
Leruka

Leruka

Master Test Plan

Version <1.5>

Revision History

Date	Version	Description	Author
11/05/2016	1.0	Dokument erstellt	Ruth W.
17/05/2016	1.1	Dokument geändert	Kassandra F., Ruth W.
18/05/2016	1.2	Testing with end user hinzugefügt und weitere Informationen hinzugefügt	Ruth W.
27/05/2016	1.3	Reports eingebunden	Ruth W.
31/05/2016	1.4	Metric reports in Appendix eingefügt	Ruth W.
08/06/2016	1.5	Anpassungen	Ruth W.

Table of Contents

[Introduction](#)

[Purpose](#)

[Scope](#)

[Intended Audience](#)

[Document Terminology and Acronyms](#)

[References](#)

[Evaluation Mission and Test Motivation](#)

[Background](#)

[Evaluation Mission](#)

[Test Motivators](#)

[Target Test Items](#)

[Outline of Planned Tests](#)

[Outline of Test Inclusions](#)

[Outline of Other Candidates for Potential Inclusion](#)

[Outline of Test Exclusions](#)

[Test Approach](#)

[Initial Test-Idea Catalogs and Other Reference Sources](#)

[Testing Techniques and Types](#)

[Functional Testing \(User Interface Testing\)](#)

[Unit Test](#)

[Testing with end user](#)

[Deliverables](#)

[Test Evaluation Summaries](#)

[Reporting on Test Coverage](#)

[Perceived Quality Reports](#)

[Incident Logs and Change Requests](#)

[Smoke Test Suite and Supporting Test Scripts](#)

[Additional Work Products](#)

[Detailed Test Results](#)

[Testing Workflow](#)

[Environmental Needs](#)

[Base System Hardware](#)

[Base Software Elements in the Test Environment](#)

[Productivity and Support Tools](#)

[Test Environment Configurations](#)

[Responsibilities, Staffing, and Training Needs](#)

[People and Roles](#)

[Staffing and Training Needs](#)

[Iteration Milestones](#)

[Risks, Dependencies, Assumptions, and Constraints](#)

[Appendix](#)

[Cyclomatic Complexity](#)

[Code Style](#)

Master Test Plan

Introduction

Purpose

The purpose of the Master Test Plan is to gather all of the information necessary to plan and control the test effort for a given master test plan. 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 Leruka supports the following objectives:

- Functional testing for main menu
- Unit test for the implemented functions in unity
- Testing with end user

Scope

Functional Testing

Will blackbox test the application behaviour by using cucumber.

Unit tests

Will test the internal application logic.

Testing with end user

Will test the user interface if it is easy to learn.

Intended Audience

- Students
- Professors
- Programmer

Document Terminology and Acronyms

n/a

References

n/a

Evaluation Mission and Test Motivation

Automated Testing is good for detect coding errors and will show them. Programmer are lazy people and test only things they have changed. To prevent error in the productive run, we have to detect error in the development process.

Background

Because our team includes some programmers, it could happen that while changing, deleting or updating code the already existing code is negatively affected. The automatic testing should discover these.

Evaluation Mission

Testing is done to provide a stable software. And we will fulfill the goal by the following points.

- find as many bugs as possible
- find important problems

- certify a standard
- verify software specification

Test Motivators

- technical risks
- functional requirements
- non-functional requirements

Target Test Items

The listing below identifies those test items _software, hardware, and supporting product elements _that have been identified as targets for testing. This list represents what items will be tested.

- Game logic
- Controller
- Models

Outline of Planned Tests

Outline of Test Inclusions

- Functional Tests
- Unit Tests
- Testing with end users

Outline of Other Candidates for Potential Inclusion

n/a

Outline of Test Exclusions

- n/a

Test Approach

- Functional Tests
- Unit Tests, automatically after gradle build
- Testing with end user

Initial Test-Idea Catalogs and Other Reference Sources

Testing Techniques and Types

Functional Testing (User Interface Testing)

Technique Objective:	Navigation through user interface of the app.
Technique:	<ul style="list-style-type: none">• Testing the API• User<ul style="list-style-type: none">○ Create User○ Login User○ Is logged in• Highscore<ul style="list-style-type: none">○ view personal highscore○ view public highscore• Instruction<ul style="list-style-type: none">○ view instruction• Change settings<ul style="list-style-type: none">○ change display name

	○ change password
Oracles:	The Test is successfull, if all correct answeres are running through.
Required Tools:	<ul style="list-style-type: none">● Cucumber● Calabash● End device with Android (smartphone, tablet, etc.)
Success Criteria:	All tests run successful.

Unit Test

Technique Objective:	Testing the functionality of the code
Technique:	<ul style="list-style-type: none">● Testing the server functions of the testable classes.<ul style="list-style-type: none">○ User○ Score○ Game
Oracles:	The Test is successfull, if all correct answeres are running through
Required Tools:	<ul style="list-style-type: none">● Android Studio● JUnit
Success Criteria:	All tests pass.

Testing with end user

Technique Objective:	Testing the simplicity of the app
Technique:	<ul style="list-style-type: none">● Testing the menu for simplicity● Testing the app for easy understanding● User fill out a survey
Oracles:	The test user are happy with the app. The complete app is easy to understand, it is self explaining. The menu navigation is simple.
Required Tools:	<ul style="list-style-type: none">● End device with Android v4.2 (smartphone, tablet, etc.)
Success Criteria:	End user is happy.

Deliverables

Test Evaluation Summaries

Test evaluation is done by hand directly after the test executed.

Reporting on Test Coverage

Test coverage is reported on our [SonarQube project](#).

Perceived Quality Reports

For showing quality we also use [SonarQube](#).

Incident Logs and Change Requests

n/a

Smoke Test Suite and Supporting Test Scripts

n/a

Additional Work Products

n/a

Detailed Test Results

[Feature file results](#)

[Unit Test reports](#)

[End User evaluation](#)

Testing Workflow

Beside the automatically tested unit tests we start the test manually, when we think it is necessary.

Environmental Needs

Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Test Plan*.

System Resources		
Resource	Quantity	Name and Type
Database Server	1	MySQL
Server Name	1	LerukaServer
Database Name	1	Leruka/MySQL
Test Development PCs	3	Kassandra, Leif, Ruth

Base Software Elements in the Test Environment

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

Software Element Name	Version	Type and Other Notes
Android	4.2	Operating System
MySQL		Database Server
Android Studio		IDE

Productivity and Support Tools

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

Tool Category or Type	Tool Brand Name	Vendor or In-house	Version
Project Management	JIRA	Atlassian	7.0
Test Coverage Monitor or Profiler	SonarQube		

Test Environment Configurations

The following Test Environment Configurations needs to be provided and supported for this project.

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

This table shows the staffing assumptions for the test effort.

Human Resources		
Role	Minimum Resources Recommended (number of full-time roles allocated)	Specific Responsibilities or Comments
Test Manager	1	Provides management oversight. Responsibilities include: <ul style="list-style-type: none">• 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	2	Identifies and defines the specific tests to be conducted.

		<p>Responsibilities include:</p> <ul style="list-style-type: none"> • identify test ideas • define test details • determine test results • document change requests • evaluate product quality
Test Designer	2	<p>Defines the technical approach to the implementation of the test effort.</p> <p>Responsibilities include:</p> <ul style="list-style-type: none"> • define test approach • define test automation architecture • verify test techniques • define testability elements • structure test implementation
Tester	3	<p>Implements and executes the tests.</p> <p>Responsibilities include:</p> <ul style="list-style-type: none"> • implement tests and test suites • execute test suites • log results • analyze and recover from test failures • document incidents
Test System Administrator	1	<p>Ensures test environment and assets are managed and maintained.</p> <p>Responsibilities include:</p> <ul style="list-style-type: none"> • administer test management system • install and support access to, and recovery of, test environment configurations and test labs
Database Administrator, Database Manager	1	<p>Ensures test data (database) environment and assets are managed and maintained.</p> <p>Responsibilities include:</p> <ul style="list-style-type: none"> • support the administration of test data and test beds (database).
Designer	1	<p>Identifies and defines the operations, attributes, and associations of the test classes.</p> <p>Responsibilities include:</p> <ul style="list-style-type: none"> • defines the test classes required to support testability requirements as defined by the test team

Implementer	3	<p>Implements and unit tests the test classes and test packages.</p> <p>Responsibilities include:</p> <ul style="list-style-type: none"> · creates the test components required to support testability requirements as defined by the designer
-------------	---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Staffing and Training Needs

This section outlines how to approach staffing and training the test roles for the project.

Iteration Milestones

Milestone	Planned Start Date	Actual Start Date	Planned End Date	Actual End Date
> 20% Test Coverage	11.05.2016	11.05.2016	22.05.2016	31.05.2016
Have functional tests	28.11.2015	28.11.2015	01.11.2015	01.11.2015
Have JUnit tests	20.04.2016	20.04.2016	23.04.2016	31.05.2016
Have installation test	01.06.2016		01.06.2016	
Have end user test	01.06.2016	02.06.2016	05.06.2016	06.06.2016

Risks, Dependencies, Assumptions, and Constraints

Risk	Mitigation Strategy	Contingency (Risk is realized)
Test data proves to be inadequate.	<p><Customer> will ensure a full set of suitable and protected test data is available.</p> <p><Tester> will indicate what is required and will verify the suitability of test data.</p>	<ul style="list-style-type: none"> · Redefine test data · Review Test Plan and modify components (that is, scripts) · Consider Load Test Failure
Technical Problems	<Tester> needs to make sure everything is running fine	<ul style="list-style-type: none"> · Fix the problem

Dependency between	Potential Impact of Dependency	Owners

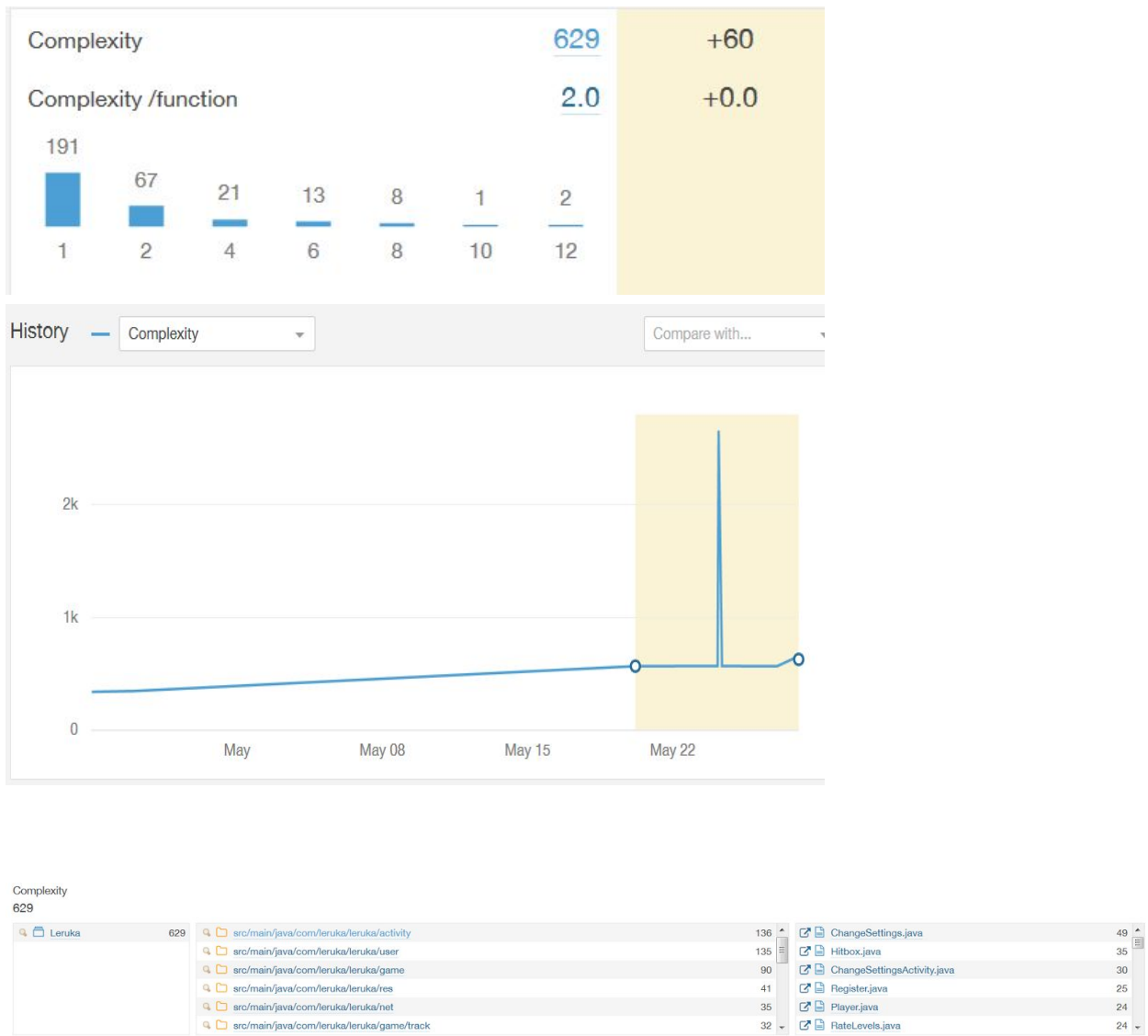
Assumption to be proven	Impact of Assumption being incorrect	Owners

--	--	--

Constraint on	Impact Constraint has on test effort	Owners

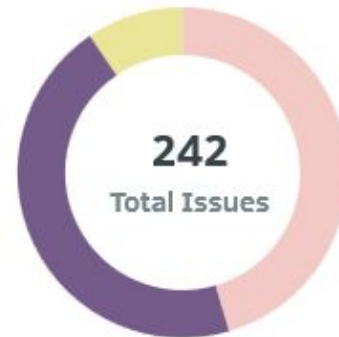
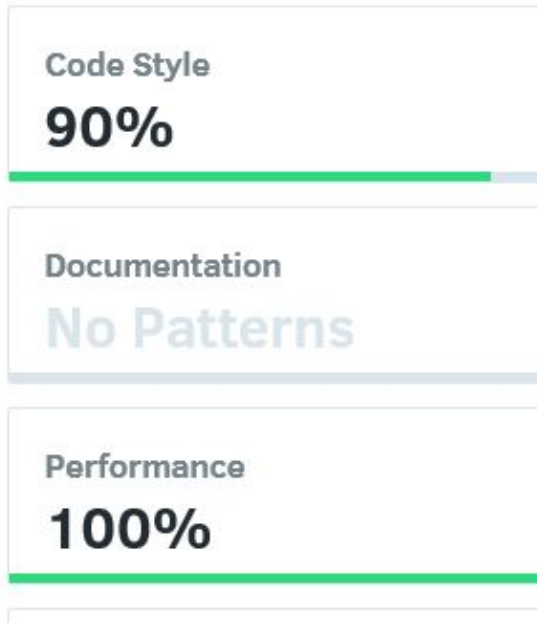
Appendix

Cyclomatic Complexity



<http://193.196.7.25/overview/structure?id=leruka>

Code Style



Code Style	110
Error Prone	109
Unused Code	23

Current Issues ▾ master ▾

Language All ▾ Category Cod... ▾ Level All ▾ Pattern All ▾ Author All ▾ X

app/src/main/java/com/leruka/protobuf/Highscore.java

Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes.	▾
117 private volatile Object levelName_;	
Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes.	▾
965 private Object sessionId_ = "";	
Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes.	▾
1418 public static final int ERRORCODE_FIELD_NUMBER = 4;	
Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes.	▾
1419 private java.util.List<Integer> errorCode_;	
Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes.	▾
2441 public static final int SCORE_FIELD_NUMBER = 3;	

app/src/main/java/com/leruka/protobuf/Rating.java

<https://www.codacy.com/app/leruka/leruka/dashboard>