Softwaredesign - Abschlussprojekt

Bachelor Studiengang - Mechatronik, Design & Innovation

3. Semester

Lektor: Matthias Panny

Gruppe: BA-MECH-22-2A

Autor: Maximilian Nachbaur, Markus Brachmann

1. März 2024

Inhaltsverzeichnis

1 Einleitung

Ziel des Abschlussprojektes war es eine Anwendung mit Web-UI zu entwickeln, mit der Musikstücke identifiziert werden können, indem nur ein kurzer Abschnitt des Musikstückes untersucht wird → wie Shazam, Soundhound, etc. Die Umsetzung erfolgte in Python mit Objektorientierung und das Web-UI war mit streamlit umzusetzen.

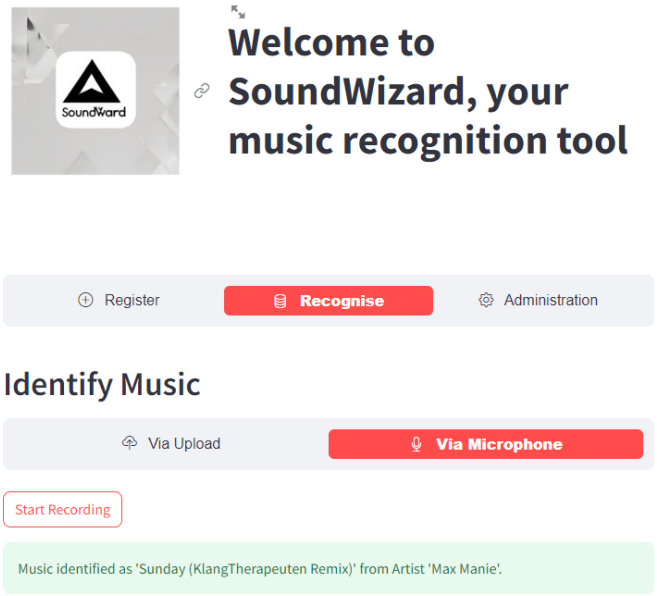

Abbildung 1.1: Screenshot der Streamlit-Oberfläche

2 Aufgabenstellung

Für das Abschlussprojekt mussten folgende Mindestaanforderungen erfüllt werden:

- Softwareprojekt in Python mit Objektorientierung
- Versionskontrolle über git & GitHub
- Anwendung mit Web-UI:
  - Register workflow → Musikstücke einlernen
  - Recognise workflow → Musikstücke identifizieren
  - Hochgeladenes Musikstück muss in Browser wiedergegeben werden können
- Dokumentation

3 Programmaufbau

Das Programm wurde in mehrere Bausteine aufgeteilt.

- Datenbank
- Register workflow
- Recognise workflow
- Streamlit Interface

3.1 DATENBANK

Zu Beginn wurde mit einer TinyDB Datenbank gearbeitet, da sich aber herausstellte das die Hashabfrage mit dieser viel zu lange dauerte, wurde als alternative Datenbank SQlite implementiert.

Die Datenbank an sich ist in 2 Tables aufgeteilt, "songs"mit den Informationen über den Song (Title, Artist, Filepath) und "hashes" in dem die einzelnen hashes und die Song-ID gespeichert wird.

Um mit der Datenbank zu interagieren und um jegliche Daten zu speichern und zu verarbeiten wurde die gleiche Struktur wie in der Case-Study Hausübung verwendet. Die einzelnen Funktionen und Module wurden entsprechend an das Abschlussprojekt und an die neue Datenbank angepasst.

3.2 REGISTER WORKFLOW

Die Datei Register.py bildet das Rückgrat unserer Anwendung und ist verantwortlich für die Verarbeitung von Audiodateien, die Aufnahme von Audio, sowie die Verwaltung und Interaktion mit der Datenbank. Diese Datei ist in verschiedene Module unterteilt, die jeweils spezifische Funktionen und Verantwortlichkeiten haben.

Das Modul zur Verarbeitung von Audiodateien umfasst die Funktionen **create_Fingerprints** und **create_hashes**. Diese Funktionen werden verwendet, um aus einer Audiodatei Fingerabdrücke zu extrahieren und Hashes zu generieren. Dabei wird eine Constellation Map erstellt, um die Peaks im Spektrum der Audiodaten zu identifizieren. Anschließend werden diese Informationen in Hashes umgewandelt und zum späteren Vergleich gespeichert. Anhaltspunkt und Hilfe bei der Implementierung dieser Funktionen waren die beiden Blogs "abracadabra: How does Shazam work?"von Cameron MacLeod [1] und "How Shazam Works - An explanation in Python"von Michael Strauss [2].

Die Funktion **process_uploaded_song** ist für die Verarbeitung von hochgeladenen Songs zuständig. Sie lädt eine hochgeladene Audiodatei, erstellt deren Fingerabdrücke und Hashes und speichert diese in der Datenbank. Dabei werden auch Informationen wie Künstlername und Titel des Songs berücksichtigt, um die Datenbank entsprechend zu aktualisieren.

Die Funktion **record_audio** ermöglichen die Aufnahme von Audioinhalten, mit spezifischen Einstellungen wie Sample-Rate und Kanalanzahl. Die aufgenommenen Audiodaten können dann weiterverarbeitet und später mit den vorhandenen hashes in der Datenbank verglichen werden.

3.3 RECOGNISE WORKFLOW

Die Datei Recognise.py ist für die Erkennung von Audiodateien und die Suche nach Übereinstimmungen in der Datenbank verantwortlich. Sie beinhaltet mehrere Funktionen, die nacheinander ausgeführt werden, um diese Aufgabe zu erfüllen. Im Folgenden wird der Programmablauf beschrieben:

Die Funktion **recognize_song** nimmt eine Audiodatei als Eingabe, generiert deren Fingerabdrücke und Hashes, und sucht dann nach der besten Übereinstimmung in der Datenbank mithilfe der Funktion **find_best_match**. Sie gibt das Ergebnis der Erkennung zurück.

Die Funktion **find_best_match** ist für die Suche nach der besten Übereinstimmung in der Datenbank anhand der berechneten Hashes verantwortlich. Sie ruft die Funktion **score_hashes** auf, um die Übereinstimmungen zu bewerten, und gibt dann den besten Treffer zurück.

Die Funktion **score_hashes** bewertet die Übereinstimmungen zwischen den berechneten Hashes und den gespeicherten Hashes in der Datenbank. Sie verwendet eine Punktzahl, um die Qualität der Übereinstimmung zu bewerten, und gibt eine geordnete Liste der Übereinstimmungen zurück.

Die Funktion **record_and_recognize** ermöglicht es, Audio aufzunehmen, es dann zu verarbeiten und nach Übereinstimmungen in der Datenbank zu suchen. Sie verwendet die Funktionen **record_audio**, **create_Fingerprints**, **create_hashes** und **find_best_match** , um diesen Prozess durchzuführen.

3.4 STREAMLIT INTERFACE

Die interface.py-Datei ist Teil des Gesamtprojekts und enthält das Skript für die Streamlit-Benutzeroberfläche, die es Benutzern ermöglicht, verschiedene Funktionen der Anwendung auszuführen.

Die Funktion **learn_workflow** ermöglicht es Benutzern, Musikstücke hochzuladen, Metadaten wie Künstler und Titel einzugeben und diese in die Datenbank einzufügen. Nach dem Hochladen und Hinzufügen eines Musikstücks kann es auch verarbeitet werden, um Fingerabdrücke zu erstellen und in der Datenbank zu speichern.

Die Funktion **recognize_workflow** bietet Benutzern die Möglichkeit, entweder eine Audiodatei hochzuladen oder über ein Mikrofon aufzunehmen, um Musikstücke zu identifizieren. Die hochgeladene Datei wird verarbeitet und mit den in der Datenbank gespeicherten Musikstücken verglichen, um eine Übereinstimmung zu finden.

Die Funktion **Administration** ermöglicht es Benutzern, die Datenbank zu verwalten, indem sie vorhandene Musikstücke löschen. Benutzer können ein Musikstück auswählen und es aus der Datenbank entfernen.

3.5 ERWEITERUNGEN

YouTube-Video-Link suchen:
Die Funktion **link_youtube** ermöglicht die Suche nach dem offiziellen Musikvideo des erkannten Songs. Sie verwendet die Parameter "Titelünd Ärtistüm diese in Kombination in einer YouTube Suchanfrage aufzurufen. Falls die Anfrage erfolgreich ist, wird die HTML-Antwort nach der Video-ID durchsucht und der Link zum gefundenen Video wird zurückgegeben. Albumcover suchen:

Die Funktion **get_album_cover** ermöglicht die Suche nach einem Albumcover eines Songs unter Verwendung von DuckDuckGo. Die Funktion verwendet die Parameter "Titelünd Ärtistüm diese in einer Suchanfrage zu kombinieren und mit der **duckduckgo_search**-Bibliothek das passende Albumcover zu finden. Das gefundene Bild wird dann als URL zurückgegeben.

3.6 INSTALLATIONSANLEITUNG

Voraussetzungen

- Python 3

- Git

- Git Large File Storage (LFS)

- SQLite Viewer Extension

Installation

1. Installieren Sie Git LFS auf Ihrem System. Die Installationsanweisungen finden Sie auf der offiziellen Git LFS Webseite(https://git-lfs.github.com/)
2. Initialisieren Sie Git LFS in GitBash: "git lfs install"
3. Klonen Sie das Repository: "git clone https://github.com/xbakerman/Abschlussprojekt-Softwaredesign.git"
4. Wechseln Sie in das Verzeichnis des geklonten Repositories: "cd Abschlussprojekt-Softwaredesign"
5. Erstellen Sie eine virtuelle Umgebung und aktivieren Sie sie: "python3 -m venv venv" "source venv/bin/activate" On Windows, use "venv\Scripts\activate"
6. Installieren Sie die benötigten Python-Pakete: "pip install -r requirements.txt"
7. Um die Datenbank zu öffnen kann eine Extension wie "SQLite Viewer"verwendet werden

Abbildungsverzeichnis

Literaturverzeichnis

[1] C. MacLeod, "abracadabra: How does shazam work?" https://www.cameronmacleod.com/blog/how-does-shazam-work, 2022.

[2] M. Strauss, "How shazam works - an explanation in python," https://michaelstrauss.dev/shazam-in-python, 2021.