A SMART SYSTEM TO HELP K-POP FANS FIND ALBUMS AND EVENTS USING FLUTTER AND ARTIFICIAL INTELLIGENCE

Xiaoya Gao [1], Rodrigo Onate [2]

[1] Fairmont Preparatory Academy, 2200 W Sequoia Ave, Anaheim, CA 92801
[2] California State University, Fullerton, 800 N State College Blvd, Fullerton, CA 92831

ABSTRACT

Collecting K-pop merchandise and tracking global concert events remains a fragmented and time-consuming process for international fans. To address this, we developed mixfan, a centralized mobile application designed to streamline the fan experience. Built using Flutter, Firebase, and the Ticketmaster API, the program integrates real-time event tracking with a personalized "bias" system and an OpenAI-powered assistant for music-related queries. Key challenges included normalizing artist data across multiple APIs and ensuring secure, real-time data synchronization via Firestore, which were resolved through strict security rules and request normalization. Experimental results revealed that while the AI achieved 100% accuracy for historical data, it struggled with identifying active artists, likely due to limitations from training data cutoffs.. Additionally, search testing highlighted a critical need for Korean-language support. Ultimately, mixfan provides a unique, label-agnostic platform that empowers fans to manage their hobby with greater efficiency and joy.

KEYWORDS

Entertainment, Music, K-pop, Flutter, Artificial Intelligence

1. INTRODUCTION

In America, Kpop albums are not easy to buy or find, but Kpop fans' amount is big, and they are also very active [1][4]. Only for some popular or new groupscan you find their albums in almost all Kpop albums websites; for old groups or some groups that are not very popular and well-known, it is super hard to find them. Also, sometimes, most Kpop albums are sold on websites. The problem is that some new fans or international fans don't know where to find albums that they really want. In addition, Kpop merch and albums are expensive sometimes. Different websites have different levels of prices and stock amounts. When fans are making their decisions, they need to spend a lot of time comparing prices on 7 or more websites such as K Place and Kpopalbums in order to buy the best. Most people do not think those are important problems because they can solve them, but they need to take more time and energy [2][3]. However, collecting albums and listening to Kpop shouldn't be a hard thing because our life is hard enough, the process should be warm, happy, and joyful.

Mylopoulos and Brodie's AI and database framework proposed integrating semantic knowledge systems to improve structured data retrieval [9]. While foundational for its era, it assumed clean, static data and had no capacity for real-time querying or natural language interaction. mixfan

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp. 77-89, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160807

addresses this directly by combining Firebase's live database with OpenAI's conversational AI, enabling dynamic and flexible artist information retrieval.

Guild's Music512 aimed to simplify live concert discovery through a location-based, browsable interface for general audiences [10]. However, it ignored dedicated fan communities entirely, offered no artist-loyalty features, and excluded international fandoms like K-pop where geography matters far less than artist allegiance. MixFan recenters the experience around artist loyalty rather than location, making it genuinely built for how K-pop fans actually behave.

Lee and Lau's study on Weverse highlighted its success in building fan-artist relationships within a label-controlled ecosystem [11]. Its critical limitation is exclusivity, only Hybe-affiliated artists are supported, locking out the majority of K-pop artists. mixfan removes this barrier entirely by being label-agnostic, while adding concert tracking and AI assistance that Weverse does not provide.

My idea is to make an app which can organize Kpop albums that have the best prices and find album types as much as I can, also, list the artists' concert events that users followed. When users open the my app — mixfan, they can add their bias, then users can check her/his events and also check the other artists' events. Even though there are a lot of apps that can find events and merchandise, they are separated, so the information is separated. mixfan is a "recombination" app, so fans can check information on one app to save more time and energy.

Mixfan's two experiments targeted the most critical blind spots in the app's core functionality. The first experiment tested the AI's accuracy in recalling tour and fan-meeting information across ten K-pop groups ranging from major acts to small-scale and disbanded artists. Using verified Ticketmaster and official profile data as the control, each AI response was scored against the actual number of tours per artist. The mean accuracy was 63.3%, with the most significant finding being that disbanded group IZ*ONE scored 100% while active artists like aespa and ARTMS scored only 33.3%, revealing that the AI's training data cutoff date is the dominant factor, fixed historical data is captured completely while ongoing active careers are perpetually underrepresented. The second experiment tested MixFan's search recognition across three input types: typo variations, Korean names, and formal English names, applied to six artist groups. The mean accuracy was 33.3%, with Korean name inputs and typos failing universally across all groups, scoring 0% without exception. Only exact formal English name entries returned successful results, exposing that MixFan's search relies entirely on strict string matching against Ticketmaster's English only database. Together, both experiments reveal two foundational weaknesses: an AI that cannot keep pace with actively evolving artist careers, and a search system that excludes the natural search behavior of its core K-pop audience.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Selecting and Coordinating Multiple Music and Event Apis

Choosing an applicable API is a big challenge for me. Because every APIs have their own advantages, some are more accurate on artists' profiles, some are more precise on events. Multiple APIs, including Spotify, YouTube, Concerts API, and Ticketmaster, were evaluated [5][6][13][14]. Spotify API is perfect for searching artists, but it cannot provide concert information. Youtube API is good for both artist and music information, but it still doesn't work for concert info. One API can be slow or fail, artist names/IDs may not match across services,

and results can arrive out of order (older search finishes after a newer search). Timeout and retry mechanisms, partial result handling, and artist identifier normalization can be implemented (prefer IDs over names), and use request “tokens” or cancellation so only the latest search updates the UI and gets saved. Concert - Artists Events Tracker API and Ticketmaster API are suitable for all the artist information and concert events.

2.2. Managing AI Cost, Safety, And Responsiveness

Cost/rate limits, unsafe or irrelevant responses, and slow streaming that makes the chat feel unresponsive. Common responses can be cached, request rates limited, and per-user usage constraints enforced. I could also add moderation/filters and prompt constraints to keep responses on-topic, plus optimistic UI updates and streaming to keep the conversation feeling fast.

2.3. Securing Data Access And Consistency In Firestore

Unauthorized reads/writes, users editing data that isn’t theirs, and inconsistent profile state between device and Firestore (offline edits, conflicts) [15]. Strict Firestore security rules tied to auth.uid can be enforced, validate updates (required fields, allowed values), and use a clear sync strategy (local state + server source of truth, conflict resolution, and offline persistence with safe merges) [12].

3. SOLUTION

The main structure of my program is API, OpenAI, Flutter, and Firebase [16][17]. I use API keys for artists and events information. In order to get the basic information of artists, and the date, time zone, and place of the concert events through JSON to use OpenAI to connect API keys to help me analyze and provide the information. AI is the most frequently used tool in my app, especially in some unobserved parts. I use OpenAI to create some visual things and figure out the problems. For example, I use AI to fill the pink color into my app's final icon’s kitty; I use AI to fix the code that I cannot find the problem with; I use OpenAI to help users to solve their problem about the app and other questions about music… I use Flutter to write the code and build my app. This framework helps me to create mixfan and compiles the code to publish it on different platforms. I use Firebase to save all the data such as user info and information which is getting in the API.

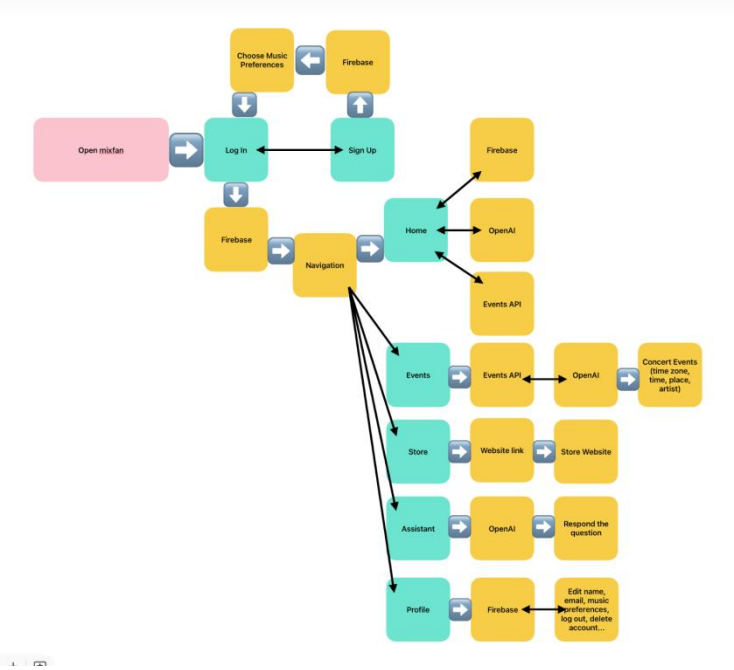
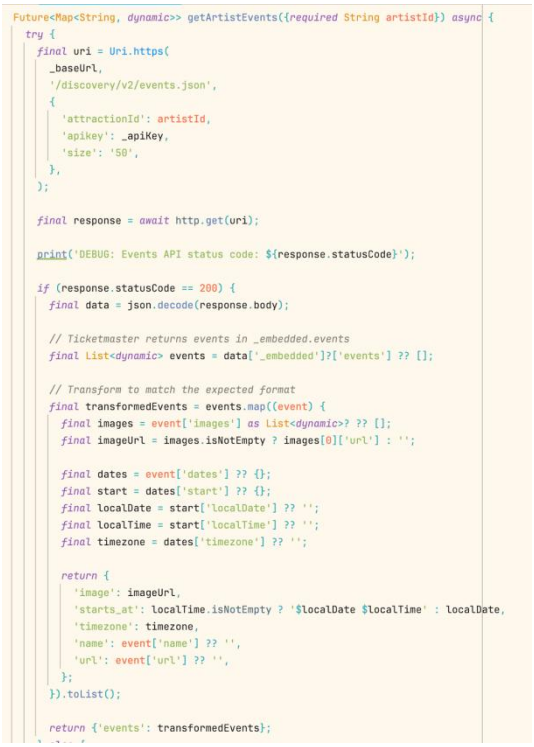


Figure 1. Overview of the solution

This component fetches upcoming concert events for a specific artist. It relies on the Ticketmaster API, chosen after testing Spotify, YouTube, and Concerts API — all of which lacked reliable event data. The component uses REST API integration as its core concept, sending authenticated HTTP requests and parsing JSON responses. Broadly, it queries Ticketmaster's database and feeds structured event data into MixFan's events screen.

Figure 2. User Interface of Event Retrieval Module



```dart
Future<Map<String, dynamic>> getArtistEvents({required String artistId}) async {
  try {
    final uri = Uri.https(
      _baseUrl,
      '/discovery/v2/events.json',
      {
        'attractionId': artistId,
        'apikey': _apiKey,
        'size': '50',
      },
    );

    final response = await http.get(uri);

    print('DEBUG: Events API status code: ${response.statusCode}');

    if (response.statusCode == 200) {
      final data = json.decode(response.body);

      // Ticketmaster returns events in _embedded.events
      final List<dynamic> events = data['_embedded']?['events'] ?? [];

      // Transform to match the expected format
      final transformedEvents = events.map((event) {
        final images = event['images'] as List<dynamic>? ?? [];
        final imageUrl = images.isNotEmpty ? images[0]['url'] : '';

        final dates = event['dates'] ?? {};
        final start = dates['start'] ?? {};
        final localDate = start['localDate'] ?? '';
        final localTime = start['localTime'] ?? '';
        final timezone = dates['timezone'] ?? '';

        return {
          'image': imageUrl,
          'starts_at': localTime.isNotEmpty ? '$localDate $localTime' : localDate,
          'timezone': timezone,
          'name': event['name'] ?? '',
          'url': event['url'] ?? '',
        };
      }).toList();

      return {'events': transformedEvents};
```

Figure 3. Implementation of Event Retrieval Function (getArtistEvents)

Fetching Concert Events getArtistEvents is a Dart function that runs whenever your app needs to display upcoming shows for a specific artist. It starts by building a URL pointed at Ticketmaster's discovery API, attaching the artist's ID, your API key, and a cap of 50 results as query parameters, then fires an HTTP GET request to Ticketmaster's backend server, which searches its events database and returns matching data as JSON. Once the response comes back with a 200 status, the function parses the JSON, digs into the _embedded.events array, and loops through each event to extract and reformat the relevant details — image URL, date, time, timezone, name, and ticket link — into a cleaner structure before returning everything as a single map under the key ‘events’.

This component provides users with an intelligent in-app assistant. It uses OpenAI's GPT-4.1-mini model, relying on Natural Language Processing (NLP) — a concept where machines understand and generate human language by recognizing patterns in text. A system prompt constrains the AI to K-pop relevant topics, keeping responses useful and on-topic. Broadly, it receives the user's conversation history, sends it to OpenAI's servers, and returns a context-aware reply directly in the chat interface.

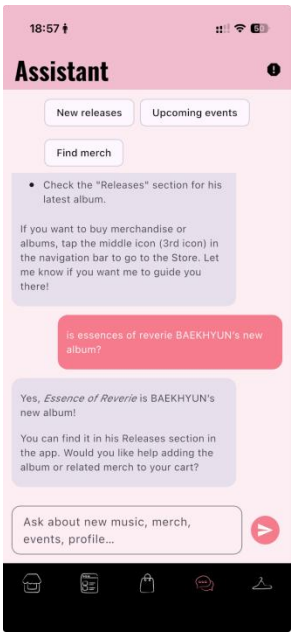


Figure 4. AI Assistant Interaction Interface

```
Future<String> _callOpenAI(List<Map<String, String>> messages) async {
  final apiKey = dotenv.env['OPENAI_API_KEY'] ?? '';
  if (apiKey.isEmpty) {
    throw Exception(
        'OPENAI_API_KEY missing. Ensure assets/.env exists and is listed in pubspec.yaml.');
  }

  const endpoint = 'https://api.openai.com/v1/responses';
  const model = 'gpt-4.1-mini';

  const systemPrompt = '''
You are the in-app assistant for a K-pop artist companion app.
- Help users discover new releases (singles/EPs/albums) for followed artists.
- Guide them to the Store (merch, which of 5 icons in the navigation bar is the middle icon (3rd icon)), Events (upcoming shows), and Profile (cart, payment).
- Be concise; use bullets for lists.
- If the user asks to "add to cart" or similar, describe the taps within the app instead of performing actions.
''';

  final inputItems = <Map<String, dynamic>>[
    {"role": "system", "content": systemPrompt},
    for (final m in messages) {"role": m["role"], "content": m["content"]},
  ];

  final body = jsonEncode({
    "model": model,
    "input": inputItems,
    "temperature": 0.5,
    // "max_output_tokens": 600, // optional
  });

  final res = await http.post(
    Uri.parse(endpoint),
    headers: {
      "Authorization": "Bearer $apiKey",
      "Content-Type": "application/json",
    },
    body: body,
  );

  if (res.statusCode < 200 || res.statusCode >= 300) {
    throw Exception("OpenAI error ${res.statusCode}: ${res.body}");
  }
```

Figure 5. OpenAI API Integration Function (_callOpenAI)

_callOpenAI is the function that powers mixfan's in-app chatbot, running every time a user sends a message. It begins by retrieving your OpenAI API key from a .env file and throwing an immediate error if it's missing, then sets the model to gpt-4.1-mini — a lightweight, cost-efficient choice for a high-traffic app [19]. It constructs the conversation by prepending a system prompt that instructs GPT to act as a K-pop companion assistant, keeping responses concise and guiding users toward merch, events, and new releases, then appends the full message history so the AI retains context across the conversation [18]. The whole thing gets JSON-encoded with a temperature of 0.5 for balanced responses and sent via HTTP POST to OpenAI's API, which processes the conversation through GPT and returns a reply — and if anything outside a success status comes back, the function throws an error rather than failing silently.

This component loads and displays personalized user data on the profile screen. It uses Firebase Authentication and Firestore, relying on authentication as its core concept — a system that verifies user identity and restricts data access accordingly. Each query is tied to the user's unique auth.uid, ensuring users only access their own data. Broadly, it retrieves identity info and artist bias counts from Firebase and progressively renders them onto the profile screen.

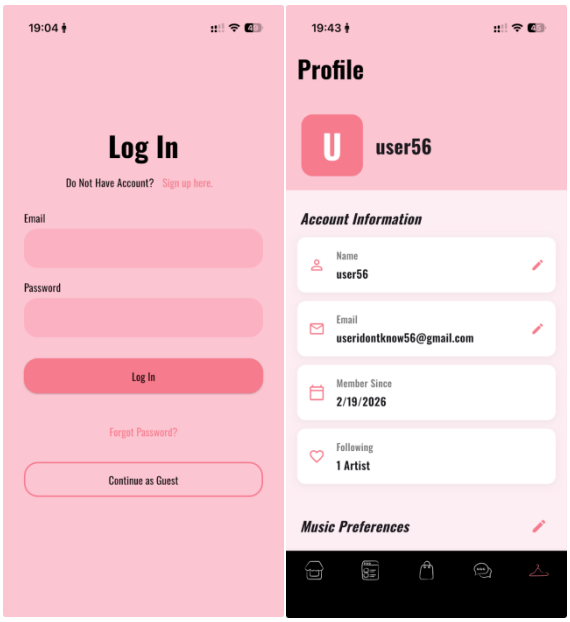


Figure 6. User Profile Interface with Personalized Data

```
Future<void> _loadUserData() async {
  if (user == null) return;

  // Get data from Firebase Auth
  setState(() {
    email = user!.email ?? 'No email';
    displayName = user!.displayName ?? 'User';
    memberSince = user!.metadata.creationTime;
  });

  // Get data from Firestore
  try {
    final userDoc = await FirebaseFirestore.instance
        .collection('users')
        .doc(user!.uid)
        .get();

    if (userDoc.exists) {
      final data = userDoc.data();
      setState(() {
        preferences = (data?['preferences'] as List?)?.cast<String>() ?? [];
        displayName = data?['name'] ?? user!.displayName ?? 'User';
      });
    }

    // Get bias count
    final biasSnapshot = await FirebaseFirestore.instance
        .collection('users')
        .doc(user!.uid)
        .collection('bias')
        .get();

    setState(() {
      biasCount = biasSnapshot.docs.length;
    });
  } catch (e) {
    print('Error loading user data: $e');
  }
}
```

Figure 7. Firebase Data Retrieval Function (_loadUserData)

_loadUserData runs when a user opens their profile page, pulling together everything needed to populate it from two Firebase sources. It first checks that a user is actually logged in before doing anything, then immediately grabs the basics — email, display name, and account creation date —

straight from Firebase Auth. From there it queries Firestore, finding the user's document by their unique ID to retrieve richer stored data like their custom name and preferences list, updating the UI progressively with each setState call rather than waiting for all data to load at once. It also queries a nested bias sub-collection to count how many artists the user follows, storing that number as biasCount — a very K-pop native data model choice. Any Firestore errors are caught and logged without crashing the app.

4. EXPERIMENT

4.1. Experiment 1

AI's accuracy is really important. Because mixfan is a Kpop fans platform for fans to follow and check their artists' events and basic information. If the AI is not able to manage the info that comes from the API, users may not find their biased artists and correct events, which is against my app's original intention. An inaccurate AI response could mislead users or surface incorrect event details, directly undermining the app's core purpose.

To test this, A control dataset table will be compiled of verified artist information and upcoming events sourced directly from the Ticketmaster API [6] [8] and official artist profiles, representing the known "ground truth." The AI will be prompted with standardized artist-related queries a typical mixfan user would ask and record its responses. Each response will be compared against the control data across categories like artist name accuracy, event date, venue, and general information correctness. Results will be organized into a comparison table showing the expected value versus the AI's response, and each answer will be marked correct or incorrect. From this I will calculate an overall accuracy percentage and assign the AI a score. The experiment is structured this way because a table format makes discrepancies easy to identify and quantify, and using real API data as the control ensures the benchmark reflects what users would actually encounter in the app.

AI Accuracy Test Table

Test groups (all tour/ fan-meeting information accuracy)	Actual (expected)	AI	Data accuracy	Grade (1 - 5 perfect)
Test 01: BAEKHYUN	3	1	33.333%	1.5
Test 02: TWICE	6	4	66.667%	3
Test 03: BLACKPINK	4	3	75.000%	4
Test 04: NCT 127	4	3	75.000%	4
Test 05: NewJeans	2	1	50.000%	2.5
Test 06: aespa	3	1	33.333%	1.5
Test 07: nmixx	1	1	100.000%	5
Test 08: EVERGLOW	3	2	66.667%	3
Test 09: ARTMS	3	1	33.333%	1.5
Test 10: I*ZONE	3	3	100.000%	5

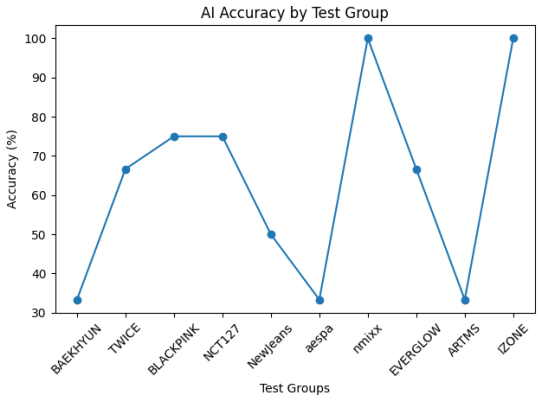


Figure 8. AI accuracy result

The ten tests produced an average accuracy of 63.3%, with a median of 66.7%. The highest accuracy was 100%, achieved by both NMIXX and IZ*ONE, while the lowest was 33.3%, shared by BAEKHYUN, aespa, and ARTMS [7][8]. The most surprising result was IZ*ONE scoring 100% despite being disbanded since 2021. Contrary to expectations, a dissolved group outperformed several active, well-known artists. This occurred because IZ*ONE's historical tour data is fixed and complete — it will never change, meaning the AI's training data captured it fully. On the other hand, aespa, despite being one of SM Entertainment's biggest active girl groups, scored only 33.3%, matching the small-scale group ARTMS. This was unexpected given Aespa's mainstream popularity. The biggest factor affecting these results is the AI's training data cutoff date. Active groups continuously add new tours and events, but the AI cannot update itself in real time, causing significant gaps for currently active artists while paradoxically performing better on groups that no longer produce new data.

4.2. Experiment 2

A blind spot I want to test is MixFan's search recognition accuracy. If users cannot find their favorite artists using Korean names or slightly misspelled inputs, they will be unable to access any features the app offers, making reliable search functionality essential to the entire user experience.

I tested five K-pop groups: BAEKHYUN, TWICE, NCT DREAM, ITZY, IVE, and Hearts2Hearts, using three search input types per group: a typo variation, the Korean name, and the correct formal name. Each search was entered into MixFan's search function and recorded as either successful (100%) or unsuccessful (0%). The correct formal name serves as the control data, establishing the baseline that the app should always return a result for. This structure was chosen because it isolates exactly which input type causes the search to fail, making the results easy to compare and analyze across all groups.

Group Name Searching Accuracy

Group name	Typo/ Korean name/Formal Name	Accuracy %	Grade (1 ~ 10)
BAEKHYUN	BEAK HYUN	0	0
BAEKHYUN	백현	0	0
BAEKHYUN	BAEKHYUN	100	10
TWICE	TWOCE	0	0
TWICE	트와이스	0	0
TWICE	TWICE	100	10
NCT DREAM	NTC DREAM	0	0
NCT DREAM	엔시티 드림은	0	0
NCT DREAM	NCT DREAM	100	10
ITZY	IZTY	0	0
ITZY	있지	0	0
ITZY	ITZY	100	10
IVE	IBE	0	0
IVE	아이브	0	0
IVE	IVE	100	100
Hearts2Hearts	Heart2Heart	0	0
Hearts2Hearts	하츠투하츠	0	0
Hearts2Hearts	Hearts2Hearts	100	100

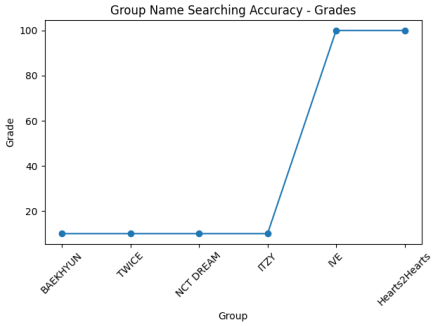


Figure 9. Group name searching for accuracy result

Across 18 tests, the mean accuracy was 33.3% and the median was 0%, reflecting that the majority of search inputs failed. The highest value was 100%, consistently achieved only when the exact formal English name was entered. The lowest value was 0%, shared by every typo and every Korean name input across all groups. The most surprising finding was that Korean name searches failed for every single group without exception — including major groups like TWICE (트와이스) and ITZY (있지) where the Korean name is widely recognized. This was unexpected because K-pop fans naturally search in Korean, and failing to support this completely excludes a significant portion of the target audience. The biggest factor affecting these results is that MixFan's search relies on exact string matching against the Ticketmaster API, which stores artist names in English only, meaning any deviation from the precise English spelling returns nothing. This reveals a critical improvement needed : implementing fuzzy search or Korean-to-English name mapping to make the app genuinely accessible to its K-pop fan user base.

5. RELATED WORK

Mylopoulos and Brodie's "Readings in Artificial Intelligence and Databases" (1989) explores how AI techniques can be integrated with database systems to improve data retrieval, knowledge representation, and intelligent querying [9]. Their framework proposed using semantic data models and knowledge-based systems to allow computers to interpret and retrieve structured information more meaningfully than traditional keyword-based database queries. While groundbreaking for its time, the solution was limited to static, structured databases and assumed that all data is clean, complete, and pre-organized. It entirely ignored real-time data retrieval, unstructured user inputs, and the challenge of handling incomplete or missing records, all of which are central problems in a modern mobile application. Additionally, the framework had no consideration for natural language interaction with end users. mixfan directly addresses these

gaps by combining Firebase's real-time database with OpenAI's natural language processing, allowing users to query artist and concert information conversationally rather than through rigid structured searches, while also handling incomplete data gracefully through fallback responses.

Guild's "Discover Live Music with Music512" (2024) presents a concert discovery application designed to help users find live music events through a streamlined, user-friendly interface [10]. The solution works by aggregating local concert and venue data into a single browsable platform, allowing users to filter events by location and genre. The app was effective at reducing the friction of finding live music for general audiences, demonstrating through user testing that simplified navigation significantly improved event discovery rates. However, Music512 is limited to location-based discovery and serves a general music audience, completely ignoring dedicated fan communities who follow specific artists across multiple cities and countries. It also excludes any AI-powered assistance, personalized artist tracking, or fan profile features. Most critically, it has no consideration for international fandoms like K-pop, where fans are highly artist-loyal rather than genre or location driven. mixfan improves on Music512 by centering the entire experience around artist loyalty rather than geography, integrating real-time Ticketmaster event data for global concert tracking, and adding an AI chatbot and personalized bias system that transforms passive event discovery into an active, personalized fan experience.

Lee and Lau's "Uniting Global Fans and Revolutionizing K-pop Marketing: Weverse" (2025) examines how Weverse functions as a global K-pop fan community platform by uniting fans and artists through direct communication, exclusive content, and merchandise integration [11]. The solution works by creating a label-controlled ecosystem where artists post updates, fans interact in community feeds, and purchases are made within the same platform. The study found Weverse effective at strengthening parasocial relationships between fans and artists, driving engagement and revenue simultaneously. However, its most significant limitation is exclusivity, Weverse only supports artists signed under Hybe Corporation or select partner labels, meaning the vast majority of K-pop artists are entirely absent from the platform. It also provides no real-time concert event tracking, no AI assistant for fan queries, and ignores fans who follow independent or smaller-scale artists. MixFan directly improves on Weverse by being completely artist-agnostic, allowing fans to follow and track any K-pop artist regardless of label affiliation, while additionally integrating live concert discovery through Ticketmaster and an AI chatbot — features Weverse does not offer at all.

6. CONCLUSIONS

Mixfan has three primary limitations identified through development and experimentation. First, the AI accuracy remains the most critical issue, as demonstrated in Experiment A, the average accuracy across ten artist tests was only 63.3%, with newly active or smaller-scale artists scoring as low as 33.3%. This means fans risk receiving outdated or incomplete information about their biased artists. Given more time, I would integrate a real-time web search layer alongside GPT to supplement its outdated training data, ensuring responses reflect current information [18]. Second, the app's UI transitions and page animations could be significantly smoother, certain screens feel abrupt when loading data from Firebase or the Ticketmaster API, which disrupts the overall user experience. I would refactor these using Flutter's animation controllers for more fluid transitions. Third, mixfan currently requires manual effort to stay updated, artist events and merchandise are not automatically refreshed. I would implement scheduled background jobs using Firebase Cloud Functions to automatically sync new events and merch data at regular intervals, making the app feel truly live and effortless for users [15].

Mixfan successfully delivers a centralized K-pop fan experience by combining artist tracking, real-time concert discovery, AI assistance, and personalized profiles in one dedicated platform.

The app brings together everything a K-pop fan needs, from finding their favorite artists to discovering upcoming concerts, making it a meaningful and practical tool for the global K-pop community.

REFERENCES

[1] Parc, Jimmyn, and Shin Dong Kim. "The digital transformation of the Korean music industry and the global emergence of K-pop." Sustainability 12.18 (2020): 7790.

[2] Andira, Nabila Putri, Raditya Adji Prasetyo, and Artha Sejati Ananda. "The impact of K-Pop idol on brand awareness, hedonic shopping motivation, and purchase intention." Kajian Branding Indonesia 5.1 (2023): 1-15.

[3] Silva, Letícia Maria Lacerda da. "Hallyu, vinil e design gráfico: releitura estética do álbumNoeasy do grupo de K-POP STRAY KIDS." (2024).

[4] Kim, Yeojin. "Examining the Global Popularity of Korean Pop Culture During the Last Three Decades." Contemporary Asian Popular Culture Vol. 2: Cultural Dynamics and Global Impact. Cham: Springer Nature Switzerland, 2024. 83-108.

[5] Apruzzese, Jeffrey. "Concerts as Catalysts: Driving Digital Streams and Physical Sales in the Modern Music Ecosystem." The Journal of Arts Management, Law, and Society (2025): 1-26.

[6] Siu, Vince S., et al. "API management in digital health: exploring IBM API connect and IBM API hub in enabling healthcare innovation." 2024 IEEE International Conference on Digital Health (ICDH). IEEE, 2024.

[7] Glott, Ruediger, Philipp Schmidt, and Rishab Ghosh. "Wikipedia survey–overview of results." United Nations University: Collaborative Creativity Group 8 (2010): 1158-1178.

[8] Farbman, Jake. "You Wanted the Best? You Got the Best! If You Can Pay for It... Antitrust Considerations and Artist-Based Solutions for Lower Concert Ticket Prices." J. Intell. Prop. L. 32 (2025): 1.

[9] Mylopoulos, John, and Michael L. Brodie, eds. Readings in artificial intelligence and databases. Morgan Kaufmann, 1989.

[10] Guild, Sophia. "Discover Live Music with Music512: Developing a user-friendly concert discovery app." Proceedings of the 42nd ACM International Conference on Design of Communication. 2024.

[11] Lee, Jong Min, and Gabriel Lau. "Uniting global fans and revolutionizing K-pop marketing: Weverse, a global fan community platform." Asia Pacific Business Review (2025): 1-23.

[12] Chougale, Pankaj, et al. "Firebase-overview and usage." International Research Journal of Modernization in Engineering Technology and Science 3.12 (2021): 1178-1183.

[13] Zhang, Boxun, et al. "Understanding user behavior in spotify." 2013 Proceedings IEEE INFOCOM. IEEE, 2013.

[14] Snelson, Chareen. "YouTube across the disciplines: A review of the literature." MERLOT Journal of Online learning and teaching (2011).

[15] Sukmana, Yuda, and Yusep Rosmansyah. "The use of cloud firestore for handling real-time data updates: An empirical study of gamified online quiz." 2021 2nd International conference on electronics, communications and information technology (CECIT). IEEE, 2021.

[16] Olsson, Matilda. "A Comparison of Performance and Looks Between Flutter and Native Applications: When to prefer Flutter over native in mobile application development." (2020).

[17] Roumeliotis, Konstantinos I., and Nikolaos D. Tselikas. "Chatgpt and open-ai models: A preliminary review." Future Internet 15.6 (2023): 192.

[18] Welsby, Philip, and Bernard MY Cheung. "ChatGPT." Postgraduate Medical Journal 99.1176 (2023): 1047-1048.

[19] Auger, Tom, and Emma Saroyan. "Overview of the openaiapis." Generative AI for Web Development: Building Web Applications Powered by OpenAI APIs and Next. js. Berkeley, CA: Apress, 2024. 87-116.