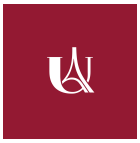
Client Acceptance Report

FFTrack

Team L2B1

Members : Jennifer Zahora, Nicolas Schuldt, Nicolas Fontaine, Ismaël Bouarfa

Supervisors : Hugo Demaret, Ioana Ileana



Université Paris Cité
UFR de Mathématiques et Informatique

 2023-2024

Summary

1 Introduction

This report provides an overview of the validation process that will be conducted on the app, FFTrack. It outlines the methodology and objectives for the future technical validation tests, and presents the tests in detail. FFTrack enables users to identify songs through audio snippets captured by their devices. The primary aim of the planned validation process is to assess the application's performance and reliability across various scenarios and environments.

1.1 Overview

The app utilises the combination of audio processing and matching algorithms and a music database to identify songs. The core features include:

- audio recording and processing,
- real-time audio recognition,
- song metadata retrieval,
- as well as saving new audio and metadata.

1.2 Goals and Objectives

In order to establish the framework for the validation process, we will define the goals and objectives to be achieved.

1.2.1 Goals

These goals will guide the overall direction and priorities of the validation process.

- **Ensuring functionality:** Validate that the application functions as intended, it meets the specified requirements, and delivers the expected functionalities.
- **Assessing performance:** Evaluate the performance of the application under different scenarios and conditions.

1.2.2 Objectives

Our main objective is to validate the correctness of the application's functionality through different testing methods. We will also lightly assess the application's performance, more specifically its response time and matching quality. Moreover, we will evaluate the usability of the application.

Through these tests, we can provide specific, measurable outcomes that help us achieve our validation goals.

2 Testing Methodology and Strategy

In the FFTrack project, ensuring that our application meets high standards for reliability, efficiency, and user satisfaction is one of our highest priorities. This section outlines our testing methodology and strategy, designed to validate every aspect of the application.

2.1 Testing Methods

Our testing approach combines manual testing, automated testing, and user testing to cover all parts of the application:

- **Manual Testing** : This involves hands-on testing of application features to identify any unexpected behaviour or bugs.
  - We will involve real users to test the application in real-world scenarios to provide feedback on its functionality and user experience.
- **Automated Testing** : Utilises tools and scripts to automatically run tests on the codebase, ensuring that new changes don't break existing functionality, and that every component works as expected, individually and as a whole.

Unit Testing

Unit testing focuses on verifying the smallest parts of the application, such as individual functions or methods. For example, we'll test audio processing functions to ensure they correctly analyse and process input audio. We use pytest, a powerful testing framework that allows us to write and run tests efficiently. Our goal is for 70-80% test coverage, meaning we want to ensure that at least 70% of our code is tested, which we track using the Python coverage tool.

In section 5 of this report, we will provide a detailed list of the different functions/features we aim to test, along with their dependencies and expected results.

Writing Unit Tests

- **Test Isolation**: Each test should cover a single "unit" of code, such as a function or method, without relying on external dependencies or the behaviour of other parts of the application. This isolation helps in identifying exactly where a problem lies when a test fails.
- **Test Cases**: For each function, we will write multiple test cases to cover various scenarios, including typical use cases, edge cases, and error conditions. For instance, when testing an audio processing function, we'll have test cases for different audio formats, and empty audio files.
- **Assertions**: Each test case will include assertions that check the function's output against expected results. Assertions are the actual test conditions and can include checks for returned values, changes to the system's state, or whether the right exceptions are raised.

Integration Testing

Integration testing examines how different parts of the application work together. Here, we'll test sequences such as the end-to-end process from audio input through fingerprint generation to song identification in the database. This ensures that components interact correctly and data flows seamlessly through the application.

Functional Testing

Functional testing assesses the application from the user's perspective, ensuring that all features work as intended. We will simulate user interactions with the application, such as uploading an audio file, to validate the entire process of identifying a song and displaying results to the user.

Perfomance Testing

Performance testing measures the application's responsiveness and stability under various conditions. We'll test for:

- **Load Times**: How quickly the application starts and responds to user input.
- **Response Times**: The speed at which the application processes audio inputs and returns results.
- **Accuracy of Song Identification**: The precision and reliability of the matching algorithm in identifying songs correctly.

2.2 Testing Environments

We use two main environments for testing:

Development Environment:Used during the development phase for initial testing. It is set up on each developer's computer, ensuring that the application works correctly in a development setting.

Staging Environment: A controlled environment that mirrors production. Before release, we deploy the application here for final testing. This environment is also where we will test the application packaged as a pip package to ensure easy installation and deployment.

2.3 Testing Tools and Frameworks

Our testing and QA tools include:

- **pytest** for writing and running our automated tests.
- **coverage** tool to measure the percentage of our code tested by unit tests, helping us maintain high test coverage.
- **flake8** for linting, ensuring our codebase adheres to PEP 8 standards and remains clean and readable.
- **venv** for managing virtual environments, ensuring that our development and testing environments are consistent and isolated from system-wide Python packages.

Commit Checklist

To ensure the robustness of our application, before committing, we must go through the following steps:

1. Run Linters: Use flake8 to check for stylistic errors and potential bugs.
2. Pass All Tests: Ensure all unit and integration tests pass with pytest. Check that the coverage level meets the standards.
3. Update Documentation: If new features are added or existing ones are changed, update the documentation accordingly.
4. Review Changes: Self-review your code changes to catch any obvious issues.

These practices, along with the test, will ensure a robust, well tested application.

3 Functional Verification

3.1 Unit Tests

To run unit tests, use the python `coverage` tool, it will run the tests, and return the current code coverage percentage. Run `coverage run -m pytest` in the terminal.

A working computer with a compatible OS is required for running the tests.

3.1.1 Module: Audio Processing

Constraints: Tests should not rely on external audio sources. Use mock audio data or synthetic tones for consistency.

ID	Purpose	Dependencies	Expected Results	Notes
AP1	Test audio file loading	None	Audio data is correctly loaded from file	Test with various file formats (WAV, MP3)
AP2	Test audio preprocessing	AP1	Audio volume is standardized and turned to mono WAV	
AP3	Test FFT conversion	AP1	Correct FFT output for given audio data	Use known audio for predictable FFT result
AP4	Test fingerprint generation	AP2, AP3	Unique fingerprint generated for input audio	Ensure fingerprints are consistent for identical inputs

Table 1: Audio Processing Unit Tests

3.1.2 Module: Database

Constraints: Tests must interact with a test database instance to avoid altering production data.

ID	Purpose	Dependencies	Expected Results	Notes
DB1	Test song record creation	None	Song record is added to the database	Include edge cases, like missing fields
DB2	Test song record retrieval	DB1	Correct song record is retrieved	Test with valid and invalid IDs
DB3	Test song record update	DB1	Song record is updated with new data	Ensure changes persist in the database
DB4	Test song record deletion	DB1	Song record is removed from the database	Verify record no longer exists post-deletion
DB5	Test fingerprint record association	DB1	Fingerprint is correctly associated with a song	Use mock fingerprint data

Table 2: Database Unit Tests

3.1.3 Module: Matching

Constraints: Matching tests should be designed to work with a limited set of known fingerprints for predictability.

ID	Purpose	Dependencies	Expected Results	Notes
MT1	Test song match retrieval	MT1, DB5	Correct song match found for query fingerprint	Include tests for no match found
MT2	Test similarity scoring	MT1, DB5	Similarity score calculated for potential matches	Scores accurately reflect match confidence
MT3	Test handling of different song versions	MT1, DB5	Variations of the same song are recognized as matches	Test with live, acoustic, and studio versions, and live recordings in noisy environments

Table 3: Matching Unit Tests

3.1.4 Module: UI

Constraints: UI tests should simulate real user interactions as closely as possible, and might require a separate script that interacts with the terminal.

ID	Purpose	Dependencies	Expected Results	Notes
UI1	Test audio file upload	AP1	Audio file is successfully uploaded through UI	Test with various file sizes and formats
UI2	Test song identification request	UI1, MT2	Correct song identification displayed to user	Include a timeout for no response
UI3	Test error handling for unsupported formats	UI1	User is notified of unsupported file format	Test with an unsupported file type

Table 4: UI Unit Tests

3.2 Integration Testing

Integration testing ensures that different components of the application work together seamlessly.

ID	Purpose	Components Involved	Expected Results	Notes
IT1	Test end-to-end song identification	Audio Processing, Database, Matching, UI	Correct song identified and displayed to the user	Test with various song versions and qualities
IT2	Test audio processing to database storage	Audio Processing, Database	Fingerprints generated and stored	Verify accessibility for matching
IT3	Test database retrieval for matching	Database, Matching	Algorithm retrieves fingerprints for comparison	Ensure correct match identification
IT4	Test UI interaction with audio processing	UI, Audio Processing	Processed audio files for fingerprint generation	Test upload errors and format issues
IT5	Test database update impacts on matching accuracy	Database, Matching	Database updates reflect in matching	Test adding new songs and modifying entries

Table 5: Integration Tests

3.3 Functional Testing

Functional testing verifies that each feature of the application operates according to the requirement specifications.

ID	Purpose	Test Steps	Expected Results	Notes
FT1	Verify known audio identification	Upload and submit a known audio file for identification	Application identifies and displays song information	Test with varied audio lengths and qualities
FT2	Verify unknown audio identification	Upload and submit an unknown audio file for identification	Application communicates no song found	Ensure user-friendly error message
FT3	Add song to database	Upload audio, add song info, submit to database	Song and fingerprint added to the database	Handle duplicates in the database

Table 6: Functional Tests

3.4 Performance Testing

Performance testing evaluates the application's speed and responsiveness under various conditions.

ID	Purpose	Test Scenario	Expected Results	Notes
PT1	Evaluate response time	Measure time from audio upload to result display	Response within predefined threshold	Test with various file sizes and formats
PT2	Test database query efficiency	Measure time for fingerprint matching queries	Queries complete within set time frame	Optimize database indices and queries
PT3	Assess audio processing performance	Evaluate speed for audio normalization and fingerprint generation	Tasks complete within acceptable time frames	Optimize for speed
PT4	Evaluate noise tolerance	Identify song with increasing noise levels	Song identifiable within noise threshold	Define noise threshold values later

Table 7: Performance Tests

4 General Validation

The technical validation is a thorough assessment aimed at ensuring the application's quality, functionality, performance, and user experience criteria. This section outlines the main acceptance criteria under which the application will be accepted or rejected. We will also describe the process by which we will ensure that the application complies with the criteria.

4.1 Validation aspects

The key areas that need to be evaluated during the validation process are the following:

Functionality: The core features of the application need to be thoroughly tested to ensure they function as intended (this includes audio recording and processing, audio recognition, metadata retrieval, and saving new audio and metadata).

Performance: The performance of the application needs to be evaluated to ensure optimal performance, this includes response time for song identification.

Compatibility: The application's compatibility should be tested with different devices and operating systems to ensure a consistent experience.

User experience: The usability and intuitiveness of the features of the application should be assessed through user testing and feedback to enhance overall user experience.

Compliance: The application's compliance with relevant regulations, such as copyright claims, needs to be verified to avoid legal risks.

4.2 Acceptance criteria

The application must meet these criteria to be considered acceptable and ready for submission. These criteria are derived from the requirements and expectations of the project owners.

4.2.1 Accuracy and reliability

Purpose
Validate the accuracy and consistency of the music identification algorithm to ensure precise song recognition.

Criteria
The application must accurately identify songs based on audio samples, with a reasonable recognition accuracy of at least 80-90%. The identified song must match the actual song title, artist, etc. retrieved from the music database.

Process of ensuring compliance
Features and functionalities will be implemented according to criteria. Thorough testing will be conducted.

- We will assess the robustness of the algorithms against noise interference, varying audio qualities, and different recording environments. This will be done by analysing and comparing the fingerprint of samples of differing qualities to the fingerprint of a HQ audio.
- By optimising the matching algorithms, we will minimise false positives and false negatives in song identification.

4.2.2 Performance

Purpose
Evaluate the application's performance.

Criteria
If the song exists in the database, a match should occur before the sample song finished playing, using a sample of at least 10-15 seconds.

Process of ensuring compliance
Thorough testing will be conducted, and the matching algorithm will be optimised to comply with the criteria.

4.2.3 Compatibility

Purpose
Verify that the application is compatible with different computer operating systems.

Criteria
The application must be compatible with Windows, Linux, and macOS.

Process of ensuring compliance
The application will be tested on computer devices running different OS.

4.2.4 User experience

Purpose
Ensure that the application provides a seamless and intuitive user experience.

Criteria
The user interface should be easy to navigate, and it should provide clear feedback to the users during the song identification process, through informative messages.

Process of ensuring compliance
The interface will be implemented according to the criteria.

- We will conduct a few user tests to get feedback on the usability of the application.
- We will identify pain points and prioritize improving them.
- We will validate that key features as audio recognition, and metadata display and input are easy to use.

5 Submission

This section will detail the submission requirements for our project, outlining the chosen delivery methods and the nature of documents to be included.

5.1 Delivery Method

To conform to the project requirements, for the submission of our project we will:

- use Subversion (SVN) for version control
- deliver documents digitally through the website, Forge
- hand-in physical copies of the project report.

Moreover, we will upload the source code and documentation to PyPI.

5.1.1 Version Control

We will upload all code and related documentation through SVN. As already mentioned in the Specifications report, the repository is structured into three main directories.

5.1.2 Forge

Most of the documents have to be uploaded on Forge, under the **Documents** section. Exceptions are the following:

- Weekly reviews, that have to be uploaded under **Forums > Comptes-rendus**
- A wiki page of our project, under **Wiki**
- The final report which will be submitted to the secretariat in the form of two physical copies

5.2 Project Documentation Delivery

In addition to the application itself, we will provide thorough documentation to support the understanding and usage of our project.

5.2.1 Technical documentation

The format of the following documents is PDF.

Specifications report: Outlines the requirements and specifications of the application in development. Defines what the system is expected to do, its functionalities, constraints, and deadlines of the project.

Client Acceptance report: Documents the validation process that will be conducted to assess whether the application meets its specified requirements and performs as expected. Includes details about the testing methodologies, the tests to be conducted, their expected results, and the validation criteria.

Detailed conception report: Provides a detailed description of the system's architecture, components, interfaces, and internal workings. Aides the implementation of the application.

5.2.2 Technical manuals

The technical manuals will be handed in as .md files, included with the code. These include a development and contribution guide, DEVELOPMENT.md and CONTRIBUTING.md, respectively.

User manual: Offers guidance on how to use the application, including navigation, and features. Aimed at the end-users. (README.md)

Installation manual: Provides instructions for installing the application, including system requirements, and installation steps. (README.md)

5.2.3 Code documentation and other

Source code with its internal documentation: The source code will be submitted in a folder, as .py files, mostly.

Executable: Along with the source code we will provide the user an OS specific script to launch the application. This will install and initialise all the program requirements, then run the application itself thus facilitating user experience.

Pip Installation: The program will also be available on pip, running `pip install fftrack` will install the source code on the computer. Then, simply running `fftrack` on the terminal will execute the application.

Test plan: Outlines the approach, scope, resources, and schedule for testing. Provides guidance on how to verify and validate the functionality, performance, and quality of the application. The test plan is a .pdf document.

Report: Contains all the information that has not been included in the other documents. Will be handed in as a .pdf file.

Wiki: A summary of the project in French and in English, with portraits of the group members included.

Presentation with audio: Presents the key aspects, progress, and outcomes of the project. Two version will be handed in, a .ppt or .pdf presentation file, and a video file of the presentation with audio.

6 Glossary

Terms and definitions used previously:

Audio Fingerprinting: Audio fingerprinting is a technique used to uniquely identify pieces of audio content. It involves analyzing the spectral content of an audio signal and extracting distinctive features that can form a ”fingerprint.” This fingerprint can resist variations such as noise, compression, and slight changes in speed or pitch.

Fast Fourier Transform (FFT): FFT is a mathematical algorithm that transforms a signal from its original time domain into a frequency domain. In the context of audio processing, FFT is used to analyze the frequencies present in an audio clip, which are necessary for generating audio fingerprints.

Version Control: Version control is a system that records changes to a group of files over time so that versions can be tracked. This project uses SVN (Subversion) for version control, it facilitates collaboration among developers by tracking and merging changes in the code.

Test Coverage: A measure used to describe the degree to which the source code of a program is executed when a particular test suite runs. A program with high test coverage has more of its source code tested by automated tests.

UI (User Interface): The point of human-computer interaction and communication in a device, application, or website, involving the presentation, appearance, and usability.