Leruka

Master Test Plan

Version <1.4>

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.

Table of Contents

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 negativly 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

- tbd

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

Technique Objective:	Navigation through user interface of the app.
Technique:	● Testing the API ● User ○ Create User ○ Login User ○ Is logged in ● Highscore ○ view personal highscore ○ view public highscore ● Instruction ○ view instruction ● Change settings ○ change display name

	○ change password
Oracles:	The Test is successfull, if all correct answeres are running through.
Required Tools:	● 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:	● Testing the server functions of the testable classes. ○ User ○ Score ○ Game
Oracles:	The Test is successfull, if all correct answeres are running through
Required Tools:	● Android Studio ● JUnit
Success Criteria:	All tests pass.

Testing with end user

Technique Objective:	Testing the simplicity of the app
Technique:	● 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:	● 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

tbd

Smoke Test Suite and Supporting Test Scripts

Additional Work Products

n/a

Detailed Test Results

Feature file results

Unit Test reports

End User evaluation -> tbd

Additional Automated Functional Test Scripts

Feature files tbd

Unit tests tbd

Test Guidelines

tbd

Traceability Matrices

tbd

Testing Workflow

Beside the automatically tested unit tests we start the test manuelly, 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	
Server Name	1	TBD
Database Name	1	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*.

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: · 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. Responsibilities include: · identify test ideas · define test details · determine test results · document change requests · evaluate product quality
Test Designer	2	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	3	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	1	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	1	Ensures test data (database) environment and assets are managed and maintained. Responsibilities include: · support the administration of test data and test beds (database).

Designer	1	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	3	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

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	
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	
Have installation test	01.06.2016		01.06.2016	
Have end user test	01.06.2016		05.06.2016	

Risks, Dependencies, Assumptions, and Constraints

Risk	Mitigation Strategy	Contingency (Risk is realized)
Test data proves to be inadequate.	<Customer> will ensure a full set of suitable and protected test data is available. <Tester> will indicate what is required and will verify the suitability of test data.	· 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	· Fix the problem

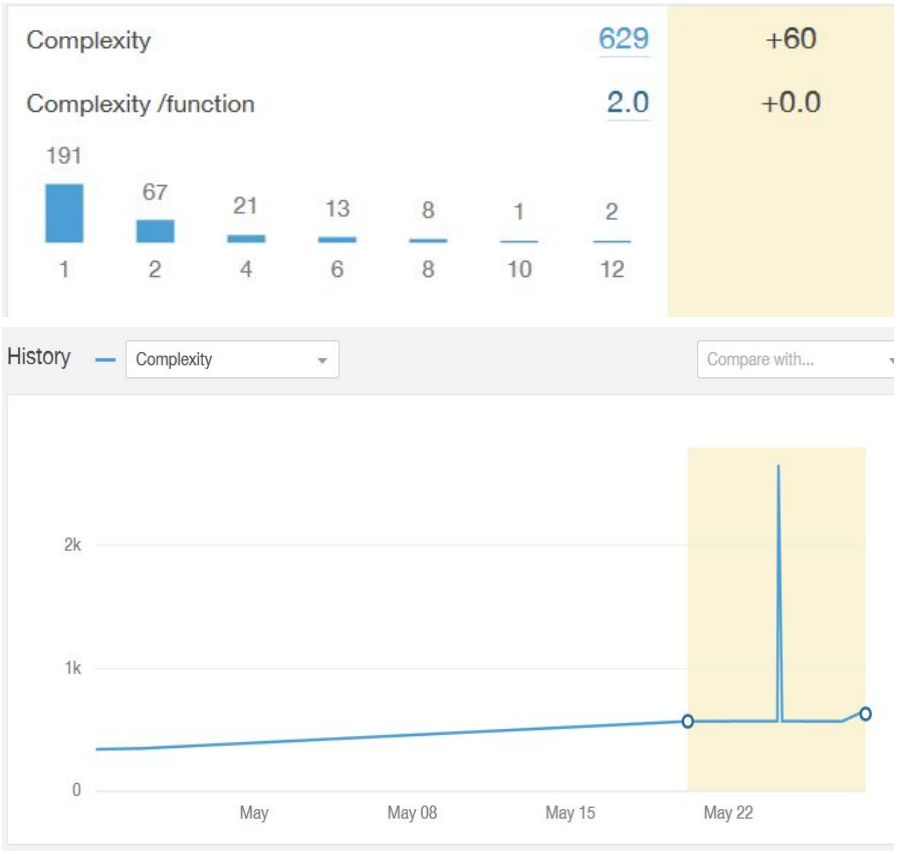
Dependency between	Potential Impact of Dependency	Owners

Assumption to be proven	Impact of Assumption being incorrect	Owners

Constraint on	Impact Constraint has on test effort	Owners

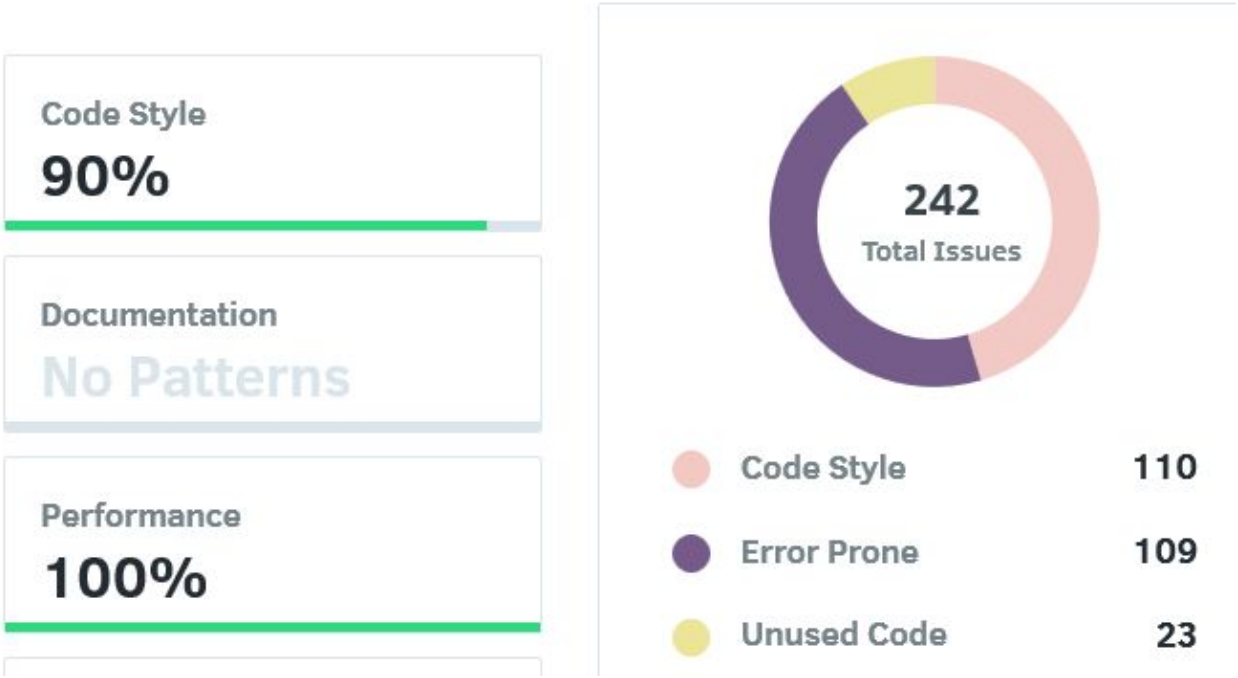
Appendix

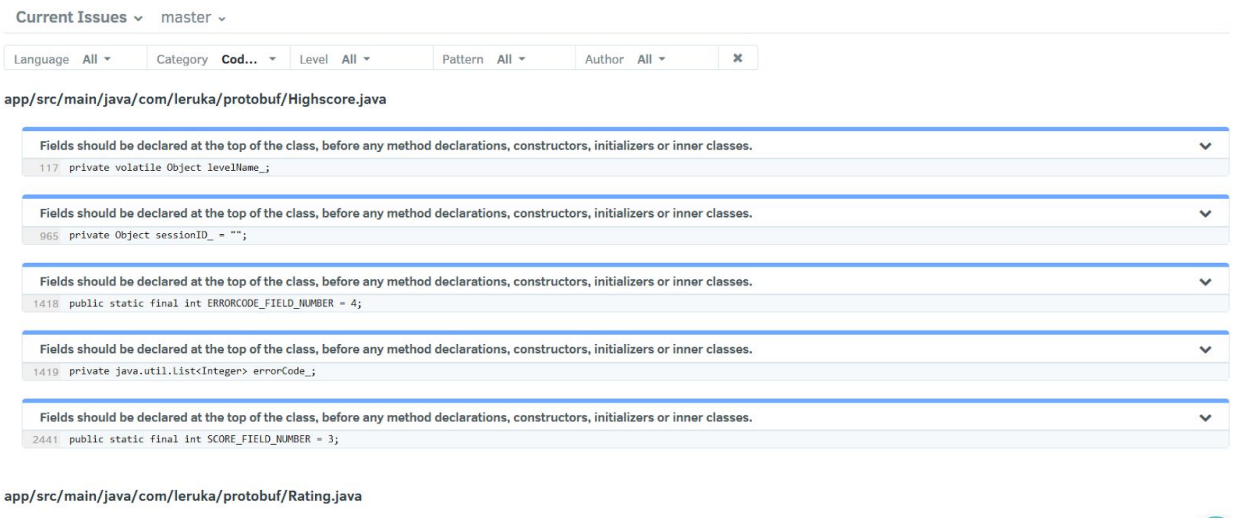
Cyclomatic Complexity



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

Code Style





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