|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <02/05/y17> | <1.0> | Init | Simon Gerhardt |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Intended Audience 4

1.4 Document Terminology and Acronyms 4

1.5 References 4

1.6 Document Structure 4

2. Evaluation Mission and Test Motivation 4

2.1 Background 4

2.2 Evaluation Mission 4

2.3 Test Motivators 4

3. Target Test Items 4

4. Outline of Planned Tests 4

4.1 Outline of Test Inclusions 4

4.2 Outline of other candidates for potential inclusion 4

4.3 Outline of Test Exclusions 4

5. Test Approach 4

5.1 Initial Test-Idea Catalogs and other reference sources 4

5.2 Testing Techniques and Types 4

5.2.1 Data and Database Integrity Testing 4

5.2.2 Function Testing 4

5.2.3 Business Cycle Testing 4

5.2.4 User Interface Testing 4

5.2.5 Performance Profiling 4

5.2.6 Load Testing 4

5.2.7 Stress Testing 4

5.2.8 Volume Testing 4

5.2.9 Security and Access Control Testing 4

5.2.10 Failover and Recovery Testing 4

5.2.11 Configuration Testing 4

5.2.12 Installation Testing 4

6. Entry and Exit Criteria 4

6.1 Test Plan 4

6.1.1 Test Plan Entry Criteria 4

6.1.2 Test Plan Exit Criteria 4

6.1.3 Suspension and resumption criteria 4

6.2 Test Cycles 4

6.2.1 Test Cycle Entry Criteria 4

6.2.2 Test Cycle Exit Criteria 4

6.2.3 Test Cycle abnormal termination 4

7. Deliverables 4

7.1 Test Evaluation Summaries 4

7.2 Reporting on Test Coverage 4

7.3 Perceived Quality Reports 4

7.4 Incident Logs and Change Requests 4

7.5 Smoke Test Suite and supporting Test Scripts 4

7.6 Additional work products 4

7.6.1 Detailed Test Results 4

7.6.2 Additional automated functional Test Scripts 4

7.6.3 Test Guidelines 4

7.6.4 Traceability Matrices 4

8. Testing Workflow 4

9. Environmental Needs 4

9.1 Base System Hardware 4

9.2 Base Software Elements in the Test Environment 4

9.3 Productivity and Support Tools 4

9.4 Test Environment Configurations 4

10. Responsibilities, Staffing and Training Needs 4

10.1 People and Roles 4

10.2 Staffing and Training Needs 4

11. Iteration Milestones 4

12. Risks, Dependencies, Assumptions and Constraints 4

13. Management Process and Procedures 4

13.1 Measuring and Assessing the Extent of Testing 4

13.2 Assessing the deliverables of this Test Plan 4

13.3 Problem Reporting, Escalation and Issue Resolution 4

13.4 Managing Test Cycles 4

13.5 Traceability Strategies 4

13.6 Approval and Signoff 4

SilverScreen - Test Plan

# Introduction

## Purpose

The purpose of the Iteration Test Plan is to gather all of the information necessary to plan and control the test effort for a given iteration. It describes the approach to testing the software, and is the top-level plan generated and used by managers to direct the test effort.

This *Test Plan* for the SilverScreen supports the following objectives:

* Connection to our DB
* Connection to the Movie DB
* Views
* Controllers
* Models

## Scope

We are using simple Unit tests for internal app functionalities and instrumented Unit tests for View and DB Access.

## Intended Audience

This document is intended for guiding team SilverScreen through the project in order to produce a reliable software product.

## Document Terminology and Acronyms

n/a

## References

n/a

## Document Structure

n/a

# Evaluation Mission and Test Motivation

## Background

Testing is very important in order to produce high quality software products. Also for continuous extending of the source code, tests verify the regression free development.

## Evaluation Mission

We want to have as many tests as possible in order to satisfy any customer needs and destroy all trust issues from customer side.

The risk of not meeting those customer needs is minimized by having a good big testing environment

## Test Motivators

Meet specifications and customer needs.

# Target Test Items

We test our DB and MovieDB. Also internal functionalities e.g. controllers.

# Outline of Planned Tests

## Outline of Test Inclusions

Testing of functionality (app internal), as well as instrumented tests for integrating the UI in additional tests.

## Outline of Other Candidates for Potential Inclusion

n/a

## Outline of Test Exclusions

n/a

# Test Approach

## Initial Test-Idea Catalogs and Other Reference Sources

n/a

## Testing Techniques and Types

### Data and Database Integrity Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise database access methods and processes independent of the UI so you can observe and log incorrect functioning target behavior or 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 and all database events have occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * base configuration imager and restorer * backup and recovery tools * installation-monitoring tools (registry, hard disk, CPU, memory, and so forth) * database SQL utilities and tools * Data-generation tools] |
| Success Criteria: | [The technique supports the testing of all key database access methods and processes.] |
| 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

|  |  |
| --- | --- |
| Technique Objective: | Exercise loading from MovieDB and having the UI setup correctly (navigation) |
| Technique: | Execute each use-case scenario’s individual use-case flows or functions and features, using valid and invalid data, to verify that:  • 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 |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | Unit Testing (Junit) already provided in Android Studio |
| Success Criteria: | Each test has success and each use-case is tested |
| Special Considerations: | n/a |

### Business Cycle Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise target-of-test and background processes according to required business models and schedules to observe and log target behavior.] |
| Technique: | [Testing will simulate several business cycles by performing the following:   * The tests used for target-of-test’s function testing will be modified or enhanced to increase the number of times each function is executed to simulate several different users over a specified period. * All time or date-sensitive functions will be executed using valid and invalid dates or time periods. * All functions that occur on a periodic schedule will be executed or launched at the appropriate time. * Testing will include 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.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * base configuration imager and restorer * backup and recovery tools * Data-generation tools] |
| Success Criteria: | [The technique supports the testing of all critical business cycles.] |
| Special Considerations: | • [System dates and events may require special support activities.  • A business model is required to identify appropriate test requirements and procedures.] |

### User Interface Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the following to observe and log standards conformance and target behavior:   * Navigation through the target-of-test reflecting 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 can be exercised–such as menus, size, position, state, and focus.] |
| Technique: | [Create or modify tests for each window to verify proper navigation and object states for each application window and object.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the Test Script Automation Tool.] |
| Success Criteria: | [The technique supports the testing of each major screen or window that will be used extensively by the end user.] |
| Special Considerations: | [Not all properties for custom and third-party objects can be accessed.] |

### Performance Profiling

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise behaviors for designated functional transactions or business functions under the following conditions to observe and log target behavior and application performance data:  • 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 that occur in each transaction.  • Scripts should be run on one machine (best case to benchmark single user, single transaction) and should be repeated with multiple clients (virtual or actual, see Special Considerations below).] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * an application performance profiling tool, such as Rational Quantify * installation-monitoring tools (registry, hard disk, CPU, memory, and so on * resource-constraining tools; for example, Canned Heat] |
| Success Criteria: | The technique supports testing:  • Single Transaction or single user: Successful emulation of the transaction scripts without any failures due to test implementation problems.]  • Multiple transactions or multiple users: Successful emulation of the workload without any failures due to test implementation problems.] |
| 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

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise designated transactions or business cases under varying workload conditions to observe and log target behavior and system performance data.] |
| Technique: | • [Use Transaction Test Scripts developed for Function or Business Cycle Testing as a basis, but remember to remove unnecessary interactions and delays.  • Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.  • Workloads should include (for example, Daily, Weekly, Monthly and so forth) Peak loads.  • Workloads should represent both Average as well as Peak loads.  • Workloads should represent both Instantaneous and Sustained Peaks.  • The Workloads should be executed under different Test Environment Configurations.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * Transaction Load Scheduling and control tool * installation-monitoring tools (registry, hard disk, CPU, memory, and so on) * resource-constraining tools (for example, Canned Heat) * Data-generation tools] |
| Success Criteria: | [The technique supports the testing of Workload Emulation, which is the successful emulation of the workload without any failures due to test implementation problems.] |
| 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

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test functions under the following stress conditions to observe and log target behavior that identifies and documents the conditions under which the system **fails** to continue functioning properly  • little or no memory available on the server (RAM and persistent storage space)  • maximum actual or physically capable number of clients connected or simulated  • multiple users performing the same transactions against the same data or accounts  • “overload” transaction volume or mix (see Performance Profiling above)] |
| 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 persistent storage space on the 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. |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * Transaction Load Scheduling and control tool * installation-monitoring tools (registry, hard disk, CPU, memory, and so on) * resource-constraining tools (for example, Canned Heat) * Data-generation tools] |
| Success Criteria: | The technique supports the testing of Stress Emulation. The system can be emulated successfully in one or more conditions defined as stress conditions and an observation of the resulting system state during and after the condition has been emulated can be captured.] |
| Special Considerations: | • [Stressing the network may require network tools to load the network with messages or packets.  • The persistent storage used for the system should temporarily be reduced to restrict the available space for the database to grow.  • Synchronize the simultaneous clients accessing of the same records or data accounts.] |

### Volume Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test under the following high volume scenarios to observe and log target behavior:  • 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) for an extended period.  • Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients are used to run queries and report transactions simultaneously for extended periods.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * Transaction Load Scheduling and control tool * installation-monitoring tools (registry, hard disk, CPU, memory, and so on) * resource-constraining tools (for example, Canned Heat) * Data-generation tools] |
| Success Criteria: | [The technique supports the testing of Volume Emulation. Large quantities of users, data, transactions, or other aspects of the system use under volume can be successfully emulated and an observation of the system state changes over the duration of the volume test can be captured.] |
| Special Considerations: | [What period of time would be considered an acceptable time for high volume conditions, as noted above?] |

### Security and Access Control Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test under the following conditions to observe and log target behavior:   * Application-level Security: an actor can access only those functions or data for which their user type is provided permissions. * System-level Security: 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] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * Test Script Automation Tool * “Hacker” security breach and probing tools * OS Security Admin Tools] |
| Success Criteria: | [The technique supports the testing of for each known actor type the appropriate functions or data affected by security settings can be tested.] |
| 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.] |

### Failover and Recovery Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Simulate the failure conditions and exercise the recovery processes (manual and automated) to restore the database, applications, and system to a desired, known, state. The following types of conditions are included in the testing to observe and log target behavior after recovery:  • power interruption to the client  • power interruption to the server  • communication interruption via network servers  • interruption, communication, or power loss to DASD (Dynamic Access Storage Devices) and DASD controllers  • incomplete cycles (data filter processes interrupted, data synchronization processes interrupted)  • invalid database pointers or keys  • invalid or corrupted data elements in database] |
| Technique: | [The tests already created for Function and Business Cycle testing can be used as a basis for creating a series of transactions to support failover and recovery testing, primarily to define the tests to be run to test that recovery was successful.  • Power interruption to the client: power the PC down.  • Power interruption to the server: simulate or initiate power down procedures for the server.  • Interruption via network servers: simulate or initiate communication loss with the network (physically disconnect communication wires or power down network servers or routers).  • Interruption, communication, or power loss to DASD and DASD controllers: simulate or physically eliminate communication with one or more DASDs or controllers.  Once the above conditions or simulated conditions are achieved, additional transactions should be executed and, upon reaching this second test point state, recovery procedures should be invoked.  Testing for incomplete cycles uses the same technique as described above except that the database processes themselves should be aborted or prematurely terminated.  Testing for the following conditions requires that a known database state be achieved.  Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * base configuration imager and restorer * installation monitoring tools (registry, hard disk, CPU, memory, and so on) * backup and recovery tools] |
| Success Criteria: | The technique supports the testing of:   * One or more simulated disasters involving one or more combinations of the application, database, and system. * One or more simulated recoveries involving one or more combinations of the application, database, and system to a known desired state.] |
| Special Considerations: | * [Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required. * Resources from the Systems (or Computer Operations), Database, and Networking groups are required. * These tests should be run after hours or on an isolated machine.] |

### Configuration Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the target-of-test on the required hardware and software configurations to observe and log target behavior under different configurations and identify changes in configuration state.] |
| Technique: | * [Use Function Test scripts. * Open and close various non-target-of-test related software, such as Microsoft Excel and Word applications, either as part of the test or prior to the start of the test. * Execute selected transactions to simulate actors 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.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * base configuration imager and restore * installation monitoring tools (registry, hard disk, CPU, memory, and so on)] |
| Success Criteria: | [The technique supports the testing of one or more combinations of the target test items running in expected, supported deployment environments.] |
| Special Considerations: | * [What non-target-of-test software is needed, is available, and what 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 system’s netware, network servers, databases, and so on, also needs to be documented as part of this test.] |

### Installation Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | [Exercise the installation of the target-of-test onto each required hardware configuration under the following conditions to observe and log installation behavior and configuration state changes:   * new installation: a new machine, never installed previously with <Project Name> * update: a machine previously installed <Project Name>, same version * update: a machine previously installed <Project Name>, older version] |
| Technique: | * [Develop automated or manual scripts to validate the condition of the target machine.   + new: never installed   + same or older version already installed * Launch or perform installation. * Using a predetermined subset of Function Test scripts, run the transactions.] |
| Oracles: | [Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.] |
| Required Tools: | [The technique requires the following tools:   * base configuration imager and restorer * installation monitoring tools (registry, hard disk, CPU, memory, and so on)] |
| Success Criteria: | [The technique supports the testing of the installation of the developed product in one or more installation configurations.] |
| Special Considerations: | [What <Project Name> transactions should be selected to comprise a confidence test that <Project Name> application has been successfully installed and no major software components are missing?] |

# Entry and Exit Criteria

## Test Plan

### Test Plan Entry Criteria

n/a

### Test Plan Exit Criteria

n/a

### Suspension and Resumption Criteria

n/a

## Test Cycles

### Test Cycle Entry Criteria

n/a

### Test Cycle Exit Criteria

n/a

### Test Cycle Abnormal Termination

n/a

# Deliverables

## Test Evaluation Summaries

n/a

## Reporting on Test Coverage

tbd

## Perceived Quality Reports

n/a

## Incident Logs and Change Requests

n/a

## Smoke Test Suite and Supporting Test Scripts

n/a

## Additional Work Products

n/a

### Detailed Test Results

n/a

### Additional Automated Functional Test Scripts

tbd

### Test Guidelines

n/a

### Traceability Matrices

n/a

# Testing Workflow

So far our testing is done locally, before every git commit (merge with master). Also by every developer on their own.

# Environmental Needs

[This section presents the non-human resources required for the **Test Plan**.]

## Base System Hardware

| **System Resources** | | |
| --- | --- | --- |
| **Resource** | **Quantity** | **Name and Type** |
| Web Server | 1 |  |
| —Network or Subnet |  | TBD |
| —Server Name |  | Cduc.su |
| —Database Name |  | MariaDB |
| Client Test PCs | 3 |  |
| —Include special configuration requirements |  | TBD |
| Test Repository | 1 | Git |
| —Network or Subnet |  | TBD |
| —Server Name |  | TBD |
| Test Development PCs | 3 | TBD |

## Base Software Elements in the Test Environment

The following base software elements are required in the test environment for this *Test Plan*.

[Note: Add or delete items as appropriate.]

| **Software Element Name** | **Version** | **Type and Other Notes** |
| --- | --- | --- |
| JDK |  | Developer Kit (with compiler and VM) |
| Android Studio |  | IDE |
| JUnit |  | Library |

## Productivity and Support Tools

The following tools will be employed to support the test process for this *Test Plan*.

[Note: Add or delete items as appropriate.]

| **Tool Category or Type** | **Tool Brand Name** | **Vendor or In-house** | **Version** |
| --- | --- | --- | --- |
| Project Management | JIRA | Atlassain |  |
| Version Control | Git |  |  |

## Test Environment Configurations

n/a

| **Configuration Name** | **Description** | **Implemented in Physical Configuration** |
| --- | --- | --- |
| Average user configuration |  |  |
| Minimal configuration supported |  |  |
| Visually and mobility challenged |  |  |
| International Double Byte OS |  |  |
| Network installation (not client) |  |  |

# Responsibilities, Staffing, and Training Needs

## People and Roles

Everybody included in this project will be developer, tester and designer.

| **Human Resources** | | |
| --- | --- | --- |
| **Role** | **Minimum Resources Recommended**  **(number of full-time roles allocated)** | **Specific Responsibilities or Comments** |
| Test Manager |  | Provides management oversight.  Responsibilities include:   * planning and logistics * agree mission * identify motivators * acquire appropriate resources * present management reporting * advocate the interests of test * evaluate effectiveness of test effort |
| Test Analyst |  | Identifies and defines the specific tests to be conducted.  Responsibilities include:   * identify test ideas * define test details * determine test results * document change requests * evaluate product quality |
| Test Designer |  | Defines the technical approach to the implementation of the test effort.  Responsibilities include:   * define test approach * define test automation architecture * verify test techniques * define testability elements * structure test implementation |
| Tester |  | Implements and executes the tests.  Responsibilities include:   * implement tests and test suites * execute test suites * log results * analyze and recover from test failures * document incidents |
| Test System Administrator |  | Ensures test environment and assets are managed and maintained.  Responsibilities include:   * administer test management system * install and support access to, and recovery of, test environment configurations and test labs |
| Database Administrator, Database Manager |  | Ensures test data (database) environment and assets are managed and maintained.  Responsibilities include:   * support the administration of test data and test beds (database). |
| Designer |  | Identifies and defines the operations, attributes, and associations of the test classes.  Responsibilities include:   * defines the test classes required to support testability requirements as defined by the test team |
| Implementer |  | Implements and unit tests the test classes and test packages.  Responsibilities include:   * creates the test components required to support testability requirements as defined by the designer |

## Staffing and Training Needs

n/a

# Iteration Milestones

tbd

| **Milestone** | **Planned Start Date** | **Actual Start Date** | **Planned End Date** | **Actual End Date** |
| --- | --- | --- | --- | --- |
| Iteration Plan agreed |  |  |  |  |
| Iteration starts |  |  |  |  |
| Requirements baselined |  |  |  |  |
| Architecture baselined |  |  |  |  |
| User Interface baselined |  |  |  |  |
| First Build delivered to test |  |  |  |  |
| First Build accepted into test |  |  |  |  |
| First Build test cycle finishes |  |  |  |  |
| [Build Two will not be tested] |  |  |  |  |
| Third Build delivered to test |  |  |  |  |
| Third Build accepted into test |  |  |  |  |
| Third Build test cycle finishes |  |  |  |  |
| Fourth Build delivered to test |  |  |  |  |
| Fourth Build accepted into test |  |  |  |  |
| Iteration Assessment review |  |  |  |  |
| Iteration ends |  |  |  |  |

# Risks, Dependencies, Assumptions, and Constraints

| **Risk** | **Mitigation Strategy** | **Contingency (Risk is realized)** |
| --- | --- | --- |
| Database is corrupt | Database Admin will have the task to keep the database clean and well setup/protected | * Restore Database |

n/a

| **Dependency between** | **Potential Impact of Dependency** | **Owners** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

n/a

| **Assumption to be proven** | **Impact of Assumption being incorrect** | **Owners** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

n/a

| **Constraint on** | **Impact Constraint has on test effort** | **Owners** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# Management Process and Procedures

## Measuring and Assessing the Extent of Testing

n/a

## Assessing the Deliverables of this Test Plan

n/a

## Problem Reporting, Escalation, and Issue Resolution

n/a

## Managing Test Cycles

n/a

## Traceability Strategies

n/a

## Approval and Signoff

n/a