Unit Test Plan & Report
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

1.1 Purpose

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.

It is essential for this system to be developed in a test driven environment since a minor fault in one of the services could lead to large part of the system not functioning as expected. Thus fully automatic white-box testing will be applied as far as possible with the exception of a few manual tests.

1.2 Test Environment

- **Programming Languages**:
  - Java
  - Javascript
- **Testing Frameworks**:
  - *JUnit* will be used for testing the backend Java implementation.
  - *Dalek.js* will be used to test the frontend Javascript, HTML and CSS.
- **Operating Systems**:
  - *Linux* (more specifically *CentOS*) is the operating system that is running on our servers that host all our services. It is thus essential that the all services function as expected on this operating system.
- **Internet Browsers**:
  - *PhantomJS and Google Chrome* will be tested during the automated unit tests conducted by *Dalek.js*.

2 Test Cases

2.1 Business Logic Test Cases

2.1.1 Main Application Bean Test

Objective Make sure all the Spring beans are created and can be injected. This will at least indicate on a high level of grandularity that the Spring Framework did detect the bean declaration and can inject an appropriote object.

Pass Criteria All the beans set up with the *Bean* annotation needs to have a non-null value when autowired (injected).

Fail Criteria Any of the beans have a null value.

2.1.2 FrontendListener Register Request Test 1

Objective Make sure the BusinessLogic service can receive a request to register a user who only chose one PIM source and send orders for other services to start some task.

Input A UserRegistrationIdentified object that contains all the neccesary information to register only a Gmail user.

Output A single UserIdentified object that will be retrieved from the RabbitMQ queue that leads to the database and a single AuthCode object that will be retrieved from the RabbitMQ queue that leads to the GmailPoller.

Pass Criteria

- UserIdentified object's fields must contain the same original values as was specified in the UserRegistrationIdentified object.
- AuthCode object's fields must correspond to the original fields specified in the UserRegistrationIdentified object.

Fail Criteria

- There is no UserIdentified object sent to the database.
- There is no AuthCode object sent to the GmailPoller.
- Any of the fields differ.

2.1.3 FrontendListener Register Request Test 2

Objective Make sure the BusinessLogic service can receive a request to register a user who chose two PIM source and send orders for other services to start some task. It is assumed that if the test for two PIM sources succeeds then more than two will also succeed since the algorithm should be generic.

Input A UserRegistrationIdentified object that contains all the neccesary information to register a user that signed up with their Gmail and Facebook account.

Output A single UserIdentified object that will be retrieved from the RabbitMQ queue that leads to the database. A single AuthCode object that will be retrieved from the RabbitMQ queue that leads to the GmailPoller. A single AuthCode object that will be retrieved from the RabbitMQ queue that leads to the FacebookPoller.

Pass Criteria

- UserIdentified object's fields must contain the same original values as was specified in the UserRegistrationIdentified object.
- AuthCode object's fields must correspond to the original fields specified in the UserRegistrationIdentified object.

Fail Criteria

- There is no UserIdentified object sent to the database.
- There is no AuthCode object sent to the GmailPoller.
- There is no AuthCode object sent to the FacebookPoller.
- Any of the fields differ.

2.1.4 FrontendListener Register Request Test 3

Objective Make sure the BusinessLogic service can receive a request to register a user who chose no PIM source. This situation should never occur.

Input A UserRegistrationIdentified object that contains all the neccesary information to register a user that did not sign up with an account.

Output A single UserIdentified object that will be retrieved from the RabbitMQ queue that leads to the database.

Pass Criteria

- UserIdentified object's fields must contain the same original values as was specified in the UserRegistrationIdentified object.

Fail Criteria

- There is no UserIdentified object sent to the database.
- Any of the fields differ.

2.1.5 FrontendListener User Update Test 1

Objective Make sure the BusinessLogic service can receive a request to update a user who removed a PIM.

Input A UserUpdateRequestIdentified object that contains all the neccesary information to update any detail about a user including the PIM that has to be removed.

Output A single UserIdentified object that will be retrieved from the RabbitMQ queue that leads to the database And an AuthCode object that will be retrieved from the RabbitMQ queue that leads to the PIM poller that should be stopped.

Pass Criteria

- UserIdentified object's fields must contain the same original values as was specified in the UserUpdateRequestIdentified object.
- AuthCode object's fields must contain the same original values as was specified in the UserUpdateRequestIdentified object.

Fail Criteria

- There is no UserIdentified object sent to the database.
- There is no AuthCode object sent to the poller.
- Any of the fields differ.

2.1.6 FrontendListener User Update Test 2

Objective Make sure the BusinessLogic service can receive a request to update a user who adds a PIM.

Input A UserUpdateRequestIdentified object that contains all the neccesary information to update any detail about a user including the PIM that has to be removed.

Output A single UserIdentified object that will be retrieved from the RabbitMQ queue that leads to the database And an AuthCode object that will be retrieved from the RabbitMQ queue that leads to the PIM poller that should be added.

Pass Criteria

- UserIdentified object's fields must contain the same original values as was specified in the UserUpdateRequestIdentified object.
- AuthCode object's fields must contain the same original values as was specified in the UserUpdateRequestIdentified object.

Fail Criteria

- There is no UserIdentified object sent to the database.
- There is no AuthCode object sent to the poller.
- Any of the fields differ.

2.1.7 FrontendListener User Update Test 3

Objective Make sure the BusinessLogic service can receive a request to update a user who changed no PIM source.

Input A UserUpdateRequestIdentified object that contains all the neccesary information to update any detail about a user.

Output A single UserIdentified object that will be retrieved from the RabbitMQ queue that leads to the database.

Pass Criteria

- UserIdentified object's fields must contain the same original values as was specified in the UserUpdateRequestIdentified object.

Fail Criteria

- There is no UserIdentified object sent to the database.
- Any of the fields differ.

2.2 Database Service Test Cases

2.2.1 Main Application Bean Test

Objective Make sure all the Spring beans are created and can be injected. This will at least indicate on a high level of grandularity that the Spring Framework did detect the bean declaration and can inject an appropriote object.

Pass Criteria All the beans set up with the *Bean* annotation needs to have a non-null value when autowired (injected).

Fail Criteria Any of the beans have a null value.

2.2.2 BusinessListener Register Request Test 1

Objective Make sure the Database service can register a user where none of the PIMs occur in any of the other users in the database.

Input A UserIdentified object that contains all the neccesary information to register a user.

Output A new user in the database plus the registered UserIdentified must be sent to the frontend RabbitMQ queue.

Pass Criteria

- UserIdentified object's in the database where all fields correspond to the request's fields.
- UserIdentified object's returned to tghe frontend where all fields correspond to the request's fields.

Fail Criteria

- The user is not persisted.
- No user is returned to the frontend.
- Any of the fields in the database or the returned user differ.

2.2.3 BusinessListener Register Request Test 2

Objective Make sure the Database service can register a user where one of the PIMs occur in any of the other users in the database.

Input A UserIdentified object that contains all the neccesary information to register a user.

Output Update user in the database to contain the new PIMs plus the registered UserIdentified must be sent to the frontend RabbitMQ queue.

Pass Criteria

- UserIdentified object's in the database where all fields correspond to the request's fields.
- UserIdentified object's returned to the frontend where all fields correspond to the request's fields.

Fail Criteria

- The user is not persisted.
- No user is returned to the frontend.
- Any of the fields in the database or the returned user differ.

2.2.4 BusinessListener Register Request Test 3

Objective Make sure the Database service can register a user where the user that needs to be registered does not contain any PIMs.

Input A UserIdentified object that contains all the neccesary information to register a user excluding any PimIds.

Output No new user in the database the original request should be sent back to the frontend RabbitMQ queue.

Pass Criteria

- No new objects stored in the database.
- UserIdentified object's returned to the frontend where all fields correspond to the request's fields.

Fail Criteria

- The user is persisted.
- No user is returned to the frontend.
- No changes were made to the database.

2.2.5 BsuinessListener User Update Test 1

Objective Make sure the Database service can receive a request to update a user who removed a PIM.

Input A UserIdentified object that contains all the neccesary information to update any detail about a user including the PIM that has to be removed.

Output A single UserUpdateResponseIdentified object that will be retrieved from the RabbitMQ queue that leads to the frontend.

Pass Criteria

- Updated values must be reflected in the user that was updated in the database.
- UserUpdateResponseIdentified must be sent to the frontend.

Fail Criteria

- There is no UserUpdateResponseIdentified object sent to the frontend.
- No changes were made to the database.

2.2.6 BusinessListener User Update Test 2

Objective Make sure the Database service can receive a request to update a user who adds a PIM.

Input A UserIdentified object that contains all the neccesary information to update any detail about a user including the PIM that has to be removed.

Output A single UserUpdateResponseIdentified object that will be retrieved from the RabbitMQ queue that leads to the frontend.

Pass Criteria

- Updated values must be reflected in the user that was updated in the database.
- UserUpdateResponseIdentified must be sent to the frontend.

Fail Criteria

- There is no UserUpdateResponseIdentified object sent to the frontend.
- No changes were made to the database.

2.2.7 BusinessListener User Update Test 3

Objective Make sure the Database service can receive a request to update a user who changed no PIM source.

Input A UserUpdateRequestIdentified object that contains all the neccesary information to update any detail about a user.

Output A single UserUpdateResponseIdentified object that will be retrieved from the RabbitMQ queue that leads to the database.

Pass Criteria

- Updated values must be reflected in the user that was updated in the database.
- UserUpdateResponseIdentified must be sent to the frontend.

Fail Criteria

- There is no UserUpdateResponseIdentified object sent to the frontend.
- No changes were made to the database.

2.2.8 BusinessListener User Update Test 3

Objective Make sure the Database service can receive a request to update a user who does not exist.

Input A UserIdentified object that contains all the neccesary information to update any detail about a user.

Output A single UserUpdateResponseIdentified object that will be retrieved from the RabbitMQ queue that leads to the frontend.

Pass Criteria

- No changes must be made to the database.
- Updated UserIdentified must be sent to the frontend.

Fail Criteria

- There is no UserUpdateResponseIdentified object sent to the frontend.
- Any changes were made to the database.

2.2.9 BusinessListener User Update Test 3

Objective Make sure the Database service can receive a request to update a user who requires no changes.

Input A UserIdentified object that contains all the neccesary information to update any detail about a user.

Output A single UserUpdateResponseIdentified object that will be retrieved from the RabbitMQ queue that leads to the frontend.

Pass Criteria

- No changes must be made to the database.
- Updated UserIdentified must be sent to the frontend.

Fail Criteria

- There is no UserUpdateResponseIdentified object sent to the frontend.
- Any changes were made to the database.

2.2.10 ProcessedDataListener Test Case 1

Objective Make sure the Database service can receive ProcessedData, persist it and persist the corresponding topics.

Input A ProcessedData object with no involved contacts.

Output A ProcessedData object in the database and a corresponding Topic in the topic database.

Pass Criteria

- New ProcessedData object in the processed data repository.
- For each topic in the processed data there should be a topic in the topic repository.

Fail Criteria

- ProcessedData object is not stored in the repository.
- Some but not all topics are persisted.
- No topics are persisted.

2.2.11 ProcessedDataListener Test Case 2

Objective Make sure the Database service can receive ProcessedData, persist it and persist the corresponding topics.

Input A ProcessedData object with involved contacts.

Output A ProcessedData object in the database and a corresponding Topic in the topic database and corresponding contacts.

Pass Criteria

- New ProcessedData object in the processed data repository.
- For each topic and involvedContact in the processed data there should be a topic in the topic repository.

Fail Criteria

- ProcessedData object is not stored in the repository.
- Some but not all topics/contacts are persisted.
- No topics/contacts are persisted.

2.2.12 TopicListener Test Case 1

Objective Make sure the Database service can retrieve persisted topics based on various paths and exclude lists.

Input A TopicRequest object.

Output A TopicResponse object retrieved from the RabbitMQ queue that leads to the Business Service.

Pass Criteria

- The TopicResponse must be sent and contain unique topics.

Fail Criteria

- No TopicResponse is sent back.
- TopicResponse objects does not contain any topics.
- TopicResponse object does not contain unique topics.

2.2.13 TopicListener Test Case 2

Objective Make sure the Database service can retrieve contacts when topics are requested.

Input A TopicRequest object.

Output A TopicResponse object retrieved from the RabbitMQ queue that leads to the Business Service.

Pass Criteria

- The TopicResponse must be sent and contain unique topics and contacts.

Fail Criteria

- No TopicResponse is sent back.
- TopicResponse objects does not contain any topics or contacts.
- TopicResponse object does not contain unique topics or contacts.

2.3 FacebookPoller Service Test Cases

2.3.1 Main Application Bean Test

Objective Make sure all the Spring beans are created and can be injected. This will at least indicate on a high level of grandularity that the Spring Framework did detect the bean declaration and can inject an appropriote object.

Pass Criteria All the beans set up with the *Bean* annotation needs to have a non-null value when autowired (injected).

Fail Criteria Any of the beans have a null value.

2.3.2 ItemRequestListener Test Case 1

Objective Make sure an array of Facebook posts can be retrieved and sent to the frontend.

Input An ItemRequest object.

Output A ItemResponse object retrieved from the RabbitMQ queue that leads to the Business Service.

Pass Criteria

- The ItemResponse object must have an array of facebook posts in the form of HTML iframe elements that will be rendered by the browser in the frontend.

Fail Criteria

- No ItemResponse is sent back.
- The items in the ItemResponse is not the correct format (HTML iframe format).

2.4 GmailPoller Service Test Cases

2.4.1 Main Application Bean Test

Objective Make sure all the Spring beans are created and can be injected. This will at least indicate on a high level of grandularity that the Spring Framework did detect the bean declaration and can inject an appropriote object.

Pass Criteria All the beans set up with the *Bean* annotation needs to have a non-null value when autowired (injected).

Fail Criteria Any of the beans have a null value.

2.5 Processor Service Test Cases

2.5.1 Main Application Bean Test

Objective Make sure all the Spring beans are created and can be injected. This will at least indicate on a high level of grandularity that the Spring Framework did detect the bean declaration and can inject an appropriote object.

Pass Criteria All the beans set up with the *Bean* annotation needs to have a non-null value when autowired (injected).

Fail Criteria Any of the beans have a null value.

2.5.2 NaturalLanguageProcessor Test Case 1

Objective Given a set of strings the NLP has to show that it can extract specific topics.

Input A RawData object.

Output A ProcessedData object retrieved from the RabbitMQ queue that leads to the Database Service.

Pass Criteria

- The ProcessedData object has to consist of only a predefined set of topics.

Fail Criteria

- Any additional topics were extracted.
- Not all predefined topics were extracted.
- No topics were extracted.

2.6 Frontend Service Test Cases

2.6.1 Login Dalekjs Test

Objective Test if the a user can login successfully with the Google sign in button.

Input Username and password of Acuben Cos(test account).

Output GAPI object returned from a successful login containing Acuben Cos's information.

Pass Criteria

- GAPI object returned is not null
- GAPI object is that of Acuben Cos's
- Text of the login box must change to say "Welcome Acuben Cos"
- Must redirect to home page.

Fail Criteria

- GAPI object is null
- User is redirected to login page again.
- Text of the login box stays empty.

2.6.2 Retrieve Topics Dalekjs Test

Objective Test if the a user can ask for topics and successfully retrieve and update the mind map.

Input TopicRequest object.

Output TopicResponse object containing both topic names and PIM data lists for all topics(This is the corresponding data items from emails and posts).

Pass Criteria

- TopicResponse is not null.
- Topic names list is empty or has topics.
- Topic PIM data lists for all topics are empty or has data.
- Loading div disappears.

Fail Criteria

- Topic names list is null.
- Topic PIM data lists are null.
- Error div appears
- No response is returned after x amount of seconds.

2.6.3 Logout Dalekjs Test

Objective Test if the a user can logout successfully.

Input User input to logout.

Output None.

Pass Criteria

- User successfully navigated to login page.
- All cookies are deleted.
- User cannot get back to the home page again without logging in again.

Fail Criteria

- User stays on home page.
- Cookies are not deleted.

3 Test Results

3.1 BusinessLogic Service Test Cases

All the tests for this service can be found on Github at `https://github.com/ArmandMaree/MindMapPIM/tree/master/BusinessLogic/src/test/java/testers`.

3.1.1 Main Application Bean Test (TC 2.1.1)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.1.2 FrontendListener Register Request Test 1 (TC 2.1.2)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.1.3 FrontendListener Register Request Test (TC 2.1.3)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.1.4 FrontendListener Register Request Test (TC 2.1.4)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.1.5 FrontendListener User Update Test (TC 2.1.5)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.1.6 FrontendListener User Update Test (TC 2.1.6)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.1.7 FrontendListener User Update Test (TC 2.1.7)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2 Database Service Test Cases

All the tests for this service can be found on Github at `https://github.com/ArmandMaree/MindMapPIM/tree/master/Database/src/test/java/testers`.

3.2.1 Main Application Bean Test (TC 2.2.1)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.2 BusinessListener Register Request Test 1 (TC 2.2.2)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.3 BusinessListener Register Request Test 2 (TC 2.2.3)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.4 BusinessListener Register Request Test 3 (TC 2.2.4)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.5 BusinessListener User Update Test 1 (TC 2.2.5)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.6 BusinessListener User Update Test 2 (TC 2.2.6)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.7 BusinessListener User Update Test 3 (TC 2.2.7)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.8 ProcessedDataListener Test 1 (TC 2.2.10)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.9 ProcessedDataListener Test 2 (TC 2.2.11)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.10 TopicListener Test 1 (TC 2.2.12)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.2.11 TopicListener Test 2 (TC 2.2.13)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.3 FacebookPolling Service Test Cases

All the tests for this service can be found on Github at `https://github.com/ArmandMaree/MindMapPIM/tree/master/FacebookPolling/src/test/java/testers`.

3.3.1 Main Application Bean Test (TC 2.3.1)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.3.2 ItemRequestListener Test Case 1 (TC 2.3.2)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.4 GmailPolling Service Test Cases

All the tests for this service can be found on Github at `https://github.com/ArmandMaree/MindMapPIM/tree/master/GmailPolling/src/test/java/testers`.

3.4.1 Main Application Bean Test (TC 2.4.1)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.5 Processor Service Test Cases

All the tests for this service can be found on Github at `https://github.com/ArmandMaree/MindMapPIM/tree/master/Processing/src/test/java/testers`.

3.5.1 Main Application Bean Test (TC 2.5.1)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.6 Frontend Service Test Cases

All the tests for this service can be found on Github at `https://github.com/ArmandMaree/MindMapPIM/tree/master/Processing/src/test/javascript/loginTester.js`.

3.6.1 Frontend Login Dalekjs Test (TC 2.6.1)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.6.2 Frontend Topic Recieve Dalekjs Test (TC 2.6.2)

Result Pass

Comment All conditions tested in this test case executed successfully.

3.6.3 Frontend Logout Dalekjs Test (TC 2.6.3)

Result Pass

Comment All conditions tested in this test case executed successfully.

4 Additional Comments

All the objects that will be used in the input section of each test will be mocked in order to force certain criteria to be tested. Any RabbitMQ queues that lead out of the service that is being tested will have a dequeuer setup that only runs in the test environment.

Also note that the pollers is not tested during the unit tests due to the need for them to receive authentication codes as it is outlined in the OAUTH2 protocol. In order to gain these authentication codes a manual sign in has to occur by a user. Large parts of these services will thus be tested during the integration tests.

5 Conclusion

All unit tests passed which is an idication that the application is ready to be deployed on the server.