**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR Nagar, MSRIT post, Bangalore-54.

A **Testing Document** for

**BUILDING A WEB SERVICE FRAMEWORK (STOW-RS) FOR MEDICAL IMAGING (DICOM)**

**Submitted By**:

**NEHA PATHAPATI**

**(USN:** 1MS12CS143)

**COMPANY:** PHILIPS

**BUSINESS UNIT:** HEALTH SYSTEMS

*in partial fulfillment for the award of the degree of*

# *Bachelor of Engineering in Computer Science & Engineering*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**M.S.RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu), **April 2016**

**TESTING**

**I. INTRODUCTION**

Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product: Meets the business and technical requirements that guided its design and development.

For industry standard projects, testing is as critical as development to meet the requirements of the client.

Unit testing was carried out by developers for the code written by them. The testing team has executed manual and automated testing. However, due to the adaptation of the Scaled Agile Process Model, as the code evolves, testing has to be conducted periodically.

**II.TESTING TOOLS AND ENVIRONMENT**

1. **NUNIT TESTING FRAMEWORK – UNIT TESTING**

NUnit is an open-source unit testing framework for Microsoft .NET. It serves the same purpose as JUnit does in the Java world, and is one of many programs in the xUnit family. NUnit is a commonly used framework for all .Net languages. Initially ported from JUnit, the current production release, version 3.0, has been completely rewritten with many new features and support for a wide range of .NET platforms.

Fundamentally, NUnit testing involves writing test scripts to pass varying parameters to individual functions in the code. Then, we compare the expected return value of the function with the actual value that is returned using the static methods of the Assert class.

Some of the features of NUnit framework include:

* Tests can be run from a console runner, within Visual Studio through a Test Adapter or through 3rd party runners.
* Tests can be run in parallel.
* Strong support for data driven tests.
* Supports multiple platforms including .NET Core.
* Every test case can be added to one or more categories, to allow for selective running.

The individual functions of Storage Library were tested using NUnit testing framework by the developers.

The open-source NUnit Framework plug-in was installed using Nuget Package Manager in Microsoft Visual Studio 2015 (on Windows) for testing of the code.

1. **APACHE JMETER – PERFORMANCE TESTING**

Apache JMeter is a 100% pure Java application designed to load test client/server software, such as a web service. It may be used to test performance both on static and dynamic resources such as static files, Java Servlets, ASP.NET, PHP, CGI scripts, Java objects, databases, FTP servers, and more. JMeter can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types.

The performance of STOW-RS is tested using JMeter - the time taken by STOW-RS to process a client’s request, the responsiveness, speed, scalability, and stability characteristics of STOW-RS for varying DICOM file sizes and concurrent requests that can be processed by STOW-RS.

Apache JMeter is an open-source tool which can be installed on Windows. The only prior requirement for JMeter is that the machine open which it runs must have JDK 6 or higher installed.

1. **FIDDLER – API TESTING**

Fiddler is an HTTP debugging proxy server application. Fiddler captures HTTP and HTTPS traffic and logs it for the user to review. Any browser or web application (and most mobile devices) can be configured to route its traffic through Fiddler.

As STOW-RS forms the server side of the web service, there is no typical GUI developed. However, during development of STOW-RS, the client accessing the STOW-RS service is simulated/mocked using the Fiddler Tool. The client’s request to store a DICOM file on the STOW server is triggered using the Fiddler Tool. The HTTP response returned by the server to the client is also viewed using Fiddler. Hence, the proper functioning of the STOW-RS API (as experienced by the STOW-RS client) is testes using Fiddler.

Fiddler is an open-source tool which can installed and run on Windows.

1. **HP FORTIFY SOURCE CODE ANALYZER – SECURITY TESTING**

Fortify Source Code Analyzer (SCA) is a set of software security analyzers that search for violations of security‐specific coding rules and guidelines in a variety of languages. The rich data provided by Fortify SCA language technology enables the analyzers to pinpoint and prioritize violations so that fixes can be fast and accurate. The analysis information produced by SCA helps you deliver more secure software, as well as making security code reviews more efficient, consistent, and complete. This is especially advantageous when large code bases are involved. The modular architecture of SCA allows to quickly upload new, third party, and customer‐specific security rules.

At the highest level, using Fortify SCA involves:

* Choosing to run SCA as a stand‐alone process or integrating Fortify SCA as part of the build tool
* Translating the source code into an intermediate translated format, preparing the code base for scanning by the different analyzers
* Scanning the translated code, producing security vulnerability reports
* Auditing the results of the scan, either by transferring the resulting FPR file to Audit Workbench or Fortify 360 Server for analysis, or directly with the results displayed onscreen.

After the licensed Fortify SCA tool is installed, the plug-in appears in Visual Studio 2013 (on Windows). Then, the code of STOW-RS was analyzed using the Fortify SCA against the latest set of coding rules defined by Philips. Fortify groups the errors in the code into 4 severity categories - Critical, High, Medium, and Low. A PDF report of the Fortify findings can also be generated and circulated to the developers for them to fix parts of the code violating security coding rules.

1. **SoapUI – AUTOMATED TESTING**

SoapUI is an open-source web service testing application for service-oriented architectures (SOA) and representational state transfers (REST). Its functionality covers web service inspection, invoking, development, simulation and mocking, functional testing, load and compliance testing. It is built entirely on the Java platform, and uses Swing for the user interface. SoapUI can test SOAP and REST web services, JMS, AMF, as well as make any HTTP(S) and JDBC calls.

The core features of SoapUI include:

* Web services inspection.
* Web services invoking.
* Web services development.
* Web services simulation and mocking.
* Web services functional, load, compliance, and security testing.

The STOW-RS restful service was thoroughly tested using SoapUI on Windows environment. The defects raised by this process were then fixed by the developers.

**III.TEST CASES**

**IV.DEFECT DISTRIBUTION IN MODULE WISE**

At a very high level, the modules of STOW-RS are:

1. Accept STOW requests from third party clients.
2. Validation of STOW request and DICOM file header.
3. Ensuring security.
4. Storing the files.
5. Issuing a HTTP Response Message.

A list of the defects identified, and the modules in which these defects were found are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **DEFECT** | **DETAILS** | **MODULE** |
| 1. | STOW-RS allows the client to upload multiple studies by providing single Study UID in the request. | As per the DICOM Standard, for a Study UID in the request URL, one or more SOP instances of the same study can be uploaded by the STOW-RS client for storage by the STOW-RS server.  However, STOW-RS was allowing the client to store multiple studies for a single Study UID specified in the request URL. | Storing the files. |
| 2. | STOW-RS allows the client to upload studies using wrong Content-Type: field in the HTTP Request Message. | The DICOM Standard defines that for a DICOM file being uploaded onto PACS using STOW-RS service, the Content-Type: field of the HTTP Request header must be set to application/dicom.  However, STOW-RS was allowing the client to upload a DICOM file with the Content-Type; field set to “formdata”. | Storing the files. |
| 3. | STOW-RS allows the client to upload studies by providing incorrect study UID (Ex. 1.1.1.1.1.1..1.1) in the STOW-RS request URL. | The DICOM Standard lays down certain rules for Study, Series, and SOP Instance Unique Identifiers (UIDs). The 4 rules are:   * UIDs must be composed of only digits (0-9) and periods (.). * UIDs must not be of length greater than 64. * UIDs must not begin or end with a period. * UIDs must not contain consecutive periods. * UIDs must not begin with a zero.   STOW-RS validates the format of the Study UID as soon as the client request arrives, and logs/displays an error message if the validation test fails. However, the STOW-RS service does not get aborted as expected. | Validation of STOW request and DICOM file header |
| 4. | STOW-RS does not allow clients to upload files from the C: Drive of their local machine. | For development purposes, when the client is mocked using Fiddler, he is allowed to browse for a DICOM file from his local machine and upload it.  Technically, he must be able to upload a file from both the C: and the D: drives.  However, STOW-RS wasn’t allowing uploads from C: drive. | Storing the files. |
| 5. | Storage of large DICOM files (<2GB) failed. | The interaction between STOW-RS client and server happens via HTTP request and response messages. The client’s HTTP Request Message body, which contains the DICOM image is organized a multipart HTTP message.  Hence, there must be no restriction on the size of the payload.  However, STOW-RS was throwing an error when large files were sent by the client. | Storing the files. |
| 6. | Incorrect log messages in the log file. | For easy debugging, log statements were inserted in the code at locations of exceptions/failures and occurrences of critical conditions. These custom logs get populated at a predefined location each time the STOW-RS service runs.  However, there were some error messages getting logged even when the STOW-RS could store DICOM images successfully onto the PACS. | Logging statements distributed across all modules. |

**V.NUMBER OF DEFECTS FOUND, THEIR SEVERITY, AND THEIR STATUS**

A list of the defects encountered in STOW-RS, along with their severity and status is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **DEFECT** | **DETAILS** | **SEVERITY** | **STATUS** |
| 1. | STOW-RS allows the client to upload multiple studies by providing single Study UID in the request. | As per the DICOM Standard, for a Study UID in the request URL, one or more SOP instances of the same study can be uploaded by the STOW-RS client for storage by the STOW-RS server.  However, STOW-RS was allowing the client to store multiple studies for a single Study UID specified in the request URL. | HIGH | FIXED |
| 2. | STOW-RS allows the client to upload studies using wrong Content-Type: field in the HTTP Request Message. | The DICOM Standard defines that for a DICOM file being uploaded onto PACS using STOW-RS service, the Content-Type: field of the HTTP Request header must be set to application/dicom.  However, STOW-RS was allowing the client to upload a DICOM file with the Content-Type; field set to “formdata”. | HIGH | FIXED |
| 3. | STOW-RS allows the client to upload studies by providing incorrect study UID (Ex. 1.1.1.1.1.1..1.1) in the STOW-RS request URL. | The DICOM Standard lays down certain rules for Study, Series, and SOP Instance Unique Identifiers (UIDs). The 4 rules are:   * UIDs must be composed of only digits (0-9) and periods (.). * UIDs must not be of length greater than 64. * UIDs must not begin or end with a period. * UIDs must not contain consecutive periods. * UIDs must not begin with a zero.   STOW-RS validates the format of the Study UID as soon as the client request arrives, and logs/displays an error message if the validation test fails. However, the STOW-RS service does not get aborted as expected. | HIGH | FIXED |
| 4. | STOW-RS does not allow clients to upload files from the C: Drive of their local machine. | For development purposes, when the client is mocked using Fiddler, he is allowed to browse for a DICOM file from his local machine and upload it.  Technically, he must be able to upload a file from both the C: and the D: drives.  However, STOW-RS wasn’t allowing uploads from C: drive. | HIGH | FIXED |
| 5. | Storage of large DICOM files (<2GB) failed. | The interaction between STOW-RS client and server happens via HTTP request and response messages. The client’s HTTP Request Message body, which contains the DICOM image is organized a multipart HTTP message.  Hence, there must be no restriction on the size of the payload.  However, STOW-RS was throwing an error when large files were sent by the client. | HIGH | FIXED |
| 6. | Incorrect log messages in the log file. | For easy debugging, log statements were inserted in the code at locations of exceptions/failures and occurrences of critical conditions. These custom logs get populated at a predefined location each time the STOW-RS service runs.  However, there were some error messages getting logged even when the STOW-RS could store DICOM images successfully onto the PACS. | MEDIUM | FIXED |

**VI.TYPES OF TESTING PERFORMED**

1. **MANUAL TESTING**

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user and use most of all features of the application to ensure correct behavior. Also, it is a testing process that is carried out manually in order to find defects without the usage of tools or automation scripting. A test plan document is prepared which acts as a guide to the testing process in order to have the complete test coverage.

For STOW-RS and Storage Library, Manual testing has been done thoroughly and the defects have been brought to the notice of the developers.

For manual testing,

* First, a high level test plan where a general methodology is chosen, and resources such as machines, computers, and software licenses are identified and acquired.
* Then, detailed test cases are written, with expected outcomes.
* Finally, a test report is authored, detailing the findings of the testing process.

1. **AUTOMATED TESTING**

Automated testing tools are capable of executing tests, reporting outcomes and comparing results with earlier test runs. Tests carried out with these tools can be run repeatedly, at any time of day. The method or process being used to implement automation is called a test automation framework. Automated test scripts have been written and tested for STOW-RS using SoapUI.

1. **UNIT TESTING**

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

Unit testing is usually carried out by developers for the code written by them. This practice boosts the confidence of the developer’s in their code. Additionally, the unit tests can be used to continually test the functionality of the code, especially in cases when the code evolves incrementally.

Unit tests for the Storage Library were written in C++ and run using NUnit Testing Framework. For the tests that failed, the Visual Studio 2013 debugger was used for determine the error, either in the code or the test case itself, because of which the test case was failing.

1. **PERFORMANCE TESTING**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

JMeter is the tool used for performance testing.

1. **SECURITY TESTING**

The code of STOW-RS is tested against the custom defined security rules of the Philips Organization using the HP Fortify Code Analyzer.

1. **API TESTING**

API testing is a type of software testing that involves testing application programming interfaces (APIs) to determine if they meet expectations for functionality, reliability, performance, and security. Since APIs lack a GUI, API testing is performed at the message layer.

The STOW-RS REST API is tested using the Fiddler tool.

**VII.CONCLUSION**

The details of the various types of testing carried out for STOW-RS have been covered in this document. Due to the adaption of the Scaled Agile process model, as the code evolves, testing has to be done proactively.