Test Plan

for

Automated Software Engineering Student Subject Evaluations (ASESSE)

Version 1.1

Prepared by Tanaka Madyara, Deepthi Ummala, Samuel Weems



December 07, 2018

Table of Contents

Table of Contents 2

1. Introduction 1

1.1. Overview of Document 1

1.2. Overview of Product 1

1.3. Testing Process 1

2. Tests 2

2.1. Unit Testing 2

2.2. Integration Testing 2

2.3. System Testing 2

3. Test Schedule 2

4. Resources 3

4.1. Human 3

4.2. Hardware 3

4.3. Software 3

5. Recording Procedures 4

5.1. Error Report Forms 4

5.2. Unit Test Results 4

5.3. Integration Test Results 4

5.4. System Test Results 4

5.5. Non-fixed errors report 4

Appendix A: Report Forms 5

Appendix B: Glossary 8

8

# Introduction

## Overview of Product

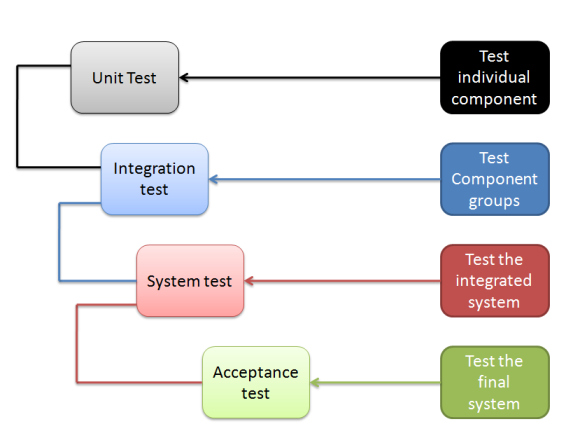
ASSESE is a web-based application which automates the student peer review system for Software Engineering Students at Midwestern State University located in Wichita Falls, Texas. The main purpose is to allow students to review other students in their course group and to collate that information automatically for the professor.

## Overview of Document

The purpose of this document is to provide and outline and overall description of the testing phase for the “Automated Software Engineering Student Subjective Evaluation” (ASSESE) project. It describes the various types of tests to be performed, lays out a schedule for the tests and lists the various resources for the testing phase. Finally the document provides a description of the recording procedures used to log and categorize test results. The appendices of this document include a glossary of terms and example report forms for testing.

## Testing Process

The testing process can be visualized in the diagram below. It shows the general flow of testing.



**Diagram 1 – Testing Process**

# Tests

Three types of testing strategies will be used in order to assess the correction and reliability of the software: unit, integration, and system testing.

## Unit Testing

Also called component testing, the purpose of unit testing is to discover any erroneous code. It is performed on standalone modules to check whether they are developed correctly.

Unit testing will be conducted as each module is completed by the developer. Matching parameters, arguments, and attributes are the criteria to consider during this phase of testing.

An example test case would be the Login module. Test cases include:

* check response for valid login and password, e.g. “Pass\_Word.”
* check response for invalid login and password, e.g. “1234”
* what happens when login is empty and login button is pressed, e.g. “ ”.

## Integration Testing

In this phase of testing, individual modules are combined and tested as a group. The data transferred between the modules is tested thoroughly to ensure that all modules undergoing integration function correctly. “Stubs” are mock modules used for testing software modules. Stubs, which are usually created to test integration in the “top-to-down” testing technique, can be created for incremental testing while each module is ready for integration simulating data that gets transferred. The top-to-down approach may be used in incremental testing where higher-level modules are tested first. A “driver,” like a “stub,” is a mock module used in software testing. Drivers can be created to simulate interactions between two modules in a bottom-up technique.

Once the individual modules and their relative data structures and methods have been tested, the integration phase will examine the interaction between the relevant modules.

## System Testing

System testing is concerned with testing the behavior of the entire system. tests complete end-to-end scenarios. Non-functional requirements, such as performance and reliability, are also tested.

Once the system construction has been fully completed and the integration testing performed, the software components are all combined and a “system test” is implemented. In this phase, the system interacts with actual user input and the complete database. An example of the type of system testing that will be taken would be to go through the entire process of a student filling out a form from login into their course, through a form submission. Database submission and all paths will be scrutinized for correct results.

# Test Schedule

Testing for the software will be done in different phases in order to assess how well the software works as it is being built. This ensure that as many defects are identified before the software is complete, in order to limit the amount of time and effort needed to debug the earliest-completed components.

**Unit Testing:** Unit testing will be done in an incremental manner. As each module is completed, testing is done in a top-down manner. This requires the creation of stubs for the various test cases available in the system.

**Integration Testing**: Integration testing will be done over the course of five days.

**System Testing:** System testing will be done over the course of two days. End-to-end testing will be implemented to ensure that the system works thoroughly. This includes using the use-cases described in the requirements document as test cases.

# Resources

## Human

The human resources for this phase of the ASSESE project consist primarily of the BOZOS team: Tanaka Madyara, Deepthi Ummala, and Samuel Weems. The customer, Dr. Stringfellow, will also be consulted during this phase. A final resource is MWSU faculty Dr. Griffin who specializes

## Hardware

Hardware for the development phases of the project consists of the personal desktop and laptop computers, as well as the computing facilities available to team members in the Midwestern State University computer science laboratories. Since the application must run over the internet, all the hardware is required to connect internet will be the hardware interface for the system.

## Software

* Software includes,
* Microsoft Windows XP
* Microsoft Visual Code
* MySQL
* Linux
* Apache
* Php
* Google Chrome, Safari, Microsoft Edge Browsers

# Recording Procedures

Errors which are detected during Unit tests, Integration tests and System tests will be recorded and entered into Error Report Form. All the errors in the Error Report Form will be validated by all the members in the team. Based on the impact on system performance and test implementation errors which are validated by the team will be categorized into two types of errors - low priority errors and high-priority errors. All the errors which are recognized during the test will be handed over to development team to make required changes to improve the system performance. For reporting purposes all the test results will be well documented in a formal approved format.

## Error Report Forms

A sample error report form that is used to record errors which are detected during the Unit test is shown in Appendix 6.1. A sample error report form that is used to record errors which are detected during the Integration test is shown in Appendix 6.2. A sample error report form that is used to record errors which are detected during the System test is shown in Appendix 6.3.

## Unit Test Results

All the errors that are recorded during the Unit test phase, and the action taken to correct these errors will be well documented and that document is called the Unit Test Result Report or Unit Test Result Document.

## Integration Test Results

All the errors that are detected during the Integration test phase, type of error, status of the error and the action taken to correct these errors will be well documented as a Integration Test Result Report or Integration Test Result Document.

## System Test Results

All the errors that are detected while applying the test cases during the System test phase along with the type of error, status of error and the action taken to correct these errors will be well documented as a System Test Result Report or System Test Result Document.

## Non-fixed errors report

After documenting all the test results we can see some errors which are not fixed will be documented as Non-Fixed Error Report. This report also contains the type of error, status of the error, chances of occurrence, the action taken to correct the error when it was occurred.

# 

# Appendix A: Report Forms

**6.1. Unit test error report form**

**UNIT TEST ERROR REPORT FORM**

ASESSE

BOZOS TEAM

Prototype # Module Class

Problem summary:

|  |
| --- |
|  |

Reported by

Action taken:

|  |
| --- |
|  |

Action taken by

**6.2. Integration test error report form**

**INTEGRATION TEST ERROR REPORT FORM**

ASESSE

BOZOS TEAM

Prototype # Class Category Status

Problem summary:

|  |
| --- |
|  |

Reported by

Action taken:

|  |
| --- |
|  |

Action taken by

**6.3. System test error report**

**SYSTEM TEST ERROR REPORT FORM**

ASESSE

BOZOS TEAM

Prototype # Class Category Status

Test case:

Problem summary:

|  |
| --- |
|  |

Reported by

Action taken:

|  |
| --- |
|  |

Action taken by

Appendix B: Glossary

ASESSE - Automated Software Engineering Student Subject Evaluation

*Unit Testing* - Smallest testable parts of an application, called units, are independently and individually are tested for proper operation.

*Integration Testing -* A level of software testing where individual software modules are combined and tested as a group.

*System Testing* - A testing conducted on complete and integrated software to evaluate its compliance with specified requirements.

*High priority Errors* - Major errors that must be resolved as soon as possible.

*Low Priority Errors* - Minor errors that can be resolved after resolving the high priority errors.

*Test Report* - A document that record the data obtained from an evaluation experiment.