HI-Chat – Final Report

: A Chatbot for Hongik University Information Guidance

지도교수	팀원	학번	C011147	이름	이승연
이장호			C111038		김지윤

<1. Introduction>

This report presents the final outcome of the HI-Chat project, a chatbot system designed for Hongik University students. The goal of HI-Chat is to provide a convenient, centralized, and interactive way for students to access a wide range of campus-related information such as course registration, academic schedules, cafeteria menus, scholarship details etc. By enabling quick keyword-based queries, HI-Chat reduces the time and effort required for students to find important academic and administrative information, streamlining their campus experience through a simple and accessible interface.

<2. Requirements Recap & Updates>

In the initial phase, HI-Chat was designed to include the following core features:

- Answering Frequently Asked Questions (FAQ)
- Providing course registration info and academic schedules
- Access to scholarship and certificate information
- Campus map and facility usage info

These features were mapped to 13 user-facing chatbot menu options:

While the initial plan included 14 chatbot menu items, the "Mobile Student ID" feature was removed from the implementation scope due to system access restrictions and technical constraints. All remaining features were implemented in the final prototype.

1. FAQ
2. Course Registration Info
3. Today's Cafeteria Menu
4. Campus Contacts
5. Scholarship Info
6. Request Certificate
7. Check Academic Calendar
8. Library Info
9. View Grades
10. Campus Map
11. Exam Schedule
12. Facility Guide
13. Admission Info

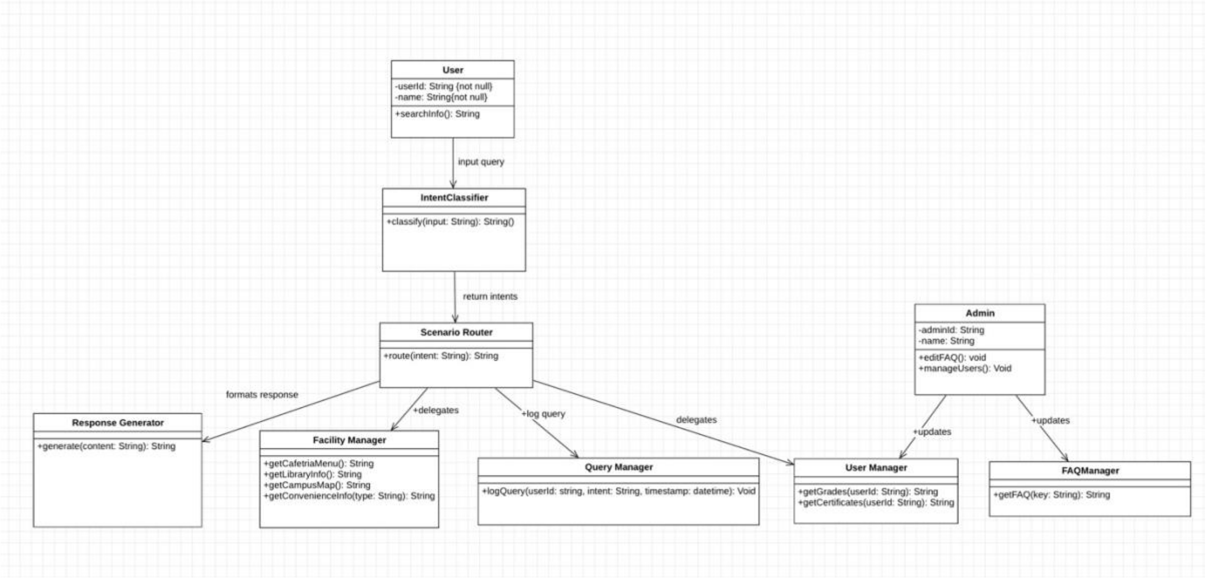
<3. Final Architecture Design>

The finalized architecture includes modular components:

- **UI**: A simple web-based interface styled after KakaoTalk
- **Backend (Spring Boot)**: Manages request routing, logic processing, and response generation
- **IntentClassifier**: Determines user intent using keyword matching rules

- **ScenarioRouter**: Delegates the identified intent to the appropriate manager class
- **Managers**: FAQManager, UserManager, and FacilityManager handle specific data requests such as retrieving FAQs, verifying student academic data like grades and certificates, and accessing facility information including cafeteria menus or campus maps. (Additional managers can be added in the future for features like campus contacts or academic calendars.)
- **ResponseGenerator**: Formats the output into a human-readable message before sending it back to the frontend.

<4. Class Diagram & Internal Design>


User
-userId: String {not null}
-name: String{not null}
+searchInfo(): String
input query
IntentClassifier
+classify(input: String): String()
return intents
Scenario Router
+route(intent: String): String
Admin
-adminId: String
-name: String
+editFAQ(): void
+manageUsers(): Void
formats response
+delegates
+log query
delegates
+updates
+updates
Response Generator
+generate(content: String): String
Facility Manager
+getCafetriaMenu(): String
+getLibraryInfo(): String
+getCampusMap(): String
+getConvenienceInfo(type: String): String
Query Manager
+logQuery(userId: string, intent: String, timestamp: datetime): Void
User Manager
+getGrades(userId: String): String
+getCertificates(userId: String): String
FAQManager
+getFAQ(key: String): String

The system is structured around clear roles for each component:

- User, Admin represent system actors
- IntentClassifier detects user intent
- ScenarioRouter dispatches the request to managers
- Managers fetch relevant data and return results
- ResponseGenerator composes responses for display to the user

Each component is modular and aligns with the single-responsibility principle. This separation of concerns ensures that the system is easy to maintain and extend. For instance, the IntentClassifier focuses solely on interpreting user input, the ScenarioRouter determines the appropriate logic path, and each Manager specializes in one domain (e.g., FAQManager handles questions, FacilityManager provides cafeteria/menu info). This modularity makes it easier to update or expand specific parts of the system independently. For example, if we want to newly support features such as campus contacts or academic calendar lookup, a corresponding Manager class (e.g., ContactManager or CalendarManager) can be added without modifying the existing intent classification or routing logic. This architectural clarity supports long-term scalability and facilitates collaborative development by separating responsibilities across the team.

<5. Database Design>

Our ERD includes the following tables:

1) User: Stores student data including user ID, name, major, and account status. This table supports personal data retrieval features such as grade checking and certificate issuance.
Example:*'account status '* refers to whether a user is currently considered active within the system. This field typically indicates if the student is eligible to use the chatbot features. It may be implemented as a boolean (e.g., true for active, false for inactive) or as an enumerated type (e.g., active, inactive, suspended, deleted). In this project, it is used to determine whether a user can access personalized features such as grade inquiries or certificate issuance.

2) Admin: Manages administrator credentials and authorizations for maintaining the system, such as editing FAQs or updating chatbot content.
Example: An admin can log in to revise an outdated response to a scholarship inquiry

3) FAQ: Contains frequently asked questions and their corresponding answers. It supports intent-based FAQ retrieval by keyword.
Example: A query like "When is course registration?" maps to a FAQ that explains registration dates and portal links.

4) Facility: Represents campus-related resources such as cafeteria menus, library hours, and convenience facility info.
Example: A query like "When is course registration?" maps to a FAQ that explains registration dates and portal links.

5) QueryLog: Records user queries, classified intents, and timestamps. This table can be used for usage analytics and future chatbot improvement.
Example: If many users ask about certificate reissue, developers can create a shortcut button or clarify the response.

Entity relationships:

- Each QueryLog references the User table via a foreign key to trace which user made which query.
- Admin has update authority over the FAQ table.
- The 'FAQ' table can be updated or deleted only by authenticated administers
- `Facility` entries are accessed by multiple use cases such as library info, cafeteria menu, and convenience facility data, and are mapped through keyword-based queries.

- All response-generating actions are logged in `QueryLog`, establishing a traceable intent-response history for future analytics.

This database design supports both functional requirements of the chatbot and provides scalability for future personalization and analytics features.

6. Implementation

The HI-Chat system is composed of a simple yet functional frontend and a rule-based backend. The backend was developed using Java and Spring Boot, and it exposes a single RESTful API endpoint /chat, which receives a user message via HTTP GET requests. Upon receiving input, the backend routes the message through a series of internal components: IntentClassifier, ScenarioRouter, and a set of domain-specific Manager classes such as FAQManager, UserManager, and FacilityManager.

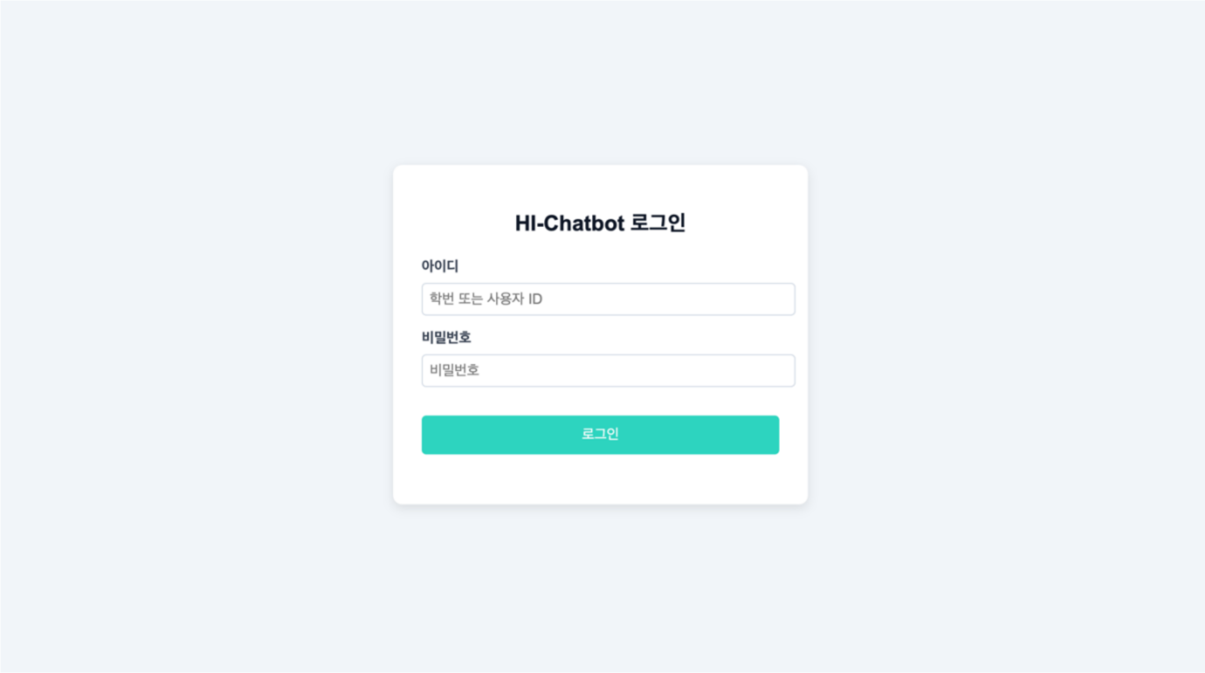
The frontend was built using basic HTML, CSS, and JavaScript to mimic a familiar chat interface like KakaoTalk. The chatbot screen displays message bubbles for both user and system responses, along with a grid of quick-access buttons representing common queries such as "수강신청 정보", "장학금 정보", and "오늘의 메뉴". Although the input field and buttons are disabled in the prototype, they demonstrate the structure and user flow of an interactive campus assistant.

Internally, intent classification is handled through simple keyword matching rather than complex natural language processing (NLP). For instance, if a user inputs "수강신청 언제예요?", the classifier detects the keyword "수강신청" and categorizes the intent as "Course_Registration". This intent is then passed to the ScenarioRouter, which dispatches the request to the appropriate manager (in this case, FAQManager). The manager fetches a pre-defined response, which is formatted by the ResponseGenerator and returned to the frontend.

All chatbot functions are handled statelessly without login or database integration. Instead, mock data and hardcoded keyword-to-response mappings are used to simulate the 13 core menu features. This implementation strategy simplifies testing and allows the system to be deployed and demonstrated easily without external dependencies.

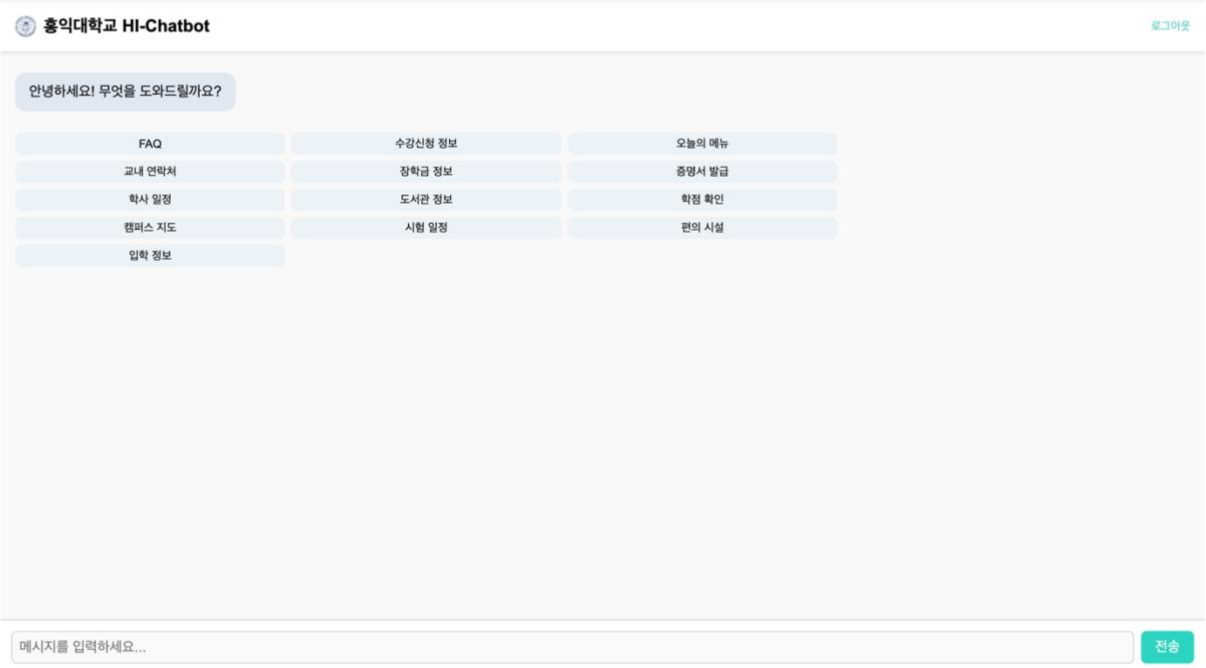
<7. Screenshots>

Below is an example of the user interface design and response interaction:



7-1) Login Interface

The HI-Chat login screen is a minimal and clean interface where users are prompted to enter their student ID and password. Although the current prototype does not implement actual authentication logic, this screen provides a conceptual foundation for future login functionality. This feature is essential for enabling personalized services, such as grade lookup or certificate issuance, in the next development phase.

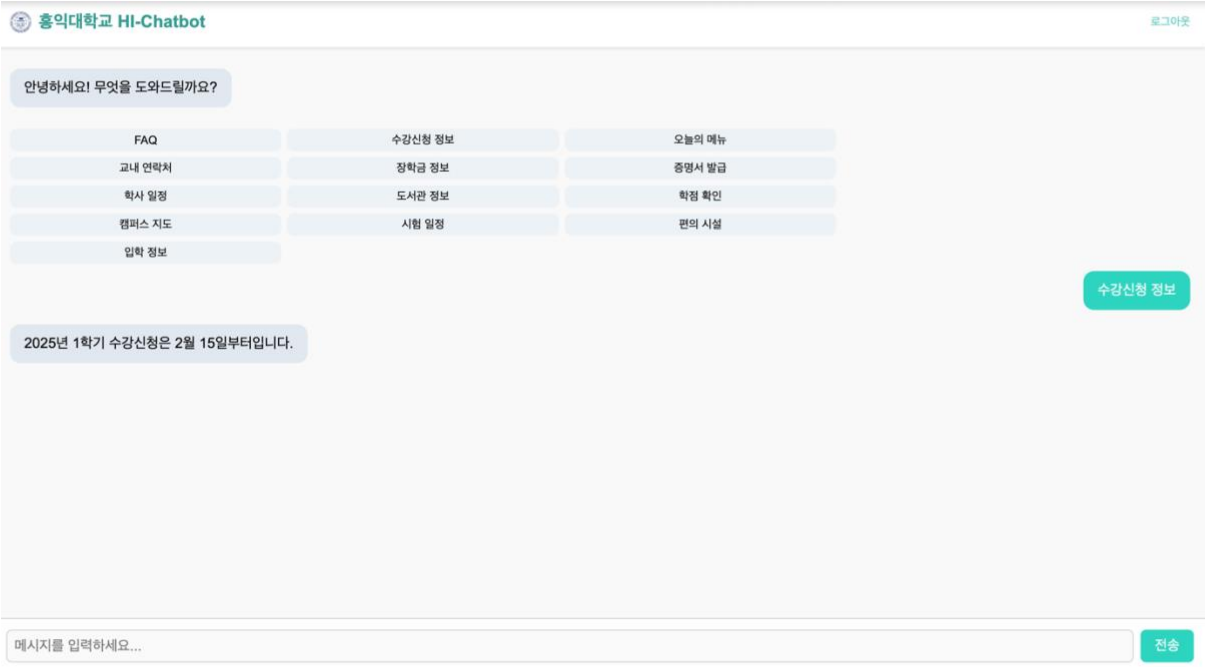


홍익대학교 HI-Chatbot

안녕하세요! 무엇을 도와드릴까요?

7-2) Main Chatbot Interface

The main interface replicates a familiar messenger-style chat layout. At the top, the chatbot greets the user and offers a list of 13 pre-defined menu buttons corresponding to frequently requested information such as course registration, cafeteria menu, scholarships, and campus map, etc.



안녕하세요! 무엇을 도와드릴까요?

수강신청 정보

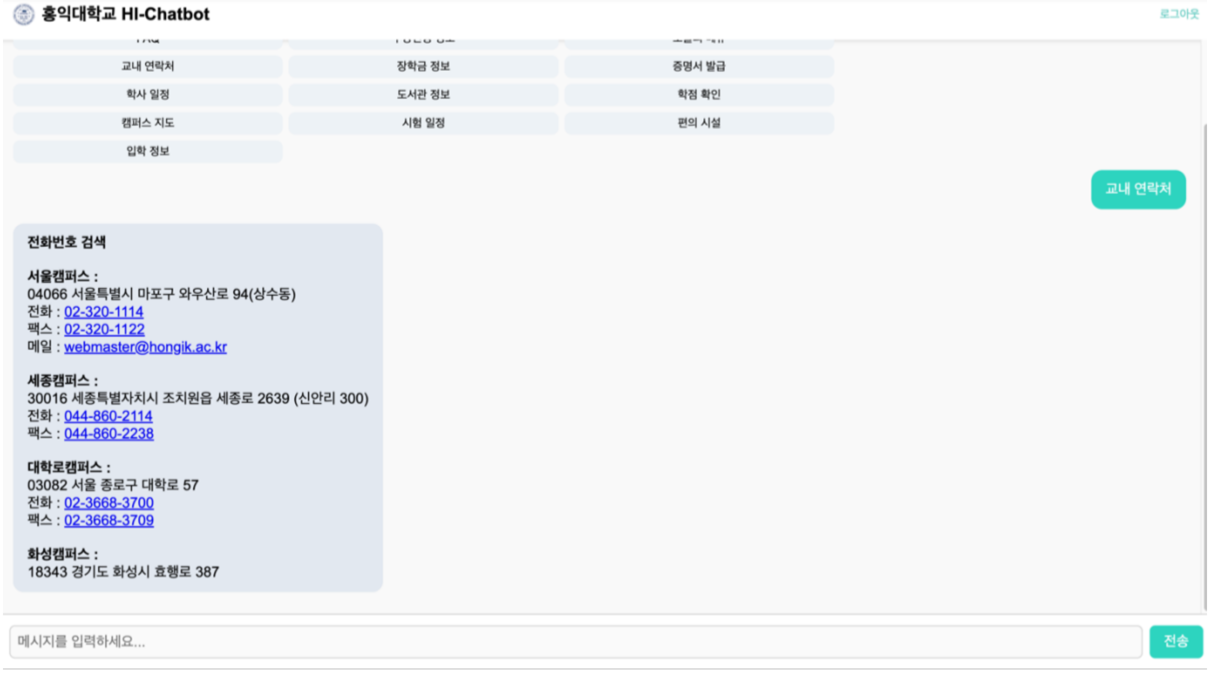
2025년 1학기 수강신청은 2월 15일부터입니다.

7-3) Course Registration Info

The user selects "수강신청 정보 (Course Registration Info)" and the chatbot responds with a predefined message:

"2025 년 1 학기 수강신청은 2 월 15 일부터입니다."

This interaction demonstrates the intent-based keyword matching and response flow between the frontend and backend components of the system.



전화번호 검색

서울캠퍼스 :
04066 서울특별시 마포구 와우산로 94(상수동)
전화 : 02-320-1114
팩스 : 02-320-1122
메일 : webmaster@hongik.ac.kr

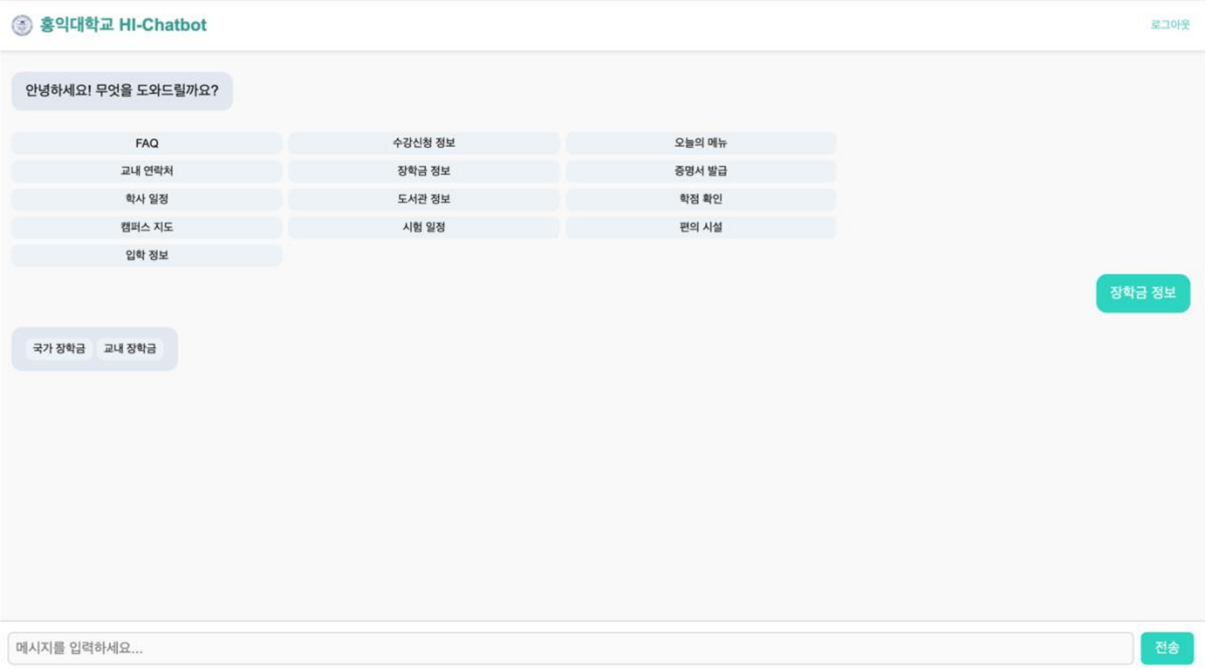
세종캠퍼스 :
30016 세종특별자치시 조치원읍 세종로 2639 (신안리 300)
전화 : 044-860-2114
팩스 : 044-860-2238

대학로캠퍼스 :
03082 서울 종로구 대학로 57
전화 : 02-3668-3700
팩스 : 02-3668-3709

화성캠퍼스 :
18343 경기도 화성시 효행로 387

7-4) Campus Contacts

Upon clicking the "교내 연락처 (Campus Contacts)" button, the chatbot retrieves and displays important institutional contact information directly within the chat window. This includes the postal addresses, phone numbers, fax numbers, and email addresses of Hongik University’s Seoul, Sejong, and Daehakro campuses. Instead of linking out to an external directory page, the chatbot provides the information immediately through keyword-based retrieval handled by the FacilityManager. This improves accessibility and ensures students can conveniently find campus communication details when needed.

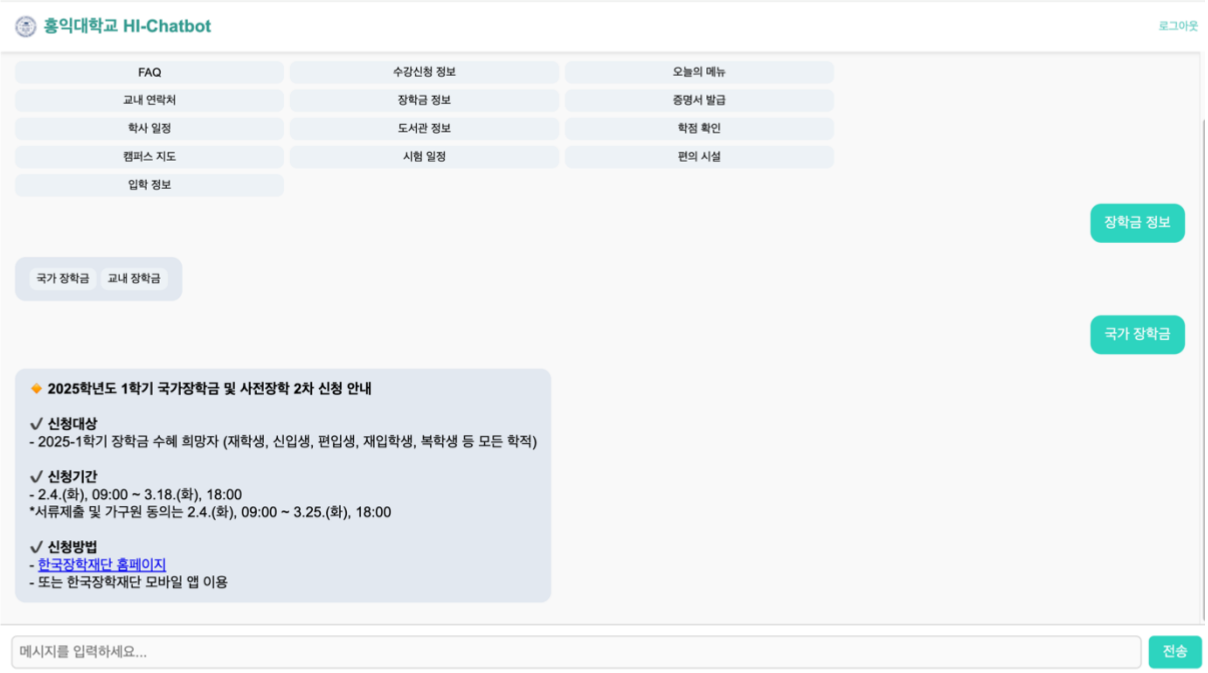


안녕하세요! 무엇을 도와드릴까요?

장학금 정보

7-5) Scholarship Information

The scholarship inquiry feature demonstrates the chatbot’s ability to guide users through multi-step intent handling. When the user clicks the “장학금 정보 (Scholarship Info)” button (see screenshot ①), the chatbot responds with two refined sub-options: “국가 장학금 (National Scholarship)” and “교내 장학금 (School-based Scholarship)”.



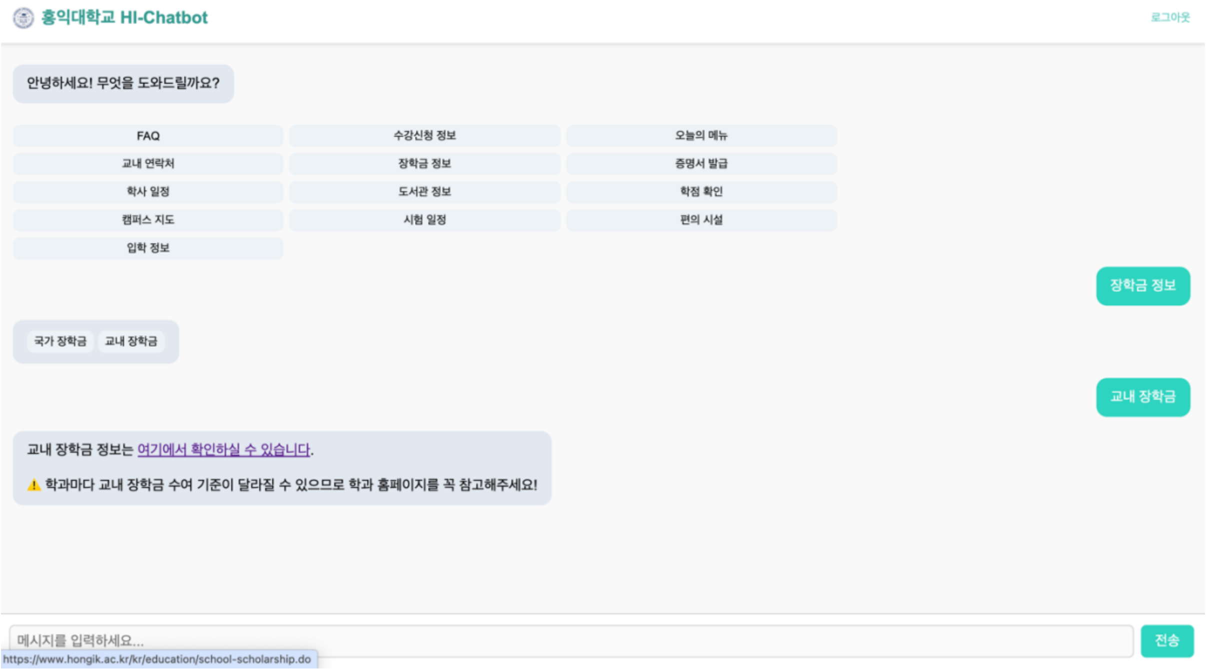
🔶 **2025학년도 1학기 국가장학금 및 사전장학 2차 신청 안내**

✓ **신청대상**
- 2025-1학기 장학금 수혜 희망자 (재학생, 신입생, 편입생, 재입학생, 복학생 등 모든 학적)

✓ **신청기간**
- 2.4.(화), 09:00 ~ 3.18.(화), 18:00
*서류제출 및 가구원 동의는 2.4.(화), 09:00 ~ 3.25.(화), 18:00

✓ **신청방법**
- 한국장학재단 홈페이지
- 또는 한국장학재단 모바일 앱 이용

If the user selects “국가 장학금 (National Scholarship)” (screenshot ②), the chatbot returns detailed application information including eligibility criteria, submission period, and instructions for using the Korea Student Aid Foundation (KOSAF) system.



안녕하세요! 무엇을 도와드릴까요?

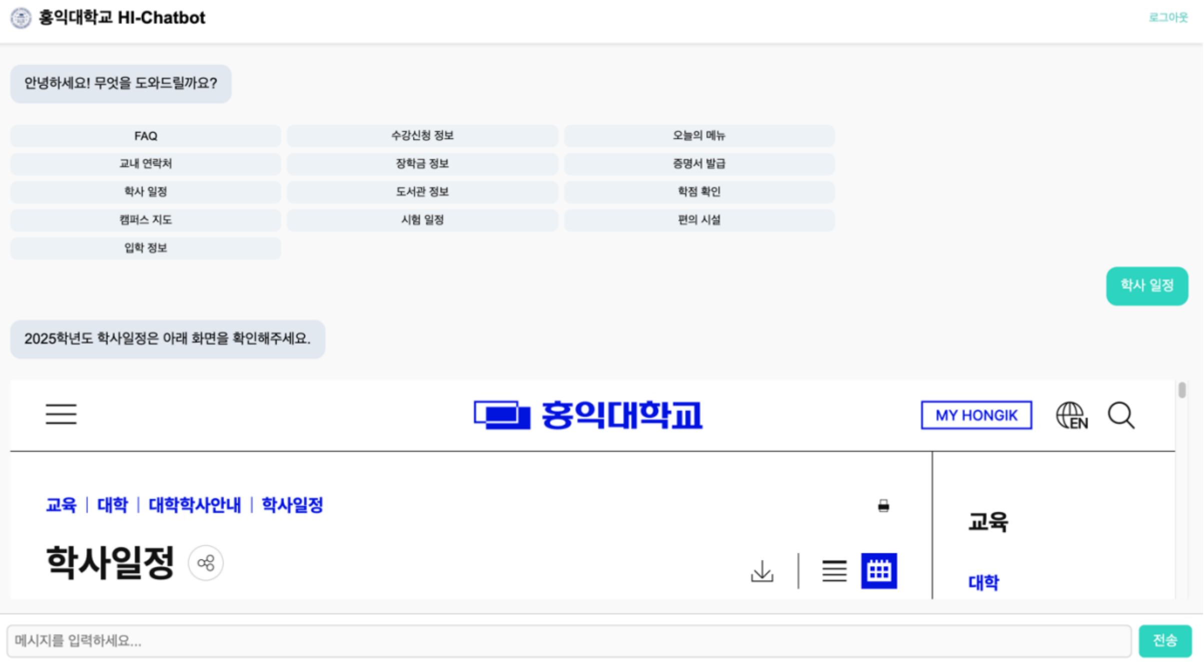
교내 장학금 정보는 여기에서 확인하실 수 있습니다.

⚠ 학과마다 교내 장학금 수여 기준이 달라질 수 있으므로 학과 홈페이지를 꼭 참고해주세요!

https://www.hongik.ac.kr/kr/education/school-scholarship.do

If the user instead chooses “교내 장학금 (School-based Scholarship)” (screenshot ③), the chatbot provides a direct link to the university's internal scholarship guide webpage, along with a reminder that scholarship conditions may differ by department.

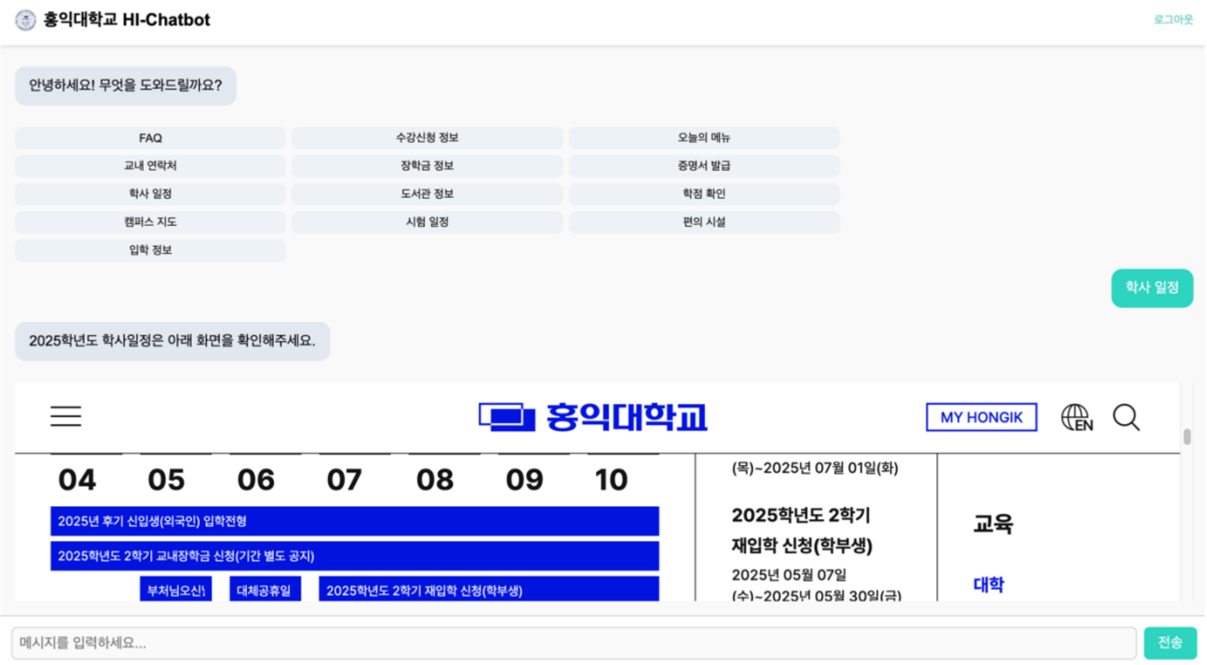
This flow highlights HI-Chat’s capability to offer not only predefined answers but also dynamic content tailored to the user’s follow-up intent.



안녕하세요! 무엇을 도와드릴까요?

학사 일정

2025학년도 학사일정은 아래 화면을 확인해주세요.



안녕하세요! 무엇을 도와드릴까요?

학사 일정

2025학년도 학사일정은 아래 화면을 확인해주세요.

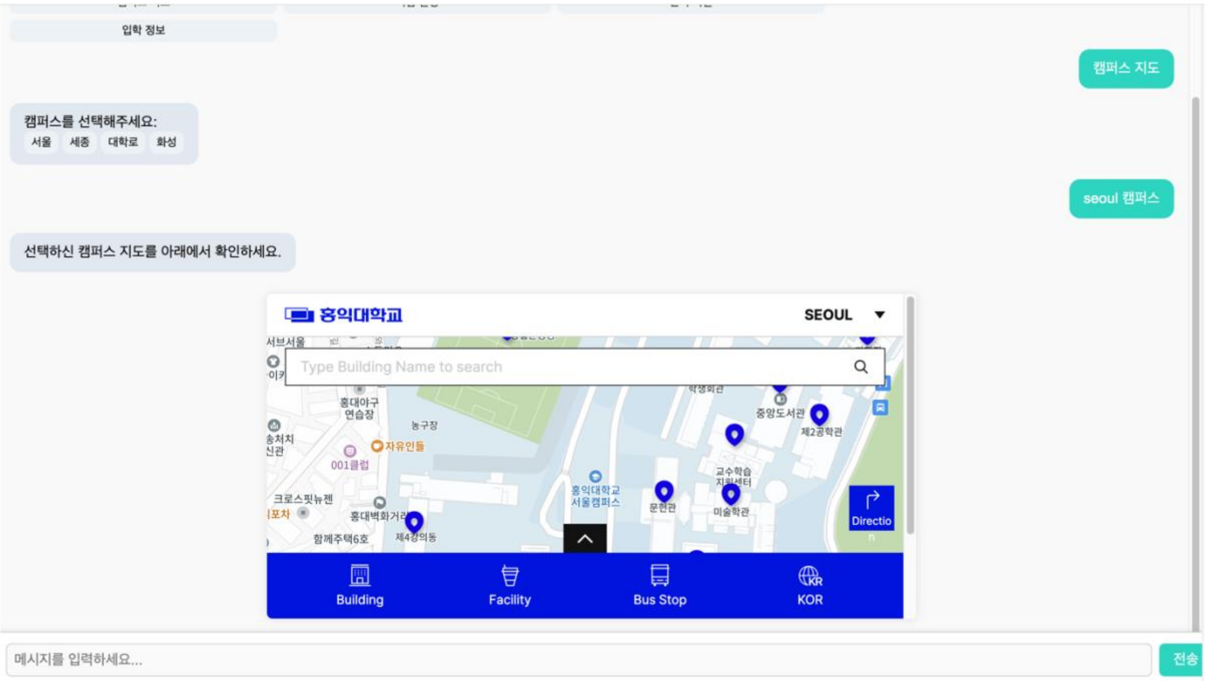
(목)~2025년 07월 01일(화)

2025학년도 2학기 재입학 신청(학부생)

2025년 05월 07일 (수)~2025년 05월 30일(금)

7-6) Academic Calendar Integration

When the user clicks the "학사 일정 (Academic Calendar)" button, the chatbot responds by embedding the official Hongik University academic calendar page within the chat interface. This approach enhances the usability of the system by directly presenting reliable, up-to-date institutional data without requiring users to manually search external websites. The academic calendar is retrieved as a live view from the university portal and allows students to instantly confirm important dates such as enrollment periods, holidays, and grade release schedules.



입학 정보

캠퍼스 지도

캠퍼스를 선택해주세요:

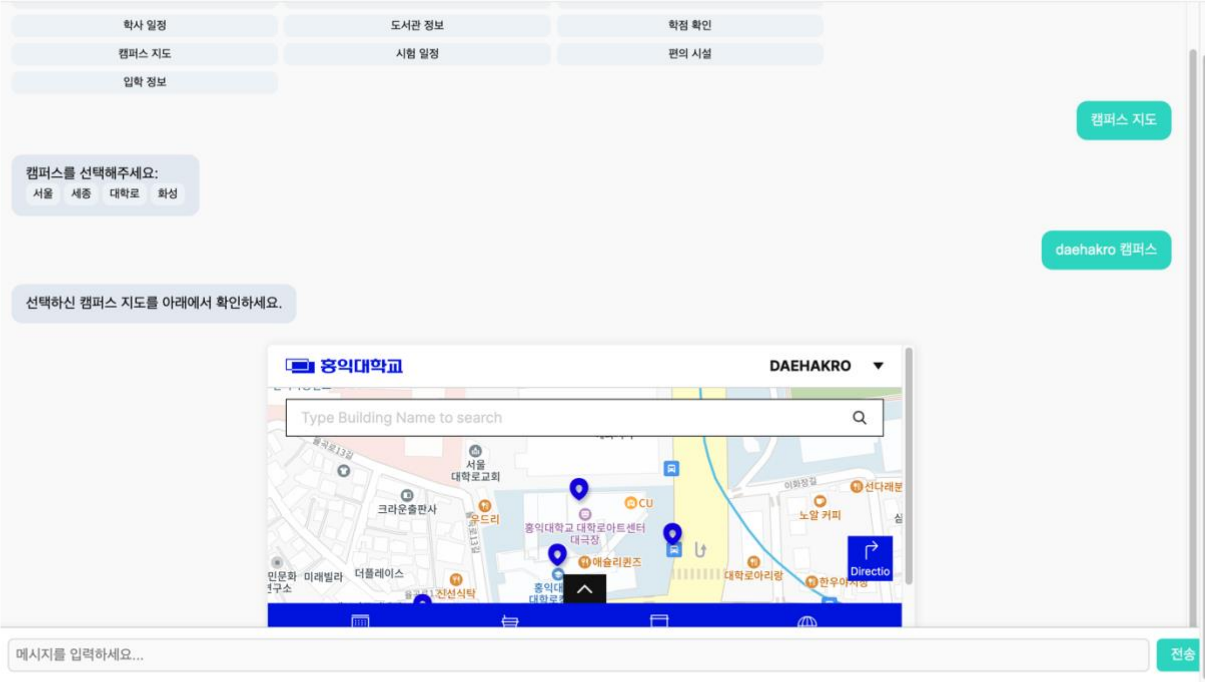
서울 세종 대학로 화성

seoul 캠퍼스

선택하신 캠퍼스 지도를 아래에서 확인하세요.

메시지를 입력하세요...

전송



학사 일정 도서관 정보 학점 확인

캠퍼스 지도 시험 일정 편의 시설

입학 정보

캠퍼스 지도

캠퍼스를 선택해주세요:

서울 세종 대학로 화성

daehakro 캠퍼스

선택하신 캠퍼스 지도를 아래에서 확인하세요.

메시지를 입력하세요...

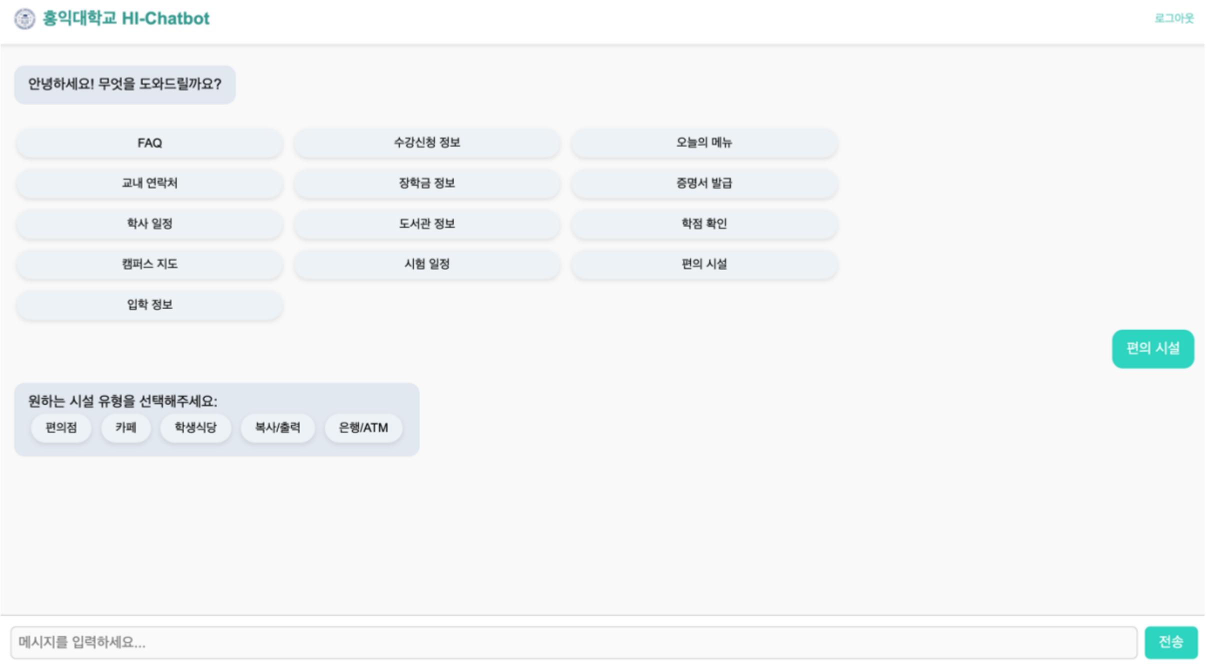
전송

7-7) Campus Map

When the user selects the “캠퍼스 지도 (Campus Map)” button, the chatbot presents four campus options: Seoul, Sejong, Daehakro, and Hwaseong. Each option corresponds to a specific campus of Hongik University.

For example, in the previous screenshot, the user has chosen the “Seoul” campus, and the chatbot displays the corresponding interactive map embedded directly in the chat. Similarly, if the user selects “Daehakro,” the chatbot immediately loads the Daehakro campus map in the same location. These maps include searchable building names, icons for facilities and bus stops, and navigation tools.

By supporting all four campus maps in an integrated format, the system allows users to quickly find physical locations without needing to visit separate web pages. This enhances the practicality of the chatbot for students navigating different campuses.



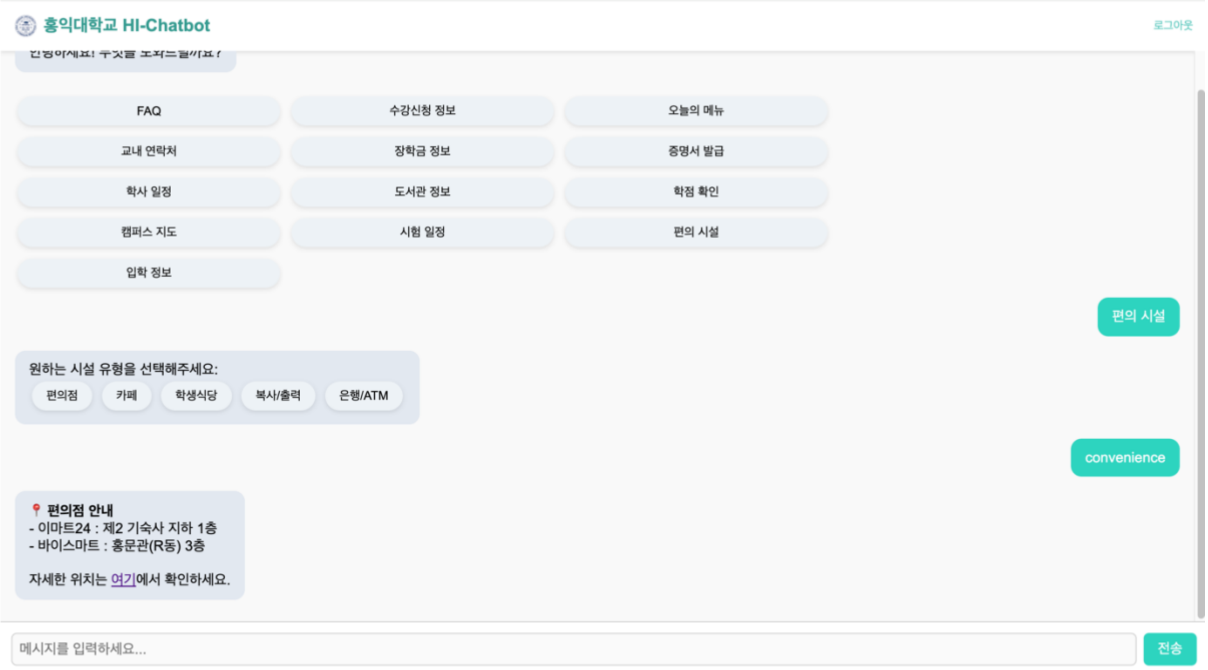
안녕하세요! 무엇을 도와드릴까요?

편의 시설

원하는 시설 유형을 선택해주세요:

7-8) Facility Guide

When the user selects the “편의 시설 (Convenience Facility)” button, the chatbot presents five facility type options: Convenience Stores, Cafés, Student Dining, Printing/Copying, and Banks/ATMs. These options allow students to quickly find the on-campus services they are looking for.

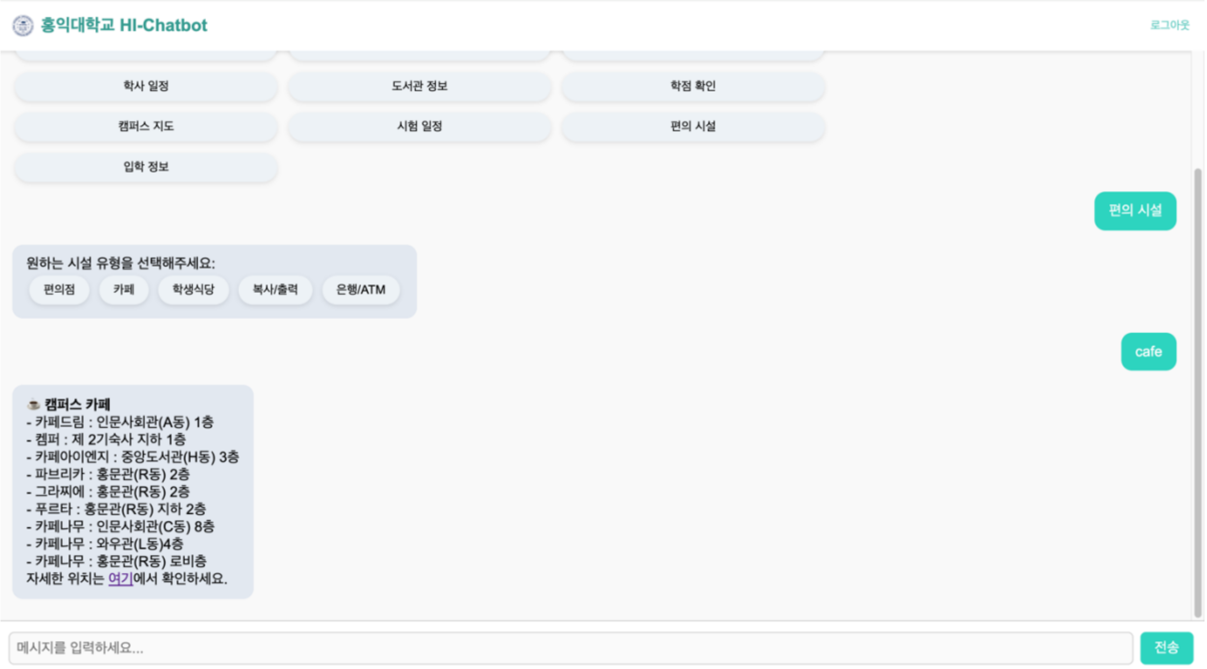


원하는 시설 유형을 선택해주세요:

📍 **편의점 안내**
- 이마트24 : 제2 기숙사 지하 1층
- 바이스마트 : 홍문관(R동) 3층

자세한 위치는 여기에서 확인하세요.

If the user chooses “편의점 (Convenience Store),” as shown in the second screenshot, the chatbot responds with a list of locations, including store names and building locations, along with a link to view a detailed map.



원하는 시설 유형을 선택해주세요:

☕ **캠퍼스 카페**
- 카페드림 : 인문사회관(A동) 1층
- 켐퍼 : 제 2기숙사 지하 1층
- 카페아이엔지 : 중앙도서관(H동) 3층
- 파브리카 : 홍문관(R동) 2층
- 그라찌에 : 홍문관(R동) 2층
- 푸르타 : 홍문관(R동) 지하 2층
- 카페나무 : 인문사회관(C동) 8층
- 카페나무 : 와우관(L동)4층
- 카페나무 : 홍문관(R동) 로비층

자세한 위치는 여기에서 확인하세요.

If “카페 (Café)” is selected, the chatbot displays various cafés on campus with their corresponding building names and floor levels, allowing students to easily identify where to go.

This layered interaction demonstrates the chatbot’s ability to handle multi-level facility inquiries through keyword-driven decision flows. It offers both informative text and external links for map access, thereby increasing accessibility for users unfamiliar with campus layout.

<8. Testing & Validation>

Manual testing was conducted based on interaction with the chatbot’s button-based interface, rather than free-form text input. Since the current prototype is designed around predefined menu selections, test scenarios were structured to simulate realistic user flows initiated through clicks. The goal was to ensure that each button correctly triggered the intended logic flow and returned an accurate and appropriately formatted response.

8-1) Test Methodology

Users were presented with 13 main menu buttons corresponding to key use cases, including “FAQ,” “수강신청 정보 (Course Registration Info),” “장학금 정보 (Scholarship Info),” “캠퍼스 지도 (Campus Map),” and “편의 시설 (Facility Info).” Clicking each button invoked a chain of internal processes: IntentClassifier mapped the selected label to a defined intent string; ScenarioRouter dispatched the request to the corresponding Manager class; and the response was generated and rendered in the UI.

8-2) Core Feature Tests

1. **FAQ Handling**

   ① Test: Clicked "FAQ" → "입학 정보"
   ② Expected: Response with brief entry guide and link to admission homepage
   ③ Result: Accurate predefined response retrieved from FAQManager

2. **Course Registration Info**

   ① Test: Clicked "수강신청 정보"
   ② Expected: "2025 년 1 학기 수강신청은 2 월 15 일부터입니다."
   ③ Result: FAQManager returned correct static message

3. **Scholarship Information**

   ① Test: Clicked "장학금 정보" → "국가 장학금"
   ② Expected: Multi-line response including application period, eligible students, and application link
   ③ Test 2: "장학금 정보" → "교내 장학금"
   ④ Expected: Short notice with clickable link to the school's scholarship page
   ⑤ Result: Both flows routed through FAQManager and returned accurate responses with formatting and hyperlinks

4. **Campus Contacts**

   ① Test: Clicked "교내 연락처"
   ② Expected: Listing of contact info for Seoul, Sejong, Daehakro campuses (phone, fax, email)
   ③ Result: Displayed neatly formatted text block from FacilityManager

5. **Academic Calendar**

① Test: Clicked "학사 일정"

② Expected: Embedded academic schedule view from the university's official website

③ Result: Correct link loaded and displayed within chatbot window

6. **Campus Map**

① Test: Clicked "캠퍼스 지도" → "서울"

② Expected: Interactive campus map displayed inside the UI

③ Test 2: "캠퍼스 지도" → "대학로"

④ Expected: Map of Daehakro campus

⑤ Result: FacilityManager mapped sub-intent and embedded correct view

7. **Facility Information**

① Test: Clicked "편의 시설" → "편의점"

② Expected: List of on-campus convenience stores and building info

③ Test 2: "편의 시설" → "카페"

④ Expected: 8 locations with floor and building names

⑤ Result: Accurate formatting, bullet list, and optional map links included

8-3) Edge Case Tests

- Switching quickly between categories (e.g., "장학금 정보" → "캠퍼스 지도") without system crash
- Clicking "전송" without input — properly disabled input prevented unintended request
- No login validation required; tested with mock user "홍길동" assumed in backend logic

8-4) Outcome

All tested use cases functioned as expected within the defined scope of the prototype. The chatbot maintained robust flow control through button-triggered state transitions. Even without dynamic NLP or database integration, the system demonstrated reliable routing, intent matching, and structured response formatting using keyword-based logic.

These results validate the effectiveness of the current rule-based chatbot structure for delivering a campus information assistant service, especially in a low-resource development environment.

<9. Challenges & Lessons Learned>

During the development process, several practical challenges emerged that influenced our technical decisions and project scope.

- **NLP and Free-form Input Handling**: Initially, we explored the possibility of supporting natural language queries. However, implementing flexible sentence understanding without external NLP libraries or pre-trained models proved to be unfeasible for our 2-person team. We recognized the limitation early and shifted to a more maintainable keyword-based approach.
- **Parallel Frontend-Backend Development**: Managing both the frontend UI and backend logic concurrently within a tight schedule created coordination challenges. Design decisions had to be made quickly, often requiring compromise between interactivity and scope control.
- **Input Ambiguity and Misclassification**: Without NLP, user intent had to be inferred strictly from button labels and keyword strings. This required

careful design of the intent taxonomy and response structure to avoid overlap or confusion.

- **Time-Boxed MVP Strategy**: Instead of over-engineering, we focused on delivering a well-functioning Minimum Viable Product (MVP) that emphasized clarity and consistency across 14 pre-defined chatbot features.

These challenges taught us valuable lessons in **scope management**, **modular interface design**, and the importance of **iterative development**. The decision to simplify intent handling and rely on a button-based interaction model allowed us to complete a usable, testable system within our constraints, while still laying the groundwork for future expansion.

<10. Conclusion & Future Work>

HI-Chat successfully demonstrates a functional rule-based chatbot prototype designed for Hongik University students. The system provides campus-related information such as course registration, cafeteria menus, and facility usage through a structured, button-driven interface. This minimized the reliance on complex natural language inputs while maximizing usability and stability.

Due to scope and resource limitations, the current version does not include user authentication, personalized data access, or advanced NLP (natural language processing). All queries are routed via keyword-matched intents, and responses are returned using predefined content and simulated user context.

Future work may include:
- User login and personalization for services such as grade viewing and certificate issuance
- FAQ and facility data management tools for administrators
- Optional database integration to support persistent logs and dynamic content

- Possibly, simple NLP extensions such as typo tolerance or semantic keyword expansion — but full-scale NLP adoption is not currently planned

This approach reflects our focus on delivering a stable and usable prototype under real-world constraints. The design prioritizes clarity, modularity, and scope control — all of which serve as a solid foundation should further development continue beyond this course.