UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

Longsword Notification

Phase 4 Testing Broadsword Notification

Quinton Swanepoel	15245510
Frederick Ehlers	11061112
Tshepo Malesela	14211582
Kyle Erwin	15015302

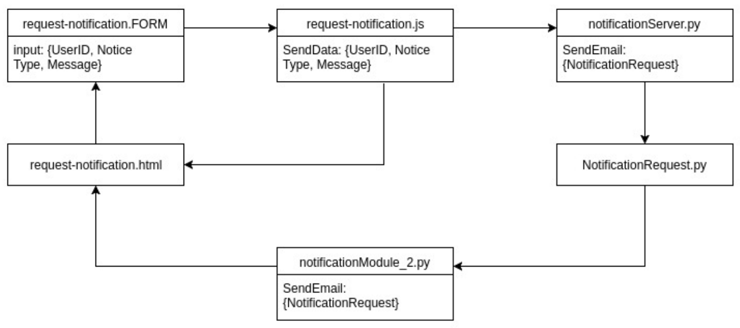
Stakeholders

Computer Science Department of University of Pretoria:	Vreda Pieterse
Team being tested:	Broadsword Notification

Contents

1 High level functional level requirements test models

Find below the figure for the high level requirements test model


request-notification.FORM
input: {UserID, Notice Type, Message}
request-notification.js
SendData: {UserID, Notice Type, Message}
notificationServer.py
SendEmail: {NotificationRequest}
request-notification.html
NotificationRequest.py
notificationModule_2.py
SendEmail: {NotificationRequest}

2 Lists of service contracts tested

2.1 Pre-conditions

- User details such as email address do exist and are valid.
- Notification request is complete with details of the email for example the subject and body of the email.
- The user receiving the email exists in the system

2.1.1 Post-conditions

- At least one notification has been sent to the appropriate user/users.
- System should log who Notifications are sent to, if the Notification request is not successful, the error message should be logged as well.

The Notifications services should only fail under Three conditions that are:

- Request has failed to meet the validation criteria specified in the post conditions.
- The request information is incomplete or has an error.
- There actually is no notification for the user.

2.2 Team Broadsword Notification Services

For this section we inspected the code of the Team Broadsword email services.

- In the code, the user details are supplied to send an email, this includes email, subject and body.

- The email is not verified that it truelly is a real email address, therefore for fake user email, the code will produce exceptions.
- The notification services of the Broadsword team does work, There are however no batch emails have been implemented.
- The system does have a method to log the email notifications, this is done by saving he notifications request object.

3 Evaluate the test cases of the previous team for non-functional requirements using your own criteria

3.1 Create/Submit a Notification Request Through Browser

Mark: 10/10
Description: The HTML page accepts an email address and message which can then be submitted through a javascript function which retrieves the information entered and creates a notification request. The information is sent to this request object using a JSON string. This request is submitted using the ”Post” method.

3.2 Verify Email Address

Mark: 4/10
Description: The email address is not verified that it truly is a real email address, therefore for fake user emails, the code will produce exceptions. However these exception are somewhat accounted for.

3.3 Handle Incoming Notification Request From Browser

Mark: 8/10
Decription: The notification module does handle the incoming requests submitted from the browser using a python script.This script uses and SMTP server to send the email through to the supplied email address. The is no way of handling batch email requests however, so several email notification requests have to be created in order to mimic this batch functionality.

3.4 Create Web Server To Handle Incoming Requests

Mark: 10/10
Description: A notification HTTP server is created and implemented to handle all notification requests. This sever is implemented using a python script and once again makes use of SMTP.

3.5 Log Notifications

Mark: 10/10
Description : The system does have a method to log the email notifications, this is done by saving he notifications request object.

4 Create a list of non-functional requirements tested

4.1 Integrability

Mark: 8/10
Comments : The functions are well defined making them easy to understand resulting in a notification module that can easily be plugged into another application/program. The file structure can be improved on however as there were a lot of redundant files in the final Git repository.

4.2 Maintainability

Mark : 9/10
Comments : The code is readable and simple to understand. The modularity also makes the maintenance of the program simpler as errors in a certain part of the program are easier to locate and rectify.

4.3 Scalability

Mark: 8/10
Comments : Due to the fact that a Gmail account is being used to send the emails, a maximum limit of 500 emails can only be sent per day. This can be rectified by using the university's email server.

4.4 Reliability

Mark: 6/10
Comments : If all required fields have the correct values such as email,body and subject then the email will go through, but there is no error checking to check if the parameters actually contain valid values, and if the email address supplied is incorrect then the code runs into errors.

4.5 Security

Mark: 6/10
Comments : Since they are making use of the Google API, a certain level of encryption is provided in the API itself. The password to the account is however visible in the code and not encrypted.

4.6 Accessibility

Mark: 9/10
Comments : The email services can be used and are always accessible through the notification server as long as the account has not reached the limit of 500 emails for the day, after which the system will be completely unresponsive or result in errors.

5 Evaluate the test cases of the previous team for non-functional requirements using your own criteria

5.1 Modularity

Mark: 8/10
We felt that the broadsword notification team developed well written and modularized code. Each class has designed for a unique purpose and did nothing outside of that purpose. The functions themselves were not over complicated However they could have broken them up to into separate tasks.

5.2 In-code Comments

Mark: 6/10
The comments for their implementation were decent however there were places were a comment could have been added or expanded upon. The classes themselves each have a description of what the classes are supposed to do but again were not explained upon.

5.3 Capacity

Mark: 6/10
The team implemented the ability to send a single a email however they didn't provide the ability to send batch emails.

5.4 Error Handling

Marks: 7/10
The error handling was implemented very well with in their python script, which is the dominant language they used. Every error thrown was caught and handled such as unsupported http requests. Their java script, however, did not provided any error handling on some object creations. This could lead to the function ending unexpectedly.

6 Add more tests if required

There were no tests found in the team Broadsword's Github repository. We would use the following python's built in libraries to test different cases

6.1 Test cases for the NSQ server

- A test to ensure the NSQ server has succesfully started and is ready to be used:
  - Did it output "Starting"
  - Can you connect to the 127.0.0.1:4160 address
  - Did it output "Message service is running..."
- A test to see if the server failed to start but it is expected to not start:
  - Did it output "Starting"
  - HTTP status code 404 when accessing the 127.0.0.1:4160 address

6.2 Test cases for the notification server

- A test to see that the Notification server has successfully started and is ready to be used
  - In the test's setup create a JSON object with correct data.
  - Call sendEmail(self, request)
  - Check that the email's body was what is expected as per the setup process
  - Ensure all other aspects such as the To addres, From address and subject is also correctly set
  - Confirm with the writer.py's output that the email was sent succesfully
- A test to see if sendEmail failed as expected with incorrect data
  - In the test's setup create a JSON object with incorrect data to cause the email to fail.
  - Call sendEmail(self, request)
  - Check that all aspects of the email are as expected with the incorrect data in mind.
  - Confirm that the function hit the error and was caught in the try catch
- Test cases for the logNotification function's implementation
  - In the test's setup, create and send a successful email
  - Print out the latest message in the log and it confirm that it is what is expected to be logged