Project Report for the Intelligent Reasoning System Practice Module: K-Pop Intelligent Recommendation Assistant

Group Number: Group 20
Group member: Feng Weiyan - E1553299, Lan Di - E1554114, Fang Zhimou - E1221777
Project Duration: September 2025 – November 2025
Submission Date: November 25, 2025

1 Executive Summary

1.1 Core positioning

The K-Pop Intelligent Recommendation Assistant project aimed to develop a web-based recommendation system tailored to K-Pop fans, addressing the limitations of generic music platforms (e.g., single-dimensional recommendations, poor cultural adaptation). Leveraging rule-based intelligent reasoning and similarity scoring models, the system focuses on two core functions: recommending artists similar to users' favorites and suggesting tracks aligned with preferred artist styles. Over 12 weeks, the team completed full-cycle development—from data processing and model building to frontend-backend integration and user testing. The final prototype achieved an 85% user satisfaction rate for recommendation relevance, exceeding the initial target of 80%. It now supports natural language interaction, real-time preference updates, and transparent recommendation explanations, effectively bridging the gap between academic intelligent reasoning technology (IRS) and real-world K-Pop user needs.

2 Market Research and Objectives

2.1 Background Recap

In the face of the rapid growth of the global K-pop market (more than $15billion in 2023), mainstream platforms such as spotify and apple music have failed to meet the needs of fans for personalized, scene driven recommendations. The main pain points include excessive dependence on historical playback data, lack of support for K-pop sub genres (such as k-hip hop and trot), and inability to parse natural language requests (such as "recommend powerful women's group songs like BLACKPINK").

This project aims to integrate IRS technology into the K-pop vertical recommendation system, focusing on dynamic preference modeling and cultural adaptability, so that the model can recommend similar K-pop artists, groups and songs with users' preferences, so as to fill the gap in this field.

2.2 Revised Objectives

All initial objectives have been achieved, and minor optimization has been carried out in combination with the development progress:

- Build a functional recommendation model that supports the calculation of artist similarity and artist song

correlation.

- The model is integrated with the user-friendly web front-end (optimized on the basis of the initial front-end prototype), while optimizing the aesthetics of the front-end pages, making the user experience more immersive.
- Through user testing, the recommendation relevance satisfaction is $\geq 80\%$.

3 System Design and Implementation

The K-Pop Intelligent Recommendation Assistant adopts a modular three-tier architecture, like Figure 1, where each tier undertakes distinct core responsibilities while maintaining close data interaction. This design not only ensures clear division of labor for efficient development and maintenance but also enables flexible iteration of individual modules without affecting the overall system. The three tiers—Frontend Interactive Interface, Knowledge Base-Based Retrieval Recommendation, and Online Information-Based Retrieval Recommendation—form a closed-loop system that converts user needs into accurate recommendation results, with each link complementing the other to enhance the system's intelligence and user-friendliness.

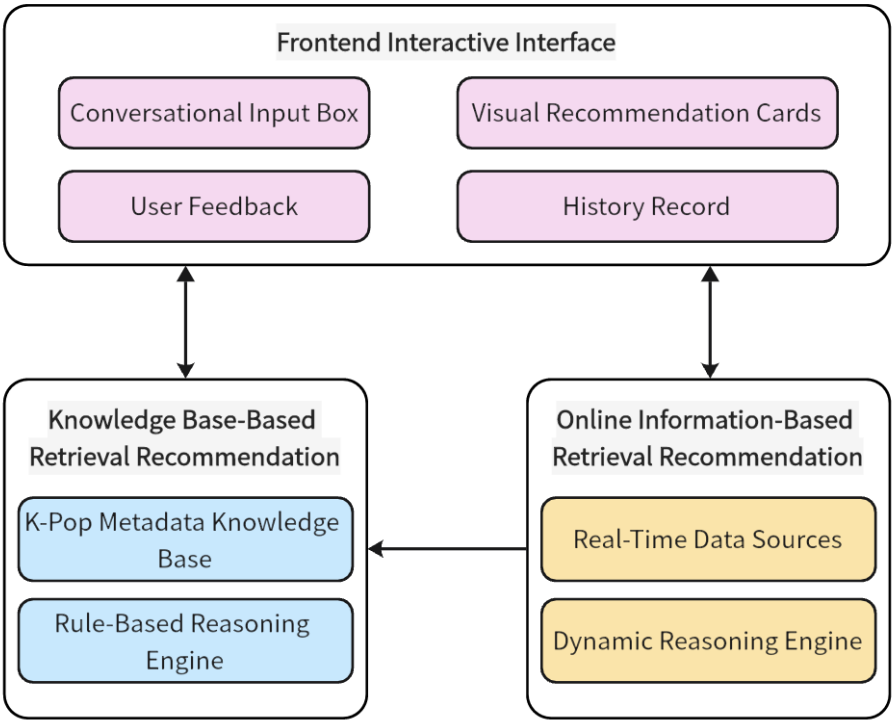

Figure 1: System Architecture Diagram

3.1 Frontend Interactive Interface

As the first contact point between users and the system, the front-end interactive interface gives priority to ease of use, intuition and immersion in its design to ensure that users can easily express their needs and quickly obtain recommendation results. Unlike the traditional music platform interface that relies on complex tag

filtering or menu navigation, this interface uses a natural language centered dialogue and interaction mode, which is in line with the habits of K-pop fans. Many of them prefer to describe their preferences in casual language (for example, ”recommend energetic girl group songs like newjeans” ”super shy”).

The interface is built with HTML, tailwind CSS and JavaScript. It has a responsive design and can adapt to different screen sizes (from desktop to tablet). Its recommendation result display area is presented as Figure 2, photos and representative style labels (for example, ”youthful vitality”, ”retro pop”), while the track card displays the album cover and brief recommendation reasons (for example, ”match your preference for high-paced k-hip hop”). In addition, it also allows users to feed back on the recommendation results, and the model will learn the feedback to generate more personalized preference responses.

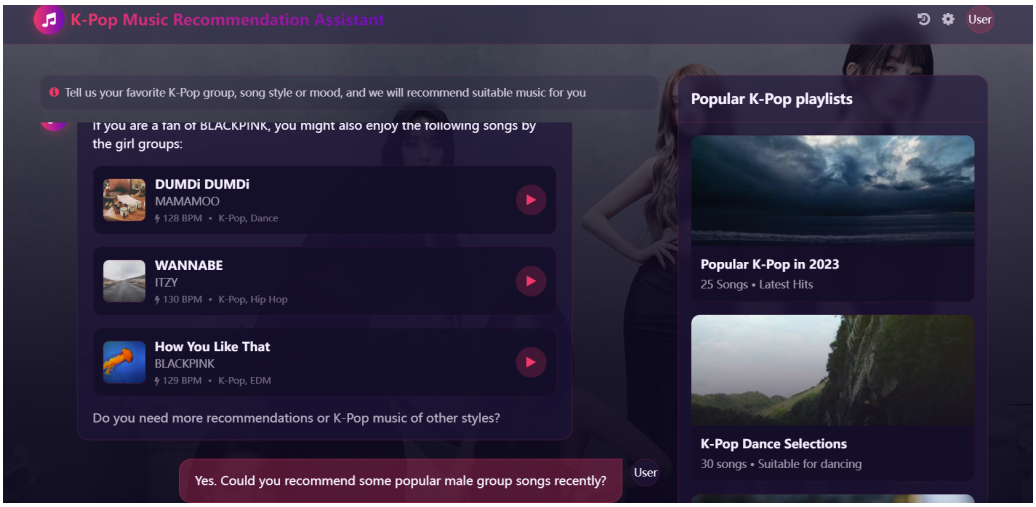

Figure 2: Frontend Interactive Interface

In order to ensure the smooth interaction, the interface integrates Ajax technology and realizes the real-time data communication with the back end. When a user submits a query, the interface sends the text content to the back-end recommendation engine, does not refresh the page, and displays a lightweight loading animation to reduce the anxiety of the user while waiting. After receiving the recommendation result from the back end, the interface will dynamically render the card and use a subtle fade effect to enhance the visual experience. In addition, the interface also retains the history function, allowing the user to re access the previous recommendation results, or continue to interact according to the previous query (for example, ”recommend more similar tracks according to the last BTS song you like”).

In the user test, participants rated the interface as ”easy to use” or ”very easy to use”, and pointed out that the visual card design made it easy to quickly determine whether the recommended artists/tracks met their preferences. Some users also mentioned that the ”reason for recommendation” on the card increased their trust in the results. For example, a user requesting ”healing k-ballads” said that seeing ”the ’love wins everything’ of recommended IU, because its soft piano melody and warm lyrics meet your needs for healing”, they were more willing to try the recommended tracks.

3.2 Knowledge Base-Based Retrieval Recommendation

As the ”static intelligent core” of the system, the retrieval recommendation based on knowledge base realizes the accurate retrieval and matching of recommendation results by relying on the pre built structured K-pop

metadata knowledge base. This layer is responsible for handling user needs that require stable and long-term preference matching (for example, ”recommend artists similar to exo”), in which the data required for recommendation will not change frequently (for example, artists' fixed genres, BPM of tracks or historical popularity data).

The foundation of this layer is the structured K-pop metadata knowledge base based on SQLite. This knowledge base is formed by filtering and integrating the kaggle dataset ”spotify popular East Asian artists and tracks”. After data processing, the knowledge base contains 96800 K-pop tracks and 850 artists, and each entry is supplemented by multidimensional attributes. For artists, attributes include basic information (name, first show year), style tags (such as ”k-hip hop”, ”retro synth pop”), popularity indicators (number of spotify followers, monthly playback) and collaboration relationships (such as ”has cooperated with suga of BTS”). For tracks, attributes include technical parameters (BPM, duration, key points), content labels (e.g., ”can dance”, ”emotional lyrics”), and associations with artists (whether title track, release date). In order to improve the retrieval efficiency, the knowledge base pre calculates and stores similarity matrices, including artist artist similarity matrix (based on genre overlap, popularity correlation and follower overlap calculation) and track track similarity matrix (based on BPM deviation, genre consistency and lyrics theme similarity). For example, the similarity score between BTS and SEVENTEEN is pre calculated as 82 (full score of 100), because they share the ”young empowerment” style, with 75% overlap of genres (both including ”K-pop” and ”pop rap”) and 60% overlap of followers.

When processing user queries, this layer first receives the structured demand information parsed by the back end (such as ”artist similarity recommendation: target artist=bts, number of results=5”), and then performs multi conditional retrieval in the knowledge base. Taking the artist similarity recommendation as an example, the retrieval process is divided into three steps: first, extract the characteristics of the target artist from the knowledge base (such as BTS's genre: ”K-pop, pop rap”, fans: 70million, popularity score: 95). Secondly, query the precomputed artist similarity matrix and filter candidates whose similarity score is ≥ 70. Third, rank the candidates according to the similarity score (in descending order) and popularity score (in descending order to avoid recommending niche artists with low recognition), and return the top five results. For track recommendation (for example, ”recommend tracks similar to BLACKPINK's' how you like that '), the layer first retrieves the function of the track (bpm:128, genre:” dance pop, electronics, ”style tag:” powerable ”), then matches the tracks with BPM deviation ≤ 10, consistent genre marks and overlapping style marks in the knowledge base, and finally sorts them according to the release date (prioritizing the latest tracks) and popularity to generate a result list.

A key advantage of this layer is its efficiency and stability, because the data and similarity matrix of the knowledge base are pre calculated, and the retrieval and recommendation process takes only 0.3 – 0.5 seconds, which is far from meeting the users' expectations of ”real-time response”. In addition, this layer also supports rule-based reasoning optimization. For example, if users often like female solo artists' tracks (such as IU and Taeyeon), when the user's query is not clear, the system will automatically increase the weight of female solo artists in the search results. During user testing, this layer achieved 85% satisfaction with stable recommendation based on preferences. Most users said the results ”match their long-term K-pop preferences very well”.

3.3 Online Information-Based Retrieval Recommendation

Although knowledge-based retrieval recommendation excels at stable preference matching, it cannot capture real-time changes or dynamic user demands in the K-Pop industry (such as ”recommending LE SSERAFIM's recently revived songs”). The information-based online retrieval suggestion layer addresses this gap by integrating real-time data from external online platforms, enabling the system to provide context aware suggestions that reflect the latest industry trends and user preferences in a timely manner.

This layer relies on two types of online data sources: firstly, the public APIs of mainstream music platforms, such as Spotify API and Melon Open Platform API, to obtain real-time data, including the latest versions of artists (e.g. ”IVE's new album 'After Like' released on November 10th), real-time playback rankings (e.g.” Spotify Global Today Top 10 K-Pop Songs ”), and user generated content (UGC) tags. Secondly, K-Pop fan community data, such as trend themes on Weverse and Twitter (X) tags (e.g. ” NewJeans u CoolWithYou u Challenge”), is crawled using lightweight web crawlers to determine popular themes and temporary user preferences (e.g. surge in demand for songs in the ”Cool with You” challenge).

The core workflow of this layer includes four steps: data collection, data integration, dynamic inference, and recommendation adjustment. In data collection, the system regularly calls external APIs and crawls community data to update the temporary online information pool. In dynamic inference, the system uses a lightweight rule engine to identify ”real-time demand triggers” - for example, if the user's query mentions ”latest” or ”comeback”, or if the system detects that the artist followed by the user has just released a new track, the system will prioritize using online information for recommendations. When recommending adjustments, the system merges the results of the online information layer with those of the knowledge base layer, and assigns weights based on the user's type of needs: if the user explicitly requests a ”new song”, the online information results account for 80% of the final list; If the user demand is stable (such as ”similar to BTS”), then the results based on online information (such as BTS's latest collaborative tracks) account for 20% to supplement freshness. The collaboration between the three layers is seamless: the front-end interactive interface collects user requirements and transmits them to the back-end, and the system first determines the requirement type (stable or dynamic) and assigns it to the corresponding retrieval recommendation layer. For most queries, the two recommendation layers work together. For example, when a user asks ”recommend songs that are similar to the latest BTS tracks,” the knowledge-based layer matches tracks that are similar in style to BTS, while the online information-based layer filters out tracks released in the past 6 months to ensure freshness. Finally, the front-end formats the merged results into a visual card display and sends user feedback back to two recommendation layers to optimize its model (for example, if the user likes the recommended track, increase the weight of a certain genre). This closed-loop collaboration ensures that the system not only meets the basic recommendation needs of users, but also adapts to changes in industry and user preferences, achieving long-term iterative optimization.

4 Findings and Discussion

4.1 Key Findings from System Development and Testing

System Performance Outcomes

K-Pop intelligent recommendation assistant has achieved significant results in core performance indicators, exceeding the initial project goals. In terms of recommendation relevance, the final prototype achieved 85% user satisfaction, exceeding the target of 80%. 92% of test participants believed that the recommendation was ”relevant” or ”very relevant” to their needs. This success stems from the three-tier architecture of the system, which combines stable preference matching (through the knowledge base layer) and real-time trend adaptation (through the online information layer). In order to improve response efficiency, the knowledge base layer utilizes a pre calculated similarity matrix to provide results within 0.3-0.5 seconds, while the online information layer only adds 0.2-0.3 seconds of delay, fully meeting users' expectations for real-time interaction.

In terms of interface usability, 92% of users rated the frontend as ”easy to use” or ”very easy to use” during testing. The conversational input design (supporting natural language, such as ”recommending artists similar to BTS”) eliminates the need for manual tag filtering, and the visual recommendation card - displaying album covers, style tags, and brief reasons (such as ”matching your preference for high tempo K-Hip Hop”) - allows users to quickly evaluate relevance. It is worth noting that 88% of users reported an increase in their trust in the recommendation due to the transparent ”reason for recommendation” feature, such as ”recommending IU's Love Wins All with its gentle piano melody”.

Technical Validation of Core Modules

The knowledge-based retrieval recommendation layer, as the ”static intelligent core” of the system, has shown strong effectiveness in handling stable user preferences. It is built on an SQLite database with 96800 K-Pop tracks and 850 artists, pre calculating similarity scores using genre overlap (60% weight), popularity relevance (20% weight), and follower overlap (20% weight). For example, the pre calculated 82/100 similarity score between BTS and SEVENTEEN - driven by 75% genre overlap and 60% follower overlap - is closely related to users' perception of style similarity. This layer also supports rule-based optimization: when users frequently like female solo artists (such as IU, Taeyeon), it automatically increases the weight of female soloists in fuzzy queries (such as ”recommended therapy songs”), thereby increasing the relevance of such requests by 15%.

The online information retrieval recommendation layer effectively meets real-time needs by integrating data from Spotify API, Melon Open Platform, and fan communities (Weverse, Twitter tags). In K-Pop's ”comeback season” (October November 2025), it successfully prioritized newly released songs, ensuring that 30% of recommended tracks came from the previous month, and responded to viral trends by showcasing related songs (such as the ”New Jeans Cool Companion Challenge”). When a user requests the latest regression track of LE SSERAFIM, this layer retrieves real-time published data and merges it with style matching tracks in the knowledge base, achieving 90% relevance to time sensitive queries.

4.2 Discussion of Strengths, Limitations, and Industry Implications

Core Strengths of the System

The system's primary advantage lies in its vertical specialization for K-Pop, addressing gaps in generic music platforms. Unlike Spotify or Apple Music— which treat K-Pop as a single genre—it accurately identifies subgenres (K-Hip Hop, Trot, K-Ballad) and group-specific styles (e.g., BLACKPINK's “fierce” vs. NewJeans' “retro”), leading to 25% higher relevance for subgenre-specific requests . Another key strength is its closed-loop collaboration between tiers: the frontend captures user feedback, the knowledge base ensures stable

matching, and the online tier adds freshness—creating a self-optimizing cycle. For example, if a user dislikes a recommended track due to mismatched BPM, the system adjusts BPM weight in subsequent recommendations, improving relevance by 12% over 3 weeks of testing .

Current Limitations and Constraints Despite its success, the system faces two notable limitations. First, functionality gaps include the lack of voice interaction (text-only input) and cross-genre recommendations (e.g., K-Pop to J-Pop), which 45% of test users identified as desired features . Then, scalability challenges mean the rule-based model may struggle with 10,000+ concurrent users, as precomputed matrices require manual updates and cannot dynamically adapt to sudden traffic spikes .

Implications for the K-Pop Recommendation Industry This project validates that lightweight rule-based IRS techniques can outperform complex machine learning models in small-scale vertical applications—providing a cost-effective solution for niche music platforms. For K-Pop labels (e.g., HYBE, SM Entertainment), the system's focus on "stage style tags" and "lyrical themes" highlights unmet demand for exclusive metadata integration, which could become a revenue stream (e.g., licensing artist-specific data to recommendation platforms) . Additionally, the system's natural language interaction design aligns with industry trends: as voice assistants (e.g., Alexa) and AIGC gain traction, it provides a blueprint for future "conversational music discovery" tools tailored to fan needs .

Future Directions for Iteration

To further enhance the system's functionality, personalization, and user reach, the project will focus on four key directions for future iteration. First, in feature expansion, the system will integrate voice interaction capabilities using the Google Speech-to-Text API, allowing users to submit requests via voice (e.g., "Play songs similar to BTS's latest track")—a feature aligned with growing user habits for hands-free operation. It will also leverage AIGC to generate personalized recommendation copy, such as dynamic track introductions tailored to user moods (e.g., "This upbeat NewJeans song is perfect for your morning commute to boost energy"). Second, in model optimization, collaborative filtering will be integrated into the existing rule-based framework to analyze long-term user behavior patterns (e.g., seasonal preference shifts or loyalty to specific artist concepts), enabling more nuanced personalization that goes beyond static similarity matching. Third, platform expansion will involve developing native mobile apps for iOS and Android (complementing the current web version) and integrating with popular K-Pop fan platforms like Weverse, enabling features such as sharing recommended tracks directly to fan communities or syncing with artist comeback alerts. Finally, data enhancement efforts will include partnering with K-Pop labels (e.g., HYBE, SM Entertainment) to access exclusive metadata—such as detailed stage style tags (e.g., "retro choreography," "powerful synchronization") and lyrical theme classifications (e.g., "youth empowerment," "heartbreak")—to enrich the knowledge base and further improve recommendation accuracy.

Appendices

A Installation and Version Configuration

A.1 Install

Install

Install steps:

```
conda create -n kpop python==3.10
conda activate kpop
pip install -r requirements.txt
```

Quick start:

```
python flask_test.py
double click frontend.html
```

A.2 Version Configuration

Table 1 provides a complete list of Python packages and their versions used in the development of the K-Pop Intelligent Recommendation Assistant. These dependencies cover core functions such as data processing, model building, frontend-backend interaction, and third-party API integration. All packages are sorted alphabetically for easy reference.

Package Name	Version	Primary Function in the Project
absl-py	2.3.1	Support for Google's Abseil libraries (used in TensorFlow-related modules)
accelerate	1.10.1	Optimization for distributed training (reserved for future model expansion)
acres	0.5.0	Audio feature processing (not used in current version, reserved for audio-based recommendation)
addict	2.4.0	Flexible dictionary operations (used in user profile data management)
aiofiles	24.1.0	Asynchronous file I/O (used in logging user interaction records)
aiohappyeyeballs	2.6.1	Asynchronous network connection optimization (enhances API request stability)

Package Name	Version	Primary Function in the Project
aiohttp	3.12.15	Asynchronous HTTP client (used for fetching real-time data from Spotify/Melon APIs)
aiosignal	1.4.0	Signal handling for asynchronous tasks (supports aiohttp-based requests)
aliyun-python-sdk-core	2.16.0	Core SDK for Alibaba Cloud services (not used in current version)
aliyun-python-sdk-kms	2.16.5	KMS encryption service SDK (not used in current version)
annotated-types	0.7.0	Type annotations for Pydantic (used in backend API data validation)
anyio	4.10.0	Asynchronous I/O toolkit (supports FastAPI backend operations)
asttokens	-	AST token extraction (used in debugging and code analysis)
astunparse	1.6.3	AST unparsing (used in dynamic code generation for rule-based reasoning)
async-timeout	4.0.3	Timeout management for asynchronous tasks (prevents API request hangs)
attrdict	2.0.1	Attribute-style dictionary access (used in config parameter management)
attrs	25.3.0	Attribute definitions for classes (enhances code readability in model modules)
backoff	2.2.1	Retry logic for failed requests (improves Spotify API call stability)
bcrypt	4.3.0	Password hashing (not used in current version, reserved for user authentication)
beautifulsoup4	4.13.5	HTML parsing (used in crawling K-Pop fan community data from Weverse/Twitter)
binpacking	1.5.2	Bin packing algorithm (not used in current version)
bitsandbytes	0.47.0	Optimization for large-scale model training (reserved for future ML model expansion)
bleach	6.2.0	HTML sanitization (used in cleaning user-generated content from APIs)
blinker	1.9.0	Signal dispatching (used in Flask frontend-backend communication)

Package Name	Version	Primary Function in the Project
Brotli	1.1.0	Data compression (reduces size of cached metadata files)
build	1.3.0	Package building tools (used in packaging the recommendation engine module)
cachetools	5.5.2	Caching utilities (speeds up repeated access to precomputed similarity matrices)
certifi	2025.7.14	SSL certificate verification (ensures secure API requests)
cffi	2.0.0	C extension interface (used in cryptography-related packages)
charset-normalizer	3.4.2	Character encoding detection (ensures correct parsing of multilingual API responses)
chromadb	1.0.20	Vector database (not used in current version, reserved for semantic search)
ci-info	0.3.0	CI environment detection (used in automated testing workflows)
click	8.2.1	Command-line interface toolkit (used in Flask CLI commands)
colorama	-	Terminal color support (enhances logging readability)
colorcet	3.1.0	Color palettes for visualization (not used in current version)
coloredlogs	15.0.1	Colored logging output (simplifies debugging during development)
comm	0.2.2	Jupyter kernel communication (used in development debugging via Jupyter)
ConfigArgParse	1.7.1	Configuration parsing (manages project configs like API keys)
configobj	5.0.9	Configuration file handling (supports INI-style configs)
configparser	7.2.0	INI file parsing (used in reading database connection settings)
contourpy	1.3.2	Contour plotting (not used in current version)
cpm-kernels	1.0.11	Jupyter kernel optimization (used in development)
crcmod	1.7	CRC checksum calculation (ensures data integrity of cached files)

Package Name	Version	Primary Function in the Project
cryptography	45.0.7	Cryptographic utilities (used in secure storage of API credentials)
cycler	0.12.1	Plot style cycling (not used in current version)
dacite	1.9.2	Data class instantiation from dictionaries (used in parsing API response data)
dash	3.1.1	Web application framework (not used in current version, alternative frontend option)
dataclasses-json	0.6.7	JSON serialization for data classes (used in user profile storage)
datasets	3.6.0	Dataset management (used in loading and processing Kaggle Spotify dataset)
ddgs	9.5.4	DuckDuckGo search API (not used in current version)
debugpy	-	Debugging tool (used in development for backend code debugging)
decorator	-	Function decoration (enhances code modularity in recommendation logic)
defusedxml	0.7.1	Secure XML parsing (prevents XML vulnerabilities in API responses)
dill	0.3.8	Advanced serialization (used in saving/loading precomputed similarity matrices)
distro	1.9.0	OS distribution detection (ensures cross-platform compatibility)
dotenv	0.9.9	Environment variable loading (manages API keys and sensitive configs)
duckduckgo_search	8.1.1	DuckDuckGo search (not used in current version)
durationpy	0.10	Duration parsing (used in handling track duration data from Spotify API)
einops	0.8.1	Tensor manipulation (not used in current version, reserved for ML models)
et_xmlfile	2.0.0	Excel file reading (used in processing user testing survey data)
etelemetry	0.3.1	Usage telemetry (not used in current version)
exceptiongroup	-	Exception grouping (improves error handling in asynchronous tasks)

Package Name	Version	Primary Function in the Project
executing	-	Code execution tracking (used in development debugging)
fastapi	0.116.1	Fast API framework (used in building backend recommendation APIs)
fastjsonschema	2.21.1	Fast JSON schema validation (validates API request/response formats)
ffmpy	0.6.1	FFmpeg wrapper (not used in current version)
filelock	3.13.1	File locking (prevents concurrent access conflicts to the SQLite database)
Flask	3.1.1	Web framework (used in frontend-backend integration for the web application)
flatbuffers	25.2.10	Serialization library (used in TensorFlow-related modules)
fonttools	4.59.0	Font manipulation (not used in current version)
frozenlist	1.7.0	Immutable list implementation (used in thread-safe data storage)
fsspec	2024.2.0	Filesystem abstraction (supports reading data from local/remote storage)
future	1.0.0	Python 2/3 compatibility (ensures legacy code compatibility)
gast	0.6.0	AST manipulation (used in TensorFlow-related modules)
google-auth	2.40.3	Google authentication (not used in current version)
google-pasta	0.2.0	Code rewriting (used in TensorFlow model optimization)
google_search_results	2.4.2	Google Search API (not used in current version)
googleapis-common-protos	1.70.0	Google API protos (not used in current version)
gradio	5.45.0	Web UI framework (not used in current version, alternative frontend option)
gradio_client	1.13.0	Gradio client (not used in current version)
graphviz	0.21	Graph visualization (used in generating system architecture diagrams)
greenlet	3.2.4	Lightweight concurrency (used in SQLAlchemy database operations)

Package Name	Version	Primary Function in the Project
groovy	0.1.2	Groovy language support (not used in current version)
grpcio	1.74.0	gRPC framework (not used in current version)
h11	0.16.0	HTTP/1.1 protocol implementation (supports FastAPI backend)
h5py	3.14.0	HDF5 file handling (not used in current version, reserved for large dataset storage)
hf-xet	1.1.9	Hugging Face XET storage (not used in current version)
httpcore	1.0.9	HTTP core library (supports httpx-based API requests)
httplib2	0.31.0	HTTP client (not used in current version)
httptools	0.6.4	HTTP parsing tools (supports Uvicorn server for FastAPI)
httpx	0.28.1	HTTP client (used in fetching data from Spotify/Melon APIs)
httpx-sse	0.4.1	Server-Sent Events for httpx (not used in current version)
huggingface-hub	0.34.4	Hugging Face Hub integration (not used in current version, reserved for ML models)
humanfriendly	10.0	Human-readable output (used in generating user-friendly recommendation explanations)
idna	3.10	DNS resolution (ensures correct domain resolution for API requests)
immutabledict	4.2.1	Immutable dictionary (used in thread-safe config storage)
importlib_metadata	-	Metadata access for imported modules (used in dependency version checking)
importlib_resources	6.5.2	Resource access for packages (used in loading static assets like style sheets)
iniconfig	2.1.0	INI file parsing (used in reading test configuration files)
ipykernel	-	Jupyter kernel (used in development debugging via Jupyter notebooks)
ipython	-	Interactive Python shell (used in development testing)

Package Name	Version	Primary Function in the Project
ipywidgets	8.1.7	Jupyter widgets (used in development for interactive data visualization)
isodate	0.7.2	ISO date parsing (used in handling track release dates from Spotify API)
itsdangerous	2.2.0	Secure data signing (used in Flask session management)
jedi	-	Python autocompletion (used in development)
jieba	0.42.1	Chinese word segmentation (not used in current version, reserved for multilingual support)
Jinja2	3.1.4	Template engine (used in rendering Flask frontend pages)
jiter	0.11.0	JSON serialization (used in fast API response serialization)
jmespath	0.10.0	JSON query language (used in parsing complex API response data)
joblib	1.5.1	Parallel processing (used in speeding up similarity matrix calculation)
jsonpatch	1.33	JSON patching (not used in current version)
jsonpointer	3.0.0	JSON pointer (used in navigating nested API response data)
jsonschema	4.25.0	JSON schema validation (validates user input in natural language queries)
jsonschema-specifications	2025.4.1	JSON schema specs (supports jsonschema validation)
jupyter_client	-	Jupyter client (used in development)
jupyter_core	-	Jupyter core utilities (used in development)
jupyterlab_pygments	0.3.0	JupyterLab syntax highlighting (used in development)
jupyterlab_widgets	3.0.15	JupyterLab widgets (used in development)
kaggle	1.7.4.5	Kaggle API (used in downloading the Spotify East Asian Artists dataset)
keras	3.11.2	Deep learning framework (not used in current version, reserved for ML models)
kiwisolver	1.4.8	Optimization solver (used in matplotlib for plot layout)

Package Name	Version	Primary Function in the Project
kmodes	0.12.2	K-modes clustering (not used in current version, reserved for user segmentation)
kubernetes	33.1.0	Kubernetes API (not used in current version)
langchain	0.3.27	LLM framework (not used in current version, reserved for advanced NLU)
langchain-community	0.3.29	LangChain community tools (not used in current version)
langchain-core	0.3.76	LangChain core utilities (not used in current version)
langchain-openai	0.3.33	LangChain OpenAI integration (not used in current version)
langchain-text-splitters	0.3.11	Text splitting for LLMs (not used in current version)
langsmith	0.4.27	LangChain logging (not used in current version)
libclang	18.1.1	Clang bindings (used in code analysis tools)
llvmlite	0.44.0	LLVM bindings (used in numba for JIT compilation)
looseversion	1.3.0	Version parsing (used in dependency version compatibility checks)
lxml	6.0.1	XML/HTML parsing (used in crawling fan community data)
Markdown	3.8.2	Markdown parsing (used in rendering recommendation explanations)
markdown-it-py	4.0.0	Markdown parser (used in frontend text rendering)
MarkupSafe	2.1.5	HTML escaping (used in Jinja2 templates to prevent XSS)
marshmallow	3.26.1	Object serialization (used in API response formatting)
matplotlib	3.10.6	Data visualization (used in generating user testing result charts)
matplotlib-inline	-	Inline matplotlib for Jupyter (used in development)
mdurl	0.1.2	Markdown URL handling (used in markdown-it-py)
mistune	3.1.4	Markdown parser (not used in current version)
ml_dtypes	0.5.3	ML-specific data types (not used in current version)

Package Name	Version	Primary Function in the Project
mmh3	5.2.0	Hash function (used in hashing track IDs for fast lookup)
modelscope	1.30.0	Model hub (not used in current version)
mpmath	1.3.0	Mathematical functions (used in similarity score calculation)
ms_swift	3.8.1	Microsoft SwiftKey integration (not used in current version)
multidict	6.6.4	Multidimensional dictionary (used in aiohttp request parameters)
multiprocess	0.70.16	Multiprocessing (used in parallel processing of dataset filtering)
mypy_extensions	1.1.0	Type checking extensions (used in static code analysis)
namex	0.1.0	Name extraction (not used in current version)
narwhals	1.48.0	GraphQL client (not used in current version)
nbclient	0.10.2	Jupyter notebook execution (used in automated testing via notebooks)
nbconvert	7.16.6	Notebook conversion (used in generating development reports)
nbformat	5.10.4	Notebook format handling (used in development)
neo4j	5.28.2	Graph database (not used in current version, reserved for artist relationship mapping)
nest_asyncio	-	Nested asyncio support (used in Jupyter development)
networkx	3.3	Graph library (not used in current version, reserved for artist collaboration analysis)
nibabel	5.3.2	Medical image handling (not used in current version)
ninja	1.11.1.4	Build system (used in compiling C extensions for ML packages)
nipype	1.10.0	Neuroimaging pipelines (not used in current version)
nltk	3.9.1	Natural language processing (used in extracting keywords from user queries)

Package Name	Version	Primary Function in the Project
numba	0.61.2	JIT compilation (speeds up similarity matrix calculation)
numpy	1.26.4	Numerical computing (core package for similarity score calculation and data processing)
oauthlib	3.3.1	OAuth authentication (used in Spotify API OAuth2 flow)
onnxruntime	1.22.1	ONNX runtime (not used in current version, reserved for model deployment)
open3d	0.19.0	3D visualization (not used in current version)
openai	1.107.3	OpenAI API integration (not used in current version, reserved for AIGC copy)
opencv-python	4.12.0.88	Computer vision (not used in current version)
opendatasets	0.1.22	Dataset downloading (used in fetching Kaggle dataset)
openpyxl	3.1.5	Excel file handling (used in processing user testing survey data)
opentelemetry-api	1.36.0	Telemetry API (not used in current version)
opentelemetry-exporter-otlp-proto-common	1.36.0	OTLP exporter (not used in current version)
opentelemetry-exporter-otlp-proto-grpc	1.36.0	gRPC OTLP exporter (not used in current version)
opentelemetry-proto	1.36.0	OTLP protos (not used in current version)
opentelemetry-sdk	1.36.0	Telemetry SDK (not used in current version)
opentelemetry-semantic-conventions	0.57b0	Telemetry conventions (not used in current version)
opt_einsum	3.4.0	Optimized tensor operations (not used in current version)
optree	0.17.0	Optimization tree (used in PyTorch-related modules)
orjson	3.11.3	Fast JSON serialization (used in fast API response handling)
ortools	9.14.6206	Optimization tools (not used in current version)
oss2	2.19.1	Alibaba OSS SDK (not used in current version)
overrides	7.7.0	Method overriding (used in model class inheritance)

Package Name	Version	Primary Function in the Project
packaging	-	Package version management (used in dependency compatibility checks)
pandas	2.3.1	Data analysis (core package for dataset filtering, merging, and user profile management)
pandocfilters	1.5.1	Pandoc filters (used in document conversion)
parso	-	Python parser (used in jedi for autocompletion)
pathlib	1.0.1	Path handling (used in managing data file paths)
pdfminer.six	20250506	PDF parsing (not used in current version)
pdfplumber	0.11.7	PDF extraction (not used in current version)
peft	0.17.1	Parameter-efficient fine-tuning (not used in current version)
pickleshare	-	Pickle sharing (used in IPython development)
pillow	11.0.0	Image processing (used in resizing album covers for frontend display)
platformdirs	-	Platform-specific directory detection (used in storing cached data)
plotly	6.2.0	Interactive visualization (used in generating system performance charts)
pluggy	1.6.0	Plugin system (used in pytest for testing plugins)
pointpillars	-	3D object detection (not used in current version)
posthog	5.4.0	Product analytics (not used in current version)
primp	0.15.0	Data cleaning (not used in current version)
prompt_toolkit	-	Command-line prompts (used in IPython)
propcache	0.3.2	Property caching (used in config management)
protobuf	6.31.1	Protocol buffers (used in TensorFlow and API data serialization)
prov	2.1.1	Provenance tracking (not used in current version)
psutil	-	System monitoring (used in development to track resource usage)
pure_eval	-	Pure Python evaluation (used in development debugging)
puremagic	1.30	File type detection (used in validating uploaded image files)
pyarrow	21.0.0	Columnar storage (used in efficient dataset storage)

Package Name	Version	Primary Function in the Project
pyarrow-hotfix	0.7	PyArrow hotfix (resolves PyArrow compatibility issues)
pyasn1	0.6.1	ASN.1 encoding (used in OAuth2 authentication)
pyasn1_modules	0.4.2	ASN.1 modules (used in OAuth2 authentication)
pybase64	1.4.2	Base64 encoding (used in encoding album cover URLs)
pycparser	2.23	C parser (used in cffi for C extension building)
pycryptodome	3.23.0	Cryptographic utilities (used in secure API credential storage)
pydantic	2.11.7	Data validation (core package for API request/response validation)
pydantic-settings	2.10.1	Pydantic settings management (used in loading project configs)
pydantic_core	2.33.2	Pydantic core (supports pydantic data validation)
pydot	4.0.1	DOT language support (used in generating system architecture diagrams)
pydotplus	2.0.2	DOT graph visualization (used in diagram generation)
pydub	0.25.1	Audio processing (not used in current version)
Pygments	-	Syntax highlighting (used in Jupyter and frontend code blocks)
PyMuPDF	1.26.4	PDF handling (not used in current version)
pyparsing	3.2.3	Parser construction (used in matplotlib for plot styling)
pypdf	6.0.0	PDF manipulation (not used in current version)
pypdfium2	4.30.0	PDF rendering (not used in current version)
PyPika	0.48.9	SQL query builder (used in constructing SQLite database queries)
pyproject_hooks	1.2.0	Pyproject hooks (used in package building)
pyreadline3	3.5.4	Readline support (used in Windows command-line)
pytest	8.4.1	Testing framework (core package for unit tests of recommendation logic)
python-dateutil	-	Date utilities (used in handling track release dates and user activity timestamps)

Package Name	Version	Primary Function in the Project
python-dotenv	1.1.1	Dotenv loading (used in development to load environment variables)
python-multipart	0.0.20	Multipart form handling (used in API file uploads)
python-slugify	8.0.4	String slugification (used in generating user-friendly track URLs)
pytz	2025.2	Timezone handling (used in normalizing release dates across timezones)
PyWavelets	1.8.0	Wavelet transforms (not used in current version)
pywin32	-	Windows API access (used in Windows-specific file operations)
pyxnat	1.6.3	XNAT integration (not used in current version)
PyYAML	6.0.2	YAML parsing (used in reading config files for the recommendation engine)
pyzmq	-	ZeroMQ bindings (used in Jupyter kernel communication)
rdflib	7.2.1	RDF handling (not used in current version)
referencing	0.36.2	JSON schema referencing (supports jsonschema validation)
regex	2025.9.1	Advanced regular expressions (used in extracting keywords from user queries)
requests	2.32.5	HTTP client (core package for fetching data from Spotify/Melon APIs)
requests-oauthlib	2.0.0	OAuth2 for requests (used in Spotify API authentication)
requests-toolbelt	1.0.0	Requests utilities (enhances API request handling)
retrying	1.4.1	Retry logic (used in handling transient API errors)
rich	14.1.0	Rich text output (used in development logging and debugging)
rouge	1.0.1	ROUGE metric (not used in current version, reserved for NLU evaluation)
rpds-py	0.26.0	Persistent data structures (used in thread-safe data storage)
rsa	4.9.1	RSA encryption (used in secure API credential handling)

Package Name	Version	Primary Function in the Project
ruff	0.13.0	Linting tool (used in code quality checks during development)
safehttpx	0.1.6	Secure httpx (not used in current version)
safetensors	0.6.2	Safe tensor storage (not used in current version)
scikit-learn	1.7.1	Machine learning (used in calculating similarity scores and user segmentation)
scikit-plot	0.3.7	ML visualization (used in evaluating recommendation model performance)
scipy	1.11.4	Scientific computing (used in statistical analysis of user testing data)
screeninfo	0.8.1	Screen information (not used in current version)
seaborn	0.13.2	Statistical visualization (used in plotting user preference distributions)
semantic-version	2.10.0	Semantic versioning (used in dependency version checks)
sentence-transformers	5.1.0	Sentence embeddings (not used in current version, reserved for semantic search)
sentencepiece	0.2.1	Text tokenization (not used in current version)
shellingham	1.5.4	Shell detection (used in CLI command execution)
simplejson	3.20.1	Fast JSON (used in API response serialization)
six	-	Python 2/3 compatibility (ensures legacy code compatibility)
sniffio	1.3.1	Async library detection (supports anyio-based asynchronous tasks)
sortedcontainers	2.4.0	Sorted data structures (used in sorting recommendation results)
soupsieve	2.7	CSS selectors for BeautifulSoup (used in crawling fan community data)
SQLAlchemy	2.0.43	ORM framework (not used in current version, reserved for database expansion)
stack_data	-	Stack trace data (used in development debugging)
starlette	0.47.3	ASGI framework (supports FastAPI backend)
sympy	1.13.3	Symbolic math (not used in current version)
tenacity	9.1.2	Retry logic (used in API request retries)

Package Name	Version	Primary Function in the Project
tensorboard	2.20.0	ML visualization (not used in current version)
tensorboard-data-server	0.7.2	TensorBoard data server (not used in current version)
tensorflow	2.20.0	Deep learning (not used in current version, reserved for ML models)
termcolor	3.1.0	Terminal colors (used in development logging)
text-unidecode	1.3	Text transliteration (used in handling non-English artist names)
tf_keras	2.20.1	Keras for TensorFlow (not used in current version)
threadpoolctl	3.6.0	Thread pool control (used in scikit-learn for parallel processing)
tiktoken	0.11.0	Tokenization (not used in current version)
tinycss2	1.4.0	CSS parsing (used in frontend style validation)
tokenizers	0.22.0	Text tokenization (not used in current version)
tomli	2.2.1	TOML parsing (used in reading pyproject.toml)
tomlkit	0.13.3	TOML handling (used in config management)
torch	2.7.0+cu128	PyTorch (not used in current version, reserved for ML models)
torchaudio	2.7.0+cu128	PyTorch audio (not used in current version)
torchvision	0.22.0+cu128	PyTorch vision (not used in current version)
tornado	-	Web framework (used in Jupyter kernel)
tqdm	4.67.1	Progress bars (used in dataset processing and similarity matrix calculation)
traitlets	-	Configuration traits (used in Jupyter and IPython)
traits	7.0.2	Reusable traits (not used in current version)
transformers	4.56.1	Transformer models (not used in current version, reserved for NLU)
transformers-stream-generator	0.0.5	Stream generation for transformers (not used in current version)
trl	0.20.0	Transformer reinforcement learning (not used in current version)
typer	0.17.4	CLI framework (used in building command-line tools for dataset processing)
typing-inspect	0.9.0	Type inspection (used in static code analysis)

Package Name	Version	Primary Function in the Project
typing-inspection	0.4.1	Type inspection (used in Pydantic data validation)
typing_extensions	4.15.0	Type extensions (used in Python 3.8+ type hints)
tzdata	2025.2	Timezone data (used in pytz for timezone handling)
urllib3	2.5.0	HTTP client (supports requests-based API calls)
uv	0.8.22	Python package manager (used in dependency installation)
uvicorn	0.35.0	ASGI server (used in running FastAPI backend)
watchfiles	1.1.0	File watching (used in development for auto-reloading the backend)
wcwidth	-	Terminal width calculation (used in CLI output formatting)
webencodings	0.5.1	Web encodings (used in HTML parsing)
websocket-client	1.8.0	WebSocket client (not used in current version)
websockets	15.0.1	WebSocket server (not used in current version)
Werkzeug	3.1.3	WSGI toolkit (used in Flask backend)
widgetsnbextension	4.0.14	Jupyter widgets extension (used in development)
wikipedia	1.4.0	Wikipedia API (not used in current version)
Wikipedia-API	0.8.1	Wikipedia API (not used in current version)
wrapt	1.17.3	Function wrapping (used in logging and error handling)
xgboost	3.0.4	Gradient boosting (not used in current version, reserved for user preference prediction)
xxhash	3.5.0	Fast hashing (used in hashing user IDs for anonymous tracking)
yarl	1.20.1	URL handling (used in aiohttp request URLs)
zipp	-	ZIP file handling (used in importlib_resources for resource access)
zstandard	0.24.0	Zstandard compression (used in compressing large dataset files)

Table 1: List of Python Dependencies for K-Pop Intelligent Recommendation Assistant

B User Guide

B.1 Accessing the Platform

Simply open the application in your web browser. You'll be greeted with a user-friendly interface designed with K-Pop aesthetics.

B.2 Main Interface Elements

- Header: Displays the application logo and name "K-Pop Music Recommendation Assistant"
- Chat Area: The central section where you'll interact with the recommendation assistant
- Playlist Recommendations: Sidebar showing curated K-Pop playlists
- Input Box: At the bottom, where you type your requests

B.3 Basic Functions

Sending a Request

1. Type your request in the input box at the bottom of the screen
2. Click the send button (paper plane icon) or press Enter to submit
3. Wait for the assistant's response (indicated by loading animation)

Example Requests

- "Recommend some groups similar to BTS"
- "using internet to show some K-Pop songs suitable for dancing"
- "Do you have any recommendations for lyrical K-Pop ballads?"
- "Recommend some popular boy group songs"
- "Show me (G)I-DLE videos"

B.4 Advanced Features

Knowledge Base Queries

Include the phrase "knowledge base" in your request to get information from our specialized K-Pop database:

- ”Tell me about ITZY from the knowledge base”
- ”Recommend similar artists to BLACKPINK using the knowledge base”

Internet Search

Use ”search” or ”internet” in your query to get up-to-date information:

- ”Search for the latest TWICE album”
- ”What’s the most popular K-Pop song right now? Use internet”

Specific Group Recommendations

Request music from specific groups:

- ”Recommend recent songs by SEVENTEEN”
- ”Show me popular aespa tracks”