International Refereed Journal of Engineering and Science (IRJES)
ISSN (Online) 2319-183X, (Print) 2319-1821
Volume 15, Issue 4 (Jul. – Aug. 2026), PP. 09-18

FLYBY: A Real-Time LLM-Based Messenger for Context-Aware Travel Assistance

YunSe Lee*, YoungPyo Ko*, Songwook Lee**
** (student, Major of Railroad Data Science, Korea National University of Transportation, Uiwang-si, Korea)*
*** (professor, Major of Railroad Data Science, Korea National University of Transportation, Uiwang-si, Korea)*
Corresponding Author: Songwook Lee

Abstract: *Travel planning often requires users to search across multiple fragmented platforms for accommodations, restaurants, and attractions, resulting in inefficient information retrieval and limited contextual continuity. This study proposes FLYBY, a real-time conversational travel recommendation system that integrates a large language model (LLM), a mobile messenger interface, and external travel service APIs within a unified architecture. The proposed system employs a Kotlin-based Jetpack Compose client, Firebase for real-time synchronization and authentication, and a FastAPI server that performs semantic analysis of user input through the OpenAI API. The server extracts core travel parameters, including destination, schedule, traveler composition, and preference filters, and dynamically translates them into structured requests to external services such as Booking.com and Google Places. The resulting recommendations are delivered through a real-time chat interface that supports contextual interaction and direct access to linked service information. The system was validated through accommodation, restaurant, and attraction recommendation scenarios, and its usability was further examined through a preliminary user satisfaction survey. Across the three scenarios, the overall mean scores were 4.5 for ease of use, 4.4 for recommendation relevance, 3.9 for response speed, 4.4 for conversational convenience, and 4.5 for overall satisfaction on a five-point Likert scale. These results confirm the feasibility of transforming unstructured conversational input into service-specific API queries and suggest that the proposed system can provide a positive and practical user experience for integrated travel information retrieval within a single mobile environment.*

Keywords: *large language model, real-time communication, conversational interface, travel recommendation, context-aware system*

Date of Submission: 09-07-2026 Date of acceptance: 20-07-2026

I. INTRODUCTION

Travel planning increasingly depends on digital services that provide access to accommodations, restaurants, transportation, and tourist attractions. However, the travel information ecosystem remains highly fragmented, requiring users to move across multiple applications and web services to compare alternatives and organize their itineraries [1,2]. This fragmentation reduces search efficiency, disrupts contextual continuity, and limits the ability of digital systems to provide recommendations that reflect evolving user preferences and conversational intent.

Recent advances in large language models (LLMs) have created new opportunities for conversational travel assistance. Unlike conventional rule-based or keyword-based chatbots, LLM-based systems can interpret natural language input, preserve conversational context across multiple turns, and generate more personalized responses. In tourism and hospitality research, these capabilities have been recognized as a promising basis for intelligent travel support, especially in situations that require flexible and user-centered exploration of travel information [3].

Despite these developments, many existing travel-related chatbot systems remain limited in scope or fragmented in functionality. In practice, users are still often required to switch among separate services for hotels, restaurants, and attractions. In addition, conversational interfaces frequently lack continuity across multiple requests, reducing their usefulness for sequential travel planning tasks. To address these limitations, this study proposes FLYBY, a real-time messenger-based travel recommendation system that combines LLM-driven semantic analysis, external travel service APIs, and a mobile chat interface within a unified architecture.

The main contributions of this study are as follows. First, we design a unified conversational architecture that supports accommodation, restaurant, and attraction recommendations within a single real-time messaging environment. Second, we implement a server-side semantic processing pipeline that extracts structured travel parameters from lightweight user messages and dynamically maps them to external API requests. Third, we

present a proof-of-concept mobile implementation based on Jetpack Compose, Firebase, and FastAPI, and evaluate it through representative recommendation scenarios and a preliminary user satisfaction survey.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the system architecture and implementation. Section IV presents validation scenarios and discusses system operation. Section V concludes the paper and outlines future research directions.

II. RELATED WORK

Current travel-oriented chatbot services often rely on rigid question-and-answer structures and therefore provide limited support for complex travel planning tasks [4]. Pillai and Sivathanu [5] observed that although AI chatbots have substantial potential in hospitality and tourism, many existing deployments do not maintain the contextual continuity required for comprehensive assistance. In many cases, travel-related services are implemented as separate functional modules, preventing users from exploring information in a continuous conversational flow.

The emergence of LLMs [6] has advanced conversational systems beyond simple keyword matching toward context-aware natural language interaction. Gursoy et al. [2] noted that the integration of natural language processing technologies is accelerating a shift toward more personalized service delivery in tourism. Building on this trend, LLM-based systems can support flexible interpretation of user requests and enable travel-related services to respond to more nuanced and context-dependent needs.

Wong et al. [7] further showed that generative AI can support autonomous travel decision-making by understanding user preferences expressed in natural language. This suggests that LLMs can serve as an effective semantic interface between human travel requests and structured service platforms. At the same time, Chen et al. [8] demonstrated the value of unified multi-service architectures in AI-assisted travel planning, highlighting the need for systems that can aggregate multiple information sources within an integrated interaction framework.

Nevertheless, three limitations remain in prior work. First, many systems focus on a single service domain rather than integrated multi-service recommendation. Second, conversational continuity is often insufficient for sequential travel planning tasks that require accumulated context. Third, relatively few studies describe deployable mobile architectures that combine real-time messaging, LLM-based semantic parsing, and external travel APIs within a single operational pipeline.

The proposed system addresses these limitations by integrating heterogeneous external services, including the Booking.com API [9] and the Google Places API [10], into a real-time mobile messenger environment. User input is semantically analyzed on the server side, converted into structured service requests, and returned through a persistent conversational interface. In this way, the system aims to improve continuity, convenience, and contextual relevance in travel information retrieval.

III. SYSTEM IMPLEMENTATION

3.1 Overall System Architecture

The proposed system adopts a modern client-server architecture composed of a mobile client, Firebase services, and a FastAPI-based AI server. The mobile client is implemented using Kotlin-based Jetpack Compose [11], which provides a reactive and state-driven user interface. The backend consists of Firebase for real-time synchronization, authentication, and storage, and a FastAPI server that performs semantic analysis and external API orchestration.

Firebase Firestore is used for real-time data synchronization, while Firebase Authentication manages user identity and Firebase Storage handles image transmission. The FastAPI server acts as the central natural language processing gateway. It invokes the OpenAI API [14] to analyze user messages, extract travel-related parameters, and route requests to appropriate external services such as Booking.com and Google Places. The overall pipeline follows an event-driven flow: user input → semantic analysis → API aggregation → real-time UI rendering.

This architecture was designed to reduce client-side complexity by sending only minimal request data from the mobile application and concentrating semantic interpretation and recommendation logic on the server side. As a result, the client remains lightweight while the server performs context-aware recommendation processing.

3.2 User Input Processing Flow

The system is designed to collect user input through a real-time chat interface and invoke AI-based recommendation functions when needed. Users can enter messages either in a regular chat room or in a dedicated AI chatbot screen. User messages are stored in Firebase Firestore and immediately propagated to participants in the same chat room through a Snapshot Listener. The chat interface reflects newly stored messages in real time regardless of whether the message is intended for standard user-to-user conversation or for AI processing.

In regular chat rooms, an AI toggle button allows users to determine whether a given message should be processed by the recommendation system. Only when this option is activated is the message forwarded to the FastAPI server. In contrast, the dedicated AI chatbot screen always operates with AI response generation enabled. The client transmits a lightweight payload consisting of only three fields: user ID, message text, and chat room ID. All semantic interpretation, including intent classification and filter extraction, is performed on the server side.

For example, if a user enters a request such as "Recommend a hotel in Tokyo," the server classifies the message as a travel recommendation query, identifies the service type and destination, and triggers the appropriate external API call. In contrast, casual greetings or ordinary user conversation are not treated as recommendation requests. The generated response is stored back in Firestore as a message object and rendered in real time on the Compose-based UI through the ViewModel structure.

The system also includes a conversation context reset mechanism. When a user enters a designated reset keyword, the client sends a reset request to the server, which clears the stored conversation history and extracted filters for that session. In addition, entering the dedicated AI chatbot screen automatically initializes a fresh conversational context. This design prevents accumulated context from previous interactions from interfering with new recommendation requests.

3.3 Firebase Integration Structure

The proposed system uses Google Firebase [12] as the foundation for real-time data processing, authentication, media handling, and notification delivery. User authentication is implemented through Firebase Authentication with Google account-based OAuth, enabling secure and convenient user identification. Authenticated user information functions as a unique identifier for chat participation, profile management, and user-specific interaction tracking.

Firebase Firestore serves as the real-time database and stores core data entities including chat messages, chat room information, user profiles, and AI-generated recommendation results. By using the Snapshot Listener mechanism, changes in stored data are immediately reflected in the client interface, enabling real-time chat room updates and message transmission without perceptible delay.

Firebase Storage supports image attachment and transmission. When a user uploads an image, the media file is stored in Firebase Storage and the generated download URL is linked to the corresponding Firestore document. Firebase Cloud Messaging (FCM) is used for push notifications so that new messages can be delivered reliably even when the application is running in the background.

To improve communication security, the system also integrates Firebase App Check. An App Check interceptor is attached to the HTTP client responsible for server communication, and a verified attestation token is automatically appended to outgoing requests. This mechanism helps ensure that only requests originating from the authorized mobile application are accepted by the backend, thereby reducing the risk of unauthorized API access or forged requests.

3.4 AI Server Architecture

The AI server is implemented using the Python-based FastAPI framework [13] and is responsible for semantic analysis of user input, external API integration, and session-level context maintenance. It acts as an intermediary processor between the mobile client and Firebase. The client sends a minimal payload consisting of the user ID, message text, and chat room ID, while all natural language understanding is performed on the server.

When a message is received, the server analyzes it through the OpenAI API [14] and extracts core information needed for travel recommendation, including destination, travel duration or departure date, number of travelers, and preference filters such as hotel conditions or dining requirements. These extracted parameters are then mapped to the appropriate external service.

For accommodation queries, the system uses the Booking.com API to retrieve hotel information based on the identified destination and user constraints. For restaurant and attraction recommendations, the server calls the Google Places API, combining destination information with semantically extracted preference cues. The response is organized into structured JSON data and returned to Firebase for real-time display on the client.

A key feature of the server architecture is its management of session-specific context. The server stores conversation history, prior responses, and previously selected filters in association with the user and chat room. This enables the system to supplement missing information and generate more appropriate responses for incomplete or ambiguous queries. Accordingly, the server functions not only as an API gateway but also as a context-aware orchestration layer for conversational travel recommendations.

3.5 Main UI Composition

The user interface is implemented with Jetpack Compose and organized around state management through ViewModel components and real-time data binding from Firebase. The main interface consists of a chat screen, a chat room list, a community page, and an AI response presentation layer.

The chat screen functions as the central interaction space for real-time messaging, AI-based recommendation requests, and image sharing. Messages are visually distinguished according to sender type and displayed through a Compose LazyColumn that updates dynamically via Firestore synchronization. AI-generated recommendation messages contain structured HTML with clickable hyperlinks to external services such as Booking.com hotel pages and Google Maps locations. These are rendered within the chat interface using Android's TextView through Compose AndroidView interoperability and HtmlCompat parsing. This design allows users to move directly from conversational recommendations to detailed external service pages.

The chat room page displays the set of chat rooms in which the user participates, including each room's title, most recent message, and latest timestamp. The community page supports post creation and browsing, including title, image, text content, likes, and comments.

AI responses are displayed as structured recommendation lists that include item names, ratings, and addresses with embedded links. Recommendations are categorized by service type, including accommodations, restaurants, and attractions, and are visually distinguished for easier interaction. This interface design enables travel information to be consumed within the same conversational environment in which the request was originally made.

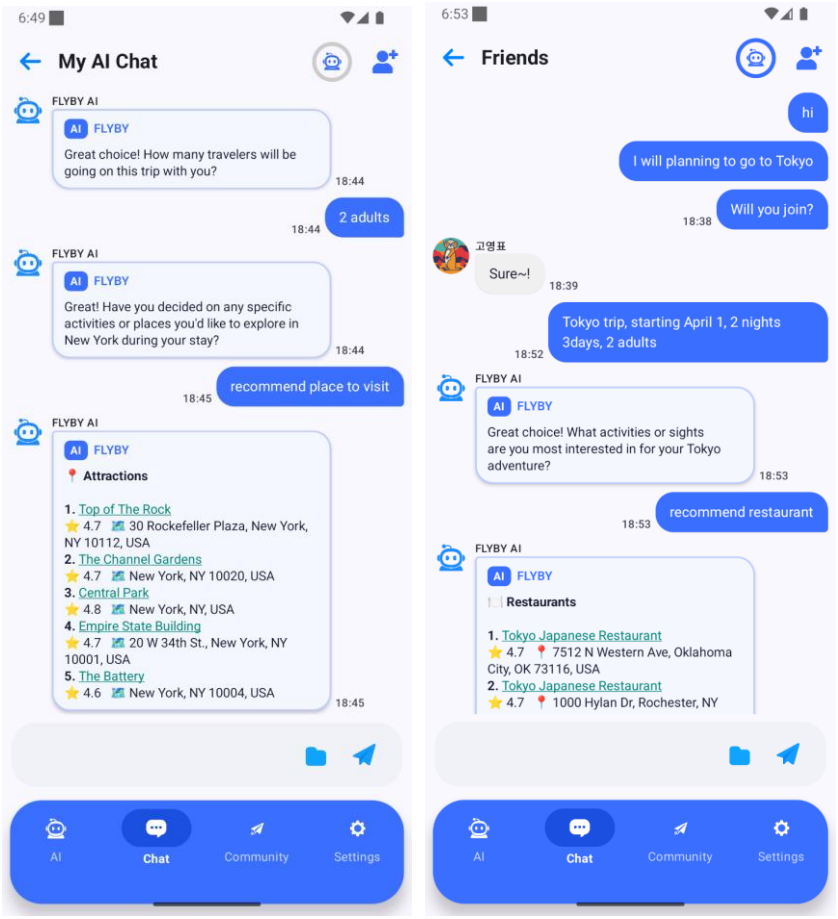

Fig. 1. An example of *Chat Screen: Real-time messaging interface with AI toggle and recommendation links. (Original UI in Korean)*

Chat Room Page: Displays a list of chat rooms in which the user is participating, including the chat room name, last message, and most recent timestamp.

Community Page: Users can write posts (title, image, body) or view others' posts, with like and comment functions included.

AI Response UI: AI recommendations are formatted as a structured list by the ViewModel receiving the server response. Each item includes the name, rating, and address with embedded links, allowing users to click and navigate to external services such as Google Maps and Booking.com. The response is categorized by service type (accommodations, restaurants, attractions), and each category is visually distinguished with corresponding emoji indicators.

IV. SYSTEM VALIDATION AND PRELIMINARY USER EVALUATION

To examine the functional feasibility and practical usability of the proposed system, this study conducted a proof-of-concept evaluation based on three representative travel recommendation scenarios: accommodation recommendation, restaurant recommendation, and attraction recommendation. The purpose of this evaluation was to verify whether the system could consistently transform natural language travel requests into service-specific API calls and return structured recommendation results through the real-time conversational interface. In addition to scenario-based validation, a preliminary user satisfaction and usability survey was conducted to assess user perceptions of the proposed interaction model.

4.1. Scenario 1: Accommodation Recommendation

In the accommodation scenario, the system processed a query such as "Find a hotel in Osaka." The semantic analysis module successfully identified the destination as Osaka and classified the request as an accommodation search task. Based on the extracted intent and location information, the server generated a Booking.com API request and retrieved relevant hotel results.

The returned response contained a structured list of recommended accommodations along with associated metadata such as hotel name, rating, and location-related information. Each item was linked to the corresponding external service page, allowing the user to access more detailed booking information directly from the conversational interface. This scenario demonstrates that the proposed system can convert unstructured user input into a structured accommodation query and provide real-time recommendations within the messenger environment.

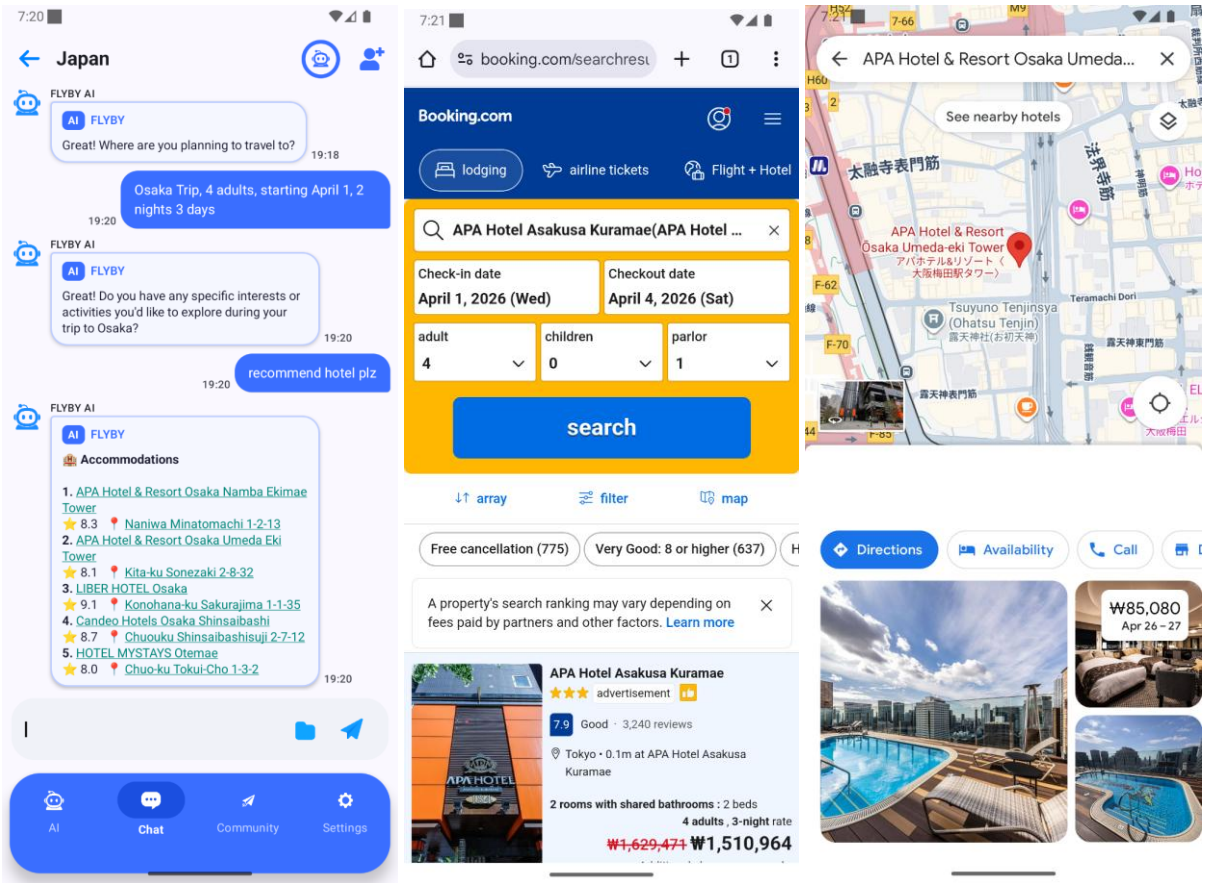

Fig. 2. Accommodation recommendation scenario: User query "Find an Osaka hotel" with AI-generated results via Booking.com API.

4.2 Scenario 2: Restaurant Recommendation

In the restaurant scenario, the user entered a dining-related request such as "Recommend a restaurant in Tokyo." The system extracted Tokyo as the destination and classified the request as a restaurant recommendation task. Based on this interpretation, the server invoked the Google Places API to retrieve dining options associated with the specified location.

The results were returned in a structured list containing restaurant names, ratings, and addresses. Each recommendation also included a direct link to its Google Maps page so that users could review additional details such as directions, images, and business information. This scenario confirms that the same conversational pipeline can support restaurant discovery tasks and provide location-aware recommendations in a consistent interaction format.

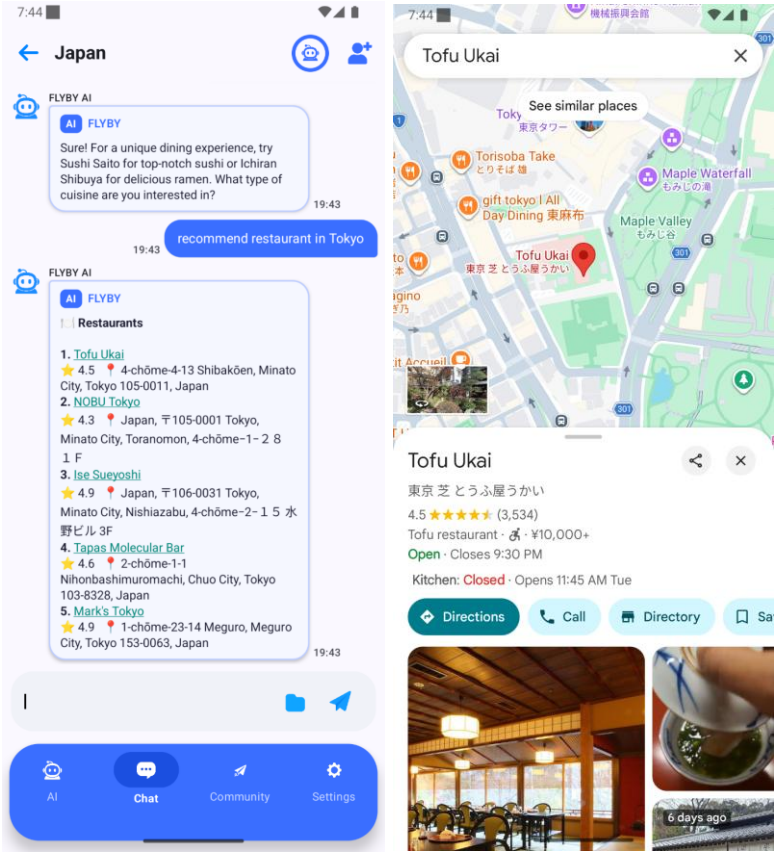

Fig. 3. Restaurant recommendation scenario: User query "Recommend a Tokyo restaurant" with AI-generated results via Google Places API.

4.3 Scenario 3: Attraction Recommendation

In the attraction scenario, the user submitted a query such as "Recommend places to visit in Melbourne." The system extracted Melbourne as the target destination and classified the request as an attraction recommendation task. It then called the Google Places API to retrieve candidate tourist attractions relevant to the identified city.

The recommended results were presented using the same structured format as in the previous scenarios, thereby maintaining interface consistency across service categories. Each attraction entry included essential descriptive information and an external link for additional exploration. This scenario demonstrates that the proposed architecture can support not only accommodations and dining but also tourist attraction discovery within a single conversational framework.

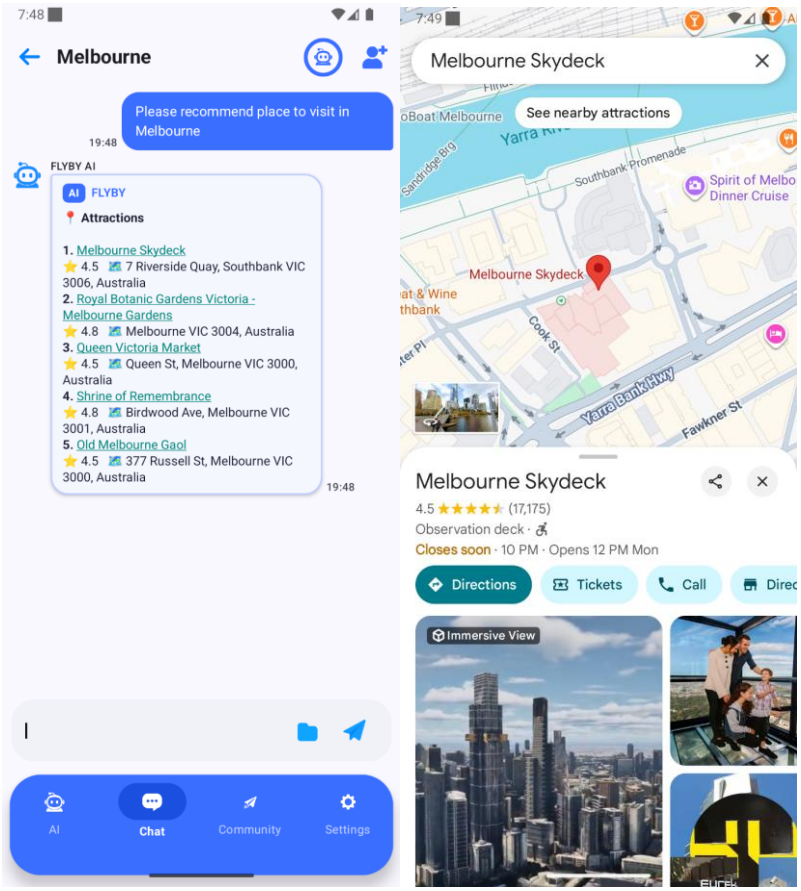

Fig. 4. Attraction recommendation scenario: User query "Recommend places to visit in Melbourne" with AI-generated results via Google Places API.

4.4 Preliminary User Satisfaction and Usability Survey

To complement the scenario-based validation, a preliminary usability survey was conducted to examine user perceptions of the proposed system. The survey focused on the same three representative scenarios: accommodation recommendation, restaurant recommendation, and attraction recommendation. For each scenario, five participants interacted with the system and then evaluated their experience using a five-point Likert scale, where 1 indicates strongly disagree and 5 indicates strongly agree.

The evaluation items were defined as follows: Q1, the system was easy to use; Q2, the recommended results were relevant to the user's request; Q3, the response was delivered quickly enough for practical use; Q4, the conversational interface was more convenient than manually searching across multiple applications; and Q5, the user was satisfied overall with the recommendation experience.

In the accommodation recommendation scenario, participants generally reported favorable experiences. The mean scores were 4.6 for ease of use, 4.4 for recommendation relevance, 4.0 for response speed, 4.6 for convenience, and 4.6 for overall satisfaction. These results indicate that users found the accommodation recommendation function intuitive and useful, although response speed was evaluated somewhat lower than the other dimensions.

Table 1. User satisfaction survey results for the accommodation recommendation scenario

Participant	Q1	Q2	Q3	Q4	Q5
P1	5	4	4	4	5
P2	5	5	3	5	5
P3	4	4	4	4	4
P4	5	5	5	5	5
P5	4	4	4	4	4
Mean	**4.6**	**4.4**	**4.0**	**4.6**	**4.6**

In the restaurant recommendation scenario, the system also received positive evaluations. The mean scores were 4.6 for ease of use, 4.6 for recommendation relevance, 4.0 for response speed, 4.4 for convenience,

and 4.5 for overall satisfaction. These findings suggest that participants perceived the restaurant recommendation process as effective and convenient within the chat-based interface.

Table 2. User satisfaction survey results for the restaurant recommendation scenario

Participant	Q1	Q2	Q3	Q4	Q5
P1	5	5	4	5	5
P2	5	5	3	4	5
P3	4	4	4	4	3.5
P4	5	5	5	5	5
P5	4	4	4	4	4
Mean	**4.6**	**4.6**	**4.0**	**4.4**	**4.5**

In the attraction recommendation scenario, participants likewise reported favorable perceptions of usability. The mean scores were 4.4 for ease of use, 4.1 for recommendation relevance, 3.8 for response speed, 4.3 for convenience, and 4.3 for overall satisfaction. Although these values were slightly lower than those of the accommodation and restaurant scenarios, the results still indicate that users considered the attraction recommendation function practical and beneficial within the conversational environment.

Table 3. User satisfaction survey results for the attraction recommendation scenario

Participant	Q1	Q2	Q3	Q4	Q5
P1	5	4	4	4	4.5
P2	5	4	3	5	4.5
P3	4	4	4	4	4
P4	4	4.5	4	4.5	4.5
P5	4	4	4	4	4
Mean	**4.4**	**4.1**	**3.8**	**4.3**	**4.3**

To summarize the survey results across all three scenarios, the overall mean values were 4.5 for ease of use, 4.4 for recommendation relevance, 3.9 for response speed, 4.4 for convenience, and 4.5 for overall satisfaction. These aggregated results indicate that participants generally perceived the proposed system as intuitive, relevant, and convenient for travel-related information retrieval.

Table 4. Summary of mean user satisfaction scores across scenarios

Scenario	Ease of Use	Relevance	Response Speed	Convenience	Overall Satisfaction
Accommodation	4.6	4.4	4.0	4.6	4.6
Restaurant	4.6	4.6	4.0	4.4	4.5
Attraction	4.4	4.1	3.8	4.3	4.3
Overall Mean	**4.5**	**4.4**	**3.9**	**4.4**	**4.5**

4.5 Discussion

Across all three scenarios, the proposed system demonstrated a consistent end-to-end workflow from natural language input to API-driven result delivery. In each case, the system successfully interpreted the user's request, extracted relevant travel parameters, and returned structured recommendations through the real-time chat interface. These findings indicate that users can obtain multiple categories of travel information, including accommodations, restaurants, and attractions, within a single conversational environment without repeatedly switching across separate applications or services.

The preliminary user satisfaction survey further supports the practical usability of the proposed architecture. Ease of use and overall satisfaction received relatively high ratings across all scenarios, with overall mean scores of 4.5 and 4.5, respectively. These results suggest that the messenger-based interaction model provided a familiar and accessible interface for travel recommendation tasks. Recommendation relevance also received a favorable overall mean score of 4.4, indicating that the LLM-based semantic analysis pipeline was generally effective in translating natural language user requests into appropriate external API queries.

A comparison across scenarios shows that the accommodation and restaurant recommendation functions were evaluated slightly more favorably than the attraction recommendation function. The accommodation scenario achieved mean scores of 4.6 for ease of use and overall satisfaction, while the restaurant scenario showed the highest relevance score at 4.6 and an overall satisfaction score of 4.5. By contrast, the attraction scenario produced somewhat lower mean values, particularly in relevance and satisfaction. One possible explanation is

that attraction recommendations are more subjective and may depend more heavily on personal interests, travel purpose, or contextual preferences than accommodation or restaurant search. Even so, all three scenarios received positive ratings above 4.0 for most dimensions, indicating that the proposed system can support diverse travel recommendation tasks within a unified conversational framework.

Among the five evaluation dimensions, response speed received the lowest average score, with an overall mean of 3.9. Although this score remains above the neutral midpoint and suggests acceptable practical usability, it also indicates room for improvement in perceived responsiveness. Because the proposed system depends on multiple sequential processes, including server-side semantic analysis, external API communication, database synchronization, and UI rendering, response time may be influenced by both internal processing overhead and third-party service latency. Future improvements should therefore focus on reducing end-to-end delay through techniques such as prompt optimization, caching strategies, asynchronous request handling, and streamlined response composition.

Overall, the combined results of the scenario-based validation and the preliminary usability survey indicate that the proposed system is both operationally feasible and positively received by users. The integration of LLM-based semantic parsing, real-time messaging, and multi-service API aggregation appears to provide a useful framework for conversational travel assistance. Nevertheless, the present evaluation remains preliminary because of the limited sample size and the absence of objective comparative baselines. Future work should therefore include larger user studies, more diverse query sets, systematic latency measurements, and comparative experiments against conventional travel search workflows or existing chatbot-based recommendation systems.

V. CONCLUSION

This study presented FLYBY, a real-time conversational travel recommendation system that integrates an LLM-based semantic analysis pipeline with a mobile messenger interface and external travel service APIs. The proposed architecture combines Jetpack Compose on the client side, Firebase for synchronization and authentication, and a FastAPI-based server for intent analysis, parameter extraction, and API orchestration. Through this architecture, the system enables users to obtain accommodation, restaurant, and attraction recommendations within a unified conversational environment without repeatedly switching across separate applications.

The evaluation results indicate that the proposed system is both operationally feasible and positively perceived by users. In the scenario-based validation, the system successfully transformed unstructured travel-related queries into structured API requests and returned contextually relevant results in real time across all three service categories. In addition, the preliminary usability survey showed favorable user responses, with overall mean scores of 4.5 for ease of use, 4.4 for recommendation relevance, 3.9 for response speed, 4.4 for conversational convenience, and 4.5 for overall satisfaction. These findings suggest that messenger-based travel recommendation supported by LLM-driven semantic processing can improve the accessibility, continuity, and convenience of travel information retrieval.

Nevertheless, this study has several limitations. First, the evaluation was conducted as a small-scale pilot study and therefore provides only preliminary evidence. Second, recommendation quality depends partly on the coverage and reliability of external APIs. Third, although the system maintains session-specific conversational context, more advanced long-horizon itinerary planning and personalization remain limited in the current implementation. Future work will focus on larger-scale user studies, systematic measurement of extraction accuracy and response latency, and comparative evaluation against conventional travel search workflows and existing chatbot-based recommendation systems. Additional extensions will include itinerary generation, richer personalization mechanisms, and collaborative travel planning features.

ACKNOWLEDGEMENT

This work was supported by the Glocal University 30 Project of the Korea National University of Transportation in 2025

REFERENCES

[1]. D. Buhalis and R. Leung, "Smart hospitality—Interconnectivity and interoperability towards an ecosystem*". International Journal of Hospitality Management*, vol. 71, 2018, 41–50.
[2]. D. Gursoy, Y. Li, and H. Song, "ChatGPT and the hospitality and tourism industry: an overview of current trends and future research directions". *Journal of Hospitality Marketing & Management*, 32(5), 2023, 579–592.
[3]. Y. K. Dwivedi, N. Pandey, W. Currie, et al., "Leveraging ChatGPT and other generative artificial intelligence (AI)-based applications in the hospitality and tourism industry: practices, challenges and research agenda". *International Journal of Contemporary Hospitality Management*, 36(1), 2024, 1–12.
[4]. G. Caldarini, S. Jaf, and K. McGarry, "A literature survey of recent advances in chatbots". *Information*, 13(1), 2022, 41.
[5]. R. Pillai and B. Sivathanu, "Adoption of AI-based chatbots for hospitality and tourism". *International Journal of Contemporary Hospitality Management*, 32(10), 2020, 3199–3226.

[6]. T. Brown et al., "Language models are few-shot learners," in *Proceedings of Neural Information Processing Systems*, 33, 2020, 1877–1901.
[7]. I. A. Wong, Q. L. Lian, and D. Sun, "Autonomous travel decision-making: An early glimpse into ChatGPT and generative AI," *Journal of Hospitality and Tourism Management*, 56, 2023, 253–263.
[8]. A. Chen, X. Ge, Z. Fu, Y. Xiao, and J. Chen, "TravelAgent: An AI Assistant for Personalized Travel Planning," arXiv preprint arXiv:2409.08069, 2024.
[9]. Booking.com, "Affiliate Partner API," [Online]. Available: https://www.booking.com/affiliate-program/v2
[10]. Google, "Google Places API Documentation," [Online]. Available: https://developers.google.com/maps/documentation/places/web-service/overview
[11]. Google, "Jetpack Compose Official Documentation," [Online]. Available: https://developer.android.com/jetpack/compose
[12]. Google, "Firebase Official Documentation," [Online]. Available: https://firebase.google.com/docs
[13]. S. Ramírez, "FastAPI: Modern, fast web framework for building APIs with Python," [Online]. Available: https://fastapi.tiangolo.com
[14]. OpenAI, "OpenAI API Documentation," [Online]. Available: https://platform.openai.com/docs