C4GT 2025: Integrate Internet Archive Into BrainzPlayer

Hey Everyone 👋!

I am Rayyan Seliya (AKA rayyan_seliya123 on IRC and RayyanSeliya on GitHub), a prefinal year student at Indian Institute of Information Technology Agartala, India, studying Computer Science. I was thrilled to be selected as a contributor in the C4GT (Code For Govt Tech) 2025 program under the MetaBrainz Foundation. My project focused on integrating music streaming from Internet Archive into BrainzPlayer. It was mentored by Kartik Ohri (lucifer on IRC) and Nicolas Pelletier (monkey on IRC).

Let's start

Project Overview

ListenBrainz has a number of music discovery features that use BrainzPlayer to facilitate track playback. BrainzPlayer (BP) is a custom React component in ListenBrainz that uses multiple data sources to search and play a track. As of now, it supports Spotify, YouTube, Apple Music, SoundCloud, and Funkwhale as music services. It would be useful for BrainzPlayer to support the Internet Archive, which hosts a vast collection of digitized recordings from physical releases of the early 20th century, including 78 RPMs and Cylinder Recordings.

Let's Deep Dive into My Coding Journey!

My journey with MetaBrainz started during the community bonding period when I was exploring good first issues on the ListenBrainz tickets. That's when lucifer suggested me to contribute to this specific issue - adding an "add another" checkbox to the Submit Listens modal.

My First Contribution to the community !

The problem was simple but also annoying ! when users wanted to add multiple listens in a row (like for each side of a record), they had to reopen the modal every single time. I added a checkbox that kept the modal open after submission, making it much easier to add multiple listens.

This first PR was quite a journey with monkey's reviews! Through his feedback, I learned a lot about TypeScript, React best practices, and how big codebases handle accessibility and forms. The reviews went through several rounds from fixing form submission logic to implementing proper state reset using React hooks.

After several improvements, the PR was finally merged successfully on May 20th. This experience taught me about code quality, accessibility, and working with maintainers who care about the codebase.

After this successful contribution, lucifer suggested me to create a demo showing how Internet Archive recordings could be made playable locally. I built a simple full-stack app that could search and stream audio from Internet Archive collections. This project helped me understand how metadata indexing works and how to integrate external music services. I also studied existing handlers for Spotify, Apple Music, and SoundCloud to understand the patterns used in ListenBrainz's system.

You can check out the demo repository here to see how it all worked!

Integrate Music Streaming from Internet Archive

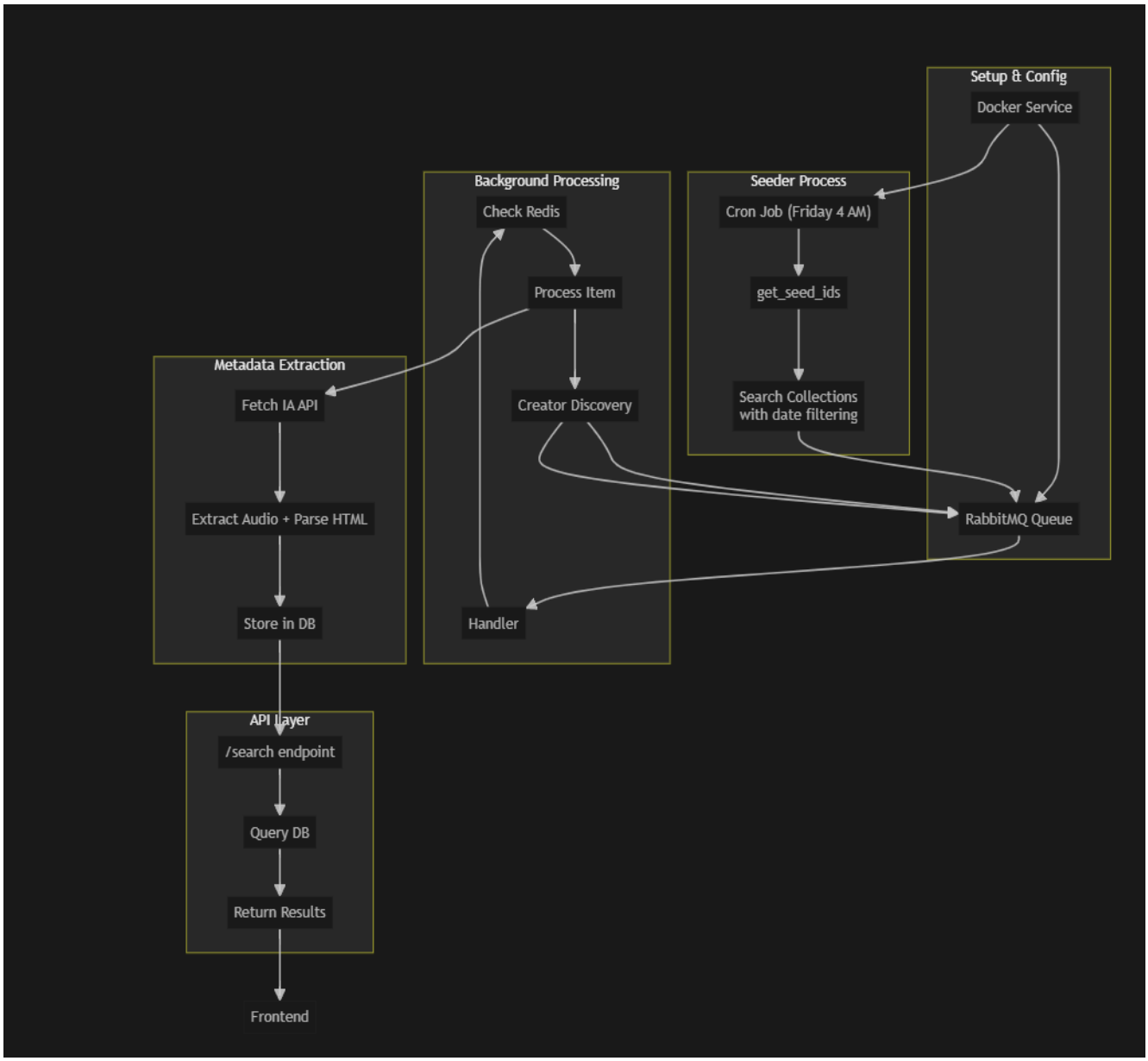
Internet Archive provides a rich collection of historical audio recordings that are freely available to the public. The main challenge here was to create an efficient indexing system that could crawl through the vast collections of 78rpm and cylinder recordings, extract metadata, and provide a seamless search experience for users.

Backend Architecture

- Metadata Cache Handler: A Python-based handler that crawls Internet Archive collections

- Database Schema: TimescaleDB tables to store track metadata efficiently
- Search API: RESTful endpoints for BrainzPlayer to query the cached metadata
- Background Processing: Queue-based system for continuous indexing

The following flowchart explains the various steps taken to integrate Internet Archive with ListenBrainz.



The database schema was designed to efficiently store Internet Archive track information:

```
CREATE TABLE internetarchive_cache.track (
    id              INTEGER GENERATED ALWAYS AS IDENTITY NOT NULL,
```

```
    track_id      TEXT UNIQUE NOT NULL,
    name          TEXT NOT NULL,
    artist        TEXT[] NOT NULL,
    album         TEXT,
    stream_urls   TEXT[] NOT NULL,
    artwork_url   TEXT,
    data          JSONB NOT NULL,
    last_updated  TIMESTAMPTZ DEFAULT NOW()
);
```

Frontend Integration

I added the InternetArchivePlayer component to BrainzPlayer by taking reference from the existing BrainzPlayer architecture and DataSourceType interface to understand how to properly integrate with the existing services. It detects when a listen originates from Internet Archive, handles search-based matching, and manages audio streaming.

I also created a custom icon component for Internet Archive that emulates the FontAwesome icons exported from react-fontawesome. it helps us to avoid messy and hardcoded styles for the icon and yeah thai was suggested by monkey and he helped me in this ! :

```
export const faInternetArchive: IconDefinition = {
  prefix: "fas" as IconPrefix,
  iconName: "internetarchive" as IconName,
  icon: [
    420, 480,
    [],
    "",
    "m 0,457.074 h 423.26 v 21.71 H 0 Z m 16.7,-41.74 h 390.7 v 30.05 H 16.7 Z..."
  ],
};
```

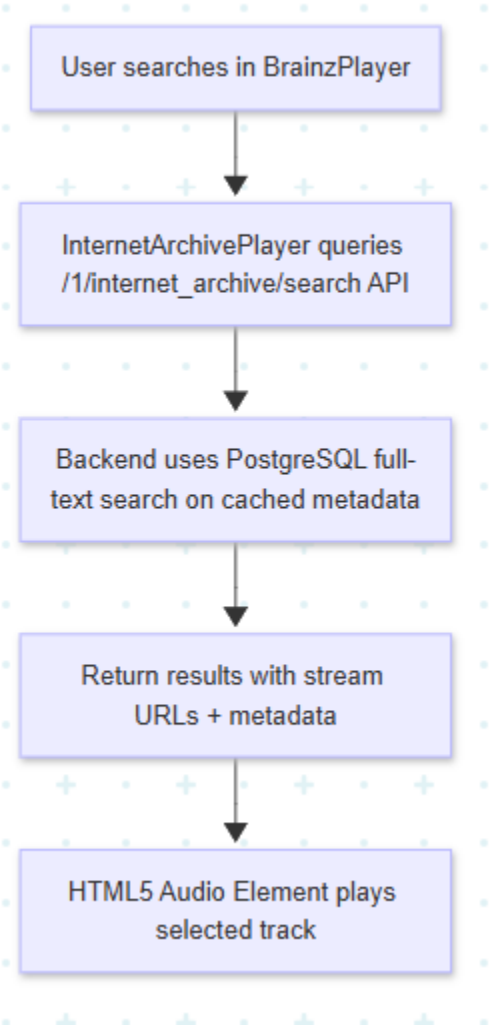
Search and Playback Flow

The search flow works as follows:

- User searches for a track in BrainzPlayer

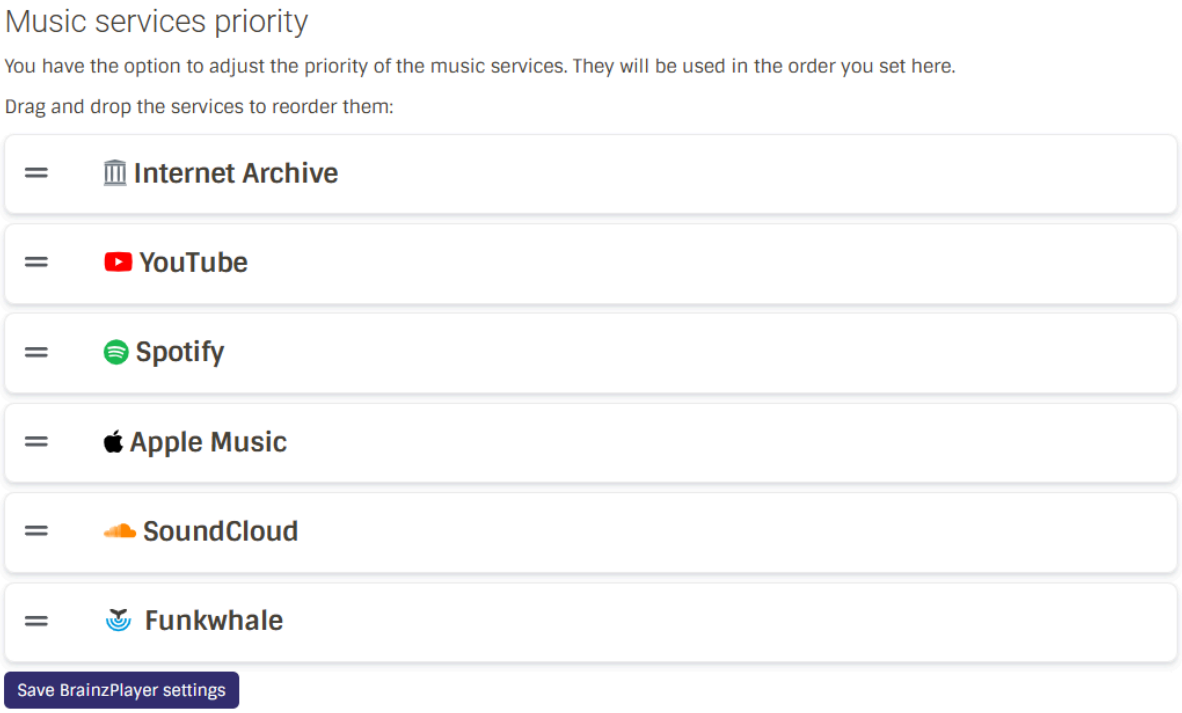
- InternetArchivePlayer queries the `/1/internet_archive/search` API
- Backend searches the cached metadata using PostgreSQL full-text search
- Results are returned with stream URLs and metadata
- HTML5 audio element plays the selected track

Flowchart explaining how a song is played through Internet Archive in BrainzPlayer

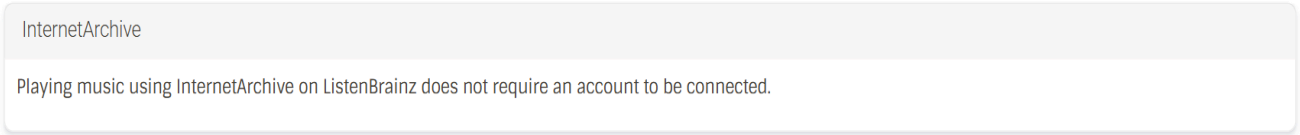


I added the option to enable Internet Archive in BrainzPlayer settings and ensure that Internet Archive service is activate and set the desired in the BrainzPlayer settings page.

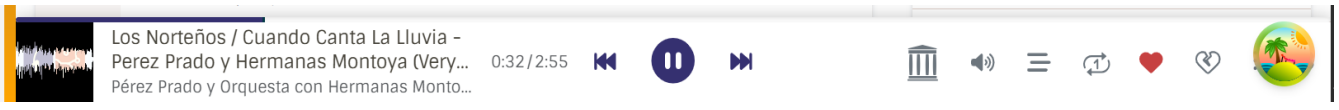
and users can control the priority of this service relative to other music services.



Users don't need to connect the Internet Archive as a music service ,since it's a free public-domain service does not require any account ! so i have added this UI in the connect services page !



And finally our brainzplayer ui will look like this



Implementation Challenges and Solutions

One of the complex parts was handling Internet Archive's diverse metadata formats. The metadata extraction had to handle various HTML descriptions, multiple artist formats, and different audio file types. I implemented a robust parsing system using BeautifulSoup that could extract artist, album, and track information from various description formats. So we have implemented this to tackle this challenge

```
def extract_from_description(soup: BeautifulSoup | None, field) ->
str | None:
    """
    Extracts a field (e.g. 'Artist', 'Album') from the IA description
HTML using BeautifulSoup.
    Handles both string and list input.
    """
    if not soup:
        return None

    for element in soup.find_all(["div", "p", "span"]):
        _text = element.get_text(strip=True)
        if _text.startswith(f"{field}:"):
            return _text[len(field) + 1:].strip()

    return None
```

Another Challenge: Seeding Limits and Discovery

Initially, we faced a challenge with the sheer volume of Internet Archive's collections. The 78rpm and cylinder collections contain millions of recordings, and trying to index everything at once would overwhelm the system and hit API rate limits.

The Initial Approach:

```
def get_seed_ids(self, limit_per_collection=1000) -> list[str]:
    """Fetch identifiers for 78rpm and cylinder collections."""
    collections = [
        {'name': '78rpm', 'query': 'collection:78rpm AND
mediatype:audio'},
        {'name': 'cylinder', 'query': 'cylinder mediatype:audio'}
```

```
    ]
    # ... limited to 1000 per collection
```

The Problem was Setting a hard limit of 1000 recordings meant we were missing out on discovering new and interesting content, and the system wasn't really "learning" about the collections over time.:

So After discussing with lucifer, we implemented a smarter approach combining date-based seeding with discovery-based fetching:

```
def get_seed_ids(self, limit_per_collection=1000) -> list[str]:
    """Fetch identifiers for 78rpm and cylinder collections with date
filtering."""
    today = datetime.today().strftime("%Y-%m-%d")
    last_week = (datetime.now() -
timedelta(days=7)).strftime("%Y-%m-%d")
    date_filter = f"[{last_week} TO {today}]"

    collections = [
        {
            "name": "78rpm",
            "query": f"collection:78rpm AND mediatype:audio AND
publicdate:{date_filter}"
        },
        {
            "name": "cylinder",
            "query": f"collection:cylinder AND mediatype:audio AND
publicdate:{date_filter}"
        }
    ]
    # ... process with date filtering
```

Another Challenge: Audio Format Detection

Initially, I was missing a vast array of file formats, which meant we were losing potential recordings that couldn't be streamed. Internet Archive hosts audio in many different formats, from modern digital formats to historical analog recordings.

So the solution i made by researching the Internet Archive library,I implemented a comprehensive audio format detection system to ensure we don't miss a single recording:

```
AUDIO_KEYWORDS = [
    "mp3", "ogg", "vorbis", "flac", "wav", "aiff", "apple lossless",
"m4a", "opus", "aac",
    "au", "wma", "alac", "ape", "shn", "tta", "wv", "mpc", "aifc",
"m4b", "m4p", "vbr",
    "m3u", "cylinder", "78rpm", "lossless", "lossy", "webm", "aif",
"mid", "midi", "amr",
    "ra", "rm", "vox", "dts", "ac3", "atrac", "pcm", "adpcm", "gsm",
"mmf", "3ga", "8svx"
]
```

Present Status and Future Improvements

The entire implementation for the Internet Archive integration is contained in the following components:

- Backend Handler: `listenbrainz/metadata_cache/internetarchive/handler.py`
- Search API: `listenbrainz/webserver/views/internet_archive_api.py`
- Frontend Player: `frontend/js/src/common/brainzplayer/InternetArchivePlayer.tsx`
- Settings Integration: Updated BrainzPlayer settings and music services pages

The implementation has been reviewed, The backend part has been merged can be seen here in these two prs :- pr1 , pr2 .Only frontend pr is left to be merge see here pr

As future improvements, it would be useful to implement more sophisticated search algorithms, add support for more Internet Archive collections beyond 78rpm and

cylinder recordings, and implement user preference-based content filtering. More thorough unit and integration tests would also be useful in preventing regressions.

Testing

It was my first time writing tests for a project of this scale, but I was successfully able to write basic tests for both frontend and backend. The existing tests were easy to read and served as a great reference! In the future, I will add more functional tests and integration tests.

Overall C4GT Experience

This summer has been an incredible journey for me to work with the MetaBrainz Foundation, and I'm deeply grateful to C4GT for this amazing opportunity. Contributing to ListenBrainz and implementing Internet Archive music service integration has been both challenging and rewarding, seeing my work now live in production for users worldwide. Being a part of MetaBrainz is an incredible feeling. I will be continuing to fix bugs, issues or contribute other improvements.

Throughout this journey, I have learned so many things. I am now more comfortable with Git and GitHub. Initially, I didn't have much experience with large-scale web applications, but during this period, I worked on my skills, tried – failed – researched – asked for help when stuck and finally finished the implementation.I have become more comfortable with Docker, TimescaleDB, and stuff like metadata indexing and music streaming implementation etc. The extensive code reviews with monkey significantly improved my TypeScript skills and taught me best practices for large codebases. Working with Docker services, Consul templates, and the complete infrastructure setup gave me proper industrial experience that I was completely new to as a beginner!

I would like to thank lucifer, monkey, and a lot of others for helping me throughout this period, guiding me and for constantly supporting me. Whether it was the MetaBrainz chat or the code reviews, I always received detailed feedback, help and suggestions.I built some cool stuff this summer and it's going to be used by people all over the world. Thank you to all others who helped me throughout this journey and helped and guided me! I hope you guys will enjoy listening to songs more with more services.

My proposal can be found here,PDF here
All my pull requests and commits for ListenBrainz during C4GT 2025!