|  |  |
| --- | --- |
|  | «Audio Cataloger» Project |
| Test Plan |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Contents

1. [Project scope and main goals 3](#_bookmark0)
2. [Requirements to be tested 3](#_bookmark1)
3. [Requirements NOT to be tested 3](#_bookmark2)
4. [Test strategy and approach 3](#_bookmark3)
   1. [General approach 3](#_bookmark4)
   2. [Functional testing levels 3](#_bookmark5)
5. [Criteria 4](#_bookmark6)
6. [Resources 4](#_bookmark7)
7. [Schedule 4](#_bookmark8)
8. [Roles and responsibilities 4](#_bookmark9)
9. [Risk evaluation 4](#_bookmark10)
10. [Documentation 5](#_bookmark11)
11. [Metrics 5](#_bookmark12)

# Project scope and main goals

A tool for creating a catalog of audio files with the ability to detect duplicates and damaged files. It should be able to open in a browser and edit in spreadsheets. Also, it should not fail (for any reason) during its working process.

# Requirements to be tested

See referenced sections in “Audio\_Cataloger\_Recuiroments.pdf”:

* UR-1.\*: smoke test.
* UR-2.\*: critical path test.
* UR-3.\*: extended test.
* BR-1.: smoke test, critical path test.
* BR-2.: smoke test, critical path test.
* BR-3.: critical path test.
* BR-4.: critical path test.
* QA-1.\*: smoke test.
* QA-2.: smoke test.
* QA-3.: extended test.
* L-4: smoke test.
* DS-1.\*: extended test.
* DS-2.\*: extended test.
* DS-3.\*: extended test.
* DS-5.\*: extended test.

# Requirements NOT to be tested

See referenced sections in “Audio\_Cataloger\_Recuiroments.pdf”:

* L-1: the application is developed with proper java version.
* L-2: See DS-1 for JRE version and configuration details.
* L-3: JRE setup and configuration process are out of this project scope and therefore are NOT described in any product/project documentation.

# Test strategy and approach

## General approach

The application should be console-based, supporting smart comparison of algorithms and the following audio formats: mp3, flac, wav, ogg, wma. HTML (for browser) and CSV (for spreadsheet editor) output support. Deep testing for negative and critical situations should be implemented to ensure the application reliability. Therefore, issues of usability, security, etc. not explored during testing.

## Functional testing levels

* Smoke test: automated with batch files under Windows and Linux.
* Critical path test: executed manually.
* Extended test: executed manually.

Due to the cross-functional nature of the team, a significant contribution to quality improvement can be expected from a code review combined with manual testing using the white box method. Unit testing will be applied through the TDD software development technique

# Criteria

* Acceptance criteria: 100% success of test cases on smoke test level and 90% success of test cases on critical path test level (see “[Test cases success](#_bookmark13) percentage” metric) if 100% of critical and major bugs are fixed (see “[Overall defects](#_bookmark14) fixed percentage” metric). Final requirements coverage by tests (see “[Requirements](#_bookmark18) coverage by tests” metric) should be at least 80%.
* Testing start criteria: new build.
* Testing pause criteria: critical path test must begin only after 100% success of test- cases on the smoke test (see “[Test cases success percentage](#_bookmark13)”); test process may be paused is with at least 25% test-cases executed there is at least 50% failure rate (see “[Stop-factor](#_bookmark16)” metric).
* Testing resumption criteria: more than 50% of bugs found during the previous iteration are fixed (see “[Ongoing defects fixed percentage](#_bookmark15)” metric).
* Testing finish criteria: more than 80% planned for the current iteration test cases are executed (see “[Test-cases execution percentage](#_bookmark17)”).

# Resources

* Software: four virtual machines (four with Windows 10 Ent x64, four with Linux Ubuntu 20.04 LTS x64), four Intellij IDE licenses (latest version available).
* Hardware: four standard workstations (8GB RAM, i7 3GHz).
* Personnel:
  + Two senior developer with testing experience (100% workload during all project time). Roles: team lead, senior developer.
  + Two tester with java knowledge (100% workload during all project time). Role: tester.
* Time: one workweek (40 work hours).
* Finances: according to the approved budget.

# Schedule

* 25.06-27.06 – requirements testing and finalizing.
* 28.06-02.07 – test-cases and scripts for automated testing creation.
* 03.07-09.07 – main testing stage (test-cases execution, defect reports creation).
* 10.07 – testing finalization, reporting.

# Roles and responsibilities

* Senior developers: participation in requirements testing and code review.
* Testers: documentation creation, test-cases execution, participation in code-review.

# Risk evaluation

* Personnel (low probability): if any team member is inaccessible, we can contact the representatives of the “Cataloger” project to get a temporary replacement (the commitment from the “Pocemon” PM John Smith was received).
* Time (high probability): the customer has indicated a deadline of 10.07, therefore time is a critical resource. It is recommended to do our best to complete the project by 09.07 so that one day (10.07) remains available for any unexpected issues.
* Complexity of accurate parsing of some audio formats.
* Complexity (or impossibility) to detect encoding for non-English tags in files.

# Documentation

* Requirements. Responsible person – tester Ignatiev Ihor, deadline – 27.06.
* Test cases and defect reports. Responsible – tester Kramar Ignat, creation period – 28.06-02.07.
* Test result report. Responsible person – tester 28.06-02.07, deadline – 10.07.

# Metrics

* Test cases success percentage:

?𝑆𝑃 = 𝑇𝑆𝑢𝑐𝑐𝑒𝑠𝑠 ∙ 100%, where

𝑇𝑜𝑡𝑎𝑙

𝑇

𝑇𝑆𝑃 – percentage of successfully passed test cases,

𝑇𝑆𝑢𝑐𝑐𝑒𝑠𝑠 – quantity of successfully passed test cases,

𝑇𝑇𝑜𝑡𝑎𝑙 – total quantity of executed test cases. Minimally acceptable borders:

* Beginning project phase: 10%.
* Main project phase: 40%.
* Final project phase: 80%.
* Overall defects fixed percentage:

?𝐹𝑇𝑃 = 𝐷 𝐶𝑙𝑜𝑠𝑒𝑑 ∙ 100%, where

𝐿𝑒𝑣𝑒𝑙

?𝑒𝑣𝑒𝑙

𝐷

𝐹𝑜𝑢𝑛𝑑

𝐿𝑒𝑣𝑒𝑙

?𝐹𝑇𝑃 – overall defects fixation percentage by 𝐿𝑒𝑣𝑒𝑙 during all project lifetime,

𝐿𝑒𝑣𝑒𝑙

?𝐶𝑙𝑜𝑠𝑒𝑑 – quantity of defects of 𝐿𝑒𝑣𝑒𝑙 fixed during all project lifetime,

𝐿𝑒𝑣𝑒𝑙

?𝐹𝑜𝑢𝑛𝑑 – quantity of defects of 𝐿𝑒𝑣𝑒𝑙 found during all project lifetime.

𝐿𝑒𝑣𝑒𝑙

Minimally acceptable borders:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Defect severity | | | |
| Minor | Medium | Major | Critical |
| Project phase | Beginning | 10% | 40% | 50% | 80% |
| Main | 15% | 50% | 75% | 90% |
| Final | 20% | 60% | 100% | 100% |

* Ongoing defects fixed percentage:

?𝐹𝐶𝑃 = 𝐷 𝐶𝑙𝑜𝑠𝑒𝑑 ∙ 100%, where

𝐿𝑒𝑣𝑒𝑙

?𝑒𝑣𝑒𝑙

𝐷

𝐹𝑜𝑢𝑛𝑑

𝐿𝑒𝑣𝑒𝑙

?𝐹𝐶𝑃 – defects fixation percentage by 𝐿𝑒𝑣𝑒𝑙 (defects found in the previous build and fixed in the current build),

𝐿𝑒𝑣𝑒𝑙

?𝐶𝑙𝑜𝑠𝑒𝑑 – quantity of defects of 𝐿𝑒𝑣𝑒𝑙 fixed in the current build,

𝐿𝑒𝑣𝑒𝑙

?𝐹𝑜𝑢𝑛𝑑 – quantity of defects of 𝐿𝑒𝑣𝑒𝑙 found in the previous build.

𝐿𝑒𝑣𝑒𝑙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Defect severity | | | |
| Minor | Medium | Major | Critical |
| Project phase | Beginning | 60% | 60% | 60% | 60% |
| Main | 65% | 70% | 85% | 90% |
| Final | 70% | 80% | 95% | 100% |

* Stop-factor:

𝑆 = {𝑌𝑒𝑠, 𝑇𝐸 ≥ 25% && 𝑇𝑆𝑃 < 50% , where

𝑁𝑜, 𝑇𝐸 < 25% || 𝑇𝑆𝑃 ≥ 50%

𝑆 – decision to pause the testing process,

𝑇𝐸 – current 𝑇𝐸 value,

𝑇𝑆𝑃 – current 𝑇𝑆𝑃 value.

* Test-cases execution percentage:

? 𝑇𝐸𝑥𝑒𝑐𝑢𝑡𝑒𝑑

𝑇 = ∙ 100%, where

𝑇𝑃𝑙𝑎𝑛𝑛𝑒𝑑

𝑇𝐸 – test-cases execution percentage,

𝑇𝐸𝑥𝑒𝑐𝑢𝑡𝑒𝑑 – quantity of executed test-cases,

𝑇𝑃𝑙𝑎𝑛𝑛𝑒𝑑 – quantity of planned (to execution) test-cases. Levels (borders):

o Minimal: 80%.

o Desired: 95%-100%.

* Requirements coverage by tests:

?𝐶 = 𝑅𝐶𝑜𝑣𝑒𝑟𝑒𝑑 ∙ 100%, where

𝑇𝑜𝑡𝑎𝑙

𝑅

𝑅𝐶 – requirements coverage by tests (percentage),

𝑅𝐶𝑜𝑣𝑒𝑟𝑒𝑑 – quantity of requirements covered with test-cases,

𝑅𝑇𝑜𝑡𝑎𝑙 – overall quantity of requirements. Minimally acceptable borders:

* Beginning project phase: 40%.
* Main project phase: 60%.
* Final project phase: 80% (90%+ recommended).