FaceRecognition

Test Plan

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 22.05.2016 | 1.0 | First work | Pascal Treptow |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 5

1.1 Purpose 5

1.2 Scope 5

1.3 Intended Audience 5

1.4 Document Terminology and Acronyms 5

1.5 References 5

1.6 Document Structure 5

2. Evaluation Mission and Test Motivation 6

2.1 Background 6

2.2 Evaluation Mission 6

2.3 Test Motivators 6

3. Target Test Items 6

4. Outline of Planned Tests 6

5. Test Approach 6

5.1 Initial Test-Idea Catalogs and Other Reference Sources 6

5.2 Testing Techniques and Types 6

5.2.1 Function Testing 6

5.2.2 User Interface Testing 7

5.2.3 Load Testing 7

5.2.4 Stress Testing 8

5.2.5 Installation Testing 10

6. Entry and Exit Criteria 11

7. Deliverables 11

7.1 Test Evaluation Summaries 11

7.2 Reporting on Test Coverage 11

7.3 Perceived Quality Reports 11

7.4 Incident Logs and Change Requests 11

7.5 Smoke Test Suite and Supporting Test Scripts 11

7.6 Additional Work Products 11

7.6.1 Detailed Test Results 11

7.6.2 Additional Automated Functional Test Scripts 11

7.6.3 Test Guidelines 11

7.6.4 Traceability Matrices 11

8. Testing Workflow 11

9. Environmental Needs 12

9.1 Base System Hardware 12

9.2 Base Software Elements in the Test Environment 12

XAMPP for Windows 5.6.14 12

9.3 Productivity and Support Tools 13

9.4 Test Environment Configurations 13

10. Responsibilities, Staffing, and Training Needs 13

10.1 People and Roles 13

10.2 Staffing and Training Needs 15

11. Iteration Milestones 15

12. Risks, Dependencies, Assumptions, and Constraints 15

13. Management Process and Procedures 16

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 FaceRecognition supports the following objectives:

* The tests are aligned with the interface and functionality.
* The motivation is to develop an application with the fewest number of bugs.

## Scope

This document addresses the following types and levels of testing:

* Unit Tests
* Functional Tests
* User Interface and Usability Tests
* Stress Test

## Intended Audience

n/a

## Document Terminology and Acronyms

tba

## References

|  |  |
| --- | --- |
| **Title** | **Date** |
| [Overall Use Case Diagram](https://raw.githubusercontent.com/sapacaFaceRecognition/Documentation/master/Use%20Cases/UCD.jpg) | 07.04.2016 |
| [Software Requirements Specification](https://github.com/sapacaFaceRecognition/Documentation/wiki/Software-Requirements-Specification) | 07.04.2016 |
| [Software Architecture Documentation](https://github.com/sapacaFaceRecognition/Documentation/wiki/Software-Architecture-Documentation) | 11.05.2016 |
| [Use Case: Browse Image](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case:-Browse-Image) | 04.05.2016 |
| [Use Case: Delete Image](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case:-Delete-Image) | 04.05.2016 |
| [Use Case: Detect Face](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case:-Detect-Face) | 04.05.2016 |
| [Use Case: Label Image](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case:-Label-Image) | 04.05.2016 |
| [Use Case: Upload Image](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case:-Upload-Image) | 04.05.2016 |
| [Use Case (S2): Bind to Google Search](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case-(S2):-Bind-to-Google-Search) | 04.05.2016 |
| [Use Case (S2): Eye Detection](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case-(S2):-Eye-Detection) | 04.05.2016 |
| [Use Case (S2): Gender Classification](Use%20Case%20(S2):%20Gender%20Classification) | 04.05.2016 |
| [Use Case (S2): Statistics](https://github.com/sapacaFaceRecognition/Documentation/wiki/Use-Case-(S2):-Statistics) | 04.05.2016 |

## Document Structure

n/a

# Evaluation Mission and Test Motivation

## Background

n/a

## Evaluation Mission

In general our mission is to improve our design and code quality.

This contains to find as many bugs as possible, find quality risks and so forth.

## Test Motivators

Tests reduce bugs in new features and in existing features. Also tests are good documentation and reduce the cost of work if something needs to be changed.

# 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.

* Operation System
* WebInterface
* And so forth

# Outline of Planned Tests

n/a

# Test Approach

## Initial Test-Idea Catalogs and Other Reference Sources

## Testing Techniques and Types

### Function Testing

|  |  |
| --- | --- |
| Technique Objective: | This tests should ensure that the functions of our project are working correctly |
| Technique: | The program starts functions and compares the results with an expected condition |
| Oracles: | If the result and the condition matches the test was successful |
| Required Tools: | [Junit](http://junit.org/junit4/) |
| Success Criteria: | If the result and the condition matches the test was successful |
| Special Considerations: | The face detection can´t be test, because the result picture is never exactly the same, there are always differences. |

### User Interface Testing

|  |  |
| --- | --- |
| Technique Objective: | Functionalities of the userintface got emulated |
| Technique: | User interaction, like clicks, got emulated. The tool compared therefore the condition before and after the emulation. |
| Oracles: | The test are successful, if the userinterface acts like the emulation predicted it. |
| Required Tools: | [Selenium](http://www.seleniumhq.org/) |
| Success Criteria: | All tests need to run through successfully |
| Special Considerations: | We could not test the whole userinterface, same functionalities require a special user input |

### Load Testing

[Load testing is a performance test that subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and abilities of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time-sensitive issues).]

[**Note**: Transactions in the following table refer to “logical business transactions”. These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.]

|  |  |
| --- | --- |
| 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

[Stress testing is a type of performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources, like database locks or network bandwidth, although some of these tests are usually addressed under functional and load testing.]

[**Note**: References to transactions in the following table refer to logical business transactions.]

|  |  |
| --- | --- |
| 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.] |

# Entry and Exit Criteria

# Deliverables

## Test Evaluation Summaries

n/a

## Reporting on Test Coverage

Our Test Coverage can be looked up at [Coverall.io](https://coveralls.io/github/sapacaFaceRecognition/FaceRecognition?branch=master)

## Perceived Quality Reports

n/a

## Incident Logs and Change Requests

We used [SonarQube](http://193.196.7.25/overview?id=sapaca) and [Travis-Ci](https://travis-ci.org/sapacaFaceRecognition/FaceRecognition) to trake and log our Incident Logs and Change Request

## Smoke Test Suite and Supporting Test Scripts

n/a

## Additional Work Products

n/a

### Detailed Test Results

n/a

### Additional Automated Functional Test Scripts

Our Repository can be found under [GitHub](https://github.com/sapacaFaceRecognition/FaceRecognition)

### Test Guidelines

n/a

### Traceability Matrices

n/a

# Testing Workflow

tba

# Environmental Needs

tba

## 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 |  |  |
| —Network or Subnet |  | TBD |
| —Server Name |  | TBD |
| —Database Name |  | TBD |
| Client Test PCs |  |  |
| —Include special configuration requirements |  | TBD |
| Test Repository |  |  |
| —Network or Subnet |  | TBD |
| —Server Name |  | TBD |
| Test Development PCs |  | 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** |
| --- | --- | --- |
| Windows | 10 | Operating System |
| XAMPP for Windows 5.6.14 | 5.6.14 | Virtuell Server and Database |
| Intellij | Community Editon 2016.1.1 | Editor |
| Spring Tool Suite | 3.6.4 | Editor |
| Internet Explorer |  | Internet Browser |
| Mozilla Firefox |  | Internet Browser |

## 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** |
| --- | --- | --- | --- |
| Test Coverage Monitor | Coverall.io | Lemur Heavy Industries |  |
| Code Climate, Metrics | Sonarqube | SonarSource S.A | 4.5.7 |
| Build Tool | Travis.ci | Travis CI GmbH |  |

## 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 | Number of users who are accessing the application at the same time | 5000 Useres |
| Minimal configuration supported | Speed and power of the internet connection provided by the server host | Not relevant, because it is running local |
| Test Computer | Specs of a test computer | Processor: AMD A10-8700P Radeon R6 1,8 GHz  RAM: 8GB  HDD: 1 TB  OS: Windows 10 |

# 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 | Sascha Kühne | 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 | Carolina Mehret | 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 | Sascha Kühne | 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 | Pascal Treptow | 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 | Sascha Kühne | 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 | Sascha Kühne | Ensures test data (database) environment and assets are managed and maintained.  Responsibilities include:   * support the administration of test data and test beds (database). |
| Designer | Pascal Treptow | 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 | Carolina Mehret | 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

n/a

# Risks, Dependencies, Assumptions, and Constraints

| **Risk** | **Mitigation Strategy** | **Contingency (Risk is realized)** |
| --- | --- | --- |
| Project crashes and can´t be started | Bevor every commit it must be sure the project is working | * Testing bevor committing |
| The port for the server is occupied | Make sure no program is running on the port the server is using | * Give the server a special port * Restart your comupter |
| Changes influence existing functionalities | Chances, especially functionally chances, have to be tested bevor | * Reverse a commit * Testing bevor committing |

# Management Process and Procedures

n/a