to a trending list. “Mentions: “. i.e. embedded Twitter IDs will be added to the mentions list.
* The Tweet must be saved in a JSON file.

**Help**

* Must display a message dialog that’ll contain instructions on how to use the program.

## Non-Functional Requirements

This section includes a list of requirements that define what the system is supposed to be. Non-functional requirements are listed below:

**Scalability**

* The program must be able to be used by multiple users

**Response Time / Performance**

* The program must respond to every action in a reasonable amount of time
* The program must save files very quickly

**Storage**

* The program must ensure there’s a reasonable amount of storage to store any files associated with it.

## User Interface Requirements

These are list of requirements for the GUI:

**System Wide User Interface Requirements:**

* The user interface must take form of an input using WPF (Windows Presentation Foundation).
* The system must be able to identify the message type and process it accordingly.
* All buttons font colour must be black
* All buttons font must be set to Arial
* All buttons font size must be set at 20 and set to bold
* All text boxes font must be set to Arial
* All text boxes must have a font size of 16

**Main Window**

* Must have a button called btnCreateMessage and content set to “Create Message”
* Must have a button called btnFilterMessage and content set to “Filter Message”
* Must have a button called btnHelp and content set to “Help”
* The Main Window must display the Napier Bank Logo
* Must have a frame called pageFrame to allow Create Message Page or Filter Message page to populate it

**Create Message**

* Must be displayed in the pageFrame.
* Must have a frame called messageFrame to allow the Enter Message Page to populate it
* Must have a button called btnClose and content set to “Close”

**Enter Message**

* Must be displayed in the messageFrame
* Must have a text block called lblMessageCredentials with text set to “Please enter your message credentials:”
* Must have a text block called lblSubject with text set to “Subject”   
  - The subject text block will only show if a valid email address is typed into the message credentials text box.
* lblSubject font must be Arial
* lblSubject font size must be 20
* Must have a text block called lblSortCode with content set to “Sort Code”   
  - The sort code text block will only show if a valid Serious Incident Report is typed into the subject text box
* Must have a text box called txtMessageCredentials with blank text
* Must have a text box called txtSubject with blank text
* Must have text box called txtSortCode with blank text  
  - The sort code text box will only show if a valid Serious Incident Report is typed into the subject text box
* lblSortMessage font must be set to Arial and have a font size of 16
* Must have a text block called lblBody with content set to “Body”
* lblBody font must be Arial
* lblBody font size must be 20
* Must have a text box called txtMessageContainer with blank text  
  - This text box must have its scroll bar enabled
* Must have a button called btnClear and content set to “Clear”
* Must have a button called btnSend and content set to “Send”

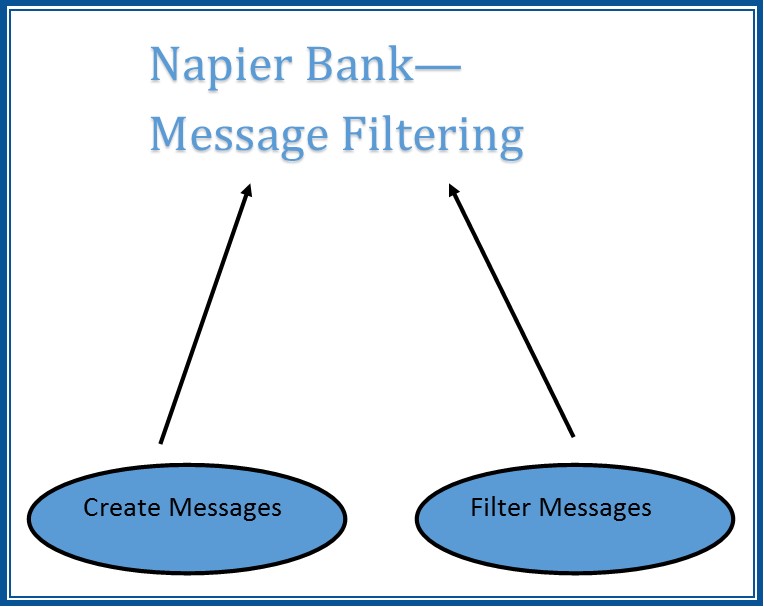
**Filter Message**

* Must be displayed in the Main Window’s frame
* Must have a frame to allow the ShowMessagePage to populate it
* Must have a text box called txtMessageWall with blank text  
  - This text box must have its scroll bar enabled
* Must have a button called btnShowAll with content set to “Show All”
* Must have a button called btnNext with content set to “Next”
* Must have a button called btnPrevious with content set to “Previous”
* Must have a button called btnClear with content set to “Clear”
* Must have a listView called lstHashtags with a column called Hashtags with data binding
* Must have a listView called lstMentions with a column called Mentions with data binding
* Must have a listView called lstSIR with two columns called SIR and the other called Sort Code both with data binding
* Must have a listView called lstUrls with a column called Urls with data binding

# Class Diagram

This is the class diagram for the Napier Bank Message Filtering Service

# Logo

Here is the logo for the Napier Bank Message Filtering Service.

# Test Plan

## Version History

This is a list that shows all the versions of the Napier Bank Message Filtering Service

|  |  |  |
| --- | --- | --- |
| Application Name: | Version | Created |
| Napier Bank Message Filtering Service | 1.0 | 10/11/16 |

## Purpose

This test plan describes the overall framework and testing approach that will drive the testing of the Napier Bank Version 1.0. It identifies the items and features to be tested; types of testing.

## Objectives

The objective of the test plan is to show the full functionality and features defined in the Requirements Specification.

## Scope

The scope of this test plan is to ensure that all the requirements are met and developed properly. Scheduled and unscheduled tests will be done and errors found will be documented.

## Environmental Needs & Tools

These are tools and software that will be needed to run and document the tests for Napier Bank Message Filtering Service.

**Software:**

* Visual Studio 2013
* Microsoft Word

**Hardware:**

* Desktop PC

## Testing Strategy

The following types of tests will be performed by using the information on this document such as: Requirements Specification and Detailed Design. All tests must pass.

### Unit Testing

The primary goal of unit testing is to take the smallest piece of testable software in the application, then isolate it form the remaining code, and determine whether it behaves exactly as you expect (Microsoft, n.d.).

Therefore, unit testing will be used to verify the implementations of each necessary requirement changes. The requirements will be tested and verified. After new features are added unit tests will be done for those features.

### Functional Testing

Functional Testing is a technique that is used to test the features / functionality of the system or software, this should cover all the possible scenarios including failure paths and boundary cases (tutorialspoint, n.d.).

This test will be performed to ensure the Napier Bank Message Filtering Service functions work correctly and respond to failure paths.

### Acceptance Testing

Acceptance Testing is a testing technique performed to determine whether the software system has met the requirements specification. The purpose of this test is to evaluate the system’s compliance with the client’s requirements and verify if it has met the required criteria for the delivery to end users (tutorialspoint, n.d.).

This test will be conducted by Napier Bank to ensure the program does what they have requested.

## Test Cases

### Unit Test Cases and Logs

Here are the results for the unit test cases.

#### testabbreviate1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| LOL | abbreviate() | LOL<Laughing out loud> | LOL<Laughing out loud> | Pass | Works as intended |

#### testabbreviate2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| B4 | abbreviate() | B4<Before> | B4<Before> | Pass | Works as intended |

#### Testabbreviate3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| B4 LOL | abbreviate() | B4<Before> LOL<Laughing out loud> | B4<Before> LOL<Laughing out loud> | Pass | Works as intended |

#### Testabbreviate4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Class** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| lol | abbreviate() | lol | lol | Pass | Does not expand text that is not capped Works as intended |

#### testURL1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| www.bbcnews.co.uk | quarantine () | <URL Quarantined> | <URL Quarantined> | Pass | Works as intended |

#### testURL2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| www.skynews.co.uk | quarantine () | <URL Quarantined> | <URL Quarantined> | Pass | Works as intended |

#### testURL3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| www.skynews.co.uk www.bbcnews.co.uk | quarantine () | <URL Quarantined> <URL Quarantined> | <URL Quarantined> <URL Quarantined> | Pass | Works as intended |

#### testURL4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Method Tested** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| www.skynews | quarantine () | www.skynews | www.skynews | Pass | Does not quarantine unfinished urls. Works as intended. |

## Functional Test and Logs

Here are the results for the functional test cases.

**SMS Messages:**

**Test 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| 0773990755 | LOL | SMS Saved! | SMS Saved! | Pass | Works as intended |

**Test 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| 01506856467 | Hello | SMS Saved! | SMS Saved! | Pass | Works as intended |

**Test 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| 123456789 |  | Message Box: Please Enter a message! | Message Box: Please Enter a message! | Pass | Error message shows as intended |

**Test 4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| 07739900755 | 141 character long message | Message Box: SMS messages can only be 140 characters’ long | Message Box: SMS messages can only be 140 characters | Pass | Error message shows as intended |

**Email Messages:**

**Test 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| lee@live.co.uk | Money | Money | Email Saved! | Email Saved! | Pass | Works as intended |

**Test 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| ben@live.co.uk |  | Hello | Message Box: The subject box is empty | Message Box: The subject box is empty | Pass | Error message shows as intended |

**Test 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| lee@live.co.uk | Hello |  | Message Box: Please enter a message! | Message Box: Please enter a message! | Pass | Error message shows as intended |

**Test 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| lee@live.co.uk | Help | 1028-character message | Message Box: Emails can only be 1028 characters long! | Message Box: Emails can only be 1028 characters long! | Pass | Error message shows as intended |

**SIR Messages:**

**Test 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| lee@live.co.uk | SIR 09/1193 | Theft | SIR Saved! | SIR Saved! | Pass | Works as intended |

**Test 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| ben@live.co.uk | SIR 09/11/93 | 09-09 | Message Box: Sort Code format not recognised. Must be in format of XX-XX-XX! | Message Box: Sort Code format not recognised. Must be in format of XX-XX-XX! | Pass | Error message shows as intended |

**Test 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Email Address** | **Subject** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| lee@live.co.uk | SIR 09/11/16 | Hello | Message Box: SIR not recognised. Must be one contained in the list! | Message Box: SIR not recognised. Must be one contained in the list! | Pass | Error message shows as intended |

**Tweet Messages:**

**Test 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| @Lee | Hello | Tweet Saved! | Tweet Saved! | Pass | Works as intended |

**Test 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| @Ben | 140 character- message | Message Box: Tweets can only be 140 characters long! | Message Box: Tweets can only be 140 characters long! | Pass | Error message shows as intended |

**Test 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| @Liam |  | Message Box: Please enter a message! | Message Box: Please enter a message! | Pass | Error message shows as intended |

**Test 4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phone Number** | **Message** | **Expected Results** | **Actual Results** | **Results Pass** | **Comment** |
| @CraigBenLiamLees | Hello | Twitter address' must not exceed 15 characters! | Twitter address' must not exceed 15 characters! | Pass | Error message shows as intended |

## Acceptance Test and Logs

The system was demonstrated to a Peter a representative of Napier Bank and has stated that the application meets all the requirements needed for launch.

## Test Schedule

This part of the report shows the schedule that will be used to during testing.

|  |  |  |
| --- | --- | --- |
| **Test No:** | **Test Name:** | **Test Date:** |
| 1 | Unit Tests | 10/11/16 |
| 2 | Functional Test | 15/11/16 |
| 3 | Acceptance Test | 24/11/16 |

## Risks and Solutions

Here is a list of risks and solutions that are likely to appear when testing.

**SCHEDULE:**

* Testing schedule is tight. If the start of testing is delayed due to design tasks, the test cannot be extended beyond the scheduled start date.

Probability: Low

Impact: High

Solution: The testing team can start preparing the unit tests earlier to help lower the risk of falling behind.

**RESOURCES:**

* If resources prove to be insufficient, then it could take up to 15 days for them to be reacquired.

Probability: Medium

Impact: High

Solution: Holidays and vacations can be built into the schedule to mitigate this.

**DEFECTS:**

* A defect is found at the late stage of the cycle. Defects discovered are most likely due to unclear specifications and are often time consuming to solve.
* Probability: High
* Impact: High
* Solution: Extra tests are performed to ensure this does not happen.

# Version Control Plan

## Introduction

A version control system is a software package that observes your files for changes and allows you to bookmark the changes that are needing done at different levels so that they can be revisited whenever needed.

## Purpose

The version control plan documents and shows the necessary information required to effectively manage the software change. The document will explain what type of version control that is being used, what it is and how it will be used.

## Version Control System

The version control system that has been selected is Centralised. The Team Foundation Server has been selected to store this project.

Centralized version control systems are based on the idea that there is a single “central” copy of the project that is being developed sored somewhere (normally on a server) and the software team will “commit” their changes to this central copy.

“Committing” a change means recording the change in the central system. Other members of the software development team will also see the change and can elect to download the change and then the version control tool will automatically update the contents of any files that were edited (Lionetti, 2012).

## Connect to Version Control Server

Before they can connect to the server the software development team must set each member to a specific security group. After the security groups, have been set up they must open the Tools menu in Visual Studio and click Connect to Team Foundation Server. Then the software team should select the relevant Team Foundation Server from the drop down list.

## Primary Tasks

This section explains what the software development must do after connecting to the Team Foundation Server.

1. The first thing the software development team must do is Set Team Foundation version control as the default version control plugin for Visual Studio System Team Foundation Server.
2. Create the Napier Bank Message Filtering Service in visual studio
3. Add the project to version control and then create a local workspace where the software team can store, edit and test the uploaded version-controlled files.

## Secondary Tasks

This section explains what the software development team can do after they have completed the primary tasks.

1. The software development can check the solution for changes.
2. The software development team can add a file to the version control project and edit an existing one
3. The software team can commit any pending changes.

## Useful Features

The team will also be able to use any of the following features:

### Branching

The team will able to separate the projects to allow experimentation of code (basically a parallel universe). In doing so it will prevent the software development team having to freeze the current development and giving the QA team stability. The software development team members will be able to see the source code it contains and operate independently of the people that may be working on other branches.

File revision numbers within a branch have extra levels in their numbers e.g. 1.0 and 1.0.1.

### Merging

If the software development team fixes a bug in the release branch and knows the same bug will be present in the mainline code – it is often nice to merge the changes back to the mainline code.

The team will be able to choose between the two types of merging:

#### Strict Locking

All files that are checked out are flagged as being “read only”, until they are checked back in.

#### Optimistic Locking

Every developer gets to edit any checked out file. But, the repository will not allow a team member to check in a file that has been updated since the team member last checked it out. Instead, it asks the team member to update the local copy of the file to include the latest repository changes before checking it in.

If the software team were to use strict locking, it will be a lot of extra hassle with no payback. Each team member will probably be working on different areas of the application so optimistic locking will be the best option for merging.

## How it will be used

The software development team will have regular meetings to discuss how to effectively use this system. Any developers that make changes or make a new branch will create a changelog document to document what the change is and show where the change has been made.

The changelog will follow this format:

|  |  |  |
| --- | --- | --- |
| YYYY-MM-11 | Team Member’s Name | Email Address |
| 2016-11-16 | Lee | lee@napierbank.com |

Example Notes:

Validate Class

* Validation for twitter address length has been included in the sendMessage method and now should only allow twitter address’ that are below 15 characters in length.

# Software Evolution Strategy

## Introduction

Software change for the Napier Bank Message Filtering Service is inevitable. The system may meet new requirements when the software is used, the business environment may change, errors that appear must be repaired, new equipment may be needing added, or the performance and reliability of the system may have to be improved.

## Purpose

The software evolution plan documents and shows the necessary information required to effectively evolve the system or fix defects. The document will explain what software evolution is, discuss the cost analysis, maintenance activities and software re-engineering.

## Software Evolution

Software maintenance is now a widely-accepted part of SDLC. It stands for all the modifications and updates done after the delivery of a software product (tutorialspoint, n.d.). There are many reasons as stated above to do software evolution. The software development team may need to use the following types of maintenance stated below to perform the evolution.

### Types of Maintenance

These are the types of maintenance the software team may use to evolve the Napier Bank Message Filtering Service in the future:

#### Corrective Maintenance

This type of maintenance involves modifications or updates done to correct or fix problems, which are normally discovered by the user or by concluded user reports.

#### Adaptive Maintenance

This type of maintenance involves modifications or updates applied to keep the software up-to date and ready for the changing world of technology and the business environment.

#### Perfective Maintenance

This type of maintenance involves modifications or updates done to keep the software usable over very long periods of time. It usually includes new features, new user requirements for making minor changes to the software and improves the software’s reliability and performance.

#### Preventive Maintenance

This type of maintenance involves modifications or updates to prevent any problems cropping up in the future. It aims to service problems, which are not significant at this moment but may cause serious issues in the future.

## Cost Analysis

The cost of maintenance is high. A study on the estimating software maintenance found that the cost of maintenance can be as high as two thirds the cost of the entire software development cycle.

### Factors That May Affect Cost

There’re two factors that may affect the cost of maintenance and they are: Real World and Software End factors.

#### Real World Factors

* The standard age of any software is considered up to 10 to 15 years
* Older software that was designed to work on an older platform may not be able to challenge themselves against newly coming enhanced software’s on modern hardware.
* As technology advances, it then becomes too costly to maintain old software

#### Software End Factors

* Structure of the software program
* Programming Language
* Dependence on external environment
* Staff reliability and availability

## Software Maintenance Activities

The software development team will use these phases to perform maintenance of the application.

**Identification & Tracing**

This involves activates pertaining to identification of requirement of modification or maintenance. It often generated by the user or the system may sometime create report logs itself or error message. Also, the maintenance type needed is stated here also.

**Analysis**

The modification will be analysed for its impact on the current system including safety and security implications. If probable impact is severe, an alternative solution is looked for. A list of required changes or maintenance is analysed and estimation is concluded.

**Design**

New modules, which will need to be replaced or changed, are designed against requirement specifications set in the previous stage. Test cases will be created for validating and verification.

**Implementation**

The new modules are coded with the help of the structured design created in the design stage. All programmers are expected to do unit testing in parallel.

**System Testing**

Integration testing is done among the newly created modules. Integration testing will also be carried out between new modules and the system. Then the whole system will be tested.

**Acceptance Testing**

After testing the system internally, it is tested for acceptance with the help of users. If a user complains at this stage, they are noted to address in the next sprint.

**Delivery**

After acceptance test, the system is deployed all over the business either by using a small update package or fresh installation of the system. The final testing takes place at client end after the software is delivered and installed.

A Training facility may also be provided if required, in addition to the hard copy of user manual and installation manual.

**Maintenance Management**

Configuration management is an essential part of system maintenance. It is aided with version control tools to control versions, semi-version or patch management.

# Research

The software development team was required to research a different way of storing files that was different from text files and xml files. The alternative that was chosen was JavaScript Object Notation (JSON). JSON is a lightweight data-interchange format. It is easy for humans to read and write and it is also easy for computers to parse and generate (json, n.d.).

## Structure

JSON is built on two structures:

* A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
* An ordered list of values. This is normally realized as an array, vector, list or sequence.

Here is an example of a JSON file’s structure:

{"messageID":"E615131617","messageType":"SIR","sendType":"leesives@live.co.uk","subject":"SIR 11/06/16","message":"09-09-09\nCash Loss"}

## Appropriate API

The software development team decided to use the Java Script Serializer Class as it’s tool for processing JSON files (Microsoft, n.d.).

This API allows JSON files to be serialized and deserialized.

# Conclusion

The software team has been working hard over the last two months to ensure that Napier Bank is satisfied with the analysis, design and implementation of the product. Napier Bank will start to automate the filtering of messages more easily and allow the Napier Bank to save money with less staff being required to do the task manually.

# Bibliography

json, n.d. *Introducing JSON.* [Online]   
Available at: http://www.json.org/  
[Accessed 25 11 2016].

Lionetti, G., 2012. *What is Version Control: Centralized vs. DVCS.* [Online]   
Available at: http://blogs.atlassian.com/2012/02/version-control-centralized-dvcs/  
[Accessed 19 11 2016].

Microsoft, n.d. *JavaScriptSerializer Class.* [Online]   
Available at: https://msdn.microsoft.com/en-us/library/system.web.script.serialization.javascriptserializer(v=vs.110).aspx  
[Accessed 25 11 2016].

Microsoft, n.d. *Unit Testing.* [Online]   
Available at: https://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx  
[Accessed 07 11 2016].

Sourceforge, n.d. *StarUML - The Open Source UML/MDA Platform.* [Online]   
Available at: http://staruml.sourceforge.net/v1/  
[Accessed 07 11 2016].

tutorialspoint, n.d. *Functional Testing.* [Online]   
Available at: https://www.tutorialspoint.com/software\_testing\_dictionary/functional\_testing.htm  
[Accessed 23 11 2016].

tutorialspoint, n.d. *Software Maintenance Overview.* [Online]   
Available at: https://www.tutorialspoint.com/software\_engineering/software\_maintenance\_overview.htm  
[Accessed 24 11 2016].

tutorialspoint, n.d. *What is Acceptance Testing?.* [Online]   
Available at: https://www.tutorialspoint.com/software\_testing\_dictionary/acceptance\_testing.htm  
[Accessed 07 11 2016].

utdallas, n.d. *Download and Installation.* [Online]   
Available at: http://www.utdallas.edu/~supakkul/tools/RE-Tools/download-installation.html  
[Accessed 07 11 2016].