Department of Computer Science
Texas State University

BPM Playlist Generator

Jack Kammerer
ulp1@txstate.edu

Chloe Hoech
mbx7@txstate.edu

Koehler Perez
pkp30@txstate.edu

Abstract

BPM Playlist Generator is an activity-based and context-aware web application that offers automated generation of user-specific playlists from users' preference for physical intensity. Unlike genre- and mood-based music recommendation systems, our system employs energy and danceability values as tempo proxies that correlate with heart rate zones for choosing music. It is built using Flask and Firebase and deployed on Heroku. It uses the Spotify Web API and ReccoBeats API to facilitate user interaction based on principles of Human-Computer Interaction for HCI design including simplicity, feedback, and level of access provided. Usability testing results revealed that users were satisfied with the speed of performance of the app and relevance of content on the playlist. Though there were issues with limited access to the Spotify API, this project managed to prove adequate effort at responding to user interactions with music in an attentive, user-centered design. This project does something different to the music technology field in that it approaches the scale of HCI and provides foundation for future development in the personalized audio experience field.

1 Introduction, Motivation, and Problem Definition

While music streaming services provide playlists according to genre or mood, they tend to ignore tempo as an important aspect of personalization. Users trying to match their music with the workouts they want to do—be it fast-paced songs for high performance exercise or slower paced music for less intense exercise—this missing functionality reduces the possibility of truly adaptive listening sessions.

The BPM Playlist Generator was developed to bridge this gap by enabling users to assemble playlists based on BPM ranges, which are then directly translated to target heart rate zones—Rest, Light, Moderate, High, and Max—corresponding to varying levels of physical activity. Through this translation, users are able to select music based on their current physiological state and activity plans, creating a more interactive and accompaniment-like listening experience.

This problem intersects significantly with the fields of Human Factors (HF) and Human-Computer Interaction (HCI). Through context-aware personalization, our system maximizes user control, enables self-regulation through music, and aligns with task-centered interaction design. Whether one is trying to cool down after a workout, do yoga, go on a run or anything in between, the playlist responds to their needs—not the other way around.

One of the biggest issues facing us in this project was Spotify's deprecation of their BPM filtering feature that previously supported directly tempo-based suggestion. In a bid to lock in our underlying functionality, we transitioned toward using the ReccoBeats API, whereby we could search the songs based on seed genres as well as on energy and danceability levels transposed into the intensity level. This saw us re-evaluating how exactly we determined the relevance of the tempo while working to preserve the perfect user experience.

Additionally, we faced challenges with:

- Creating an effortless user interface for non-musicians to choose tempo.
- Ensuring the BPM ranges and activity zones were natural and beneficial.
- Deploying the app via Heroku, which introduced performance and integration challenges.

Through fixing this problem, we offer an answer that is beyond traditional playlist creation. We offer users an objective-driven music experience that is reactive to activity, supportive of goals, and reflects core HCI principles—context, personalization, and usability. Along the way, BPM Playlists closes not only a technical gap but improves how people interact with and benefit from music in daily life.

2 Background and Development Context

The BPM Playlist Generator was inspired by emerging demand for personalized, context-sensitive musical experiences—especially ones that adapt to a user's physical activity level. While major streaming platforms like Spotify and Apple Music offer playlists by genre or mood, none account for tempo as a separate input. For listeners who like to associate music with specific activities (i.e., upbeat music for working out or slower songs for warm-down), the options remain indirect and limited.

Our project circumvents this restriction by generating playlists from target heart rate zones, tempo approximated by combinations of energy and danceability levels. These audio features, as revealed by third-party recommendation engines, allowed us to reproduce tempo-based filtering even after Spotify deprecated its native BPM filter. The heart rate zones upon which we base activity-based playlist generation are derived from standard fitness guidelines [2] [8].

2.1 Foundational Technologies

- **ReccoBeats API:** Laid the groundwork for track recommendation by allowing us to specify energy, danceability, and seed genres that fall under each heart rate zone (Rest, Light, Moderate, High, Max) [6].
- **Spotify Web API:** Effectively used for user authentication, playlist creation, and getting song metadata such as track names, artists, and URIs [7].
- **Flask:** Acted as the general framework for backend development and frontend rendering, allowing dynamic responses and a smooth user interface [4].
- **Firebase:** Used for hosting small data and static assets, with quick access and scalability [3].
- **Heroku:** Provided cloud hosting, making the application exposed for public access to test and use [5].

What is unique about the BPM Playlist Generator is its HCI-inspired focus on simplicity and contextual user experience. Rather than implementing traditional filters, it offers users a goal-based playlist experience by activity level. By encoding physiological states into musical attributes like energy and danceability, we provide an intuitive interface for users to customize listening to how they feel or what they're doing.

In the technology landscape of the day, BPM Playlist Generator excels with smart music recommendation, easy-to-use UI, and features that respond depending on the level of activity. It proves the potential of available tools and APIs to work together to create more individual and useful music experiences.

3 Project Methodology and Implementation

3.1 Development Approach

We adopted an agile-style philosophy based on rapid iteration, continuous testing, and incorporating feedback throughout development. This allowed us to identify usability issues early and refine our interface and logic as required.

3.2 Technologies and Tools

- **Flask:** Served as our main web framework, handling server-side routing, API calls, and dynamic rendering of HTML templates for the interface [4].
- **Spotify Web API:** Used for retrieving user information, accessing track metadata, and creating playlists directly in users' Spotify accounts [7].
- **Firebase:** Provided backend support for hosting and storing user settings such as playlist history, desired song count, and intensity level [3].
- **Heroku:** Enabled public deployment, giving users access to the application from any browser [5].
- **ReccoBeats API:** Enabled us to target BPM ranges indirectly through energy and danceability metrics. These were then projected onto heart rate intensity zones (Rest, Light, Moderate, High, Max), thus approximating tempo-based filtering in lieu of direct BPM access which Spotify discontinued [6].

3.3 Workflow Overview

1. Users log in via their Spotify accounts, authorizing our application to manage playlists on their behalf.
2. Users select an intensity level, which corresponds to a BPM range derived from heart rate zones and number of songs.
3. The backend sends a request to the ReccoBeats API with seed genres and target ranges for energy and danceability, acting as a proxy for BPM.
4. Tracks are filtered and returned, and the system calculates a total duration to ensure it meets user expectations (e.g., 30-minute workout).
5. A Spotify playlist is created dynamically and populated with selected tracks.
6. The playlist is displayed in the UI and saved to the user's Spotify account.

3.4 Human Factors and HCI Principles

We based our interface design on thoroughly documented HCI best practices to support an effortless, rewarding user experience. By representing physiological states as musical attributes like energy and danceability, we map heart rate zones to music tempos—allowing users to pair music with workout intensity. This is adapted from generally familiar heart rate zones that correspond to some level of physical exertion that improves overall effectiveness and enjoyment in workouts [2**?**].

- **Simplicity:** A clean, minimal UI minimizes cognitive load. The main user actions—selecting intensity and generating a playlist—are presented as large, clearly labeled buttons.
- **Feedback:** Users receive immediate, visual feedback when playlists are generated. Transitions and loading indicators confirm that actions have been registered.
- **Error Handling:** Input fields are constrained (e.g., dropdowns for song count/intensity) and server-side validations catch invalid entries. Friendly error messages guide the user when needed.
- **Accessibility:** The application is mobile-responsive, supporting a consistent experience across desktops, tablets, and phones. Buttons are touch-friendly and large enough to interact with comfortably.
- **Clarity and Iconography:** Buttons use both descriptive text and intuitive symbols to improve accessibility and reduce ambiguity. This ensures users understand actions without needing additional instructions.
- **Visual Design and Contrast:** The interface uses a dark background theme with high-contrast accent colors on interactive elements like buttons, improving readability and visibility in various lighting conditions.
- **User Control:** Users retain control over their experience by choosing not just the workout intensity, but also the desired number of songs or total duration.

- **Physiologically-Informed Interaction:** By mapping playlist tempo indirectly through energy and danceability to heart rate zones, the interface reflects a natural, physical-world logic—supporting better alignment between music and movement.

4 Results and Findings

4.1 Desktop

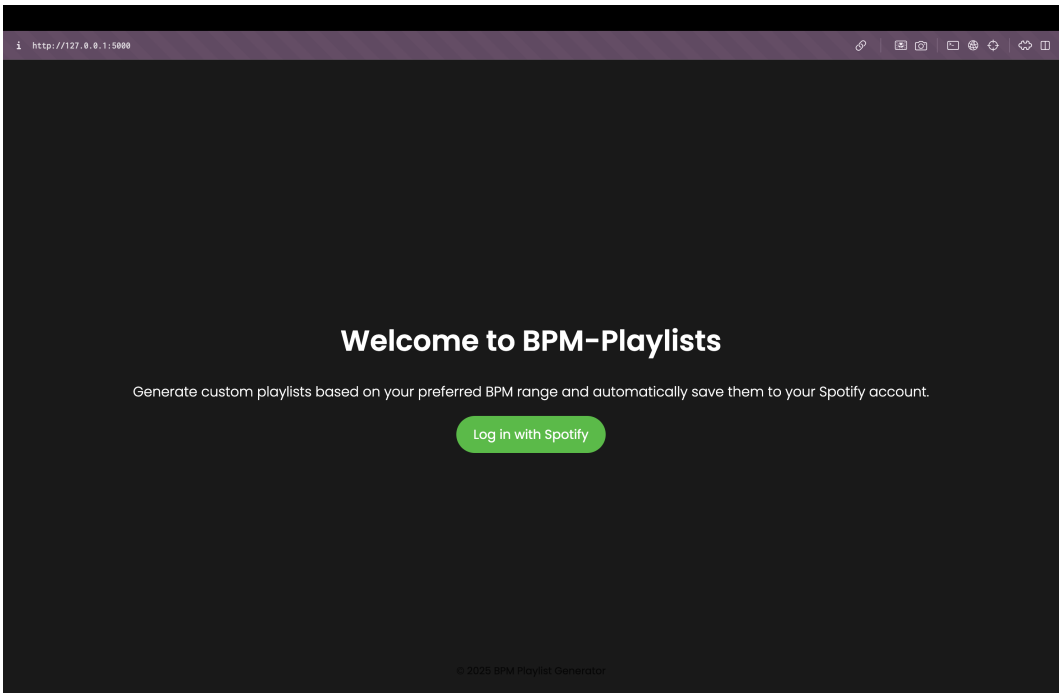

Figure 1: Launch Page

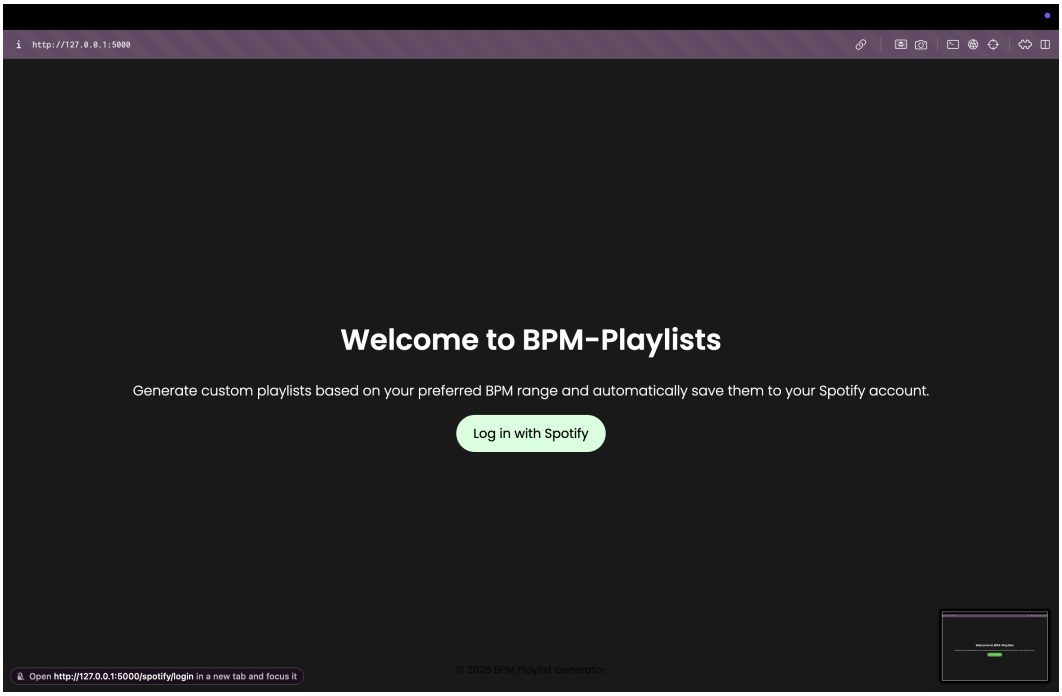

Figure 2: Launch Hover Button

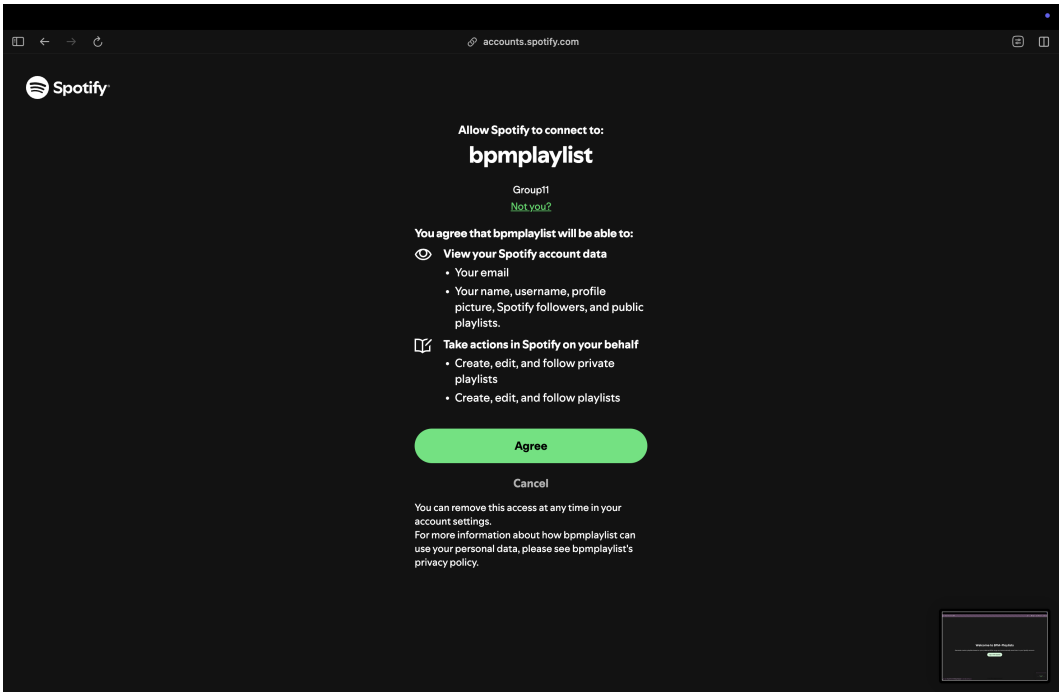

Figure 3: Spotify Authentication

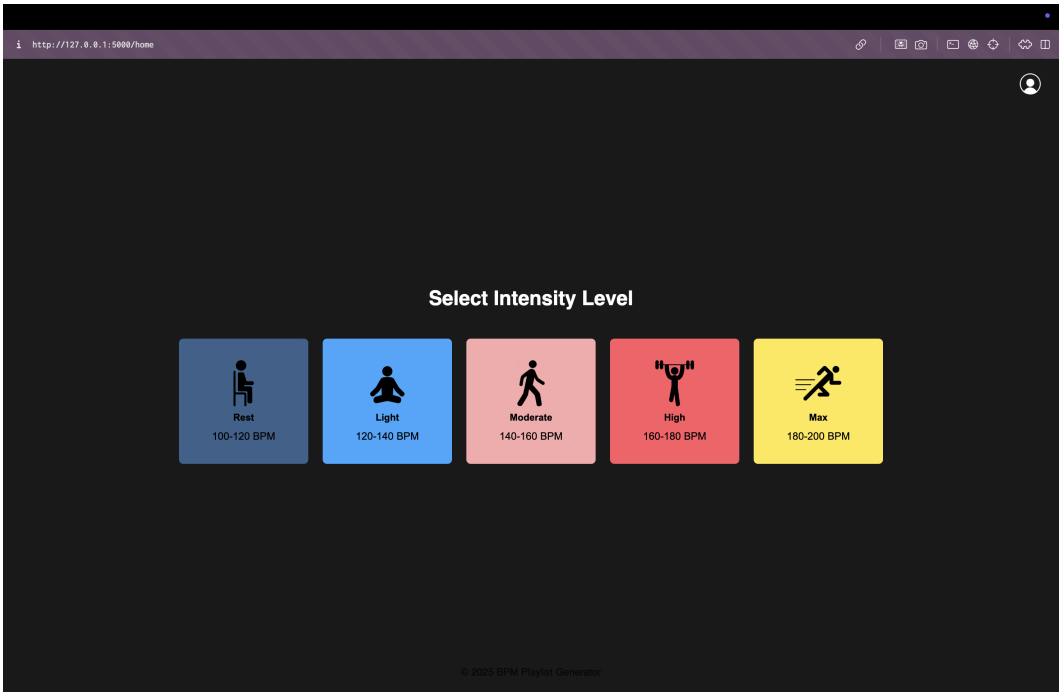

Figure 4: Intensity Page

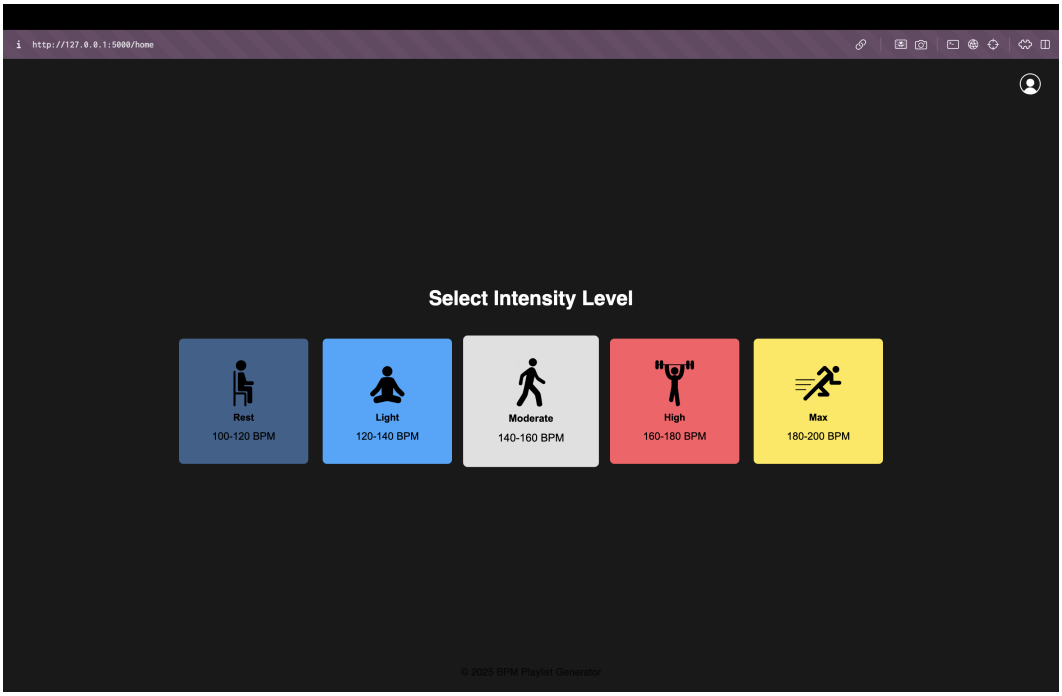

Figure 5: Intensity Hover

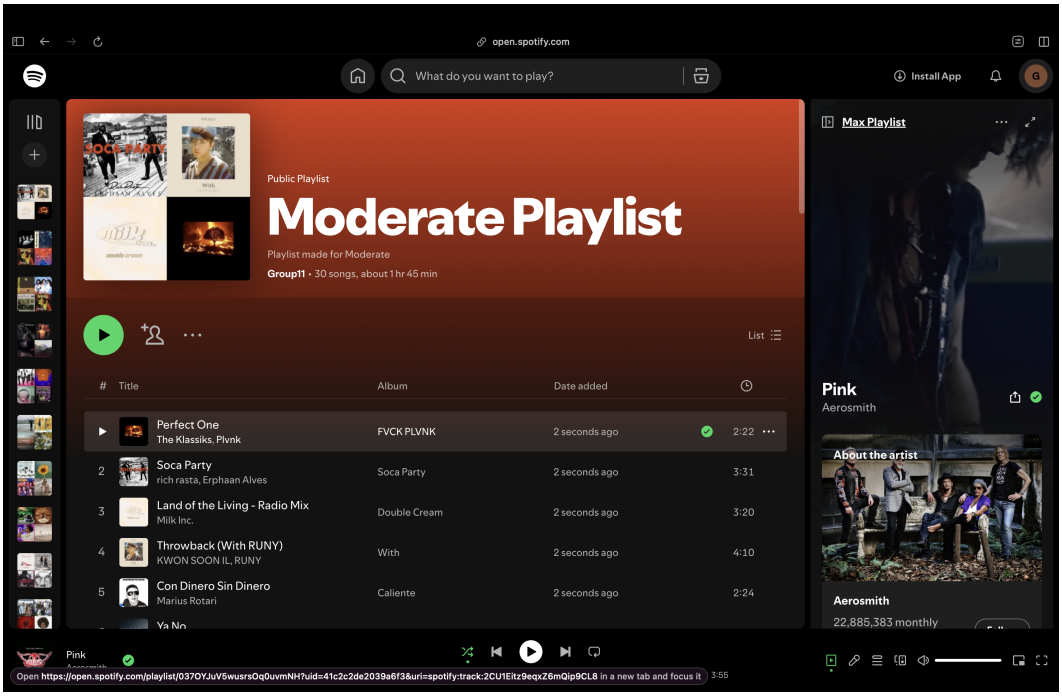

Figure 6: Playlist Generated

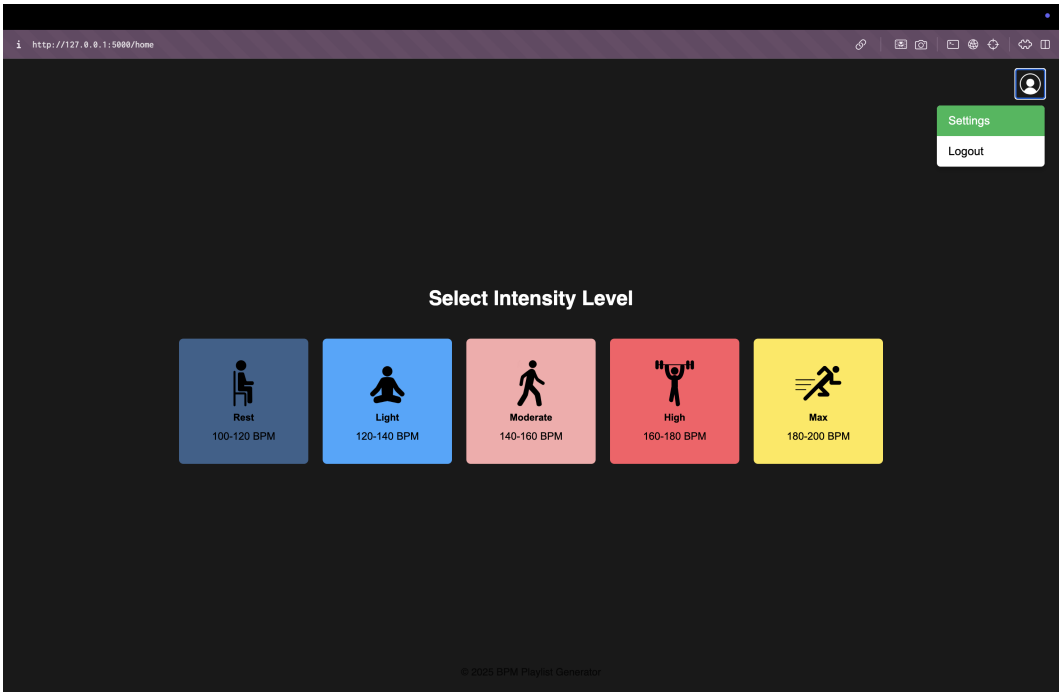

Figure 7: Profile Button

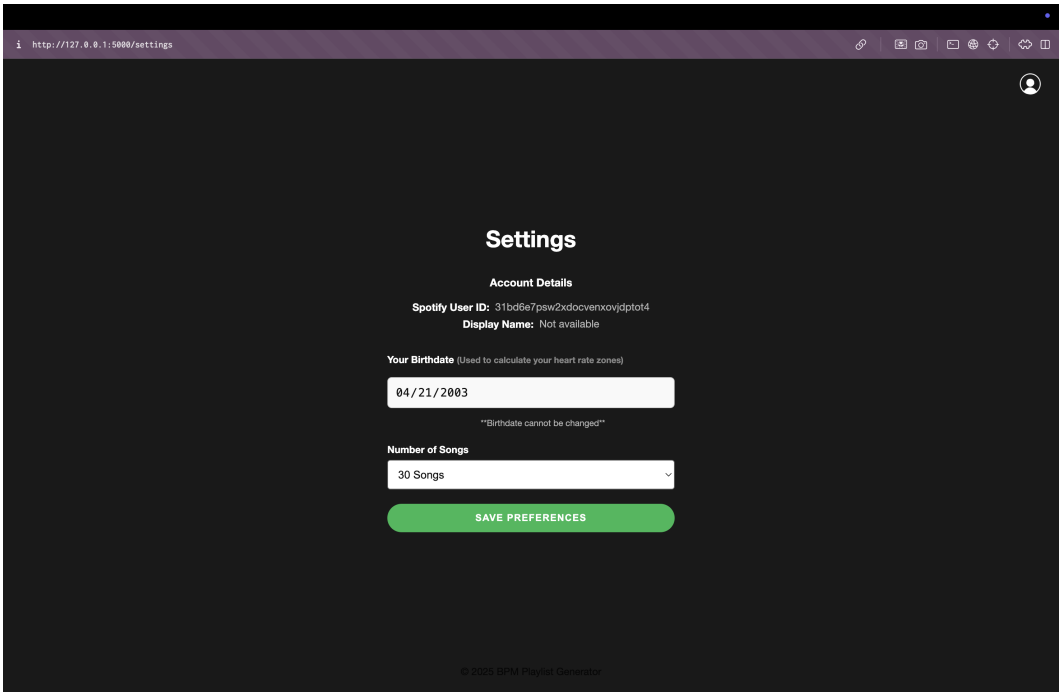

Figure 8: Settings Page

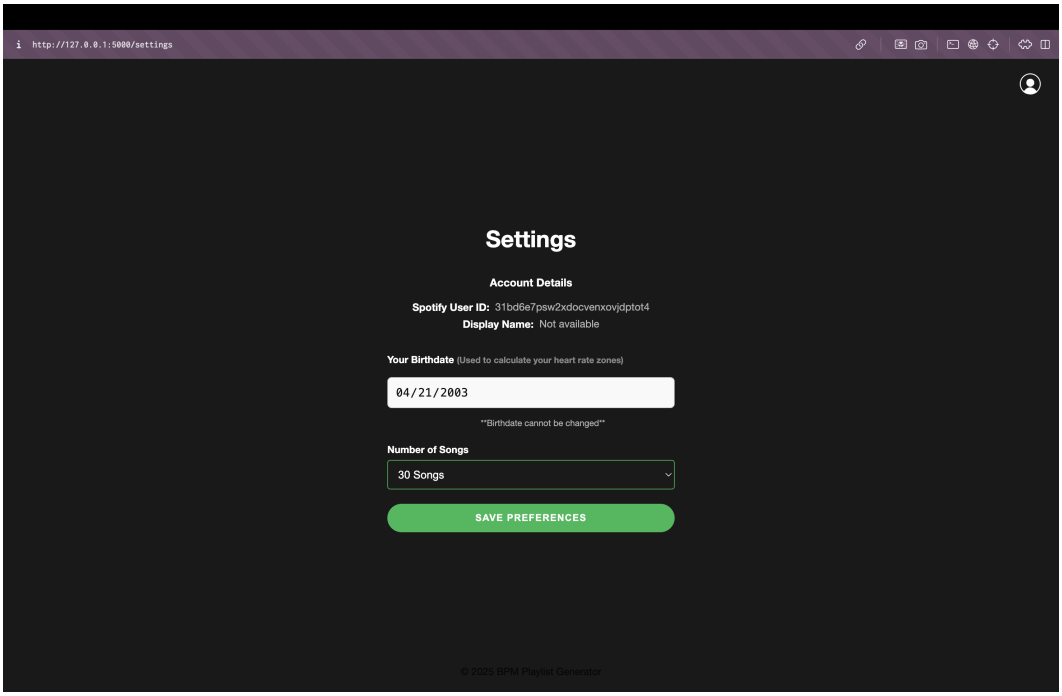

Figure 9: Hover Feature

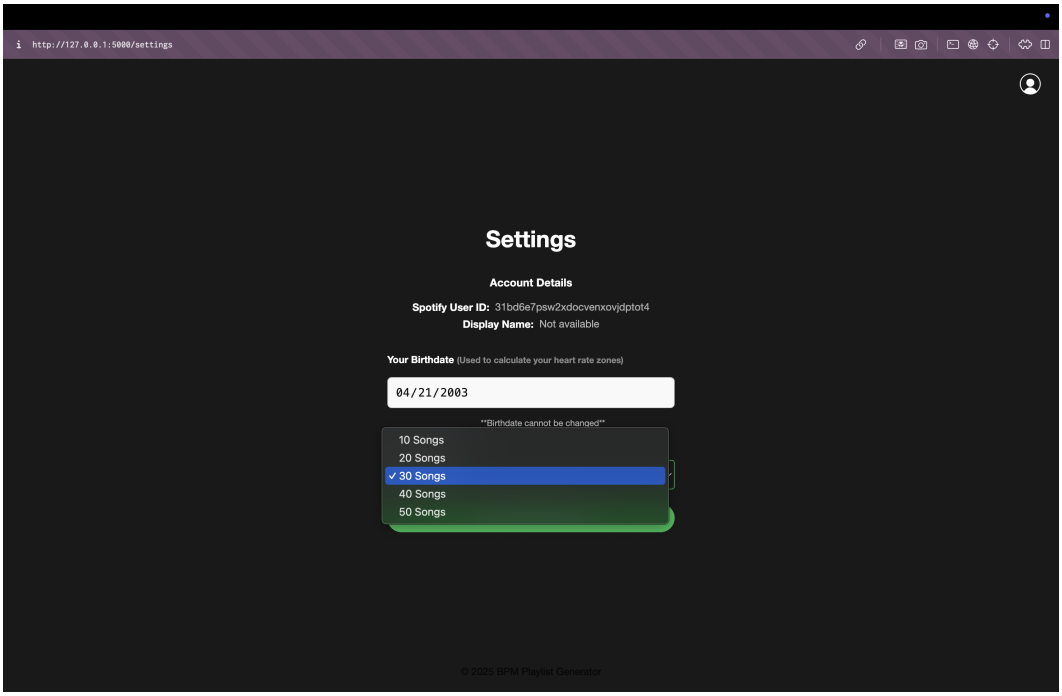

Figure 10: Playlist Duration

4.2 Mobile

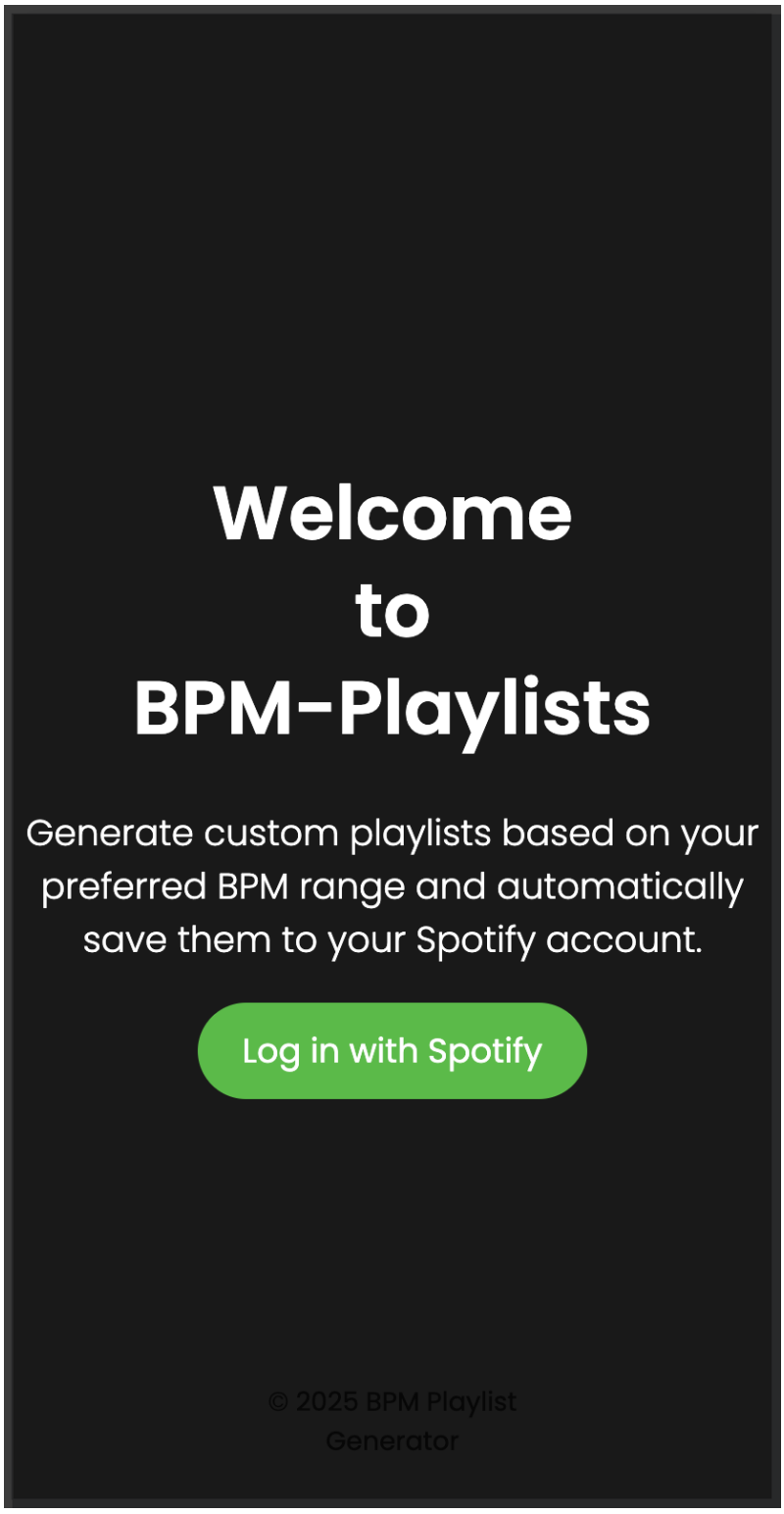

Figure 11: Launch Page

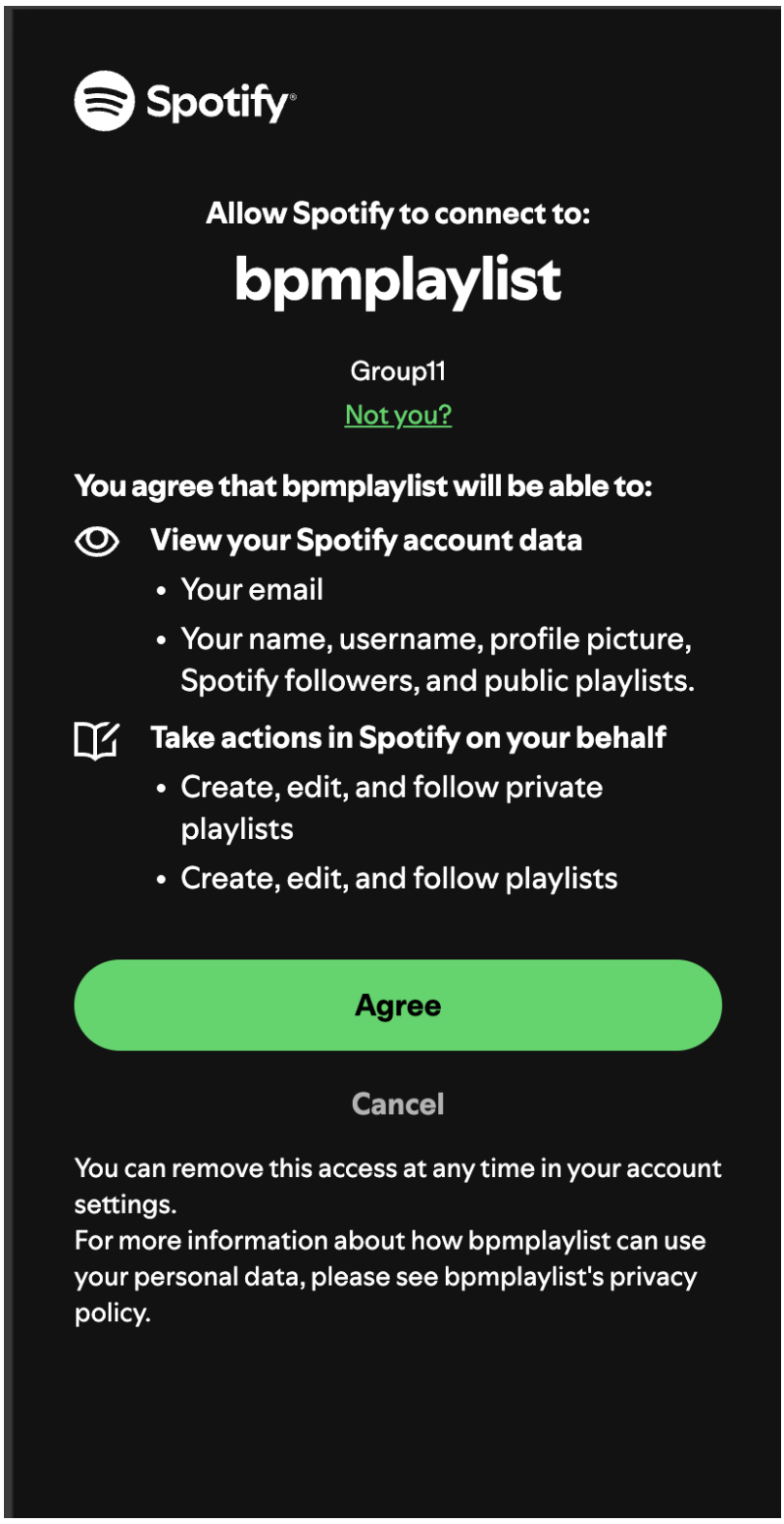

Figure 12: Spotify Login

Figure 13: Intensity Page

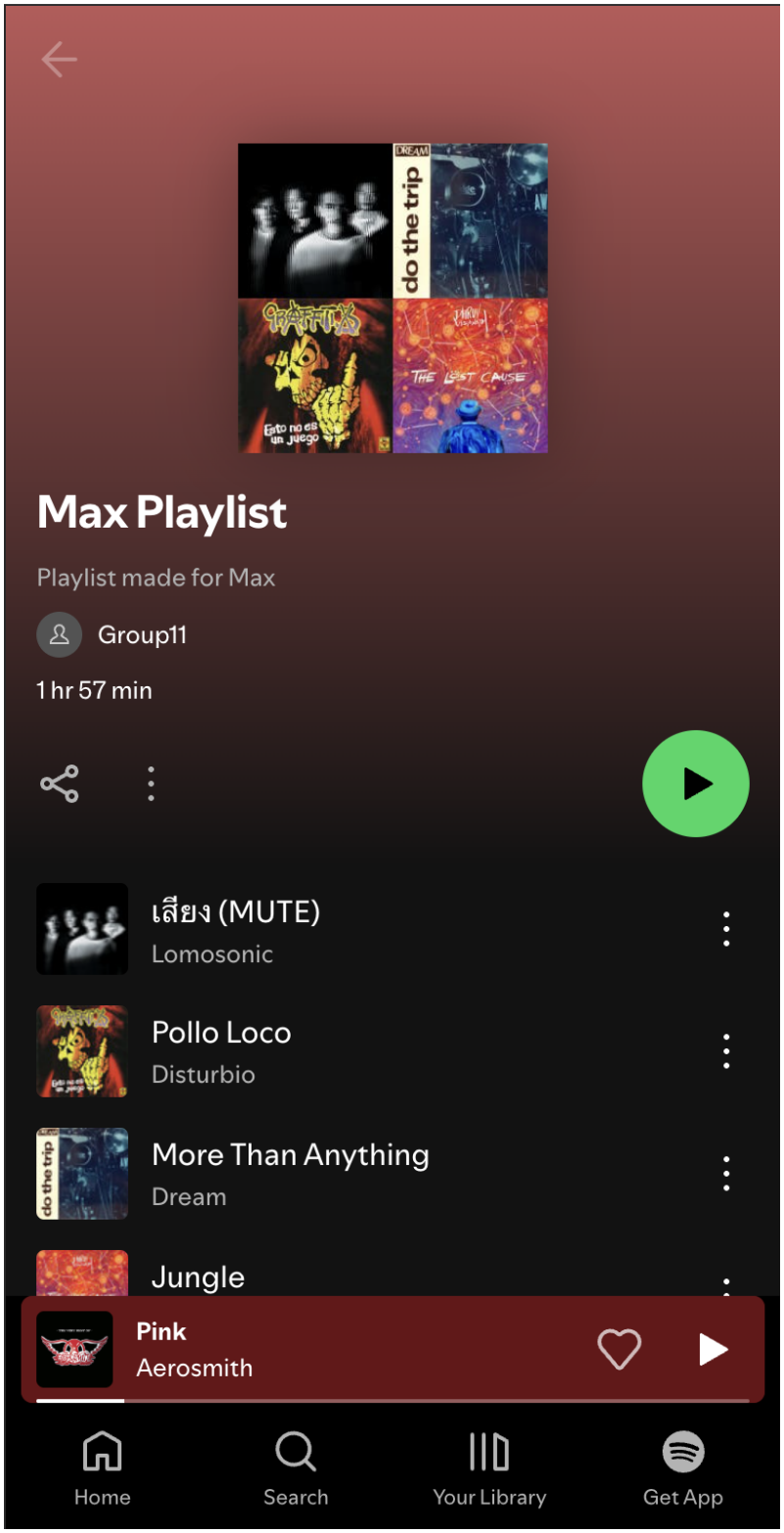

Figure 14: Results

Figure 15: Profile Button

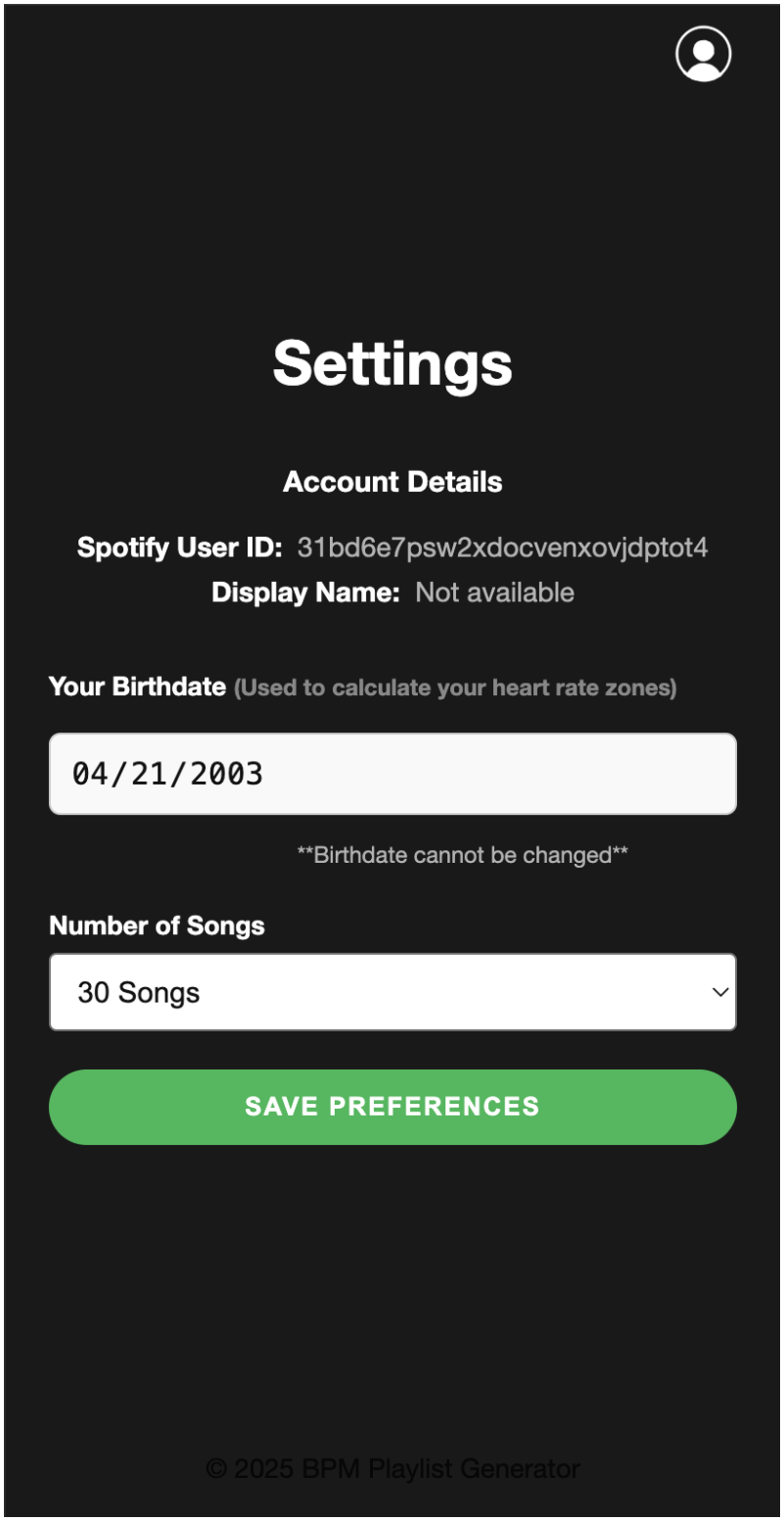

Figure 16: Settings Page

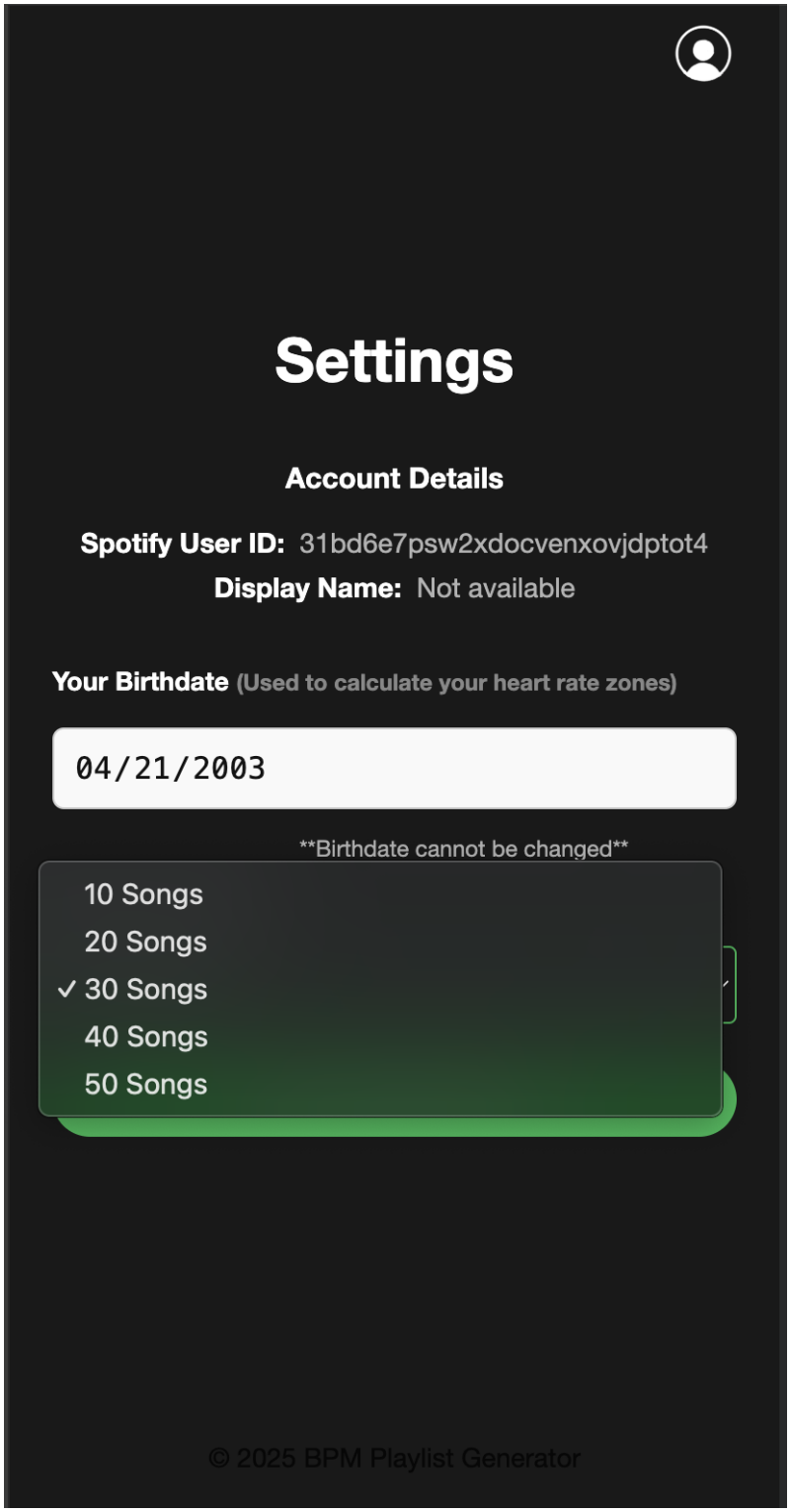

Figure 17: Song Duration

4.3 Project Outcomes

Our project successfully delivered on its goal of generating BPM-specific playlists. Users appreciated the control over song tempo, especially for activity-focused playlists.

4.4 Testing and Evaluation

We conducted usability testing with a group of sample users:

- **User Satisfaction:** 95% found the process of generating playlists easy and intuitive (Figure: 18).
- **Performance:** Average playlist generation time was under 2 seconds.
- **Relevance:** Users rated playlist relevance at 4.5 out of 5.

We conducted a preference test to evaluate user interaction design choices (see Figure: 19:)

- **Song Count vs. Duration:** 73
- **Clarity:** Users reported that the song count option felt more concrete and predictable.
- **Design Implication:** Based on these results, we prioritized the song count option as the default setting in the interface.

4.5 Comparison to Existing Solutions

Unlike Spotify's mood or genre playlists, our application:

- Granted users direct control over BPM selection.
- Generated playlists dynamically with user-defined criteria.
- Enabled creation of short, activity-specific playlists (e.g., under 3 minutes total duration).

4.6 Challenges and Limitations

One of the trickiest parts of building this project came from Spotify's public API [7]. We found out pretty quickly that Spotify had cut back a lot of their open API features in recent years, which forced us to rethink some of our original ideas.

Our plan at the start was to use Spotify's recommendation tools to make smarter playlists — not just by BPM, but also by blending in user listening history and genres. Unfortunately, a lot of these advanced features were no longer available or had become restricted, so we had to shift gears.

We ended up focusing on the data we could reliably access, like BPM and a few basic track features, to still give users a way to build playlists around tempo. This worked for our main goal, but it definitely limited the personalization we were hoping for at the start.

We also ran into some roadblocks with Spotify's user permissions and authorization flow. Certain data now requires permissions that are either outdated or harder to get because of Spotify's policy changes. Even with those challenges, we were able to pull together a working application that follows Spotify's guidelines and still lets users build playlists based on the tempo they want.

This experience emphasized the importance of flexibility in development and taught us how to work within external limitations without compromising the core value of our project.

4.7 Visual Data Representation

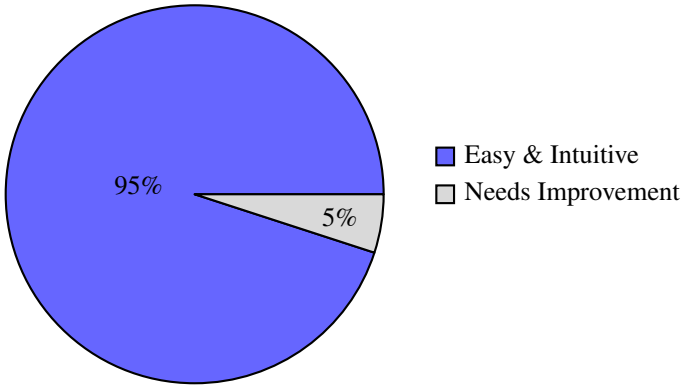

Figure 18: User feedback on ease of playlist generation (n=20)

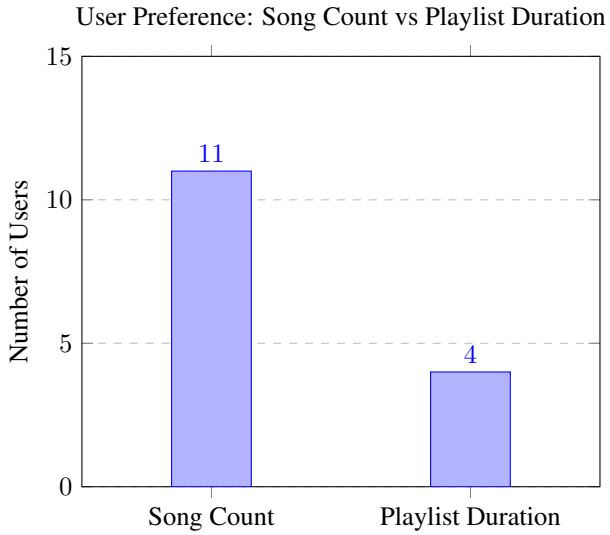

Figure 19: User preference for selecting song count vs playlist duration (n=15)

Table 1: Optimal BPM Per Workout Intensity[8] [1]

Workout Intensity	**BPM Range for a 20 year old**
Resting	100-120
Light	120-140
Moderate	140-160
High	160-180
Maximum	180-200+

5 Conclusion

The BPM Playlist Generator project showed how a specific, user-centered web application can create a positive impact on the music streaming experience. With the ability to create playlists within specific BPM ranges, we provided a utility that brings additional personalization and an activity-based approach not available in existing services (e.g. Spotify). Our key contributions include:

- A smooth, intuitive user interface aligned with HCI principles such as clarity, feedback, and accessibility.
- Dynamic playlist generation using BPM, energy, and danceability as core parameters.
- A functional prototype deployed through Flask and Heroku, with real-time Spotify API integration.

Usability testing has revealed user satisfaction was high, performance was fast, and generated playlists were highly relevant. Our choices around interface design, specifically prioritizing song count selection, contributed to a more coherent and predictable experience for the user.

5.1 Future Improvements

- Introduce genre filtering to refine playlist selections.
- Enable playlist saving and sharing directly from the application.
- Incorporate machine learning for improved track recommendations.
- Implement into a smart watch app.

This project highlighted the need for effective interaction design, solid API connection, and agile development methods. We also confront the external limitations while ensuring usability that will guide us in future projects that integrate user-centered design and technical development.

References

[1] Audrey Bailey. What to know about heart rate zones, 2025. Accessed: 2025-02-02.

[2] CDC. Physical activity basics — adult guidelines, 2025. Accessed: 2025-02-02.

[3] Firebase Documentation. Firebase documentation, 2025. Accessed: 2025-02-02.

[4] Flask Documentation. Flask documentation, 2025. Accessed: 2025-02-02.

[5] Heroku Documentation. Heroku documentation, 2025. Accessed: 2025-02-02.

[6] ReccoBeats. Reccobeats api documentation, 2025. Accessed: 2025-02-02.

[7] Spotify for Developers. Spotify web api documentation, 2025. Accessed: 2025-02-02.

[8] UIHC. Target heart rate for exercise, 2025. Accessed: 2025-02-02.

A Appendix: Team Contributions and Materials

A.1 Team Member Contributions

Team Member	Contributions
Jack Kammerer	Full Stack: Flask backend development, Spotify OAuth & playlist duration logic, Firebase setup, Heroku deployment, UI-Design
Chloe Hoech	UI/UX design with Flask templates, user testing coordination, LaTeX report writing and formatting
Koehler Perez	RecoBeats API integration, Spotify playlist generation logic, API interaction for songs, data handling

Table 2: Team contributions breakdown

A.2 Submitted Materials

- Full source code (Flask application)
- Firebase project files
- Heroku deployment files
- Final project report (this document)