|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <02/05/y17> | <1.0> | Init | Simon Gerhardt |
| <20.06/y17> | <1.1> | Added Stresstest | Fabian Blatz |
|  |  |  |  |
|  |  |  |  |

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. Another sort of test we are performing is a stress test of the running app.

## 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

### 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

### User Interface Testing

n/a

### Performance Profiling

n/a

### Load Testing

n/a

|  |  |
| --- | --- |
| Technique Objective: | Verify appropriate behavior and response time under high load. |
| Technique: | Swamp endpoints with concurrent requests. Observe CPU load on server and measure response times. Commands are entered manually on the console or can alternatively be automated with a shell script. |
| Oracles: | We assume that the endpoints we test actually do the advertised work. |
| Required Tools: | - Apache Bench tool |
| Success Criteria: | reports success responses only with mean response time < 100ms |
| Special Considerations: | n/a |

### Stress Testing

|  |  |
| --- | --- |
| Technique Objective: | Ensure App is working correctly under high workload |
| Technique: | Trying to crash the app by switching fast in the app |
| Oracles: | We assume that the app does not crash. |
| Required Tools: | - Android SDK and AndroidDebugBridge |
| Success Criteria: | App does not crash |
| Special Considerations: | n/a |

### Volume Testing

n/a

### Security and Access Control Testing

n/a

### Failover and Recovery Testing

n/a

### Configuration Testing

n/a

### Installation Testing

n/a

### Unit Testing

|  |  |
| --- | --- |
| Technique Objective: | Testing the functionality of the code |
| Technique: | Testing using Junit tests |
| Oracles: | If all tests pass we have success (assert is true) |
| Required Tools: | Junit |
| Success Criteria: | All tests pass |
| Special Considerations: | n/a |

# 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

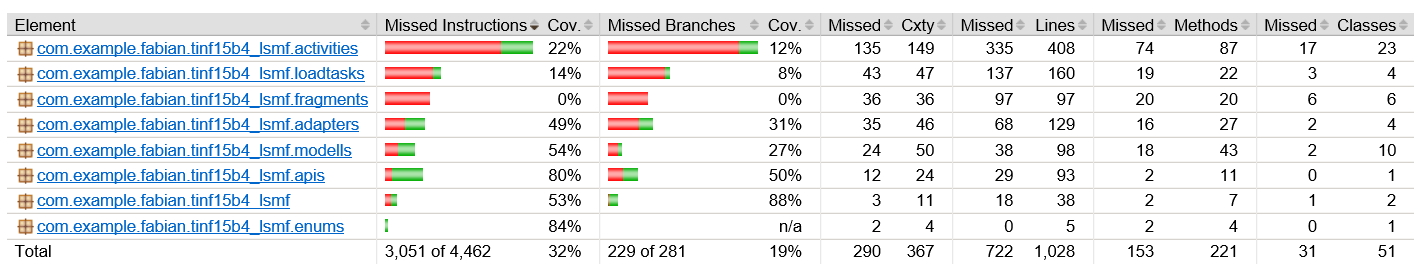
n/a

### Test Cycle Abnormal Termination

n/a

# Deliverables

## Test Evaluation Summaries



## Reporting on Test Coverage

The reporting on our test-coverage is always live monitorable through the badge on our github repo. The test is run during the build testing of Travis.ci and the results are uploaded to codecov.

Stress tests can only be performed locally since Travis.ci has problem with the ADB, which means it is not automatically performed after each commit, it’s the developers task.

## Perceived Quality Reports

The amount of results we are getting is sufficient to ensure the apps functionality.

## Incident Logs and Change Requests

n/a

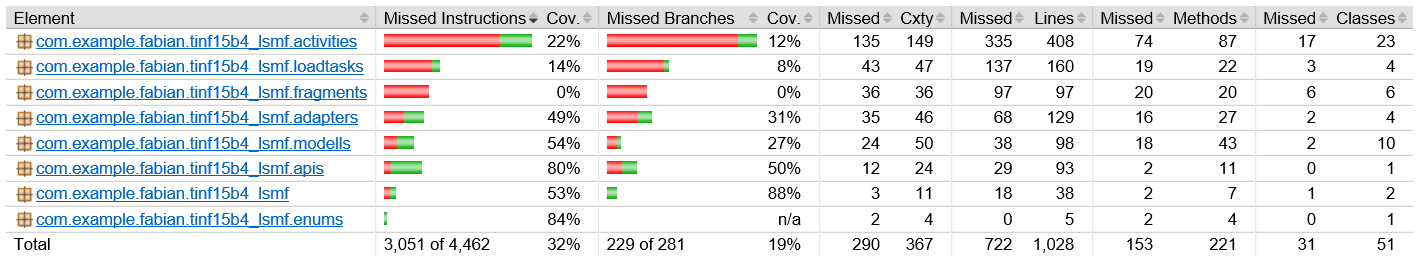
## Smoke Test Suite and Supporting Test Scripts

n/a

## Additional Work Products

n/a

### Detailed Test Results



And the output of the stress test performed by SDK monkey:

C:\Users\blatzfab\AppData\Local\Android\sdk\platform-tools>adb shell monkey -p com.example.fabian.tinf15b4\_lsmf -v 100

:Monkey: seed=1498212514806 count=100

:AllowPackage: com.example.fabian.tinf15b4\_lsmf

:IncludeCategory: android.intent.category.LAUNCHER

:IncludeCategory: android.intent.category.MONKEY

// Event percentages:

// 0: 15.0%

// 1: 10.0%

// 2: 2.0%

// 3: 15.0%

// 4: -0.0%

// 5: -0.0%

// 6: 25.0%

// 7: 15.0%

// 8: 2.0%

// 9: 2.0%

// 10: 1.0%

// 11: 13.0%

:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.example.fabian.tinf15b4\_lsmf/.activities.LoginActi

vity;end

// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.fabian.tinf15b4\_lsmf/.activities.LoginActivity } in package

com.example.fabian.tinf15b4\_lsmf

:Sending Touch (ACTION\_DOWN): 0:(532.0,1237.0)

:Sending Touch (ACTION\_UP): 0:(562.5359,1224.78)

:Sending Touch (ACTION\_DOWN): 0:(586.0,831.0)

:Sending Touch (ACTION\_UP): 0:(581.2366,900.5496)

:Sending Touch (ACTION\_DOWN): 0:(855.0,892.0)

:Sending Touch (ACTION\_UP): 0:(857.1568,887.71375)

:Sending Touch (ACTION\_DOWN): 0:(964.0,1576.0)

:Sending Touch (ACTION\_UP): 0:(959.88617,1544.8258)

:Sending Trackball (ACTION\_MOVE): 0:(-5.0,3.0)

:Sending Trackball (ACTION\_MOVE): 0:(-2.0,4.0)

:Sending Touch (ACTION\_DOWN): 0:(782.0,1173.0)

:Sending Touch (ACTION\_UP): 0:(773.2234,1172.1191)

:Sending Trackball (ACTION\_MOVE): 0:(0.0,-4.0)

:Sending Trackball (ACTION\_MOVE): 0:(-4.0,1.0)

:Sending Touch (ACTION\_DOWN): 0:(642.0,677.0)

:Sending Touch (ACTION\_UP): 0:(633.97864,672.3119)

:Sending Trackball (ACTION\_MOVE): 0:(4.0,-3.0)

Events injected: 100

:Sending rotation degree=0, persist=false

:Dropped: keys=0 pointers=0 trackballs=0 flips=0 rotations=0

## Network stats: elapsed time=605ms (0ms mobile, 0ms wifi, 605ms not connected)

// Monkey finished

### Additional Automated Functional Test Scripts

n/a

### 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.

We also added Travis to automatically test every commit and recognize failures immediately.

To have an overview concerning our code coverage percentage we also integrated codecov in our project.

# Environmental Needs

## 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

Android OS with SDK>15 and Version>4.1

# 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

| **Milestone** | **Planned Start Date** | **Actual Start Date** | **Planned End Date** | **Actual End Date** |
| --- | --- | --- | --- | --- |
| Iteration starts | 25.04.2017 |  |  |  |
| >20% Test Coverage | 25.04.2017 | 25.04.2017 | 01.06.2017 | 02.06.2017 |
| Have Functional Tests | 25.04.2017 | 25.04.2017 | 24.05.2017 | 24.05.2017 |
| Have Unit Tests | 25.04.2017 | 25.04.2017 | 24.05.2017 | 24.05.2017 |
| Have Stess Tests | 25.05.2017 | 15.06.2017 | 20.06.2017 | 20.06.2017 |
| Tests integrated in CI | 25.04.2017 | 25.04.2017 | 02.05.2017 | 02.05.2017 |
| 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 |
| Test data proves to be inadequate | Tester will indicate what is required and will ensure to use a full set of suitable and protected test data. | * Redefine test data |

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