Proiect PIU
Raport Intermediar

Proiect: Editor Audio Digital
Student: Jireadă Teodor

1. Obiectivele Proiectului

Obiectivul principal al acestui proiect este dezvoltarea unei aplicații desktop de tip Digital Audio Workstation (DAW) simplificat, care să permită utilizatorilor să editeze și să mixeze fișiere audio.

Obiectivele specifice pentru această etapă includ:

- **Interfață Grafică Intuitivă:** Crearea unei interfețe moderne și responsive folosind biblioteca Qt (PySide6), care să ofere acces rapid la funcțiile de bază.
- **Motor Audio Performant:** Implementarea unui engine audio capabil să redea simultan mai multe track-uri audio, cu latență scăzută.
- **Manipulare Track-uri:** Posibilitatea de a importa, șterge și gestiona (Mute/Solo) track-ui audio individuale.
- **Vizualizare:** Reprezentarea grafică a formelor de undă (waveform) pentru fiecare fișier audio încărcat.
- **Navigare:** Implementarea unui timeline interactiv cu funcționalități de zoom și scroll sincronizat.

2. Prezentarea Aplicației Parțiale

În stadiul curent, aplicația este funcțională și îndeplinește cerințele fundamentale ale unui editor audio. Interfața este împărțită în trei zone principale:

1. **Ribbon (Zona Superioară):** Conține butoanele globale de control al redării (Play, Stop) și alte instrumente esențiale.
2. **Panoul de Control (Stânga):** Afișează lista de track-uri, fiecare având controale dedicate (Nume fișier, butoane Mute/Solo/Delete). Tot aici se află butonul principal de "Add Track" notat cu semnul ‘+’.
3. **Timeline și Workspace (Dreapta):**
    - **Timeline Ruler:** O riglă gradată care indică timpul și permite navigarea rapidă (seek) prin click.
    - **Track Lanes:** Zona unde sunt afișate vizual formele de undă ale fișierelor audio, sincronizate cu timeline-ul.

Funcționalități Implementate:

- **Import Audio:** Suport pentru formatele comune (WAV, MP3, OGG, FLAC, OPUS) prin intermediul bibliotecii “soundfile”. Importul se realizează pe un thread separat pentru a nu bloca interfața.

- **Playback & Mixing:** Redarea simultană a tuturor track-urilor active. Engine-ul audio mixează datele în timp real.
- **Mute & Solo:** Fiecare track poate fi închis (Mute) sau izolat (Solo) în timpul redării.
- **Zoom Dinamic:** Utilizatorul poate face zoom in/out pe timeline pentru a vedea detalii fine sau o privire de ansamblu.
- **Sincronizare UI:** Scroll-ul vertical și orizontal este sincronizat între lista de track-uri și zona de vizualizare a formelor de undă.

3. Implementare și Utilizare

Arhitectura Aplicației

Proiectul este structurat modular pentru a separa logica de interfață:

- `**main.py**`: Punctul de intrare în aplicație. Inițializează `QApplication` și fereastra principală.
- `**core/**`:
    - `audio_engine.py`: Gestionează stream-ul audio către placa de sunet folosind `sounddevice`. Se ocupă de mixarea sample-urilor și menținerea poziției de redare (playhead).
    - `track_loader.py`: Clasă `QThread` care încarcă și procesează fișierele audio în background, calculând inclusiv forma de undă pentru vizualizare.
    - `models.py`: Definește structurile de date, cum ar fi `AudioTrackData`.
- `**ui/**`:
    - `main_window.py`: Clasa principală care asamblează interfața și leagă evenimentele UI de logica din `core`.
    - `widgets/`: Componente UI personalizate (`TimelineRuler`, `TrackLane`, `TrackHeader`, `TrackContainer`).

Utilizare:

1. **Pornire:** Se rulează scriptul `main.py`.
2. **Adăugare Track:** Se apasă butonul mare "+" din zona din stânga. Se deschide un dialog de fișiere pentru a selecta un fișier audio.
3. **Redare:** Se apasă "Play" sau tasta `Space`.
4. **Navigare:** Click pe rigla de sus (Timeline) pentru a sări la un anumit moment. Scroll pentru a naviga în listă.
5. **Editare:** Folosiți butoanele "M" (Mute) și "S" (Solo) de pe fiecare track pentru a controla mixajul. Butonul "X" șterge track-ul.

4. Contribuția Membrilor

Echipa: 1 membru (Lucru individual)
Realizări:
- Proiectarea arhitecturii software.
- Implementarea engine-ului audio (mixing, timing).
- Dezvoltarea interfeței grafice complete folosind PySide6.

- Implementarea logicii de încărcare asincronă a fișierelor.
- Testarea și debugging-ul funcționalităților curente.

5. Dependențe și Execuție

Tehnologii Utilizate
- **Limbaj:** Python 3.10+
- **GUI Framework:** Qt for Python (PySide6)
- **Procesare numerică:** NumPy, SciPy

Biblioteci Necesare
Lista completă a dependențelor se află în fișierul `requirements.txt`:
- `PySide6`: Pentru interfața grafică.
- `sounddevice`: Pentru redarea audio în timp real.
- `numpy`: Pentru manipularea eficientă a bufferelor audio.
- `soundfile`: Pentru citirea fișierelor audio în diverse formate
- `scipy`: Pentru re-eșantionare (resampling) de înaltă calitate.

Instalare și Rulare

1. **Cerințe preliminare:** Python instalat.
2. **Instalare dependențe:**
Deschideți un terminal în folderul proiectului. Pentru a instala global dependențele rulați:

```
pip install -r requirements.txt
```

Pentru a utiliza un mediu virutal, mai întâi rulați:

```
python -m venv /your_path
source /your_path/bin/activate
```

3. **Rulare Aplicație:**

```
python main.py
```

Mediu de Dezvoltare
- **IDE:** Visual Studio Code
- **Sistem de Operare:** Dezvoltat și testat pe Linux (compatibil Windows/macOS).