Test Document
Mindmap PIM

Client: IMINISYS

Team: A-Cube-N

Grobler, Arno 14011396

Lochner, Amy 14038600

Maree, Armand 12017800

Department of Computer Science, University of Pretoria

Contents

1 Introduction

This document provides details on the tests that were conducted on the system to increase stability, integity and scalability. The tests detailed in this document will be continuously updated as more tests are completed and more testing phases are entered.

2 Unit Tests

2.1 Processing Service

This section provides details on the unit tests that were conducted on the NaturalLanguageProcessor (NLP) service.

2.1.1 data.ProcessedData

The processed data class provides a class in which topics extracted by the NLP can be contained.

Test Name: testConstructor	**Test Date:** 20/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Ensure the conversion between RawData and ProcessedData occurs correctly.			
Expected result: Conversion occurred successfully.		**Actual Result:** Conversion occurred successfully.	
Steps: Copy member variables that occur in both RawData and ProcessedData and assign the given topics. Null as a topic array or userId should throw NullPointerException.			
Status: Pass			

2.1.2 listeners.RawDataListener

A class that is intended to dequeue RawData from RabbitMQ and send it to the NLP. Afterwards it should produce ProcessedData and add it to RabbitMQ for persistence.

Test Name: testRawDataListenerBean	**Test Date:** 25/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Ensure rawDataListener bean is initialized successfully.			
Expected result: RawDataListener is not null and no queue length.		**Actual Result:** RawDataListener is not null and no exception is thrown.	
Steps: Assert that rawDataListener is not null.			
Status: Pass			

Test Name: receiveRawData	**Test Date:** 25/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Ensure RawData is dequeued from RabbitMQ.			
Expected result: RawData dequeued successfully.		**Actual Result:** RawData dequeued successfully.	
Steps: Add RawData to RabbitMQ and check the queue to make sure it gets dequeued by monitoring the queue length.			
Status: Pass			

Test Name: testProcess	**Test Date:** 25/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Pass the raw data to the NLP and make sure the expected topics is extracted.			
Expected result: Expected topics are returned from NLP.		**Actual Result:** Expected topics are returned from NLP.	
Steps: Construct a RawData object and give it to the processor and evaluate the returned topics.			
Status: Pass			

2.1.3 main.Application

The main Spring Boot class that starts up the application.

Test Name: testBeans	**Test Date:** 14/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Make sure RabbitTemplate and NaturalLanguageProcessor beans are instantiated correctly.			
Expected result: Both beans are not null and no exception is thrown.		**Actual Result:** Both beans are not null and no exception is thrown.	
Steps: Assert the bean is not null.			
Status: Pass			

2.2 Database Service

This is the service responsible for database interaction.

2.2.1 data.Topic

A single topic in the database.

Test Name: testGetWeight	**Test Date:** 28/07/2016	**Type:** Manual	**Tester:** Armand Maree
Description: Method uses avaliability heuristic to calculate the weight a topic should have.			
Expected result: Test the weight of various topics and make sure the weight of each is correct compared to other topics based on temperal and fequency components.		**Actual result:** Most topics were placed in the correct position.	
Steps: Create a few topics and calculate their weight. Use the weight to sort the topics and then check if the topics are in the correct order.			
Comments: The accuracy for this component is what is measured rather than its ability perfectly order topics.			
Status: Pass			

2.2.2 listeners.ProcessedDataListener

This class dequeues ProcessedData from RabbitMQ and persists it in the database. It also creates a topic object for each topic and persists that.

Test Name: testReceiveProcessedData	**Test Date:** 24/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Make sure the processedData is dequeued, correctly persisted and the corresponding topics are correctly extracted and persisted.			
Expected result: No duplicated topics in topic database. ProcessedData stored correctly.		**Actual result:** If two threads concurrently try to store the same topic then a duplicate topic is found in database.	
Steps: Create a processedData object and send it to RabbitMQ. Check the processedData repository for the correct data. Check the topic repository for no duplicates and correct content.			
Status: Fail			

2.2.3 main.Application

The main application that starts Spring Boot.

Test Name: testBeans	**Test Date:** 25/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Make sure the beans for all the repositories are correctly set up and the RabbitTemplate bean is correct.			
Expected result: No bean must be null and no exception mut be thrown.		**Actual result:** No bean must be null and no exception mut be thrown.	
Steps: Assert that the repositories and rabbitTemplate is not null.			
Status: Pass			

2.3 Business Service

This service serves as a middle man between the frontend and the backend.

2.3.1 listeners.FrontendListener

This class receives all messages from the frontend by dequeuing from RabbitMQ.

Test Name: testReceiveTopicRequest	**Test Date:** 28/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Method receives a TopicRequeuest from a RabbitMQ queue and sends it to the database.			
Expected result: One queue is dequeued and another is enqueued.		**Actual result:** One queue is dequeued and another is enqueued.	
Steps: Create a TopicRequest and send it to RabbitMQ and monitor both queues.			
Status: Pass			

Test Name: testReceiveRegistration	**Test Date:** 25/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Receives a user registration and extracts a user to be sent to the database and an auth token to be sent to the pollers.			
Expected result: Correct user and auth token information is extracted and sent to RabbitMQ.		**Actual result:** Correct user and auth token information is extracted and sent to RabbitMQ.	
Steps: Create a UserRegistration object and send it to RabbitMQ. Monitor Database queue and polling queue to make sure correct information is extracted.			
Status: Pass			

Test Name: testBeans	**Test Date:** 25/07/2016	**Type:** Automatic	**Tester:** Armand Maree
Description: Make sure all beans are instantiated correctly.			
Expected result: No bean is null and no exception is thrown.		**Actual result:** No bean is null and no exception is thrown.	
Steps: Assert that no bean is null.			
Status: Pass			

2.4 Gmail Polling Service

This service retrieves emails from Gmail and sends them for processing by the NLP.

2.4.1 poller.GmailPoller

This class exchanges auth tokens for access codes and polls Gmail for emails of the user.

Test Name: testOldEmails	**Test Date:** 21/07/2016	**Type:** Manual	**Tester:** Armand Maree
Description: A loop iterates through all of the emails already present a user's mail box. After it received these emails it parses them into RawData and sends them to RabbitMQ.			
Expected result: All emails should be processed and all text should be extracted from emails.		**Actual result:** After one page of processing the server encounters a NullPointerException.	
Steps: Provide the poller with a valid auth token and monitor the raw data it produces.			
Status: Fail			

Test Name: testNewEmails	**Test Date:** 21/07/2016	**Type:** Manual	**Tester:** Armand Maree
Description: After old emails has been processed the poller most wait for new emails and process them into raw data.			
Expected result: All new emails should be processed and all text should be extracted from emails.		**Actual result:** All new emails should be processed and all text should be extracted from emails.	
Steps: Provide the poller with a valid auth token, send a new email to the email address and monitor the raw data it produces.			
Status: Pass			

2.5 Frontend Service

This is the Spring MVC that provides the web clients with the content and server interaction they need.

2.5.1 pim.LoginController

This class handles all the login/sign up request from web clients via websockets.

Test Name: testReceiveUserRegistration	**Test Date:** 27/07/2016	**Type:** Automatic	**Tester:** Arno Grobler

Description: When a user signs up the a user registration is sent to the app and this must be sent to the business. It then has to wait and reply with an ID the client can use to later rquest topics.

Expected result: Successfully receive and parse the incoming JSON object and send it to Business. The service must wait until it receives an ID and then return that.

Actual result: Successfully received and parsed the incoming JSON object and sent it to Business. The service waited until it received an ID and then returned that.

Steps: Provide it with a UserRegistration object and make sure it is sent to RabbitMQ, then return an ID and make sure that is returned to the web client.

Status: Pass

2.5.2 pim.TopicController

This class handles all the requests for new topics

Test Name: testReceiveTopicRequest	**Test Date:** 27/07/2016	**Type:** Automatic	**Tester:** Arno Grobler

Description: When a user sends a topic request this method must send it to the business and wait for the toics to be returned. It must then return the topics to the web client.

Expected result: Successfully receive and parse the incoming JSON object and send it to Business. The service must wait until it receives the topics and then return that.

Actual result: Successfully received and parsed the incoming JSON object and sent it to Business. The service waited until it received the topics and then returned that.

Steps: Provide it with a TopicRequest object and make sure it is sent to RabbitMQ, then return an array of Topics and make sure that is returned to the web client.

Status: Pass