



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85308>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

StreamHub - Multi - Agent Framework for Movie Recommendation System Using Langchain

Vanitha Mani K.¹, Vidhyavarshini K. K. H.², Vaishnavi N.G.³

¹Assistant Professor/CSE, K.L.N. College of Engineering, Pottapalayam, Sivagangai, Tamil Nadu, India

^{2,3}UG Scholars, CSE, K.L.N. College of Engineering, Pottapalayam, Sivagangai, Tamil Nadu, India

Abstract: *The rapid growth of digital entertainment platforms has made it difficult for users to discover suitable movies and music from a large collection of content. Conventional platforms commonly depend on separate search interfaces and recommendation features, which may not provide a unified or context-aware experience. This paper presents StreamHub, an OTT and Music Multi-Agent Recommendation System that combines movie discovery, music discovery, personalized recommendations, user interaction, and premium-content access in a single web application. A Supervisor Agent coordinates user requests and directs them to specialized agent functions. The Movie/Content Agent integrates the TMDB API to retrieve available movie and television information, while the Music Agent connects to an external royalty-free music API to obtain available tracks and related metadata. User preferences, watch history, ratings, and watchlists can be maintained through a MySQL database to support more relevant content suggestions. The system uses React.js for the frontend and FastAPI, SQLAlchemy, and Pydantic for backend services. JWT-based authentication supports protected access, while the modular multi-agent design separates responsibilities and supports future extension. The platform demonstrates how AI-assisted interaction, external APIs, persistent user information, and coordinated agents can simplify entertainment discovery. The system is intended to improve usability and personalization by bringing movie and music exploration together through one interface.*

Keywords: *StreamHub, Multi-Agent System, OTT Platform, Music Discovery, Movie Recommendation, Artificial Intelligence, LangChain, TMDB API, FastAPI, React.js, MySQL, JWT Authentication.*

I. INTRODUCTION

The rapid growth of online entertainment has made a wide variety of movies, television shows, and music tracks available to users. Although this provides greater choice, it can also make it difficult to identify content that matches individual interests. Users often need to browse different categories, search several platforms, and compare content manually before finding something suitable. A unified intelligent platform can reduce this effort by bringing content discovery and recommendations together. [8]

Traditional entertainment applications commonly provide search bars, category filters, trending lists, and recommendations based on popularity or previous activity. Movie and music services may also operate separately, so users need to switch between applications to satisfy different entertainment needs. Such approaches can be less convenient when a user expresses a more specific request in natural language or expects recommendations that reflect personal interests and activity history.

Recent developments in Artificial Intelligence (AI), Large Language Models (LLMs), and agent-based architectures provide opportunities to improve digital content discovery. Recommendation systems can use information about users and content to suggest relevant items, while language models can help interpret conversational requests. Multi-agent systems divide a complex workflow into specialized responsibilities and coordinate those responsibilities through a supervisor or central routing component. [1], [2]

The proposed StreamHub system brings these capabilities into a single web-based platform. A Supervisor Agent interprets the user request and directs it to the relevant function. The Movie/Content Agent obtains movie information through the TMDB API, while the Music Agent retrieves available tracks through an external royalty-free music API. User profiles, watchlists, ratings, and activity records are maintained through the backend and database to support personalized experiences.

The platform also provides a web interface for users to sign in, explore entertainment content, and access available recommendation features. JWT-based authentication is used to protect supported application routes. The backend manages agent workflows, external API requests, database operations, and application logic, while the frontend presents the results through a common interface.

User activity is important for improving the relevance of entertainment suggestions. Information such as watched content, saved watchlists, ratings, and stated preferences can help the system identify interests. Persistent storage also allows user-related information to remain available between sessions, rather than requiring the user to repeat the same details every time.

The main objective of this work is to demonstrate a coordinated entertainment application rather than a collection of unrelated search pages. By combining movie discovery, music discovery, agent-based request routing, and user activity management, StreamHub provides a practical example of how AI and multi-agent technologies can support a more convenient and personalized entertainment experience. [5], [8]

II. PROBLEM STATEMENT

The increasing volume of digital entertainment content has made content discovery a common challenge for users. Movie platforms provide large collections of films and television shows, while music services maintain separate catalogs of tracks and artists. When users do not know the exact title they want, searching through multiple categories can take time and may not produce relevant results. [8]

Movie and music discovery are frequently separated across different services. Users may need one application to search for films and another to discover music. Existing search and recommendation features can be useful, but they may not coordinate different entertainment requests through a single conversational interface. Recommendations can also be less relevant when user preferences and previous activity are not considered together.

Another challenge is the complexity of managing different functions within one application. Content retrieval, user profiles, watchlists, recommendations, notifications, and payment-related workflows have different responsibilities. When these operations are not clearly separated, application logic can become difficult to maintain and extend.

Therefore, an integrated system is needed to provide movie and music discovery, route user requests to the appropriate functions, store supported user activities, and protect account-specific features. The system should use current information from external content services rather than relying entirely on manually hardcoded catalogs.

StreamHub addresses these requirements through a Supervisor Agent and specialized agent functions. It integrates the TMDB API for movie information, an external royalty-free music API for available tracks, a MySQL database for persistent information, and JWT-based authentication for protected access. The platform aims to make entertainment discovery more organized and convenient through one web application.

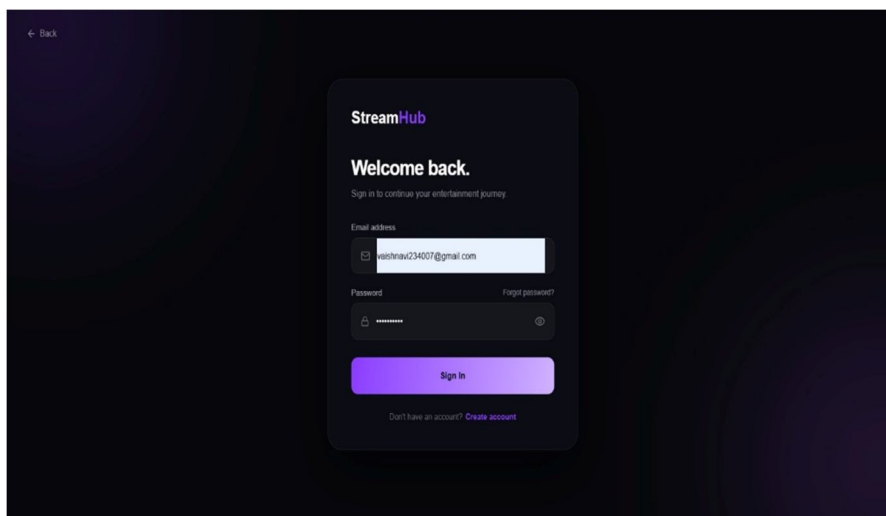


Fig 1 : StreamHub Login Page

III. PROPOSED SYSTEM

The proposed StreamHub — OTT and Music Multi-Agent Recommendation System is a web-based entertainment platform that combines movie discovery, music discovery, recommendations, user activity management, and premium-content workflows. It is designed to reduce the need to use separate interfaces for different entertainment requirements and to organize requests through a multi-agent architecture. [1]

When a user searches for a movie, the backend communicates with the TMDB API to retrieve available title information, descriptions, genres, release details, ratings, and poster information. For music-related requests, the Music Agent communicates with the configured external music API and displays the track information returned by that service. Results therefore depend on the available external API response rather than an assumed catalog.

The Supervisor Agent coordinates incoming requests and directs them to the relevant specialized functionality. Movie requests are handled by movie/content functions, music requests by music functions, and payment-related operations by the corresponding workflow. User preferences, watch history, ratings, and watchlists can support more relevant recommendations and help maintain continuity between sessions.

The application provides a user interface for accessing the main entertainment features. Users can sign in, search content, view details, explore available music, and access supported recommendations. The React frontend communicates with the FastAPI backend, which manages business logic, authentication, database access, agent coordination, and external API communication.

JWT-based authentication helps protect user-specific features and backend routes. The application can restrict protected operations to authenticated users and maintain user-related information through the database. Payment-related operations are organized as a separate workflow so that access to premium features can be controlled according to the implemented payment status.

The system stores supported user and application information in MySQL. Persistent records may include account information, preferences, watchlists, ratings, and activity data. This information supports account continuity and provides a basis for improving recommendation relevance as the application is extended.

The proposed system consists of user authentication, movie retrieval, music retrieval, Supervisor Agent coordination, recommendation functions, user activity management, payment-related workflows, and persistent storage. These components work together through a modular architecture to provide a unified entertainment experience and make future improvements easier to incorporate.

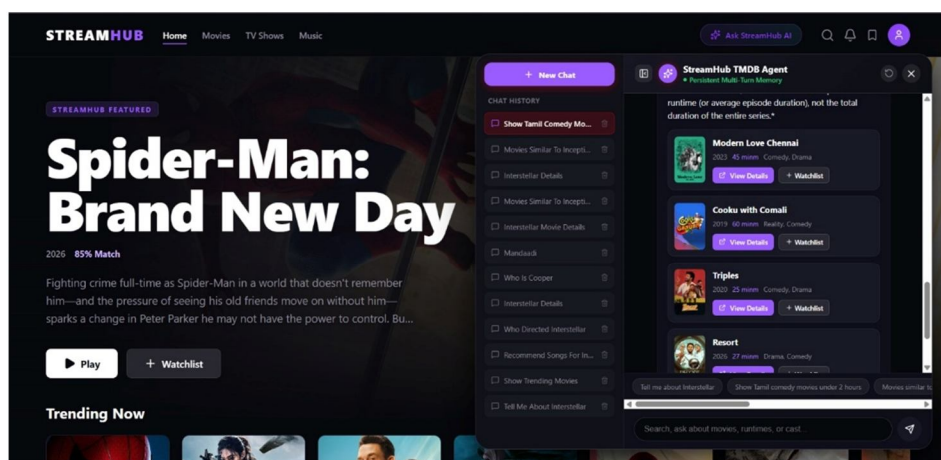


Fig 2: StreamHub User Dashboard

IV. MULTI-AGENT STRUCTURE

The StreamHub system uses a multi-agent structure to organize entertainment requests and separate the responsibilities of different application functions. The Supervisor Agent acts as the central coordinator, identifying the general type of user request and directing it to the appropriate specialized functionality. This structure helps keep request handling organized and allows modules to be updated independently.

The Supervisor Agent receives the user's request and selects the relevant workflow. It coordinates the available movie, music, recommendation, profile, notification, and payment-related functions according to the request. The supervisor does not replace each specialized module; instead, it helps coordinate how the modules are used within the application.

The Movie/Content Agent manages movie discovery and communicates with the TMDb API. It retrieves available movie and television information and returns results for display in the application. The Music Agent handles music-related requests and retrieves available royalty-free tracks and metadata through the configured external music API. The response depends on the content actually returned by the service.

The Profile and Recommendation functions manage supported user preferences and activity records. Watchlists, ratings, and watch history can be stored in MySQL and used to support more relevant content suggestions. The Notification function can support user-facing updates where implemented, while the Payment Agent organizes the workflow associated with premium content access.

The main responsibilities in the multi-agent structure can therefore be grouped as follows:

- 1) Supervisor Agent — Coordinates requests and routes them to the appropriate specialized function.
- 2) Movie/Content and Music Agents — Retrieve available movie and music information from external APIs and return suitable results.
- 3) Profile, Recommendation, Notification, and Payment Functions — Manage user activity, support personalized suggestions, provide relevant updates, and organize premium-access workflows.

This modular structure separates major responsibilities and supports maintainability. The use of JWT-based authentication and database-backed user information helps protect supported account features. The architecture also provides a basis for adding new entertainment services or specialized agents in future versions.

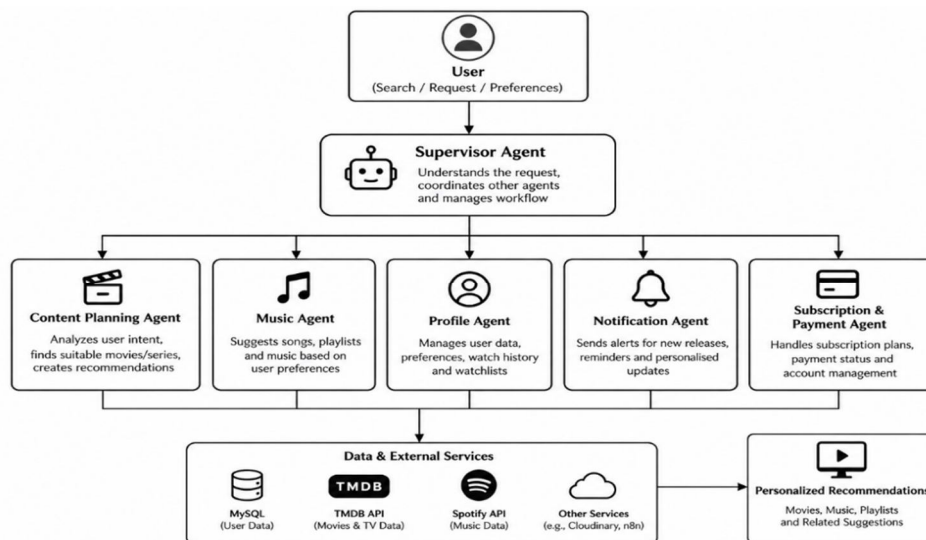


Fig 3: Multi-Agent Recommendation Workflow

V. SYSTEM ARCHITECTURE

StreamHub follows a modular client-server architecture in which the React frontend communicates with a FastAPI backend. The backend coordinates agent functions, processes external API requests, manages authentication, and interacts with the MySQL database. This arrangement separates the user interface from application logic and data management.

The first layer is the User Interface Layer. The React.js frontend provides the login page, dashboard, movie discovery pages, music discovery features, and supported user activity screens. It collects user requests and displays the results returned by backend services.

The second layer is the API and Application Layer. FastAPI exposes backend endpoints and processes requests from the frontend. Pydantic supports request and response data validation, while SQLAlchemy supports database interaction. This layer connects interface actions with the appropriate application functionality.

The third layer is the Supervisor and Agent Coordination Layer. The Supervisor Agent routes incoming requests to the appropriate movie, music, recommendation, profile, notification, or payment-related function. This division of responsibilities makes the workflow easier to organize and extend.

The fourth layer is the External Content Integration Layer. The Movie/Content Agent uses the TMDB API to retrieve available movie and television information. The Music Agent uses the configured external royalty-free music API to retrieve available music tracks and associated metadata. Results depend on the responses and availability of those services.

The fifth layer is the Authentication and User Activity Layer. JWT-based authentication supports access control for protected routes. User preferences, watchlists, ratings, and activity information are managed through backend services so that supported user-specific features can be maintained across sessions.

The final layer is the Database and Persistence Layer. MySQL stores supported user information and application records, while SQLAlchemy provides the object-relational mapping used by backend components. Persistent storage supports account continuity, activity management, and recommendation-related information.

The overall architecture can be represented as:

React.js Frontend FastAPI Backend — Supervisor Agent Movie/Music Recommendation Functions External APIs and MySQL Database Results Returned to User

This architecture combines user interaction, agent coordination, external content retrieval, authentication, and persistent storage within a single application. The modular design allows individual services to be maintained and extended without redesigning the entire platform.

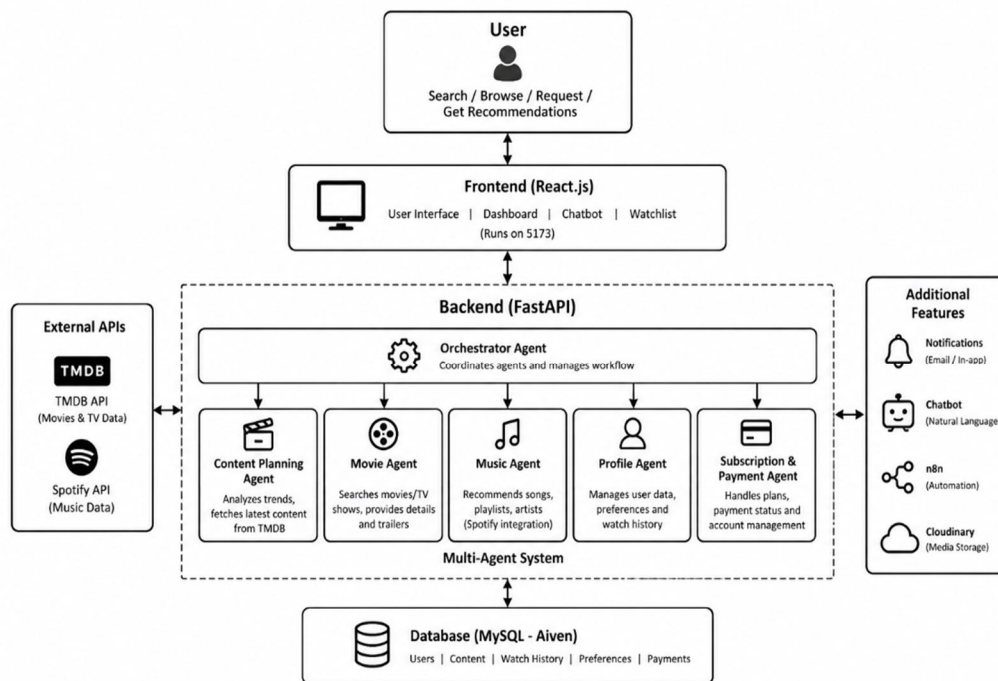


Fig. 2 System Architecture

Fig 4: System Architecture

VI. AI-ASSISTED FEATURES

The StreamHub system uses AI-oriented interaction and multi-agent coordination to make movie and music discovery more organized. Its main features include request routing, external content retrieval, preference-based recommendations, persistent user activity, and conversational access to supported entertainment functions. The available output depends on the APIs and modules configured in the application.

A. Supervisor-Based Request Routing

The Supervisor Agent receives a user request and directs it to the appropriate specialized function. Movie-related requests are routed to movie/content functionality, while music-related requests are handled by the Music Agent. This division helps avoid placing all request-processing logic inside one component.

B. Movie Discovery Through TMDB

The Movie/Content Agent connects to the TMDB API to retrieve available movie and television information. Users can explore returned titles and details such as descriptions, genres, release dates, ratings, and poster images. The displayed content is based on the API response rather than a manually fixed list.

C. Music Discovery

The Music Agent uses the configured external royalty-free music API to search for available tracks. Depending on the returned data, the interface can present track titles, artist information, genres, thumbnails, durations, and playable audio details. The system does not assume that a particular commercial track is available unless the external catalog returns it.

D. Personalized Recommendations

The system can use stored preferences, ratings, watchlists, and watch history to support more relevant recommendations. User activity provides useful signals about entertainment interests, while content information helps the application present related suggestions. The relevance of recommendations depends on the available user data and implemented recommendation logic.

E. Conversational Interaction

The conversational interface allows users to express movie, music, or recommendation requests in natural language. The backend processes the request and routes it through the supported workflow, reducing the need for users to navigate every feature manually.

F. Persistent User Activity

Supported user activity, including watchlists, ratings, preferences, and watch history, can be stored in the MySQL database. Persistent records allow user-specific information to remain available across sessions and provide a foundation for future personalization improvements.

G. Secure Access

JWT-based authentication is used to support secure access to protected backend routes and account-specific features. Authentication helps the application distinguish signed-in users and restrict supported operations according to the configured access rules.

H. Payment-Related Workflow

The Payment Agent organizes payment-related operations associated with premium content. The workflow can verify the applicable payment state before granting access to protected premium features, according to the payment functionality implemented in the application.

Overall, the AI-assisted and agent-based features combine request routing, movie and music discovery, user activity management, and secure access. The system provides a modular foundation for improving personalization and adding additional entertainment services in future versions.

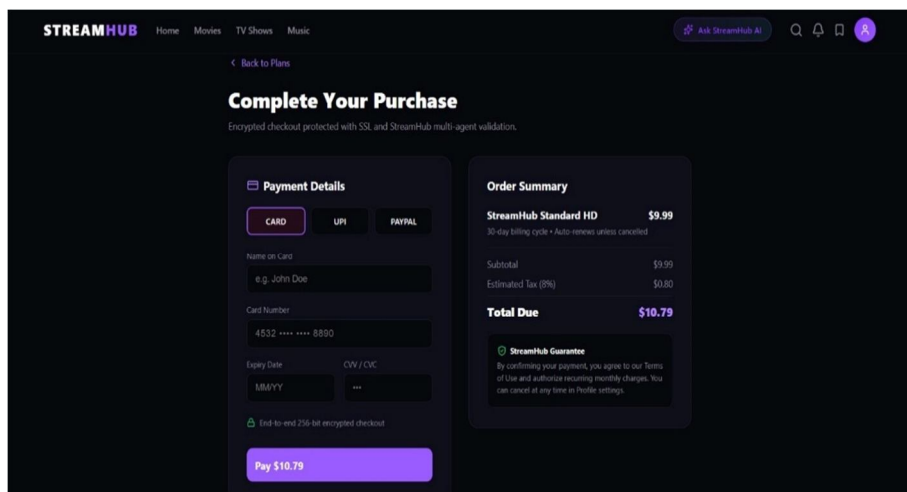


Fig 5: User Payment Page (Mock Payment)

VII. METHODOLOGY

The methodology of StreamHub is based on collecting a user request, identifying the required entertainment function, retrieving available content, and returning the result through the web interface. The process combines frontend interaction, backend request processing, agent coordination, external APIs, authentication, and database operations.

A. User Login and Authentication

The process begins when a user opens the application and signs in with valid credentials. JWT-based authentication supports access to protected backend routes. After authentication, the user can access the available dashboard and supported entertainment features.

B. User Request Submission

The user selects a feature or enters a request through the frontend. The request may relate to movie search, music discovery, recommendations, user activity, or another supported function. The frontend sends the request to the FastAPI backend for processing.

C. Supervisor Agent Coordination

The backend processes the request and uses the Supervisor Agent to identify the appropriate workflow. The request is directed to movie/content, music, recommendation, profile, notification, or payment-related functionality according to its purpose.

D. External Content Retrieval

For movie-related requests, the Movie/Content Agent communicates with TMDB and retrieves available information. For music requests, the Music Agent calls the configured external royalty-free music API. The returned response is processed by the backend before being presented to the user.

E. Recommendation Processing

When recommendation functionality is requested, the system can use available preferences, ratings, watchlists, and watch history along with content information to support relevant suggestions. Recommendation quality depends on the amount and quality of stored user activity and the implemented logic.

F. Database Operations

The backend uses SQLAlchemy to communicate with the MySQL database. Supported account details, preferences, watchlists, ratings, and activity records can be stored or retrieved as required by the requested operation. Database operations help maintain user information between sessions.

G. Result Preparation

After the relevant agent or service completes its task, the backend prepares the response and returns it to the frontend. The interface displays available movie information, music tracks, recommendations, or status information according to the result.

H. User Activity Update

When a user performs a supported action such as saving a title to a watchlist or updating a preference, the backend records the change in the database. These records maintain continuity and can contribute to later recommendation workflows.

I. Payment Workflow

When a user requests premium content, the payment-related workflow processes the applicable operation and checks the payment state according to the implemented integration. Access to protected content is controlled by the relevant application logic.

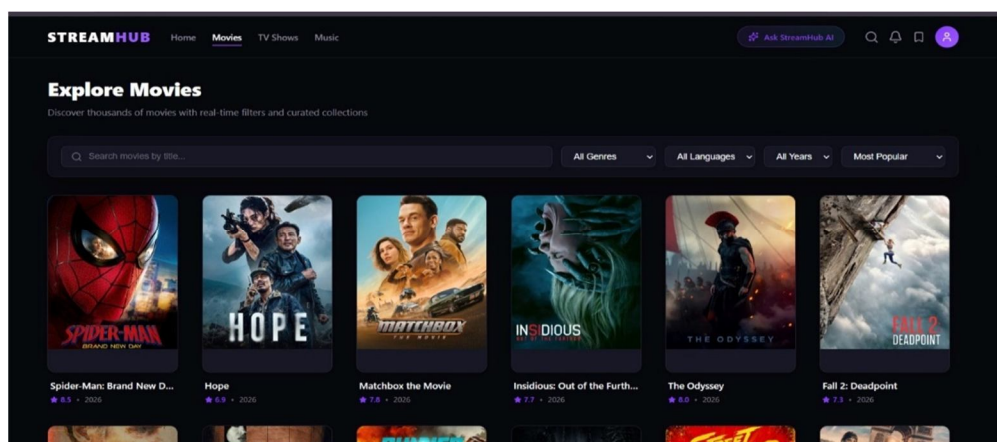


Fig 6: Movie and Music Discovery

VIII. FLOW CHART

The process begins when the user opens StreamHub and signs in. After authentication, the user accesses the dashboard and submits a movie, music, or recommendation request. The frontend sends the request to the FastAPI backend, where the Supervisor Agent identifies the appropriate workflow.

If the request concerns movies, the Movie/Content Agent retrieves available information from TMDB. If the request concerns music, the Music Agent queries the configured external music API. Recommendation requests use available user preferences and activity records where supported. The backend then prepares the relevant response.

The system also handles account-specific actions such as updating a watchlist or profile and processing payment-related requests through their corresponding functions. JWT authentication protects supported routes, and database operations maintain user information. If an external API does not return matching content or is unavailable, the displayed result is limited by that response.

Finally, the backend returns the result to the frontend, where the user can view the retrieved content or recommendation. Supported activity updates are stored in MySQL, and the process can repeat for the next user request. This workflow provides a coordinated path from user interaction to content retrieval and response display.

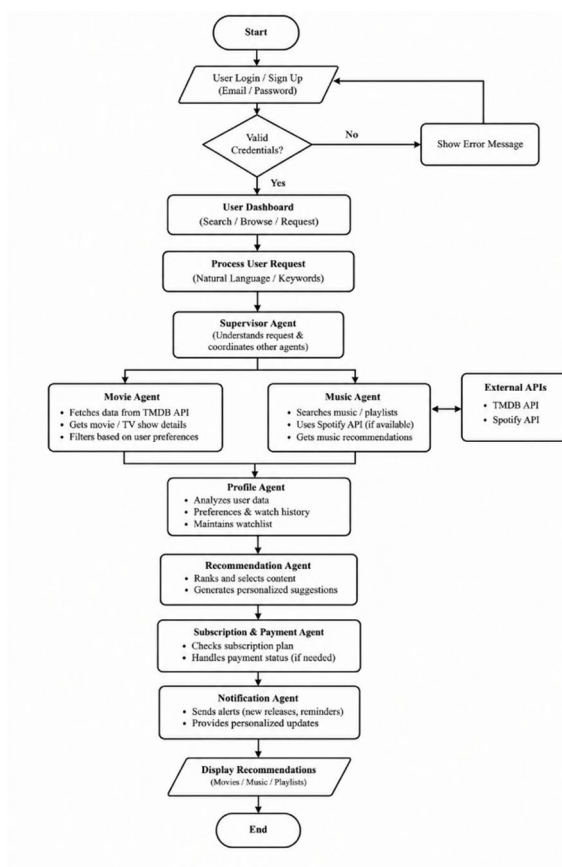


Fig 7: StreamHub Request Flowchart

IX. CONCLUSION AND FUTURE SCOPE

A. Conclusion

The StreamHub — OTT and Music Multi-Agent Recommendation System demonstrates how multi-agent coordination and modern web technologies can be used to bring movie and music discovery into a unified platform. The system provides a common interface for users to explore entertainment content and access supported recommendation and activity-management features.

The Supervisor Agent coordinates requests and directs them to the appropriate specialized functions. The Movie/Content Agent retrieves available movie information through TMDB, while the Music Agent retrieves available tracks through the configured external music API. User preferences, watchlists, ratings, and activity records can be maintained through MySQL to support continuity and personalization.

The application combines React.js, FastAPI, SQLAlchemy, Pydantic, MySQL, and JWT-based authentication. The separation between frontend, backend, agent coordination, external APIs, and database operations creates a modular structure that can be maintained and extended. The actual content displayed depends on the external services and features configured in the application. Overall, StreamHub demonstrates the potential of a multi-agent architecture for simplifying entertainment discovery and coordinating different services in one web application.

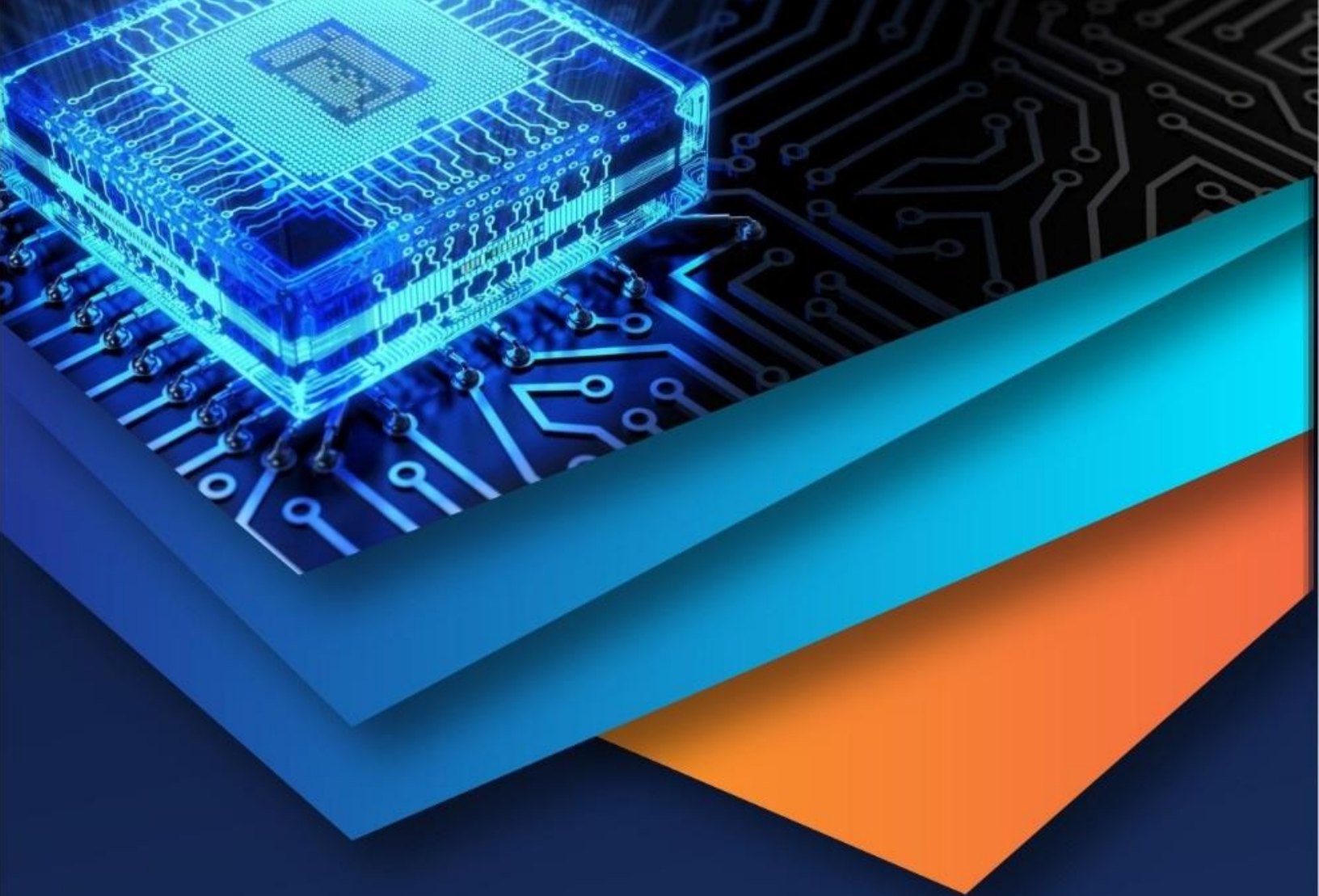
B. Future Scope

The proposed system can be further improved and expanded in several ways:

- 1) **Advanced Personalization:** Machine learning models can analyze longer-term user activity and preferences to generate more relevant movie and music recommendations.
- 2) **Additional Music Integrations:** More music services can be integrated where APIs and permissions are available, expanding the range of tracks and metadata accessible through the platform.
- 3) **Improved Conversational Agent:** The conversational interface can be enhanced to understand multi-step requests, follow-up questions, and more detailed user preferences.
- 4) **Recommendation Evaluation:** Future versions can evaluate recommendation quality using user feedback, relevance measures, and controlled testing to improve the suggestion process.
- 5) **More Specialized Agents:** Additional agents can be introduced for search refinement, content summaries, feedback analysis, notifications, and user support.
- 6) **Enhanced Security:** Future versions can strengthen access control, input validation, token handling, and secure management of external API credentials.
- 7) **Notification Improvements:** User-facing alerts can be expanded for watchlist updates, new content, subscription events, or other supported activity.
- 8) **Scalability and Caching:** Caching and optimized database queries can reduce repeated external requests and improve responsiveness when the number of users grows.
- 9) **Deployment and Integration:** After further testing, the application can be deployed with monitoring, backup, error handling, and additional integrations to support a broader user base.

REFERENCES

- [1] Jianghao Lin et al., "How Can Recommender Systems Benefit from Large Language Models: A Survey," arXiv, 2023.
- [2] Qi Wang et al., "Towards Next-Generation LLM-based Recommender Systems: A Survey and Beyond," 2024.
- [3] H. Li et al., "Recommender Systems in the Era of Large Language Models (LLMs)," IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 11, 2024.
- [4] "CD-LLMCARS: Cross Domain Fine-Tuned Large Language Model for Context-Aware Recommender Systems," IEEE Open Journal of the Computer Society, vol. 6, 2025.
- [5] "Machine Learning-Based Music Recommendation System Based on User Interest," 2024 2nd World Conference on Communication & Computing (WCONF).
- [6] Q. Peng et al., "A Survey on LLM-powered Agents for Recommender Systems," Findings of EMNLP 2025, pp. 11574— 11583, 2025. doi: 10.18653/v1/2025.findings-emnlp.620.
- [7] L. Wu et al., "A Survey on Large Language Models for Recommendation," World Wide Web, vol. 27, article 60, 2024. doi: 10.1007/s11280-024-01291-2.
- [8] S. Jayalakshmi, N. Ganesh, and R. Cep, "Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions," Sensors, vol. 22, no. 13, article 4904, 2022.
- [10] A. Yenikar et al., "Machine Learning-Based Music Recommendation System Based on User Interest," Proceedings of WCONF, 2024. doi: 10.1109/WCONF61366.2024.10692020.
- [11] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," ACM Transactions on Interactive Intelligent Systems, vol. 5, no. 4, article 19, 2015. doi: 10.1145/2827872.
- [12] [1] S. Rendle, "Factorization Machines," Proceedings of the 2010 IEEE International Conference on Data Mining, pp. 995 – 1000, 2010. doi: 10.1109/ICDM.2010.127.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)