Test Plan

Version 1.0

Revision History

Date	Version	Description	Author
10.05.2017	1.0	First version	Leon Gottschick
04.07.2017	1.1	Metrics addition	Leon Gottschick

1. Introduction

1.1 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 GottMusIg supports the following objectives:

- Unit Testing
- UI Testing
- Spring Integration Testing

1.2 Scope

Unit Tests

Unit Tests with JUnit 4 in the Database Service, the Frontend and the backend.

Integration Tests

Integration Tests with Spring in the Database Service, the Frontend and the backend.

1.3 Intended Audience

Professors, students.

1.4 Document Terminology and Acronyms

n/a

1.5 References

[This subsection provides a list of the documents referenced elsewhere within the **Test Plan**. Identify each document by title, version (or report number if applicable), date, and publishing organization or original author. Avoid listing documents that are influential but not directly referenced. Specify the sources from which the “official versions” of the references can be obtained, such as intranet UNC names or document reference codes. This information may be provided by reference to an appendix or to another document.]

2. Evaluation Mission and Test Motivation

Testing helps at developing better code by checking the functionality of already written code and whether some code changes have impact on the intended functionality.

2.1 Background

Our project is heavily dependent on external software as well as the internal communication between the three main parts of our application: the backend, the database model and the frontend. Because of this cross-project dependency we rely heavily on interface and

integration testing to guarantee a smooth interaction between all parts of our software.

But in order to have good interactions between the different part of our software project, every single component has to run smoothly and bug-free, so Unit testing is another big part of our infrastructure.

Because of the large community of World of Warcraft our project could possibly generate a lot of load on our servers so load testing is important.

To guarantee good software we have the following goals in mind:

- Code coverage above 80%
- Automated Tests and test reports with continuous integration
- Meaningful test cases

2.2 Evaluation Mission

- find as many bugs as possible
- find important problems, assess perceived quality risks
- advise about perceived project risks
- verify a specification (requirements, design or claims)
- advise about product quality, satisfy stakeholders
- advise about testing

2.3 Test Motivators

- Class grade
- Personal aim to develop good and bug-free software

3. Target Test Items

- Database Model
- UI
- Blizzard API
- Simulation Craft
- Backend

4. Outline of Planned Tests

4.1 Outline of Test Inclusions

- Junit testing (proof)
- Spring Integration testing (proof and proof)
- UI testing with cucumber and selenium (proof)
- Function Testing with external users

4.2 Outline of Other Candidates for Potential Inclusion

- Load testing (Jmeter etc.)

5. Test Approach

- Unit tests for every component of the system.
- Integration tests for Interfaces, APIs

5.1 Initial Test-Idea Catalogs and Other Reference Sources

5.2 Testing Techniques and Types

5.2.1 Data and Database Integrity Testing

Technique Objective:	Database Model is correct, and initial data is present
Technique:	Generate the database model with the entities in the model and populate it with initial data via a SQL Script
Oracles:	
Required Tools:	- Spring Data JPA - Spring - H2
Success Criteria:	Object- relational mapping runs flawlessly link to spring integration test

Special Considerations:	Model generation made on every restart of the server

5.2.2 Function Testing

Technique Objective:	User Interface fits to the Requirements specified in our feature file
Technique:	Writing cucumber tests with selenium

Oracles:	Test succeeds
Required Tools:	- Gherkin language - Cucumber - selenium
Success Criteria:	Tests succeed (proof)
Special Considerations:	Run on every deployment of the application

5.2.3 Business Cycle Testing

n/a

5.2.4 User Interface Testing

Technique Objective:	Usability tests by external users
Technique:	Supervised usability tests of our UI
Oracles:	User intuitively recognizes and appreciates key features of the UI
Required Tools:	External user
Success Criteria:	User likes the design and features of the website.
Special Considerations:	Done manually by people outside of the project

5.2.5 Performance Profiling

n/a

5.2.6 Load Testing

Technique Objective:	Test the stability of the database
Technique:	Continuous load tests of the database server
Oracles:	
Required Tools:	Apache Jmeter
Success Criteria:	The database can handle thousands of concurrent queries
Special Considerations:	Should be done regularly

5.2.7 Stress Testing

n/a

5.2.8 Volume Testing

n/a

5.2.9 Security and Access Control Testing

n/a

5.2.10 Failover and Recovery Testing

n/a

5.2.11 Configuration Testing

n/a

5.2.12 Installation Testing

n/a

6. Entry and Exit Criteria

6.1 Test Plan

6.1.1 Test Plan Entry Criteria

tbd

6.1.2 Test Plan Exit Criteria

tbd

6.1.3 Suspension and Resumption Criteria

tbd

6.2 Test Cycles

6.2.1 Test Cycle Entry Criteria

tbd

6.2.2 Test Cycle Exit Criteria

tbd

6.2.3 Test Cycle Abnormal Termination

tbd

7. Deliverables

7.1 Test Evaluation Summaries

Testergebnis

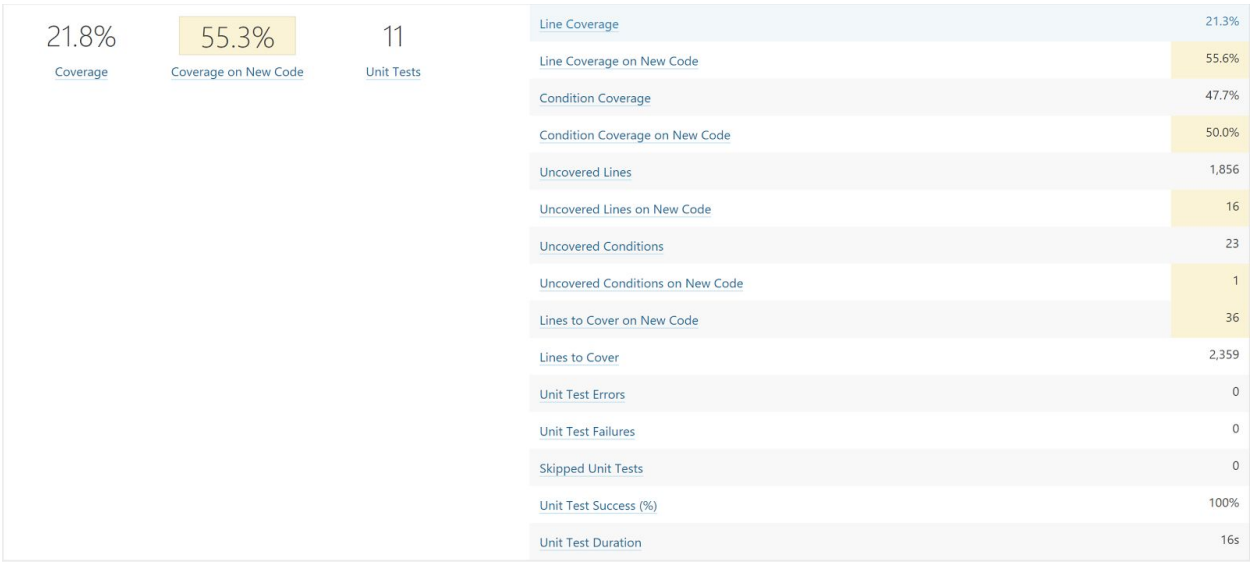
Fehlschläge (±0)

Tests (±0)

Modul	Fehlgeschlagen	(Differenz)	Gesamt	(Differenz)
com.gottmusig:database-service	0		11	

7.2 Reporting on Test Coverage

Code Coverage tools integrated in the IDE as well as SonarQube integrated in our Jenkins Build pipeline.



8. Testing Workflow

Every developer is responsible for his part of the application.

Every developer can run all tests in his IDEA.

Automated tests at deployment with maven and jenkins

9. Environmental Needs

9.1 Base System Hardware

System Resources		
Resource	**Quantity**	**Name and Type**
Database Server	1	MySQL on a cloud server
Client Test PCs	1	Laptop
Test Development PCs	3	Kamil, Christoph, Leon
Jenkins Server	1	Jenkins on a cloud server

9.2 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
Windows 10		Operating System
Linux		Operating System
Mac OS		Operating System

Firefox		Internet Browser
Microsoft Edge		Internet Browser
Google Chrome		Internet Browser
IntelliJ IDEA		IDE
Eclipse		IDE
JUnit		Testing Framework
Spring		Java Framework
H2		In-memory database for testing
Selenium		Browser testing framework

9.3 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
Test Management	IntelliJ IDEA, Maven, Jenkins

Test Coverage Monitor or Profiler	JaCoCo
Project Management	GitHub
DBMS tools	H2
Metrics Tool	SonarQube

9.4 Test Environment Configurations

n/a

10. Responsibilities, Staffing, and Training Needs

10.1 People and Roles

This table shows the staffing assumptions for the test effort.

Human Resources		
Role	**Minimum Resources Recommended**	**Specific Responsibilities or Comments**

	(number of full-time roles allocated)	
Test Manager	Leon Gottschick	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	Christoph Emig	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	Leon Gottschick	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	Kamil Kalmus	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	Leon Gottschick	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	Leon Gottschick	Ensures test data (database) environment and assets are managed and maintained. Responsibilities include: · support the administration of test data and test beds (database).
Designer	Christoph Emig	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	Kamil Kalmus	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

10.2 Staffing and Training Needs

Every developer takes time to get used to testing tools

11. Iteration Milestones

Milestone	Start Date	End Date
> 20% Test Coverage	06.04.2017	04.07.2017
JUnit tests	06.04.2017	04.07.2017

Spring Integration tests	06.04.2017	04.07.2017
UI Tests	04.07.2017	04.07.2017
80% test coverage	04.07.2017	-
Function Tests	06.04.2017	07.04.2017

12. Risks, Dependencies, Assumptions, and Constraints

Risk	Mitigation Strategy	Contingency (Risk is realized)

13. Management Process and Procedures

n/a

13.1 Measuring and Assessing the Extent of Testing

n/a

13.2 Assessing the Deliverables of this Test Plan

n/a

13.3 Problem Reporting, Escalation, and Issue Resolution

n/a

13.4 Managing Test Cycles

n/a

13.5 Traceability Strategies

n/a

13.6 Approval and Signoff

n/a