iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80016

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Aura Music: A Scalable Personalized Music Streaming System Using Time-Weighted Recommendation and Hybrid Filtering

T Deva Harsha Vardhan[1], M Bharat Srivastava[2], K. Harshini[3], Dr. D Sreenivasulu[4]
[1, 2, 3]*Department of CSE (Data Science) Institute of Aeronautical Engineering, Hyderabad, India*
[4]*Assistant Professor, Department of CSE (Data Science) Institute of Aeronautical Engineering, Hyderabad, India*

Abstract: ***The recent explosion in the online music consump- tion has resulted in a requirement of smart systems that are able to deliver users highly personalized content. The classic methods of recommendations tend to utilize fixed popularity values or a small amount of user (listener) preferences, which do not match the dynamic listening behavior. Paper presents scalable and intelligent streaming of music using Aura Music as an example, which is a hybridized recommendation framework based on collaborative filtering, behavioral analytics, and time-weighted scoring. It is developed on Android MVVM architecture-based client that will guarantee modularity and lifecycle-conscious data management, and the backend is built on Flask ensuring that it offers RESTful user request processing. Firebase Firestore constantly logs and processes user interactions (song plays, likes and the history of their listening) in real time. Time-weighted scoring system favours the recent user activity which provides context-sensitive recommendations. There is also the inclusion of diversity filtering layer to avoid redundant suggestions and further user exploration. Listening habits like artist distribution, time of day usage patterns and most popular songs are also incorporated into the system and give analytical feedback to the users. Caching strategies and effective API communication attain performance optimization. There is better recommendation accuracy, latency, and user engagement as shown by experimental evaluation. The proposed system emphasizes the usefulness of using a combination of temporal dynamics and hybrid filtering in the contemporary music recommendation system.***

Index Terms: ***Music Recommendation System, Collaborative Filtering, Time-Weighted Scoring, Personalization, Behavioral Analytics, MVVM Architecture, Firebase Firestore, Flask REST API, Music Streaming, Diversity Filtering***

I. INTRODUCTION

With the fast evolution of the digital technologies and free access to the internet, the consumption pattern of the users music has been changed greatly. Contemporary music streaming services have a huge array of songs, and it is difficult to find something that matches the preferences of the users. Consequently, user-based recommendation systems have developed into an important factor in the improvement of user engagement and experience [1]. Conventional methods of recommendations basically depend on popularity-based or fixed filtering rules that are not capable of reflecting dynamic preferences of users and situational behavior. Such systems fail to respond efficiently to the changing user interests, as time goes by resulting to redundant or irrelevant recommendations [2]. In addition, most recommendation models are not context- aware because of the temporal blindness of the models [3]. Recent studies have indicated collaborative filtering and content-based methods used to overcome these challenges. The collaborative filtering uses the patterns of interaction between user and items to predict their preferences whereas the content-based filtering uses the item attributes which include genre, artist and album details [4], [6]. Nonetheless, the methods have male disadvantages in the form of cold-start attacks, data scarce, over-specialization as well as absence of variation in the recommendations [5]. In order to address these drawbacks, the hybrid recommendation systems have been suggested as a combination of several methods to enhance the accuracy and personalization [7]. Also, the use of temporal dynamics in recommendation models has proven to be greatly effective in the processing of changing user behavior and provision of context-sensitive recommendations [8]. Strategies of recommendation based on diversity also improve the user experience by providing differentiated and new one in addition to minimising repetition [9]. Besides recommendation methods, the modern music streaming systems need highly scaled and high-performance designs to process extensive amount of user data and provide real-time responsiveness. Firebase Firestore is a proper cloud- based database that allows user interaction data to be stored, synchronized and accessed effectively and is well integrated in a dynamic recommendation set up [4]. Likewise, RESTful backend architectures

facilitate smooth intercourse among client and server, which implies modularity, scalability, as well as sustainability of the system.

In that regard, this paper introduces Aura Music, a large scale and smart music streaming platform that incorporates hybrid recommendation methods, such as collaborative filter- ing, time-adjusted scoring, and diversity filtering. The system is built in accordance with the modern MVVM-based An- droid platform and Flask-based backend to allow the efficient process of data and provide real-time personalization. The interactions with users, including plays and likes of songs perpetually receive a recording in Firebase Firestore and are processed to produce adaptive suggestions in a feedback-based process.

The primary contributions of the work are a hybrid rec- ommendation engine brought about by the combination of collaborative and time-weighted scoring to enhance the accu- racy of personalization [7], [8]. The system also incorporates behavioral analytics to capture the preferences of users and dynamically adjust them to the pattern of interaction [2]. Moreover, a diversity cognizant recommendation system is employed to improve content discovery and curb redundant recommendations [9]. A scalable cloud-based implementation based on Firebase Firestore and RESTful APIs to handle data and keep it in real time is also proposed in the system [4]. Moreover, an Android app developed based on the modular design is developed using the MVVM architecture to guar- antee separation of concerns, maintainability, and lifecycle- conscious data management. Lastly, the mechanism of contin- uous feedback loop is presented to improve over time on the recommendation introduced by users based on the behavior of the user so that the system will dynamically adapt to changing preference by the user [3].

In general, the suggested system utilizes the limitations of the current music recommendation systems through a combination of advanced music recognition algorithms and an overall scalable and efficient architecture to enhance the quality of recommendations, the performance of the system, and the interface between users.

II. RELATED WORK

Recommendation algorithms have recently become an in- separable part of any digital service and is especially relevant to music streaming websites where individuals are shown volumes of content bigger than handbooks. The main goal of these systems is to help user get relevant and more personalized content depending on their tastes and actions. Several methods of recommendation have been devised over the years such as collaborative-filtering, content-based-filtering and hybrid methods, and time based models. All these ap- proaches focus on certain parameters of the problem of the recommendations, yet each of them introduces some limi- tations. The section will examine the available literature on recommendation systems and their methodology, strengths and weaknesses and will discuss the necessity to develop an integrated approach, which will unite various techniques and enhance performance.

A. Collaborative Filtering Techniques

The effect of collaborative filtering (CF) is among the tech- niques that has been implemented most in the recommended system where the preferences of the users are forecasted by past interaction history. It is founded on the idea that people who shared similar issue in the past will probably share similar issues in the future. The Netflix Prize contest has shown how well collaborative filtering can be used to enhance the accuracy of recommendations when it utilizes massive amounts of user- item interaction data [1].

The CF techniques can broadly be categorized as an item- based or user-based technique. User-based CF finds similar users and suggests items that similar users liked and item- based recommends items similar to those that the user has interacted with in the past [2]. The techniques work well in uncovering concealed trends in the user behavior and creating individual suggestions.

Nonetheless, collaborative filtering is plagued with various drawbacks such as sparsity of data whereby the interaction matrix is not full courtesy of the limited activity of the users. Also, the cold-start problem occurs when there is a dearth of information available to make recommendations on new users or items [5]. Such constraints limit the applicability of CF techniques into practice.

B. Kwicowski Method of Content-Based Filtering

The content-based filtering gives attention to the characteris- tics of items like genre, artist, album, and metadata in order to suggest to the users similar items. This can be contrasted with a collaborative filtering method that utilizes the data of other users so it can be used in situations where the data of user interaction is minimal [6]. The system creates a user profile that is based on the past items that were consumed and will offer content with similar characteristics.

Among the greatest benefits of the content-based filtering is that it can give a recommendation of additional items without user interaction information. Nonetheless, this strategy usually encourages over-specialization of users who are constantly of- fered related kinds of material and do not have the opportunity to explore and discover new content. Therefore, users might not be exposed to different or new information which can have a negative effect on the user engagement.

C. Hybrid Recommendation Systems

The hybrid recommendation systems are multi-method sys- tems which combine several recommendation methods so as to eliminate the weakness of each of them. Burke [7] suggested a number of hybrid systems which combine collaborative filters and content based approaches showing higher quality and strength. Such systems will use various data sources, including user interactions and features in items, to produce more trustworthy recommendations.

There are several ways of deploying hybrid strategies by strategies such as weighted combinations, switching, and feature augmentation. Topics like cold-start can be mitigated by using the benefits of other methods, and the problem of sparse data can be minimized through hybrid systems. Nevertheless, most of the available hybrid systems do not remain dynamically flexible to the changing preferences of the users, particularly in real-time applications.

D. Temporal Dynamics of Recommendation Systems

Conventional recommendation systems are known to as- sume the preferences of the users as symmetric, failing to consider that the interests of users change as time goes by. In order to overcome this drawback, recommendation models have been provided with temporal dynamics. Koren [3], [8] proposed time conscious collaborative filtering algorithms in which the weight of user interactions is given varying weights depending on their recentness.

Time-based models also enhance accuracy of the recom- mendation as it has more weight on the behavior of the user recently, thus it becomes more receptive to the preferences in the system. As an illustration, the current listening behaviors of a user can vary greatly compared to the previous one and the introduction of temporal data can assist in recording the change.

Although their benefits are evident, time-conscious models add new computation complexity and need effective data processing systems to be used within real-time systems. One of the challenges in applying these models in a broad scale is still an issue in most applications.

E. Diversity-Aware Recommendation

Besides the accuracy, diversity is also an important aspect in the recommendation systems. Systems that aim at just being accurate only generate similar recommendations and this lowers the level of user satisfaction. Diversity-conscious recommendation methods seek to bring variety and novelty in recommendation list. As Castells et al. [9] upon it, accuracy is essential, and the concept of diversity can enhance user experience.

These strategies make sure that suggested items do not have too many similarities so that the user can have a greater variety of content. Such techniques like re-ranking and diversification algorithms are widely employed in order to balance this. Nevertheless, when diversity in the form of extra recommendations is increased, this can occasionally impair the accuracy of recommendations, hence, it can be difficult to strike an optimum point.

F. Architecture Scalable to Recommendation Systems

As the user base of the systems and the volume of informa- tion grows, modern recommendation systems need scalable architectures that will be able to effectively handle massive information. Cloud services and non-relation databases, in- cluding Firebase Firestore, offer the benefit of storing, syn- chronizing, and accessing data in real-time, which is why they are appropriate when a large-scale application is to be developed [4].

RESTful backend designs allow the client applications and servers to communicate efficiently, as well as facilitate the modular design and scalability. Also, it uses caching agent and distributed computing technologies to decrease latency and system performance.

In spite of these developments, scalable infrastructure and sophisticated algorithms of recommendations is a challenging issue when it comes to systems that must execute a continuous flow of information and modify systems in real-time.

G. Research Gap

Despite the example of tremendous advancements in the field of recommendation systems, the current solutions tend to shift the focus on either of the individual factors, including accuracy, diversity or scalability. Lack of unified systems that work well in making hybrid recommendation techniques, temporal dynamics, diversity filtering, and scalability building in one framework does not exist. The proposed system will bridge this gap by integrating collaborative filtering, time-weighted scoring, and diversity-conscious systems into a scalable cloud-based system. Com- bining these elements, the system offers better quality recom- mendations, flexibility and interaction with users, which solves the weaknesses of the current styles.

III. SYSTEM ARCHITECTURE

Fig 1 depicts the overall system architecture of the suggested Aura Music platform. The system adheres to the multi- layered design comprising of four main layers namely; the Presentation Layer, Application Layer, Data Layer and the External Services Layer. The layered design is designed in a manner that it is modular, scalable and the separation of concerns is achieved very well and thus the system can effectively process real-time data and generate recommendations that are tailored.

Android application is the Presentation Layer (Client) and it is the user interface of the system. It has modules like search, discovery, daily mix creation and playback of music. It is written in Kotlin and with a Model-View-ViewModel (MVVM) architecture with the ViewModel containing UI state and communicating with the backend services via an API client. It happens through this design, which offers reactive data processing and is lifecycle aware of state management. The communication between the client and the backend is done using secure HTTPS connections using authentication tokens.

Application Layer (Flask API) is an essential processing unit of the system. It is comprised of API controllers that process the requests of authentication, search, playlist manage- ment, and streaming. The fundamental services in this layer are the Recommendation Engine, Daily Mix Generator, User

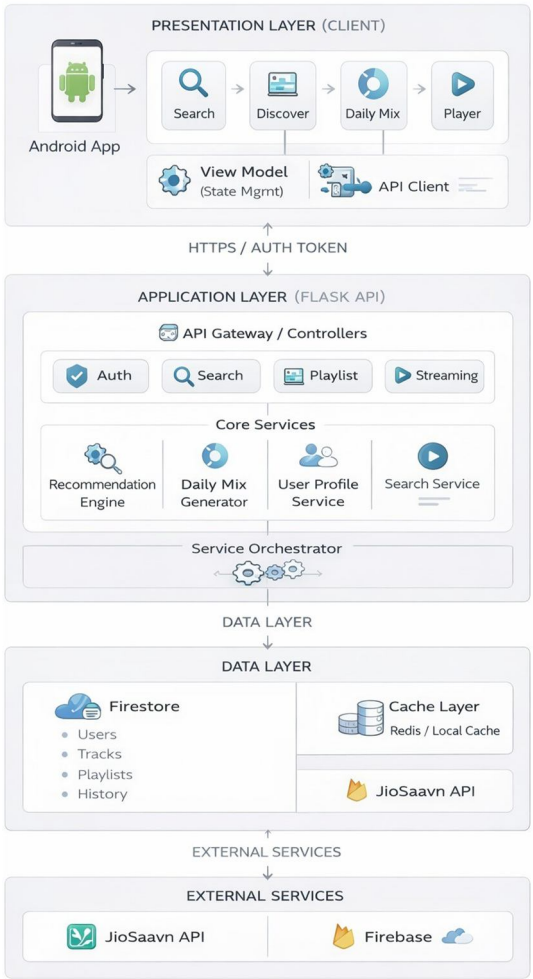


Fig. 1. Layered System Architecture of Aura Music Platform

Profile Service and Search Service. A service orchestrator coordinates such services and is responsible for data flow as well as efficient execution of business logic. The backend receives user requests and provides recommendations and communicates with the data layer and the external services.

The Data Layer will manage data and store it regard- ing application data. The main NoSQL database is Firebase Firestore, which contains the users, tracks, playlists, and the listening history among other collections. Firestore has real- time data synchronization data and efficient querying which is appropriate with dynamic recommendations systems. In addition, a caching layer of some sort, built on Redis or local cache, will be used to hold the common and popular tracks and recent searches, and thus, it will enhance performance and lessen the latency.

External Services Layer makes integration of third party services needed by the system. The metadata of music, stream- ing URL, and playlist information of music are retrieved using the JioSaavn API. Firebase services are authentication, authorization, and synchronization of data in real-time. This layer will be used to keep the seamless integration with external platforms at the same time keeping systems scalable, and flexible.

On the whole, the architecture presented in Fig 1 reflects the efficient system design that is scalable and supports the ability to generate recommendations in real-time, safe communication and smooth interaction with the user.

IV. METHODOLOGY

The proposed system adheres to a data-driven and adap- tive approach to the recommendation of personalized music based on the analysis of interactions between a user and the use of hybrid methods of recommendations. The architecture is created to facilitate the ongoing learning process, real- time information processing, and scalable recommendations. With behavioural analytics, temporal dynamics and diversity- conscious filter the system makes the recommendations rele- vant to a user, personalised and dynamically updating as time goes.

A. User Interaction Collection

Song plays, likes, dislikes and listening history are some of the examples of the information that is continuously gathered by the system when a user interacts with it. Every interaction is logged with metadata, such as timestamps, user identifiers and track details and saved in Firebase Firestore.

This well- organized form of data collection allows the system to create detailed user profiles and conduct temporal analysis of the listening behavior. The gathered information is the main input of the recommending engine and is essential in comprehending the preferences of the users.

B. Processing of Requests and API Management

Android application user requests are sent to the backend via RESTful APIs via secure HTTPS connections with the back- end. The API gateway authenticates the tokens and handles the incoming requests. Depending on the type of request e.g. search, recommendation or a streaming request, the request gets forwarded to the service module as necessary. The back- end provides effective support of the concurrent processing of requests and provides error control and validation procedures to guarantee efficiency and stability of the system.

C. Processing of Recommendation Engine

The recommendation engine works with an interaction data of users in a hybrid system, which is a combination of col- laborative and content-based filtering approach. Collaborative filtering finds similarities between users and items based on their interaction history, whereas content-based filtering mea- sures the item characteristics (e.g. genre, artist, albums and so on). Through the combination of these techniques, the system produces a system of candidate recommendations, which are comprehensive and based on similarity of the behavior of the user and content. This combination technique enhances accuracy and minimizes the weaknesses of other techniques.

D. Time-Weighted Scoring

A time-weighted scoring system is used to sort out the recent user interactions at the expense of older interaction. The method gives greater weight to the new activities and this will enable the system to focus more on the user preferences in the present moment. The scoring mechanism also picks and changes dynamically according to interaction timestamps so that obsolete preferences have less impact on recommendations. This time modeling improves the capability of the sys- tem to give context-sensitive and adaptive recommendations.

E. Generation and Diversity of Candidates

The system creates a list of possible songs across various sources such as related artists, offshoot genres, user originated patterns of listening, and what is popular around the world. In order to enhance the quality of the recommendations, the candidate list is subjected to diversity filtering layer. This layer guarantees that recommended songs are not too similar to each other thus removing redundancy and creating back content variety. The system also balances accuracy and diversity to offer more rich and engaging user experience.

F. Recommendation Delivery

The last list of suggested music is presented to the customer via the Android application interface. The system guarantees a low-latency response with the use of caching mechanism, as well as, optimized communication through the API. The recommendations are updated dynamically in real time and updated according to the interactions of users; as such, the system can continuously change itself according to user in- teractions. Lastly, the delivery mechanism will be oriented in terms of delivering a smooth and responsive user experience.

G. Feedback Loop Mechanism

The system also has a feedback loop where the interaction of the user with the recommended content is determined and examined. The user profiles are updated using this feedback and future recommendations improved. The feedback loop allows the system to learn about the behavior of the user and enhance the accuracy of recommendations in the process. This adjustive learning process will make sure that system adapts to the preference of the user as it changes.

H. Workflow Overview

The general sequence of the system is that a user inter- acts with the system through either Android application, by searching the song, or just by listening to a song or liking content. Those interactions are sent to the backend by secure RESTful APIs. The backend then requests are processed by relevant modules of service and the necessary data is accessed on Firebase Firestore and external sources like the JioSaavn API.

The recommendation engine processes user data and uses hybrid filtering strategies and time weighted score in order to produce candidate recommendations. A diversity filter mech- anism is then put in the recommendational list so that it is varied. The completed recommendations are given back to the client and presented to the user on the application interface.

The data generated is also stored in the database as the users interact with the suggested data being used to improve future recommendations. This provides a continuous feedback system which allows real time learning and adaptation of the system. The workflow is used to guarantee efficient personalized recommendations and scalability of data processing.

V. IMPLEMENTATION

The Aura Music system proposed is implemented by means of the mobile client, server services, cloud database and the external APIs that provide a scalable and customized music streaming platform. The system architecture is modular and distributed in design to support real-time processing of data, and an efficient communication as well as generation of recommendations that are dynamic. Its implementation aims at optimization of performance, scalability and smooth user experience.

A. Android Application and Playback System

The frontend is designed as an Android app in Kotlin in accordance with the Model-View-ViewModel (MVVM) design to be separate of concern and maintainable. The application offers easy-to-use interfaces to such functions like search, personalized recommendations, daily mix playlists, and music playbacks.

ViewModel layer also takes care of state of UI and commu- nicates with backend services via an API client that ensures components have efficient data flow. The reactive program- ming languages like StateFlow, deal with asynchronous up- dates of data and allow single-user interfaces to display real- time data in real-time coupled to the backend responses.

The application also uses ExoPlayer in the media play- back which supports high quality streaming, buffering and background playback. One of the services offers the player the ability to play music even when the application has been minimized which adds to the user experience. There is also optimization of rendering performance and less memory con- sumption by the use of efficient UI elements like RecyclerView and lazy loading.

B. Backend Services and Core Functional Modules

Flask framework is used to build the backend offering lightweight and scalable services of API in form of REST. It is the processing unit of the entire system and it receives the requests of clients, processes the business logic and controls the interaction with the database and external services.

The API layer has authentication endpoints, search op- erations endpoints, recommendation generation endpoints, playlist management endpoints, and streaming services. These dead ends will be engineered to support simultaneous request processing speedily and with reduced latency.

This system comprises a variety of core services amongst them the Recommendation Service, Daily Mix Service, User Profile Service, Search Service, and the Stream Service. The Recommendation Service applies hybrid filtering, as well as time-weighted scoring to make individual suggestions. The Daily Mix Service works as a creation of playlists regarding the listening preferences and habits of the users. The User Profile Service contains the data of interaction with users and model of preferences, whereas the Search and the Stream Services perform the work of processing queries and streaming of music respectively.

A mechanism of coordinating these services is service orchestration that has been able to provide seamless flow and effective implementation of activities. There is also logging and monitoring available in the backend to monitor perfor- mance of the system and detecting any possible problems.

C. Database, Caching and External Integration

Firebase Firestore is taken as the main NoSQL database to keep application data, such as user account information, tracks, playlists, recommendations, and listening history in it. Firestore is structured in such a way that it allows real time data synchronization of user information and suggestions. The database is created in such a way that it can accommodate querying and indexing, so that data in them are retrieved quickly.

Redis or local in-memory storage is used to implement a caching mechanism to improve performance. Data that are regularly viewed like popular tracks, songs that were created recently and results of suggestions are kept in the cache to lessen the burden on the database and shorten the response period. The concept of cache invalidation is employed to achieve consistency of data.

The system will be connected with the outside services like API of JioSaavn that will provide music metadata, album in- formation, and URLs to stream. This enables the application to have access to a big and dynamic music library without having local storage. Firebase is employed to do user-authentication, user-registration, and user-session management, and it guaran- tees user access to the system is secured.

D. Security, Performance Optimization and Deployment

The system has several security controls that guarantee the protection of data and safe communication. Any API calls are made using HTTPS, and this process cannot be intercepted and abused by people. Firebase Authentication and JSON Web Tokens (JWT) are used to handle authentication, and it is a secure and efficient means to manage user session. Input validation and request filtering codes are used to avoid malicious attacks like screenwriting and unauthorized access. Firestore security rules also provide a way of limiting data access to users according to their roles and permission.

Caching, efficient API design, and asynchronous process- ing are the means of performance optimization. As network overhead and better response time are sought, request batching and lazy loading techniques are employed. The system itself is optimized to accommodate a large user traffic by eliminating the unnecessary computations and the optimalizing of the database queries.

The backend is implemented within a cloud platform, and it provides horizontal scaling and high availability. Dynamic scaling of the system can be based on user demand so that the system will perform equally with different levels of workload. The modular architecture enables the system to be flexible and easily integrated to add new features and services that make it future ready.

VI. RESULTS AND DISCUSSION

The functionality and test of the proposed Aura Music sys- tem were tested in real time with the Android application and Android backend services. The system is shown to be efficient in data processing, dynamic generation of recommendations, and smooth interaction between the users. The findings that were achieved on the various modules of the system are discussed below.

A. API Performance and Backend Processing

The backend API logs during the execution of the system are provided in Fig 2. The logs show effective processing of API calls including /api/play and /api/health, response time, and status codes. The system exhibits low latency within the processing of

requests and responses normally take mil- liseconds to respond. The logs have also ensured that there has been an effective integration with Firebase Firestore and external APIs, which has guaranteed effective data retrieval and storage. This justifies the effectiveness of the Flask-based backend to support parallel requests and process data in real- time.

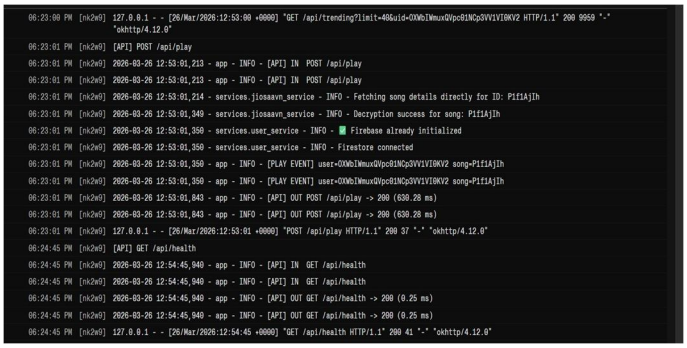


Fig. 2. Backend Processing

B. The audio playback and equalizer functionality

Fig 3 shows the audio equalizer interface installed into the music player. Various frequency controls (e.g., 60Hz, 230Hz, 910Hz, 3kHz and 14kHz) as well as preset modes (Normal, Classical and Dance) are offered by the equalizer. Also, the presence of a bass boost option will improve the user experience by giving them an option to tune the audio. This continuity of ExoPlayer also makes it possible to optimally play with a full-time adjustment of the playback; it can be easily seen that the ExoPlayer has the potential to provide customers with an enhanced audio customization experience.

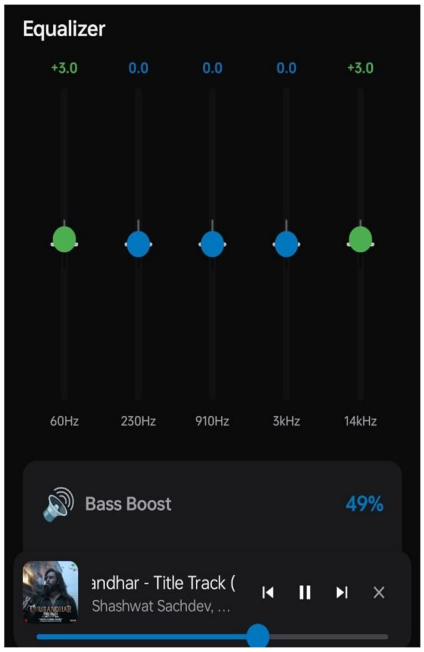


Fig. 3. audio playback and equalizer functionality

C. Music Interface and Playback Control

Fig 4 shows the interface of the music player where album art work is displayed, playback, progress, and volume control are present. Basic functions like play, pause, skip, and seek are also supported via the interface as well as updates on the real-time progress. The system guarantees continuous play out and effective buffering even when in the background. This will show the strength of media playback execution and the exhibited capability to provide a seamless user experience.

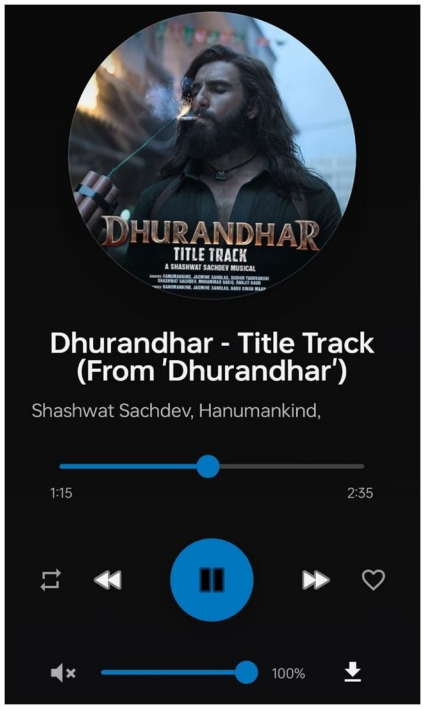


Fig. 4. Music Interface and Playback Control

D. Mood-Based Recommendation Interface

Fig 5 presents the mood selection interface, in which a user is allowed to select the following categories: Chill, Workout, Party, Focus, Romantic and Sleep. This capability is more personalized since it gives users the ability to expressly specify their preferences in listening depending on the situation or mood. The chosen mood is then employed as an added input to the recommendation engine enhancing both relevance and context accuracy of the recommended songs.

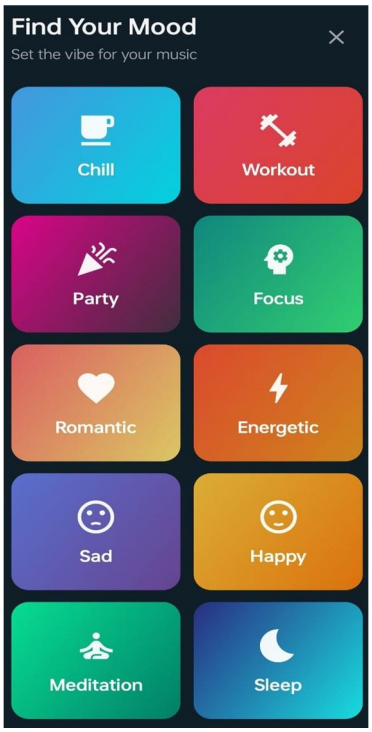


Fig. 5. Mood-Based Recommendation

E. Listening Insights and User Analytics

Fig 6 and Fig 7 give the listening insights module thus giving a detailed analytics of user behavior. The system will log the metrics of the total plays, top songs, time of day listening, and distribution of artists. This data is illustrated in charts and graphs allowing the user to determine the kind of listener he/she is. Also, the recommendation engine uses this information to personalize and enhance the accuracy of the recommendation.

F. Discussion

These findings show that the developed system is a suc- cessful combination of frontend, backend, and data modules that can bridge and bring scalable and personalized music streaming experience. The hybrid recommendation method along with the use of time-weighted scoring and diversity filtering techniques effective result in relevant and diverse recommendations.

The backend system demonstrates the high level of efficient performance; the low latency, and stable API responses, the Android application offers the responsive and user-friendly interface. Mood-based selection, listening insights, and audio equalizations are some of the features that enable better user engagement and customization.

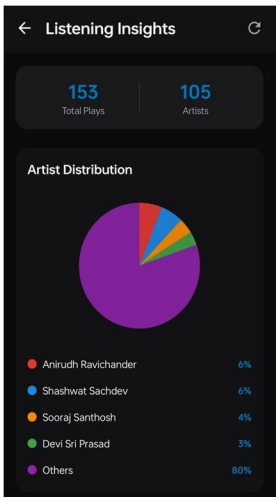


Fig. 6. Listening Insights

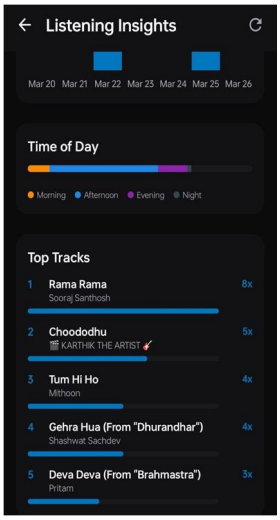


Fig. 7. User Analytics

All in all, the system meets its aims of enhancing the accuracy of its recommendations, real-time adaptability, and provision of smooth user experience. The system is compatible with the current music streaming applications due to the integration of advanced features and scalable architecture.

VII. CONCLUSION AND FUTURE WORK

This paper also introduced Aura Music, a scalable and smart music streaming system that combines hybrid recommendation methods with the current mobile and cloud-architecture. The system integrates collaboration filtering, content-based filter- ing, time-weighted scoring, and diversity-sensitive systems to produce customized and context-sensitive music suggestions. The system implementation on the basis of an Android application (MVVM) and a Flask-based Backend and Firebase Firestore could provide an effective data processing, real-time synchronization, and user-friendly interaction. The outcomes of the experiments indicate that the system is able to involve itself in the habit of the user by indicating correct and varied suggestions to the user as a result of continuous feedback.

Other ancillary functions like mood-based recommendation, listening insights and audio equalization, however, add to the user engagement and personalization. Low latency and reliable performance also characterise the system making it possible to implement the system in the real world. On the whole, the suggested system is efficient in overcoming the drawbacks of the traditional recommendation systems, combining the advanced methods with a rough-scaled system and a non- sluggish architecture.

A. Future Work

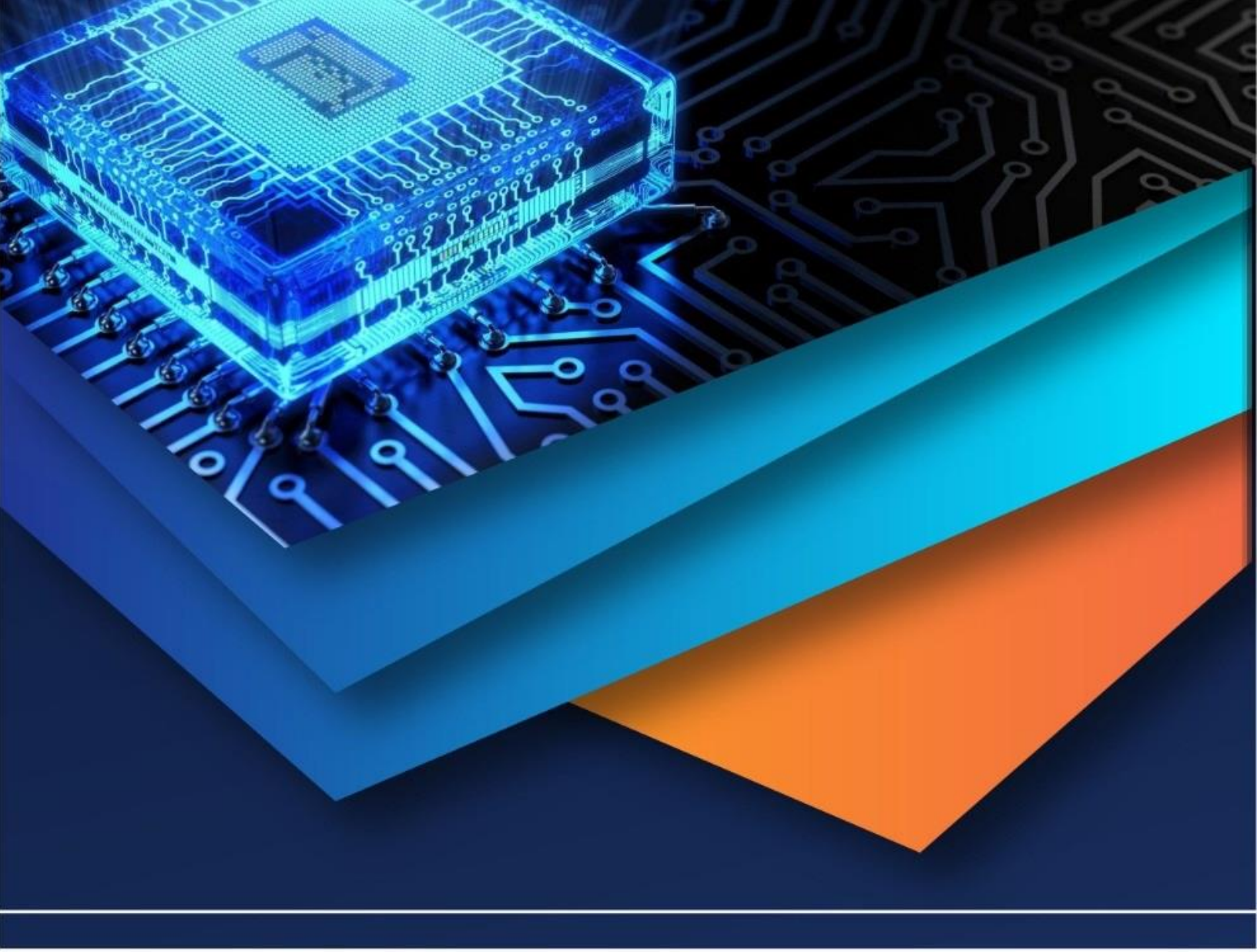
Despite the fact that the proposed framework achieved serious advances in how accurate the recommendations are and how easily they may be used, there are a number of improvements that may be addressed in future studies. Sophisticated machine learning algorithms, which include the neural collaborative filtering and deep learning-based recommendation models, may be integrated to enhance the accuracy of personalization further.

It is possible to advance the context-aware recommendations by incorporating other aspects like user location, activity, and pattern of device use. The system can also be furthered to other platforms like cross platform application, such as web and iOS platform in order to reach a larger population of users. To further minimize time latency and enhance performance, it can implement real-time streaming optimization technique and edge computing. Also, the integration of social properties, including sharing of the user, playlists, and recommendation based on the community, can contribute to the engagement of the user.

Advanced analytics dashboards, voice interaction, and wear- able integration might also improve the experience in the future and deliver a more immersive and intelligent experience of the music streaming process. Such improvements will also increase the ability of the system to provide highly adaptive and personalized recommendations.

REFERENCES

[1] Y. Koren, R. Bell and C. Volinsky, Matrix Factorization Techniques recommender systems, IEEE Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009.
[2] J. L. Herlocker, J. A. Konstan, A. Borchers and J. Riedl, An Algorithmic Framework to Perform Collaborative Filtering, Proceedings of the 22 nd Annual International ACM SIGIR Conference, pp.230237,1999.
[3] Y. Koren, Collaborative Filtering with Temporal Dynamics Communi- cations of the ACM, vol. 53, no. 4, pp. 8997, Apr. 2010.
[4] Firebase, Cloud Firestore Documentation, [Online]. Available: https://firebase.google.com/docs/firestore
[5] G. Adomavicius and A. Tuzhilin, 2005, Towards the next generation of recommender systems: a survey of the state-of-the-art system and poten- tial extensions, IEEE transactions on knowledge and data engineering, vol. 17, no. 6, 734-749.
[6] M. J. Pazzani and D. Billsus, Content-Based Recommendation Systems, in The Adaptive Web, Springer, 2007, pp. 325341.
[7] R. Burke, Hybrid Recommender Systems: Survey and Experiments, User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331370, 2002.
[8] X. Su and T. M. Khoshgoftaar, A Survey of Collaborative Filtering Techniques, Advanced of Artificial Intelligence, vol. 2009, Article ID 421425, 2009.
[9] P. Castells, N. J. Hurley and S. Vargas, Novelty and Diversity in Recommender Systems, in Recommender Systems Handbook, Springer, 2011, p. 881918.
[10] S. Rendle, "Factorization Machines," Proc. IEEE International Confer- ence on Data Mining (ICDM), pp. 995–1000, 2010.
[11] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative Deep Learning for Recommender Systems," Proc. ACM SIGKDD, pp. 1235–1244, 2015.
[12] A. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," Proc. ACM RecSys, pp. 191–198, 2016.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY