Martus Testing Project: Global Human Rights Abuse Reporting System

Jason Wilson, Steven Pilkenton, Bobby Jenkins, and Tomoko Goddard

Introduction

To enhance and practice our understanding and techniques of essential software development activities, we developed an automated requirements-based unit testing framework along with creating the system test documentation. Our goal is to develop 25 unit test cases triggered by a script on the existing open-source project, and display the test results via web browser during the time period of 9/3/2013 to 11/21/2013.

About Martus and its system requirements

Official Web Site
https://www.martus.org/

Martus is a secure information management tool that allows users to create a searchable and encrypted human rights violation cases database.

Requirements specification

Function requirements

- Records shall be protected by a unique password.
- The system shall be able to create searchable text-based bulletins about human rights violations.
- The system shall allow users to create folders and attach additional documents or other data to the bulletins
- Records shall be searched by groups or outside researchers granted access.

Non-functional requirements

- The system should support worldwide organizations.
- The software should be as easy to use as email.
- Records should be backed up to multiple locations.
- The system should encouraging easy code review to foster an atmosphere of transparency, trust and collaboration.

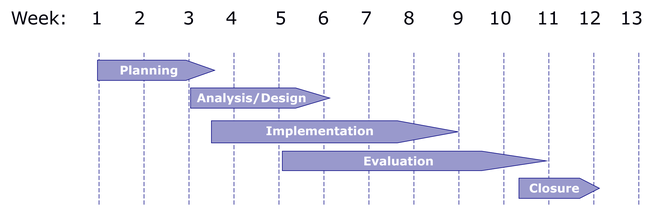
College of Charleston
Computer Science Department
Fall 2013

Test Process

Test Environments

- 64 bit hardware system
- Ubuntu Desktop 12.04
- Java 1.6
- Eclipse Indigo 3.7.2
- Mercurial Plug-in for Eclipse

Test Schedule



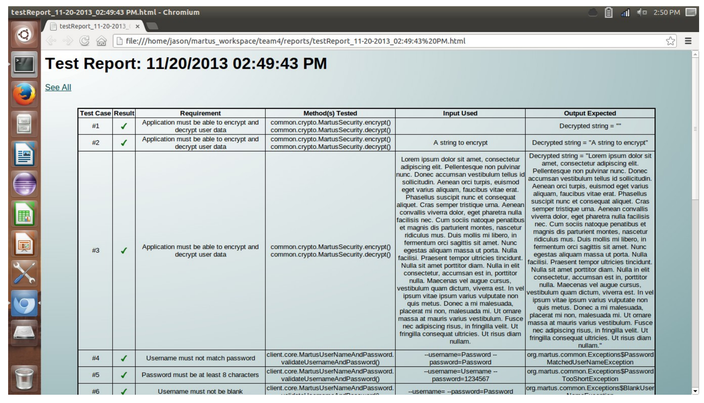
Tested Items

Function	Items	Priority
Encrypting and Decrypting	. Empty string . Standard string . Lorm ipsum document	High
Checking for an acceptable password	. The word "Username" . A short password . the empty string	Low
Checking for a strong password	. A short word with 2 nonletter/ . numbers . a long password with 1 nonletter/ . number . a long password with 2 nonletter/ . Numbers	Low
Checking for unacceptable username	. Blank username . the word "Username"	High
Setting Bulletin's status to "sealed" and "draft"	. Setting it to "draft" . Setting it to "sealed" . Comparing the two	High
Creating and deleting Martus data	. 5 automatically generated bulletin . entries . 0 generated bulletins	High
Only connecting through a secure/trusted port	. List of all secure ports . list of all unsecure ports . list of one secure port	High
Create a user ID	. Create a user ID . Account ID, local ID, prefix together	Medium
Inheritance of Data Types	. The word "tag" . The word "label" . The word "compare"	Low

Framework

The framework makes use of test case files to define parameters to run java test classes. The framework operates by looping through all valid test case files in the "testCases" directly running the java test classes defined by each test case, creating the oracle and recording the results. Evaluation of test results are made by comparing actual output of the test class to the expected output - any mismatch results in test failure. Results of each test run are generated in the "reports" directly.
Each report gives detailed testing parameters for each test case, and, in the event of failure, gives reasons for test failures. The framework may be extended by defining your own test cases that reference your java test cases.

Sample Test Results



Group Experience

The group held meetings twice a week and progress was updated and code consolidation was conducted. We found that being in a study room in the library was optimal. Due to time limitation and the deviation of knowledge level, much research was required to implement our test suite. Overall, we tackled our issues early on so there was nothing serious holding back our progress.