BugTracker

Version 1.0

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 13/10/2016 | 1.0 | Begin Test Plan | Tessa |
| 25/10/2016 | 1.1 | Edit function testing | Xinwei Jiang |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Background 4

1.3 Scope 4

1.4 Project Identification 5

2. Requirements for Test 6

3. Test Strategy 7

3.1 Testing Types 7

3.1.1 Data and Database Integrity Testing 7

3.1.2 Function Testing 7

3.1.3 Business Cycle Testing 9

3.1.4 User Interface Testing 10

3.1.5 Performance Profiling 11

3.1.6 Load Testing 13

3.1.7 Stress Testing 14

3.1.8 Volume Testing 14

3.1.9 Security and Access Control Testing 14

3.1.10 Failover and Recovery Testing 14

3.1.11 Configuration Testing 14

3.1.12 Installation Testing 14

3.2 Tools 14

4. Resources 14

4.1 Workers 14

4.2 System 14

5. Project Milestones 14

6. Deliverables 14

6.1 Test Model 14

6.2 Test Logs 14

6.3 Defect Reports 14

7. Appendix A: Project Tasks 14

# Introduction

## Purpose

This Test Plan document for Bug Tracker supports the following objectives:

* Black Box testing to ensure all functional requirements outlined in the SRS Version 1.3 are met. Sets of data that will intentionally produce errors and sets that are expected not to produce errors will be used.
* White Box testing will include unit tests that cover class variable values, statement execution, decision and conditional statements used within controller classes.
* Integration Testing to discover any faults in the interface design and make sure the program works as expected.
* Performance Testing to test the program conforms to a selection of non-functional requirements outlined in SRS Version 1.3.
* Database and data integrity Testing.

Required Resources:

* A set of erroneous data and a set of expected data.
* QT test framework.

Deliverables:

* A set of test outcomes that highlight where errors can occur together with fixes for the errors.
* An outline of improvements needed in regards to the user interface.

• [Identify existing project information and the software components that should be tested.

• List the recommended Requirements for Test (high level).

• Recommend and describe the testing strategies to be employed.

• Identify the required resources and provide an estimate of the test efforts.

• List the deliverable elements of the test project]

## Background

The target of these tests is a bug tracking software application made up of a bug report component, user profile component, search component and a MySQL database. The application interfaces are created in QT and C++ is the language used to produce the functionality.

The goal of the project is to provide an application where users can report and search for bugs found in their software. Software Developers gain reputation points for fixing bugs while reporters gain points for reporting bugs. The system generates statistical reports on bug numbers and other features of bugs.

The system uses the model view controller architectural design so includes boundary, controller and entity classes. Development of the project began in early September 2016, a working application is almost complete.

[Enter a brief description of the target-of-test (components, application, system, etc.) and its goals. Include information such as major functions and features, its architecture, and a brief history of the project. This section should only be about three to five paragraphs.]

## Scope

Stages of testing include:

* Unit testing of the major functions that produce output or receive input via a user interface. The interface will not be used, unit tests will test specific data within the unit test function. The user interface will be used as part of black box testing. Minor utility functions will not have their own unit tests.
* Integration test to determine if the interfaces work well together and no faults are detected as a user moves between each interface. This will be carried out in the form of a walk-through of the entire program.
* System tests will test whether the system meets its major functional requirements and use cases at this stage. These include:

1. A user can add a bug.
2. A user can view a bug with all its comments and history.
3. A user can log in.
4. A user can register which creates their profile.
5. A user can edit their profile.
6. A user can search for bugs using a filtering system of keywords, bug id.
7. A user can search other users using keywords, user name.
8. The system produces bug reports, of most importance is “how many bugs reported for an application”.
9. A Manager can assign a bug to a developer to fix.
10. A developer can gain reputation points by fixing a number of bugs.
11. A user can gain reputation points by reporting a number of good bugs.
12. A user can add comments to a bug.  
      
    System tests will also cover the following performance requirements:
13. The system shall take no longer than 12 seconds to upload

[Describe the stages of testing­⎯for example, Unit, Integration, or System⎯and the types of testing that will be addressed by this plan, such as Function or Performance.

Provide a brief list of the target-of-test’s features and functions that will or will not be tested.

List any assumptions made during the development of this document that may impact the design, development or implementation of testing.

List any risks or contingencies that may affect the design, development or implementation of testing.

List any constraints that may affect the design, development or implementation of testing]

## Project Identification

The table below identifies the documentation and availability used for developing the *test plan*:

[Note: Delete or add items as appropriate.]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document  (and version / date)** | **Created or Available** | **Received or Reviewed** | **Author or Resource** | **Notes** |
| Requirements Specification | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Functional Specification | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Use-Case Reports | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Project Plan | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Design Specifications | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Prototype | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Data Model or Flow | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Project or Business Risk Assessment | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |

# Requirements for Test

**Functional:**

Critical and some Essential Requirements outlined in SRS 1.3 will be tested, these are as follows:

Use Case 1: The system will provide a user of any type with a gui to log in.

Use Case 2: The system should provide a guest to register an account in the system.

Use Case 3: The system should provide any user with a GUI to modify their own profile details, name, E-mail, profile picture and subscriptions.

Use Case 3: The system should provide any user with a GUI to modify their own password.

Use Case 6: The system should provide a user with a GUI form to search for users of any type.

Search criteria include: First Name, Last Name, Email, Role, Account

Status and Key word.

Use Case 7: The system should provide a user with a GUI to view any non-administrator user profile.

Use Case 8: The system should allow a user to filter users of any type by First Name,

Last Name and Account Status.

Use Case 9: The system should allow a Reporter, Developer, Triager, Reviewer to clock up reputation points to be displayed in their user profile and whenever their name is used in commenting or bug reporting or bug solving.

Use Case 10: The system should provide a System Admin with a GUI to edit an existing non-administrator user’s profile in the Bug Tracker System.

Use Case 12: The system should provide a user with a GUI form to enter new bugs. The form should include:

* Application the reporter is reporting a bug on
* Title
* Text area to enter description of bug
* Option to receive responses from their post to their email
* Choose how often they want responses, e.g. weekly, monthly

Optional information: Version, file uploads, repeatable walkthrough, platform, severity, priority.

Use Case 13: The system should provide any user with a facility to add comments.

Use Case 11: The system should provide a user of any type (Reporter, Developer, Triager, and Reviewer) with a GUI form to subscribe to the bug. After subscribing to the bug, the user can get the newest information by E-mail once the information be updated.

Use Case 18: The system will provide a user of any type with a gui to view a bug. This includes all active and non-active bugs including duplicate bugs.

Use Case 20: The system should provide a Triager with a GUI to Assign bugs to developers. This GUI shows all currently un-assigned bugs and all developers.

Use Case 18: The system should provide a Triager a GUI to Check for duplicate and/or invalid bug submissions and take appropriate actions regarding the bug.

Use Case 19: The system should provide a Triager with a GUI to edit existing Bugs in the System. Details such as title, description, status, severity, priority, assigned developer can be changed.

Use Case 18: The system should provide a Triager with a GUI to view current bugs assigned to each developer.

Use Case 4: The system should provide a facility for the developer to change the status of the bug. A developer can only change a bug assigned to them.

Use Case 18: The system should provide a System Admin with a GUI to view current bugs assigned to each developer.

Use Case 12: The system should provide a Reporter with a GUI to export Bug Reports into a word document, pdf or html file. This can include all statistical analysis of the System.

Use Case 4: The system should provide users a GUI form to search for bugs of any type.

Search criteria include: Application, Date, Developer, Key Word, and Status.

Use Case 4: The system should allow a user to filter bugs of any type by Application,

Date, Developer and Status.

**Non-functional requirements to be tested:**

* The system should respond to any action of any user within 12 seconds.
* The system should display search results within 12 seconds.
* The system should generate a report within no more than 2 minutes.
* The system should only allow passwords that comply with the following:
  + At least one upper and one lower case letter.
  + No special characters, only alphanumeric.
  + No punctuation
  + Not a previously used password by the user
  + Minimum length of 8 characters
  + Maximum length of 64 characters
* The system should not require users to have any specific knowledge or special technical skill to be able to use it. Bug Tracker system should be appropriate for a computer-literate user community with no additional training on the system so that users can use it after at most one day of exploring the system. By providing well-formed graphical user interfaces.
* The system should provide users with appropriate feedback to recover from errors within 3 seconds by providing informative error messages.
* The system should be able to store at least 1000 user profiles and 1000 bug reports.

The listing below identifies those items⎯use cases, functional requirements, and non-functional requirements⎯that have been identified as targets for testing. This list represents what will be tested.

[Enter a high level list of the major test requirements.]

# Test Strategy

Functional and non-functional requirements will be tested by group members to provide a form of black box testing. Results of tests will be recorded. These tests ensure the main functional and non-functional requirements have been met and work as expected.

Unit tests should be written for methods included in controller classes as these provide the main functionality.

Database methods will be tested without the use of a user interface and before the database class is made available to the rest of the group members. Data and source code of tests, along with evidence of data being input and retrieved from the database will be recorded. These tests are carried out to ensure data is stored and retrieved without error and can then be used by the group creating user interfaces.

[The Test Strategy presents the recommended approach to the testing of the target-of-test. The previous section, Requirements for Test, described what will be tested⎯this describes how the target-of-test will be tested.

For each type of test, provide a description of the test and why it is being implemented and executed.

If a type of test will not be implemented and executed, indicate this in a sentence stating the test will not be implemented or executed and stating the justification, such as “This test will not be implemented or executed. This test is not appropriate.”

The main considerations for the test strategy are the techniques to be used and the criterion for knowing when the testing is completed.

In addition to the considerations provided for each test below, testing should only be executed using known, controlled databases in secured environments. ]

## Testing Types

### Data and Database Integrity Testing

The database class will use

[The databases and the database processes should be tested as a subsystem within the . These subsystems should be tested without the target-of-test’s User Interface as the interface to the data. Additional research into the DataBase Management System (DBMS) needs to be performed to identify the tools and techniques that may exist to support the testing identified below.]

|  |  |
| --- | --- |
| Test Objective: | [Ensure database access methods and processes function properly and without data corruption.] |
| Technique: | • [Invoke each database access method and process, seeding each with valid and invalid data or requests for data.  • Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons] |
| Completion Criteria: | [All database access methods and processes function as designed and without any data corruption.] |
| Special Considerations: | • [Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases.  • Processes should be invoked manually.  • Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events.] |

### Function Testing

[Function testing of the target-of-test should focus on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that is verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the testing recommended for each application:]

|  |  |
| --- | --- |
| Test Objective: | [Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval.] |
| Technique: | [Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:  • The expected results occur when valid data is used.  • The appropriate error or warning messages are displayed when invalid data is used.  • Each business rule is properly applied.] |
| Completion Criteria: | • [All planned tests have been executed.  • All identified defects have been addressed.] |
| Special Considerations: | [Identify or describe those items or issues (internal or external) that impact the implementation and execution of function test] |

### 

|  |  |
| --- | --- |
| Test Objective: | The system will provide a user of any type with a gui to log in. |
| Technique: | Black box testing   * Run test case 1 * Run test case 2 * Run test case 3 * Run test case 4 * Run test case 5   Compare the result with the expected output |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a guest to register an account in the system. |
|  | Black box testing   * Run test case 6 * Run test case 7 * Run test case 8 * Run test case 9 * Run test case 10 * Run test case 11 * Run test case 12   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI to modify their own profile details, name, E-mail, profile picture and subscriptions. |
| Technique: | Black box testing   * Run test case 13 * Run test case 14 * Run test case 15 * Run test case 16 * Run test case 17 * Run test case 18 * Run test case 19 * Run test case 20   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI to modify their own password. |
| Technique: | Black box testing   * Run test case 21 * Run test case 22 * Run test case 23   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide a System Admin with a GUI to edit an existing non-administrator user’s profile in the Bug Tracker System. |
| Technique: | Black box testing   * Run test case 24 * Run test case 25 * Run test case 26 * Run test case 27 * Run test case 28 * Run test case 29 * Run test case 30   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | The system should provide any user with a GUI to view any non-administrator user profile. |
| Technique: | Black box testing   * Run test case 31 * Run test case 32   Compare the result with the expected output. |
| Completion Criteria: | * Finishing run all the test cases * Covering all the possible cases * The pass rate is 100% |
| Special Considerations: |  |

### User Interface Testing

[User Interface (UI) testing verifies a user’s interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.]

|  |  |
| --- | --- |
| Test Objective: | [Verify the following:  • Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to- field, and use of access methods (tab keys, mouse movements, accelerator keys)  • Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.] |
| Technique: | [Create or modify tests for each window to verify proper navigation and object states for each application window and objects.] |
| Completion Criteria: | [Each window successfully verified to remain consistent with benchmark version or within acceptable standard] |
| Special Considerations: | [Not all properties for custom and third party objects can be accessed.] |

### Performance Profiling

[Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviors as a function of conditions such as workload or hardware configurations.

Note: Transactions below refer to “logical business transactions”. These transactions are defined as specific use cases that an actor of the system is expected to perform using the target-of-test, such as add or modify a given contract.]

|  |  |
| --- | --- |
| Test Objective: | [Verify performance behaviors for designated transactions or business functions under the following conditions:  • normal anticipated workload  • anticipated worst case workload] |
| Technique: | • [Use Test Procedures developed for Function or Business Cycle Testing.  • Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs.  • Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see Special Considerations below).] |
| Completion Criteria: | • [Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction.]  • [Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.] |
| Special Considerations: | [Comprehensive performance testing includes having a background workload on the server.  There are several methods that can be used to perform this, including:  • “Drive transactions” directly to the server, usually in the form of Structured Query Language (SQL) calls.  • Create “virtual” user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with “traffic”.  • Use multiple physical clients, each running test scripts to place a load on the system.  Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  The databases used for Performance Testing should be either actual size or scaled equally.] |

### Load Testing

[Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues).]

[Note: Transactions below refer to “logical business transactions”. These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.]

|  |  |
| --- | --- |
| Test Objective: | [Verify performance behavior time for designated transactions or business cases under varying workload conditions.] |
| Technique: | • [Use tests developed for Function or Business Cycle Testing.  • Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.] |
| Completion Criteria: | [Multiple transactions or multiple users: Successful completion of the tests without any failures and within acceptable time allocation.] |
| Special Considerations: | • [Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  • The databases used for load testing should be either actual size or scaled equally.] |

### Stress Testing

[Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the target-of-test can handle.]

[Note: References to transactions below refer to logical business transactions.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that the target-of-test functions properly and without error under the following stress conditions:  • little or no memory available on the server (RAM and DASD)  • maximum actual or physically capable number of clients connected or simulated  • multiple users performing the same transactions against the same data or accounts  • worst case transaction volume or mix (see Performance Testing above).  Notes: The goal of Stress Testing might also be stated as identify and document the conditions under which the system FAILS to continue functioning properly.  Stress Testing of the client is described under section 3.1.11, Configuration Testing.] |
| Technique: | • [Use tests developed for Performance Profiling or Load Testing.  • To test limited resources, tests should be run on a single machine, and RAM and DASD on server should be reduced or limited.  • For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume or mix. |
| Completion Criteria: | [All planned tests are executed and specified system limits are reached or exceeded without the software failing or conditions under which system failure occurs is outside of the specified conditions.] |
| Special Considerations: | • [Stressing the network may require network tools to load the network with messages or packets.  • The DASD used for the system should temporarily be reduced to restrict the available space for the database to grow.  • Synchronization of the simultaneous clients accessing of the same records or data accounts.] |

### Volume Testing

[Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that the target-of-test successfully functions under the following high volume scenarios:  • Maximum (actual or physically- capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period.  • Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.] |
| Technique: | • [Use tests developed for Performance Profiling or Load Testing.  • Multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume or mix (see Stress Testing above) for an extended period.  • Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries and report transactions simultaneously for extended periods.] |
| Completion Criteria: | • [All planned tests have been executed and specified system limits are reached or exceeded without the software or software failing.] |
| Special Considerations: | [What period of time would be considered an acceptable time for high volume conditions, as noted above?] |

### Security and Access Control Testing

[Security and Access Control Testing focus on two key areas of security:

• Application-level security, including access to the Data or Business Functions

• System-level Security, including logging into or remote access to the system.

Application-level security ensures that, based upon the desired security, actors are restricted to specific functions or use cases, or are limited in the data that is available to them. For example, everyone may be permitted to enter data and create new accounts, but only managers can delete them. If there is security at the data level, testing ensures that” user type one” can see all customer information, including financial data, however,” user two” only sees the demographic data for the same client.

System-level security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.]

|  |  |
| --- | --- |
| Test Objective: | * Application-level Security: [*Verify that an actor can access only those functions or data for which their user type is provided permissions.]* * System-level Security: *Verify that only those actors with access to the system and applications are permitted to access them*.] |
| Technique: | * Application-level Security: [*Identify and list each user type and the functions or data each type has permissions for.]*   • [Create tests for each user type and verify each permission by creating transactions specific to each user type.]  • Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied.   * System-level Access: *[See Special Considerations below]* |
| Completion Criteria: | [For each known actor type the appropriate function or data are available, and all transactions function as expected and run in prior Application Function tests.] |
| Special Considerations: | [Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.] |

### 

### Configuration Testing

[Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded⎯for example, applications, drivers, etc.⎯and at any one time, many different combinations may be active using different resources.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that the target-of-test functions properly on the required hardware and software configurations.] |
| Technique: | • [Use Function Test scripts.  • Open and close various non-target-of-test related software, such as the Microsoft applications, Excel and Word, either as part of the test or prior to the start of the test.  • Execute selected transactions to simulate actor’s interacting with the target-of-test and the non-target-of-test software.  • Repeat the above process, minimizing the available conventional memory on the client workstation.] |
| Completion Criteria: | [For each combination of the target-of-test and non-target-of-test software, all transactions are successfully completed without failure.] |
| Special Considerations: | • [What non-target-of-test software is needed, is available, and is accessible on the desktop?  • What applications are typically used?  • What data are the applications running; for example, a large spreadsheet opened in Excel or a 100- page document in Word?  • The entire systems, netware, network servers, databases, etc. should also be documented as part of this test.] |

### 

## Tools

The following tools will be employed for this project:

[Note: Delete or add items as appropriate.]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tool | Vendor/In-house | Version |
| Test Management |  |  |  |
| Defect Tracking |  |  |  |
| ASQ Tool for functional testing |  |  |  |
| ASQ Tool for performance testing |  |  |  |
| Test Coverage Monitor or Profiler |  |  |  |
| Project Management |  |  |  |
| DBMS tools |  |  |  |

# Resources

[This section presents the recommended resources for the project, their main responsibilities, and their knowledge or skill set.]

## Workers

This table shows the staffing assumptions for the project.

[NOTE: Delete or add items as appropriate.]

|  |  |  |
| --- | --- | --- |
| Human Resources | | |
| Worker | Minimum Resources Recommended  (number of full-time workers allocated) | Specific Responsibilities or Comments |
| Test Manager,  Test Project Manager | 1 | Provides management oversight.  Responsibilities:   * provide technical direction * acquire appropriate resources * provide management reporting |
| Test Designer | 1 | Identifies, prioritizes, and implements test cases.  Responsibilities:   * generate test plan * generate test model * evaluate effectiveness of test effort |
| Tester | 1 | Executes the tests.  Responsibilities:   * execute tests * log results * recover from errors * document change requests |
| Test System Administrator | 1 | Ensures test environment and assets are managed and maintained.  Responsibilities:   * administer test management system * install and manage worker access to test systems |
| Database Administratator, Database Manager | 1 | Ensures test data (database) environment and assets are managed and maintained.  Responsibilities:   * administer test data (database) |
| Designer | 1 | Identifies and defines the operations, attributes, and associations of the test classes.  Responsibilities:   * identifies and defines the test class(es) * identifies and defines the test packages |
| Implementer | 1 | Implements and unit tests the test classes and test packages.  Responsibilities:   * creates the test classes and packages implemented in the test model |

## System

The following table sets forth the system resources for the testing project.

[The specific elements of the test system are not fully known at this time. It is recommended that the system simulate the production environment, scaling down the accesses and database sizes if and where appropriate.]

[Note: Delete or add items as appropriate.]

|  |  |
| --- | --- |
| System Resources | |
| Resource | Name / Type |
| Database Server |  |
| —Network or Subnet | TBD |
| —Server Name | TBD |
| —Database Name | TBD |
| Client Test PC's |  |
| —Include special configuration requirements | TBD |
| Test Repository |  |
| —Network or Subnet | TBD |
| —Server Name | TBD |
| Test Development PC's | TBD |

# Project Milestones

[Testing of should incorporate test activities for each of the test efforts identified in the previous sections. Separate project milestones should be identified to communicate project status accomplishments.]

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Task** | **Effort** | **Start Date** | **End Date** |
| Plan Test |  |  |  |
| Design Test |  |  |  |
| Implement Test |  |  |  |
| Execute Test |  |  |  |
| Evaluate Test |  |  |  |

# Deliverables

[In this section list the various documents, tools, and reports that will be created, by whom, delivered to who, and when delivered.]

## Test Model

[This section identifies the reports that will be created and distributed from the test model. These artifacts in the test model should be created or referenced in the ASQ tools.]

## Test Logs

[Describe the method and tools used to record and report on the test results and testing status.]

## Defect Reports

[In this section identify the method and tools used to record, track, and report on test incidents and their status.]

# Appendix A: Project Tasks

Below are the test related tasks:

• Plan Test

* + identify requirements for test
  + assess risk
  + develop test strategy
  + identify test resources
  + create schedule
  + generate Test Plan

• Design Test

- prepare workload analysis

- identify and describe test cases

- identify and structure test procedures

- review and assess test coverage

• Implement Test

* + record or program test scripts
  + identify test-specific functionality in the Design and Implementation Model
  + establish external data sets

• Execute Test

- execute Test procedures

- evaluate execution of Test

- recover from halted Test

- verify the results

- investigate unexpected results

- log defects

• Evaluate Test

- evaluate Test-case coverage

- evaluate code coverage

- analyze defects

- determine if Test Completion Criteria and Success Criteria have been achieved