Our development experience with Tubify was a mix of smooth progress and unexpected challenges. Early on, we had to pivot from some of our original plans due to significant changes in Spotify's API, the API we had initially intended to use for much of our core functionality. This shift forced us to create alternative solutions to maintain key features, but ultimately, the process of implementing these new pipelines and researching alternatives significantly expanded our technical knowledge and skill sets.

One major area of growth was frontend development, particularly working within a framework like React. We improved in everything from page layout and styling to optimizing the structure of the app for better load times and responsiveness. React's native and third-party component libraries were crucial, they allowed us to quickly add robust functionality without spending unnecessary time building common components from scratch. As we developed the frontend, we also learned best practices for managing state, designing efficient data flows, and handling asynchronous data fetching to create a smooth and powerful user experience.
In managing both frontend and backend systems together, we gained considerable full-stack development experience. Given the large scope of our project, integrating Spotify and YouTube services, handling user authentication, managing playlists, and delivering recommendations, we needed a well-organized project structure. We learned the importance of modular design principles, such as breaking the codebase into smaller, manageable components and services, to keep our system scalable and maintainable. This modular approach helped us coordinate the complex interactions between multiple APIs, databases, and client-side views.

Another major area of growth was API integration. We worked extensively with the Spotify and YouTube APIs, along with supplementary integrations like OAuth authentication via Google and GitHub, and the Genius API for lyric data. One standout skill we developed was the ability to deeply consult and interpret API documentation, allowing us to align our feature development more closely with what the APIs actually supported. This helped avoid wasted effort and guided our system architecture to fit available resources.

In cases where APIs did not provide everything we needed, such as Spotify's API changes removing key song metadata, we had to innovate. We learned to create workarounds by designing alternative pipelines for collecting and processing data. For instance, we developed a custom search solution for fetching YouTube videos, after discovering that the YouTube Data API's quota limitations (roughly 50 songs per day) would not scale for our users' libraries. This required balancing functionality against resource constraints, often sacrificing real-time performance in favor of broader data coverage.

Throughout development, the growing size and complexity of Tubify highlighted the critical importance of effective communication within the team. One major takeaway was the need to communicate incremental progress during sprint cycles, rather than strictly relying on initial sprint plans. Regular check-ins and progress updates allowed us to adapt quickly when tasks took longer than expected or when blockers emerged. We also improved at seeking help from one another when stuck on tough technical challenges, enabling faster and more collaborative problem-solving.

Another important lesson was the necessity of flexibility and resourcefulness when external resources (like APIs) fell short. Developing creative alternatives, even at the cost of increased complexity or resource demands, became a recurring theme, especially when scaling features like our recommender system under real-world API rate limits.

Ultimately, we succeeded in achieving our Minimum Viable Product (MVP). Tubify allows users to import and compare their liked songs libraries with friends, view personalized listening analytics, import and explore playlists, and watch music videos associated with their playlists. Although we had to scope down certain features, such as playlist generation tools based on listening history, the groundwork for these future enhancements was laid. Beyond the MVP, we successfully delivered additional features like our recommendation services and significantly polished the frontend design and styling to create a complete and user-friendly experience.