Masarykova univerzita
fakulta informatiky

Musikk. A music streaming platform with social features.

Bachelor's Thesis

Kirill Vorozhtsov

Brno, 2025

Declaration

Hereby I declare that this thesis is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.
During the preparation of this thesis, these AI tools were used: - ChatGPT for debugging and small code corrections. - V0 for the initial styling config and layout. I declare that they were used in accordance with the principles of academic integrity.
I checked the content and took full responsibility for it.

Thesis Advisor: Mgr. Luděk Bártek, Ph.D

Abstract

This bachelor's thesis implements a music streaming platform with additional social features - live comment sections for songs and playlists, discussion forums, additional possibilities for interaction with the followed users etc.

A study is made beforehand in order to determine what users expect from the application; comparison and exploration of different existing platforms is presented in order to give a better insight into the market of similar applications.

The thesis leverages existing backend and frontend frameworks, such as Django and React, for the actual handling of the underlying data, logical processes and the interface of the platform. In addition, modern audio representation and streaming solutions, such as MPEG-DASH and HLS are used. In order for the application to feel responsive, Server Sent Events are added to provide two-way communication between the client and the server - this ensures that individual interactions are always synchronized with other users and instances of the program.

Keywords

Audio Streaming, Python, Django, React, MPEG-DASH, SSE

Contents

Chapter 1

Introduction

In recent years, with the rapid development of the Internet, music has become an even more integral part of everyday life. It has never been easier to experience and share music — we have come a long way from sharing physical media to simply sending a link to a streaming platform of choice. Consequently, music has integrated even deeper into social interactions between people, helping them bond and share strong emotional experiences [13].

One of the direct consequences of this trend is the fast emergence of numerous music-related platforms. While some focus on traditional music journalism or statistics, others offer unlimited access to audio content. Naturally, people have started to discover and engage with music that resonates with them more frequently.

Despite this, it is surprising that none of the major social networking platforms has integrated a fully-fledged music streaming service into its ecosystem. There are only two notable exceptions:

- **VK** — a Russian social network that includes a built-in music streaming service.
- **QQ** — a Chinese streaming platform with basic social integration, mostly tailored to content authors.

However, both platforms are limited in terms of broader functionality, which will be discussed in later chapters.

The goal of this thesis is to design a music-centric platform that supports collaboration and social interaction around music.

This work is divided into the following **six** chapters:

1. **Survey Results:** Presents the outcomes of a survey illustrating how people consume music, how prevalent it is in social interactions, and why this thesis is relevant.

2. **Platform Comparison:** Compares existing streaming solutions and explores non-musical platforms that influence people's audio habits.

3. **Specification:** Details the implementation plan, overall structure of the application, and key features to be developed.

4. **Application Outline:** Describes the application's interface and functional outline.

5. **Implementation and Design:** Explains implementation details and design choices made during development.

6. **Conclusion:** Summarizes the results and discusses possible improvements.

Chapter 2

Survey Results

Chapter 3

Existing Platforms

In order to better understand what instruments people use when interacting with music, it would be better to look at the already working solutions. The descriptions provided, instead of giving general information, will be focusing on the social aspects of the platforms.

3.1 Streaming Services

As it can be seen from the table (ref to survey table) and further confirmed by the recent study of the International Federation of the Phonographic Industry[1], nowadays, the most prevalent way of music consumption and discovery are the streaming services. There are many existing platforms, but I suggest we look only at those which are both popular and have unique features:

- Spotify. One of the most prominent social features on Spotify is the 'Spotify Jam'[2]. It lets people create a collective song queue which is then synchronized among all connected users. Moreover, volume, the order of songs and other aspects of the playback can be controlled individually. Another notable tool is the 'Blend' playlists[3]. These are playlists created automatically between two people, which contain songs matching audio preferences of both users. Lastly, 'Friend Activity'[4], which shows what the people you follow are currently listening to, and 'Listening Parties' that are live chats with limited capacity, which can be joined for a short time when new music is being released[5, 6].

- VK Music. As was mentioned previously, this a music service integrated into the VK social network. Consequently, it is possible to send songs and playlists via private messages and add audio materials to posts in groups. VK Music also supports algorithmic playlists based on the groups that a user follows. It analyzes the audio content posted in the specific groups and puts similar songs in the before mentioned playlists.

- SoundCloud. SoundCloud is one of the few platforms which lets its users leave comments and reactions on songs and playlists[7, 8]. In addition, each user has a personal feed, consisting of his personal uploads and reposts of other's content[9], which is visible when visiting his profile.

- Bandcamp. Every Bandcamp user has a profile with 4 tabs - 'collection', 'wishlist', 'followers' and 'following'. By far the most interesting feature is under the ‘following‘ tab - it is possible to list the genres which you would like to appear to other users looking at your profile. The 'whishlist' tab is also worth mentioning, as Bandcamp's model combines streaming and traditional buying of separate releases, in this tab the user is able to show what he is looking forward to listening in the future.

3.2 Forums

A lot of different resources are gathered under this umbrella term TODO

3.3 Other Platforms

TODO

Chapter 4

Specification

In this chapter the general outline of the application is specified via functional and non-functional requirements. Some of the features are based on the feedback from the survey and short in-person interviews, while others are based on the analysis of the existing platforms. The individual requirements are ordered based on the 'MoSCoW' criteria, i.e. from most to least important.

4.1 Important Terms

Below is the list of important terms that will be appearing throughout the requirements section.

- **Anonymous User** – A User which is not yet registered in the system or has not logged in.
- **Song** – An object representing an individual audio object.
- **Song Collection** – Represents a container that holds multiple Songs.
- **Playback** – A process during which the user is receiving audio information.
- **Playback Item** – A Song or a Song Collection.
- **Playback Queue** – A list of Playback Items.
- **Comment** – A small piece of text provided by the User input usually placed under a specific object to which it refers.
- **Reply Comment** – The same as Comment, but must be created in relation to another, 'parent' Comment.
- **User Profile** – An object which encompasses all of the information related to a specific User.

- **Forum** – A standalone list of User Comments. Has a unique name.

4.2 Functional Requirements

Requirement	**Priority**
The system shall support over-the-net audio Playback.	Must Have
The system shall provide a way to control the audio Playback. That includes shifting the Playback backward and forward, stopping the playback.	Must Have
The system shall provide a way to show Playback Items.	Must Have
The system shall support new Playback Item addition.	Must Have
The system shall provide a way to enqueue Playback Items and the means to manipulate the Playback Queue.	Must Have
The system shall have individual User Profiles.	Must Have
The system shall allow Anonymous Users to create a User Profile and log in to the system using email and password.	Must Have
The system shall allow Users to change their User Profile information.	Must Have
The system shall differentiate presented content based on the User. That includes Songs and Song Collections, User Profile information and other related items.	Must Have
The system shall provide a way for Users to add Comments under Song Collections, Songs and other related items.	Must Have
The system shall provide a Forum system. Forums shall be able to be created by Users. Forum Comments must be able to embed Playback Items.	Must Have
The system shall provide a way for Users to create relations to other Users. All of the Comment-related systems shall be filtered on that User relations.	Must Have
The system shall make it possible to create Reply Comments for Comments.	Should Have
The system shall provide a notification system for Reply Comments and other appropriate items.	Should Have
The system shall make it possible to filter Playback Items based on the User relations.	Should Have
The system shall provide chat capabilities.	Could Have

Table 4.1: Functional Requirements

4.3 Non-functional Requirements

Requirement	**Priority**
The system shall provide access to its contents only to authorized Users. The only exception shall be the ’Log In/Sign Up’ page.	Must Have
The system shall store and transport sensible user information only in safe manners.	Must Have
The system shall, in case of failure, remain rendered on the screen and show the appropriate non-technical message to the User.	Must Have
The system shall render new available information without refreshing the content page. That includes new Comments, Playback Items, updated Playback Queue and other related items.	Must Have
The system shall synchronize Playback across multiple devices.	Should Have

Table 4.2: Non-functional Requirements

Chapter 5

Implementation Planning

This section provides an insight into the technology choices used to implement the application. In addition, UML diagrams are provided as a high-level overview of the system.

The created platform is built using the Model-View-Controller[10] architectural pattern. It is split into three major parts: - Backend. Implements all of the data-related logic and processing. - API. Is responsible for transferring the data to the frontend and receiving commands from it. - Frontend. Presents the data to the User and accepts his commands.

5.1 Backend

Python was chosen as the main language for the Backend part of the application. It has support for all the needed instruments, such as calling external processes, file handling, and executing asynchronous code. Moreover, it has a flexible and straightforward syntax allowing for rapid development. As this project is mostly IO-driven and most of the processor-heavy operations are offloaded to external processes, i.e. no thread programming is used, Python's performance limitations[11] will not have such a drastic effect.

5.1.1 Framework Comparison

Currently there are 3 well-developed and widely-used WEB frameworks available for Python:

- **FastAPI.**
  'FastAPI is a modern, fast (high-performance), web framework for building APIs with Python based on standard Python type hints.' [12]. This framework offers a lightweight approach to API declaration, without adding almost no overhead. It has the best integration with API documentation standards such as Open API and API representation tools, e.g. Swagger.

However, it is relatively new and barebones; given the requirements and the fact that still not many extra packages are available - a lot of custom code would have to be written on top, which is a disadvantage for this project.

- **Flask.**
  Flask is a micro framework based on the WSGI interface. It is well suited for smaller apps and takes a lean approach when it comes to choosing the components that will be used with it - most of the additional functionality is written by the community and is distributed as external packages.

  One of the main problems with Flask is that it does not integrate well enough with the ASGI interface which is practically necessary for proper handling of Server-Client path of communication. Moreover, using external packages for every almost bit of additional functionality can potentially be a security risk.

- **Django.**
  Django is a 'batteries-included' framework, in a sense that almost everything - from access control and database management to API routing and admin interface, is a part of it. Moreover, it has an extensive ecosystem with support for e.g. REST-style API design via `django-rest-framework`, different methods of Authentication, even different transfer protocols, such as websockets and Server Sent Events are available via the `django-channels` package and others that build on top of it.

  Some of the notable drawback are its opinionated design choices and the size of the application, as this framework includes a lot of parts that possibly will not be used.

Out of those three frameworks Django is the most suitable for the task. The application will heavily rely on the database, frontend will be used only for data presentation, hence a JSON-based API will be implemented, moreover Django has a proper support for ASGI-based servers.

5.1.2 Audio Representation and Streaming

5.1.3 Database

5.1.4 Authorization

5.1.5 Server-Client Communication

5.2 Frontend

Sources

[1] author. *title*. URL: https://www.ifpi.org/wp-content/uploads/2023/12/IFPI-Engaging-With-Music-2023_full-report.pdf.

[2] author. *title*. URL: https://support.spotify.com/cz/article/jam/.

[3] author. *title*. URL: https://support.spotify.com/cz/article/social-recommendations-in-playlists/.

[4] author. *title*. URL: https://support.spotify.com/cz/article/friend-activity/.

[5] author. *title*. URL: https://newsroom.spotify.com/2024-09-12/fans-artists-connections-features/.

[6] author. *title*. URL: https://support.spotify.com/us/article/listening-party/.

[7] author. *title*. URL: https://help.soundcloud.com/hc/en-us/articles/115003566008-Commenting-basics.

[8] author. *title*. URL: https://help.soundcloud.com/hc/en-us/articles/31039497560731-Reactions.

[9] author. *title*. URL: https://help.soundcloud.com/hc/en-us/articles/31039497560731-Reactions.

[10] author. *title*. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

[11] author. *title*. URL: https://wiki.python.org/moin/GlobalInterpreterLock.

[12] author. *title*. URL: https://fastapi.tiangolo.com/.

[13] Peter J. Rentfrow. *The Role of Music in Everyday Life: Current Directions in the Social Psychology of Music*. Accessed: 2025-03-21. 2012.