LONGSWORD NOTIFICATION

Phase 4 Testing Broadsword Notification

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

STAKEHOLDERS

Computer Science Department
of University of Pretoria:

Vreda Pieterse

Team being tested:

Broadsword Notification

Contents

1 Create a list of service contracts tested

1.1 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

1.1.1 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.

1.2 Team Broadsword Notification Services

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

1.2.1 Create/Submit a Notification Request Through Browser

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.

Mark : 10

1.2.2 Verify Email Address

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.

Mark : 4

1.2.3 Handle Incoming Notification Request From Browser

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.

Mark : 8

1.2.4 Create Web Server To Handle Incoming Requests

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.

Mark : 10

1.2.5 Log Notifications

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

Mark : 10

2 Create a list of non-functional requirements tested

2.1 Integrability

Mark : 8

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.

2.2 Maintainability

Mark : 9

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.

2.3 Scalability

Mark : 8

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.

2.4 Reliability

Mark : 6

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.

2.5 Security

Mark : 6

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.

2.6 Accessibility

Mark : 9

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.

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

3.1 Modularity

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.
8/10

3.2 In-code Comments

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.
6/10

3.3 Capacity

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

3.4 Error Handling

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.
7/10

3.5 Use Cases

4 Add more tests if required