

ZoopZoop: An Integrated Personalized Recommendation and Notification System for Public Policy Information Services

Taeseok Lee¹, Jaegyong Song², Bosuk Yoon³, Seungjae Lee^{4*}

^{1,2,3,4} Computer Engineering Department, Sunmoon University, Korea

ABSTRACT: Public policy information is often distributed across multiple government institutions and service platforms, making it difficult for citizens to identify policies that match their individual circumstances, interests, and application schedules. Conventional keyword-based policy search requires users to actively identify relevant policy terms and repeatedly monitor policy information, which may result in missed opportunities and inefficient information exploration. To address these limitations, this paper presents ZoopZoop, an integrated personalized public policy information service that combines policy search, personalized recommendation, policy management, deadline notification, and conversational AI assistance within a unified web-based platform. The proposed system collects public policy information through government public data APIs and periodically synchronizes policy lists, detailed information, and eligibility conditions. Personalized recommendations are generated using two complementary sources of information: user behavioral information, including recent search and policy-view histories, and user profile information, including age, income range, residential area, employment status, gender, household size, and marital status. The system further provides notifications for approaching deadlines, newly registered policies, and recommended policies. To support natural-language interaction, an AI chatbot retrieves relevant policy information from the internal policy database and uses the retrieved information as contextual input for response generation. The system was implemented using React, Spring Boot, PostgreSQL, JWT-based authentication, and external public data and AI APIs. The implemented prototype demonstrates how personalized recommendation, proactive notification, and conversational interaction can be integrated into a unified public policy information service. The proposed system provides a practical foundation for improving the accessibility and personalization of digital public policy information services.

KEYWORDS: Public Policy Recommendation, Personalized Recommendation, Recommender Systems, User Behavior Modeling, Profile-Based Recommendation, Policy Information Service, Conversational AI

I. INTRODUCTION

Public policies and government support programs play an important role in addressing diverse social and economic needs, including employment, education, housing, entrepreneurship, income support, and welfare. With the expansion of digital government services, a large amount of public policy information has become available through online portals and open government data platforms. However, the increasing availability of information does not necessarily imply that citizens can easily identify and utilize the policies that are relevant to their individual circumstances. Government services are frequently distributed across multiple institutions and platforms, and individual policies may differ substantially in terms of eligibility requirements, target populations, application periods, geographic restrictions, required documents, and responsible organizations. Consequently, citizens may need to navigate a large and heterogeneous information space before identifying a policy that is applicable to them.

This problem can be understood as an information overload problem in digital government environments. Previous research on personalized e-government services has identified information overload as an important obstacle to the effective utilization of online government services and has suggested personalization and recommendation techniques as potential approaches for improving service accessibility [1], [2]. In particular, recommender systems can reduce the amount of information that users need to examine by

automatically identifying items or services that are likely to be relevant to individual users. Such approaches have been extensively studied in commercial domains, including e-commerce, entertainment, and online content services, and have increasingly been investigated in public-service and e-government environments as well [3], [4].

Conventional public policy information services, however, generally rely on keyword-based search and manually operated filtering mechanisms. Although keyword search is useful when users know the name or category of the policy they are seeking, it becomes less effective when users do not know the exact terminology used by government institutions. For example, a user may be interested in housing support or employment assistance without knowing the specific name of an applicable government program. In such cases, the user must formulate appropriate search queries, examine multiple results, compare eligibility requirements, and determine whether each policy is applicable to his or her circumstances. Moreover, even after identifying an appropriate policy, users may need to repeatedly revisit the service to monitor changes or application deadlines. This interaction model places a substantial information-search burden on citizens.

Personalized recommender systems provide an alternative approach by transforming passive information retrieval into more proactive information delivery. A recommender system analyzes information about users and available items to identify potentially relevant items and present them in a prioritized manner. Existing recommender-system research

has traditionally focused on collaborative filtering, content-based filtering, and hybrid recommendation techniques [3], [5]. Hybrid approaches combine complementary recommendation strategies to address limitations such as data sparsity and the cold-start problem [5]. More recent research has also emphasized the importance of user behavior modeling, in which interactions such as clicks, views, searches, and other behavioral signals are used to infer user interests and improve recommendation quality [6], [7].

Behavioral information is particularly useful because explicit preference information is often unavailable. Unlike domains in which users provide ratings or explicit feedback, public policy services typically do not require citizens to rate policies. Instead, implicit behavioral signals such as search queries, policy views, saved policies, and repeated interactions can provide useful evidence regarding users' interests. Research on e-government recommendation has demonstrated that implicit interaction histories can be used to generate personalized recommendations even when conventional rating information is unavailable [1]. Furthermore, recent studies of user behavior modeling indicate that different types of user-item interactions can provide complementary information about user interests and can be exploited to improve recommendation performance [6], [7].

Nevertheless, relying exclusively on behavioral information introduces another important limitation: the cold-start problem. A newly registered user has little or no historical interaction data, making it difficult to infer the user's interests from behavioral logs alone. This issue is particularly relevant in public policy services because users may access the system only when a specific social, economic, employment, or household circumstance occurs. Consequently, demographic and contextual information can provide valuable complementary information when behavioral data are sparse. Profile-based recommendation can utilize attributes such as age, income range, residential area, employment status, household size, and marital status to identify policies whose eligibility conditions correspond to the user's circumstances. Combining behavioral and profile information therefore provides a practical strategy for supporting both experienced users and newly registered users.

Another important characteristic of public policy information is its strong temporal component. Unlike many conventional recommendation domains, the usefulness of a government policy may depend not only on whether the policy matches a user's interests but also on whether the user can apply within a specific period. A highly relevant policy may become practically useless if the user is unaware of its application deadline. Therefore, a public policy information system should not be limited to recommending relevant policies; it should also proactively notify users about time-sensitive information. Proactive notification can reduce the need for users to repeatedly monitor policy services and can connect personalized recommendation with timely information delivery.

Recent developments in conversational artificial intelligence provide another opportunity to improve public policy information access. Traditional search interfaces require users to translate their needs into keywords and filters, whereas conversational interfaces allow users to describe their circumstances using natural language. Large language model (LLM)-based systems have consequently been increasingly investigated for personalized information

services and recommendation applications. However, directly generating answers using a generative AI model can introduce factual reliability problems, particularly in domains such as public policy where eligibility requirements, support amounts, application periods, and administrative conditions may change over time. For this reason, retrieval-grounded approaches that provide relevant source information to the language model can be more appropriate for policy-related question answering than unrestricted generation alone.

Despite these developments, there remains a practical gap between individual recommendation technologies and an integrated public policy information service. Existing recommender-system research has primarily focused on improving recommendation algorithms, while e-government studies have examined personalized service delivery and information overload from a public-service perspective [1], [2], [4]. In practice, however, citizens require a continuous information process rather than a recommendation algorithm alone. They need to search for policies, examine detailed eligibility conditions, save policies of interest, receive personalized recommendations, monitor deadlines, and ask questions about policy information. Integrating these functions into a single service can reduce fragmentation in the policy information-seeking process.

To address this issue, this paper presents ZoopZoop, an integrated personalized public policy recommendation and notification system. ZoopZoop combines policy search, detailed policy information, policy scraping, behavior-based recommendation, profile-based recommendation, proactive notification, and an AI chatbot within a unified web service. The system obtains public policy information through government public data APIs and periodically synchronizes policy lists, detailed policy information, and eligibility conditions. This approach allows the service to maintain a centralized policy information base while reducing the need for users to visit multiple sources.

The core recommendation mechanism of ZoopZoop combines two complementary sources of personalization. First, behavior-based recommendation analyzes users' recent search keywords and policy-view histories. Recommendation scores are calculated using information such as search terms, the support types and organizations associated with previously viewed policies, recency, activity frequency, and policy popularity. Second, profile-based recommendation compares policy eligibility conditions with user profile attributes, including age, income range, residential area, employment status, gender, household size, and marital status. The combination is intended to address the limitations of relying on a single information source, particularly when behavioral data are insufficient for new users.

ZoopZoop also extends recommendation from passive information presentation to proactive information delivery. The notification module generates notifications for approaching deadlines, newly registered policies, and personalized recommendations. Because excessive notifications can reduce their usefulness, the system incorporates duplicate-notification prevention, daily notification limits, and cooldown periods. This design reflects the need to balance the benefits of proactive information delivery with the potential burden of notification overload.

In addition, ZoopZoop incorporates a conversational AI chatbot to support natural-language exploration of public policy information. Rather than allowing the language model to independently generate policy-related information, the

system retrieves relevant policy data from its internal database and provides the retrieved information as contextual input for response generation. This retrieval-based design is intended to reduce the risk of generating policy information that is not supported by the available policy database. The main contributions of this study are therefore threefold. First, the study presents an integrated architecture that combines public policy search, personalization, notification, and conversational AI within a single service. Second, it proposes a practical hybrid personalization approach that utilizes both user behavioral information and user profile information to support policy recommendation under different levels of user interaction history. Third, it demonstrates a retrieval-supported conversational interface that connects natural-language user queries with structured public policy information. Unlike a standalone recommender system, the proposed service integrates recommendation with policy management and proactive notification, providing an end-to-end framework for personalized public policy information access. The remainder of this paper is organized as follows. Section II describes the overall system architecture and implementation of ZoopZoop, including the frontend, backend, database, policy data synchronization, recommendation system, notification mechanism, and AI chatbot. Section III presents the implementation results and discusses the major development issues and solutions. Section IV concludes the paper and discusses limitations and future research directions.

II. SYSTEM DESIGN & IMPLEMENTATION
The system consists of a React-based client, a Spring Boot-based REST API server, a PostgreSQL database, public data APIs, and the OpenAI API. Users can access policy search, policy detail viewing, policy scraping, My Page, notifications, and AI chatbot functions through a web browser, while the client communicates with the backend server using Axios. The backend server handles user authentication, policy data processing, recommendation logic, notification generation, and community functions. Major data are stored in PostgreSQL.

A. Overall System Architecture
The overall system architecture is divided into the client layer, server layer, data layer, and external service layer. The client layer is implemented using React, and it receives user input and displays policy search results, recommended policies, notifications, and chatbot responses on the screen. The server layer is implemented using Spring Boot and processes client requests through REST APIs. The server performs authentication and authorization verification, policy retrieval, policy synchronization, recommendation score calculation, notification generation, and community post processing. The data layer uses PostgreSQL to store user information, policy lists, detailed policy information, eligibility conditions, search logs, scraped policies, notifications, and community data. The external service layer consists of public data APIs that provide public policy data and the OpenAI API, which is used to generate AI chatbot responses.

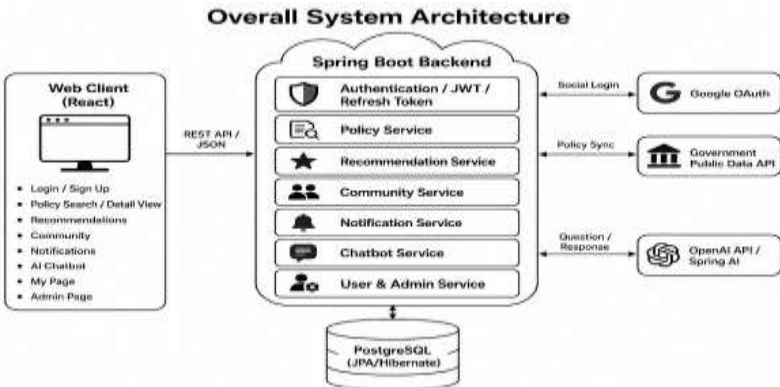


Fig. 1. Overall System Architecture of ZoopZoop

Fig. 1 shows the overall system architecture of ZoopZoop. The React-based web client communicates with the Spring Boot backend through REST APIs in JSON format, and the backend processes user requests through functional services.

In addition, PostgreSQL stores the main data, while external APIs are used for social login, policy data synchronization, and AI chatbot response generation.

TABLE I. MAIN FUNCTIONAL MODULES OF ZOOPZOOP

Component	Role
React Frontend	Provides user interfaces for policy search, recommendation, notification, chatbot, community, My Page, and admin page.
Spring Boot Backend	Handles REST APIs, authentication, policy logic, recommendation, notification generation, chatbot requests, and community functions.
PostgreSQL Database	Stores users, policies, eligibility conditions, search logs, scraps, notifications, and community data.
Government Public Data API	Provides public policy list, detail, and eligibility condition data.
Google OAuth	Supports social login.
OpenAI API / Spring AI	Generates chatbot responses for policy-related questions.

B. Frontend Implementation

The frontend was implemented using React [4], and its main screens include the main page, policy list page, policy detail page, AI chatbot page, community page, My Page, notification settings page, and admin page. Users can search for policies by entering keywords on the main page, and they can filter policies by support type, age, region, and special conditions on the policy list page. On the policy detail page, users can check detailed policy information such as eligibility criteria, application methods, application deadlines, responsible organizations, and required documents. On My Page, users can manage their profile, scraped policies, and notification settings. User profile information is used for profile-based recommendation, while scraped policy information is used to generate deadline notifications. The frontend communicates with the backend through Axios-based API modules and handles functions that require user authentication by including the JWT access token in the request header [3]. When the access token expires, the system issues a new access token using the refresh token.

C. Backend and Database Integration

The backend was implemented using Spring Boot [2], and maintainability was improved by separating domains according to their functions. The main domains include authentication, user, policy, recommendation, notification, search log, community, and admin functions. Each domain was implemented based on a Controller, Service, Repository, DTO, and Entity structure. The Controller handles client requests, the Service performs business logic, and the Repository communicates with the database through JPA. PostgreSQL was used as the database [6]. Policy data are stored separately in policy list, policy detail, and eligibility condition tables. User-related data consist of user information, refresh tokens, scraped policies, search logs, notification settings, and notification lists. In addition, posts, comments, and report data are managed separately for the community function. This structure enables efficient retrieval and utilization of data required for policy search, recommendation, and notification generation.

TABLE II. REPRESENTATIVE REST API ENDPOINTS OF ZOOPZOOP

Category	Method	Endpoint	Purpose
Authentication	POST	/api/auth/login	Authenticates users and issues tokens.
Policy	GET	/api/policies	Retrieves policy lists with search and filters.
Policy Detail	GET	/api/policies/{serviceId}	Retrieves detailed policy information.
Scrap	POST / DELETE	/api/policies/{serviceId}/scraps	Adds or removes a scrapped policy.
Recommendation	GET	/api/recommendations/personalized	Provides behavior log-based recommendations.
Profile Recommendation	GET	/api/recommendations/profile-based	Provides profile-based recommendations.
Notification	GET / PUT	/api/notifications/recent, /api/notifications/settings	Retrieves notifications and updates settings.
Chatbot	POST	/api/chatbot/ask	Generates chatbot responses.
Community	GET / POST	/api/community/posts	Retrieves or creates community posts.

D. Policy Data Synchronization

ZoopZoop implemented a policy data synchronization function using government public data APIs to provide up-to-date public policy information [1]. Policy data are divided into policy lists, detailed policy information, and eligibility condition information. The backend collects each type of data from the APIs, parses and validates the data, and stores it in PostgreSQL. To prevent duplicate storage of the same policy, an upsert method was applied based on serviceId, which is the unique identifier of each policy. In other words, new policies are inserted, while existing policies are updated with

the latest information. The synchronization process supports both manual execution by an administrator and automatic execution by a scheduler. Manual execution is performed through policy synchronization API endpoints, while automatic execution is performed periodically according to a configured schedule. In addition, retry logic was applied to improve synchronization stability because temporary errors may occur during API requests. Through this process, users can access the latest policy information without requiring administrators to manually update the data.

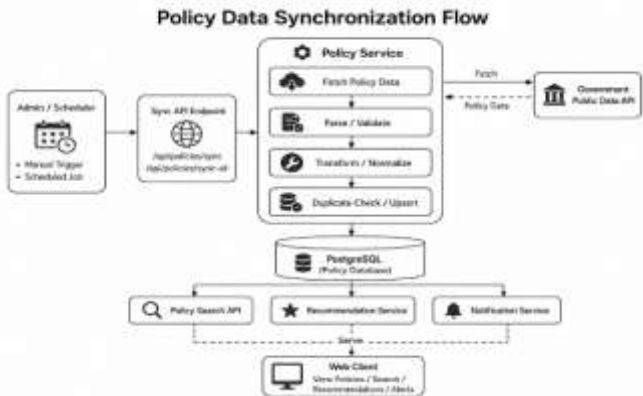


Fig. 2. Policy Data Synchronization Flow of ZoopZoop

Fig. 2 shows the policy data synchronization flow of ZoopZoop. The synchronization process starts with either manual execution by an administrator or automatic execution by a scheduler. The backend collects policy data from government public data APIs, performs parsing, validation, normalization, and duplicate checking, and then stores the data in PostgreSQL. The stored policy data are then used for policy search, recommendation, and notification functions.

E. Personalized Recommendation System

The recommendation system consists of behavior log-based recommendation and profile-based recommendation [7]-[10]. Behavior log-based recommendation uses the user's recent search history and policy view history. When a user searches for a policy or views a policy detail page, the corresponding behavior is stored as a search log. The recommendation logic calculates the scores of candidate policies by considering

search keywords, the support types of viewed policies, responsible organizations, recency, activity frequency, and policy popularity. Policies with higher scores are then prioritized and recommended to the user.

Profile-based recommendation uses user information stored on My Page. The main information used for recommendation includes age, income range, residential area, employment status, gender, household size, and marital status. The system calculates suitability by comparing the eligibility conditions of policies with the user profile. For example, it checks whether the user's age falls within the eligible age range of a policy and determines whether income range or residential area conditions match the policy requirements. Through this process, users can receive policy recommendations that are more suitable for their circumstances.

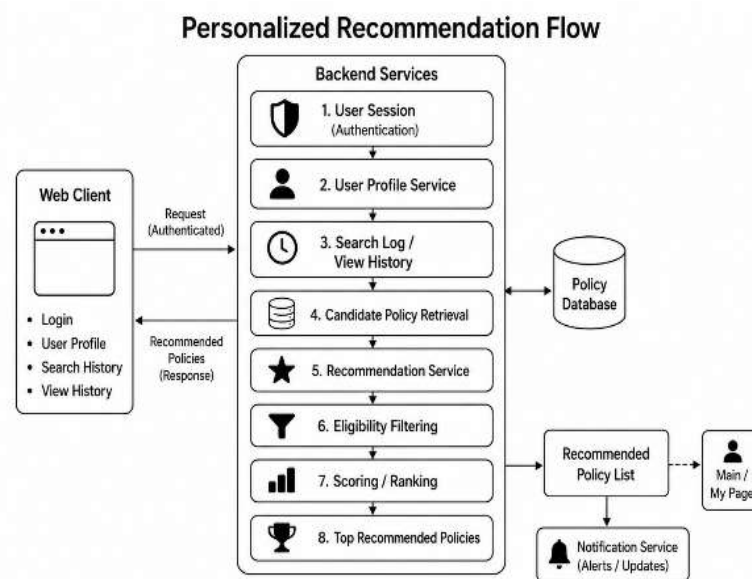


Fig. 3. Personalized Recommendation Flow of ZoopZoop

Fig. 3 illustrates the overall processing flow of the personalized recommendation function. The user's profile information, search history, and view history are used as input data to select candidate policies for recommendation. The system then performs eligibility filtering and score calculation to generate the final recommended policies. The results are displayed on the main page and My Page and are also used for notification functions.

F. Notification System

The notification system was implemented to help users avoid missing important policy updates. The notification types consist of deadline notifications for policies approaching their application deadlines, newly registered policy notifications, and recommended policy notifications. Deadline notifications are generated for scraped policies whose application deadlines are approaching. Newly registered policy notifications are generated based on recently added policies, while recommended policy notifications are generated based on the results of the recommendation system. To prevent excessive notifications, duplicate notification prevention logic and a daily notification limit were applied. In addition, a cooldown period was set so that recommended policy notifications are not repeatedly generated within a certain period. Users can configure whether to receive each

notification type on the notification settings page in My Page, and they can check the recent notification list and the number of unread notifications.

G. AI Chatbot Integration

ZoopZoop provides an AI chatbot function that allows users to ask policy-related questions in natural language. When a user enters a question on the chatbot screen, the frontend sends the message and session information to the backend API. The backend analyzes the user's question and retrieves related policy data when necessary. To improve response reliability, the chatbot uses the retrieved policy data as contextual information before generating an answer. The OpenAI API [5] is used to generate AI responses, and the generated response is displayed on the screen in a form that is easy for users to understand.

The chatbot function was designed as a supporting feature to guide users to policy information that may be difficult to find through simple search. For example, when a user describes their situation in a sentence, the chatbot can help explore related policies or suggest recommended policies. Through this function, users can obtain the information they need in a conversational manner without directly comparing complex policy conditions.

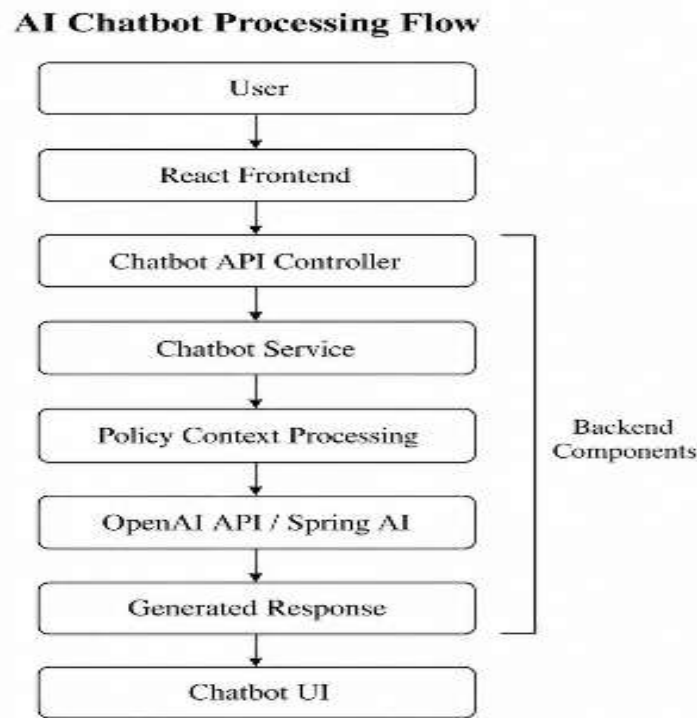


Fig. 4. AI Chatbot Processing Flow of ZoopZoop

Fig. 4 shows the AI chatbot processing flow of ZoopZoop. When a user enters a policy-related question on the chatbot screen, the React frontend sends the question to the chatbot API of the Spring Boot backend. The backend constructs policy-related context based on the question and generates a response using the OpenAI API or Spring AI. The generated response is then returned to the frontend and displayed in the chatbot interface.

III.RESULT AND DISCUSSION

A. System Execution Results

The ZoopZoop system implemented in this study runs in a web environment by integrating a React-based frontend with a Spring Boot-based backend. Users can use the service with personal accounts through registration and login. After

logging in, users can access policy search, policy detail viewing, policy scraping, personalized recommendation, notifications, AI chatbot, and community functions. Each function communicates with the backend through REST APIs, and user-specific data are stored in the PostgreSQL database. The policy search screen was implemented to allow users to search for policies based on keywords, policy types, regions, age, and special conditions. Search results are provided in the form of cards, and users can check information such as eligibility criteria, application periods, application methods, responsible organizations, and required documents on the detail page of the selected policy. Through this structure, users can comprehensively access policy information distributed across multiple institutions within a single service.



Fig. 5. Main Page of ZoopZoop

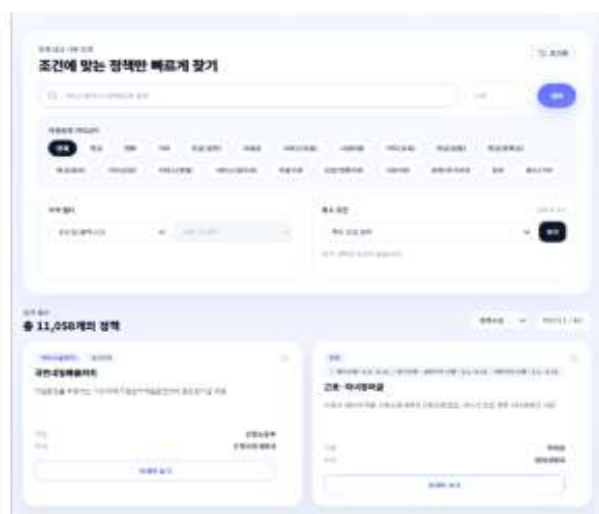


Fig. 6. Policy List and Search Result Page

The personalized recommendation function operates based on the user’s search history, policy view history, and profile information. Behavior log-based recommendation prioritizes policies similar to those that the user frequently searches for

or views, while profile-based recommendation recommends suitable policies by comparing conditions such as age, region, income range, and employment status with policy eligibility conditions. As a result, users can identify policies that match

their circumstances without relying only on simple searches. The notification function was implemented to generate notifications based on the deadlines of scraped policies, newly registered policies, and recommended policies. In particular, deadline notifications help users avoid missing the application periods of policies they are interested in. In addition, the number of unread notifications is displayed so

that users can easily check new information. The AI chatbot function was implemented so that when a user enters a policy-related question in natural language, the backend processes the question and generates an answer using an AI model. Through this function, users can check policy information that is difficult to find through simple keyword searches in a conversational manner.



Fig. 7. Personalized Recommendation Result



Fig. 8. AI Chatbot Result

B. Development Issues and Solutions

During the development process, several issues occurred in policy data synchronization, recommendation quality, AI chatbot response reliability, and notification control. The first issue was duplicate storage during policy data synchronization. Since policy data are repeatedly collected from public data APIs, the same policy could be stored multiple times. To solve this problem, an upsert method was applied based on `serviceId`, which is the unique identifier of each policy. New policies are inserted, while existing policies are updated with the latest information. This approach reduced duplicate data and helped maintain up-to-date policy information.

The second issue was related to recommendation quality. When a new user has insufficient search or view history, behavior log-based recommendation alone may not provide accurate results. To address this limitation, profile-based recommendation was applied together with behavior log-based recommendation. Even when user behavior data are limited, the system can recommend policies by using profile information such as age, region, income range, and employment status.

The third issue was related to AI chatbot response reliability. Since generative AI models may produce information that does not exist in the actual policy database, the chatbot was designed to generate responses based on internal policy search results. Policy names, eligibility conditions, support contents, application methods, and deadlines retrieved from the database were used as reference data for response generation. This reduced hallucination and improved the factual reliability of chatbot responses.

The fourth issue was excessive or duplicate notification generation. If deadline, new policy, and recommended policy notifications are generated without control, users may receive repeated or unnecessary notifications. To solve this problem, duplicate notification prevention logic, daily notification

limits, and cooldown periods were applied. As a result, the system can provide useful notifications while reducing unnecessary notification overload.

C. Discussion

As a result of the implementation, ZoopZoop was confirmed to operate not merely as a service that lists public policy information, but as an integrated service that improves policy accessibility by utilizing user conditions and behavior data. Policy search, detail viewing, scraping, recommendation, notification, and chatbot functions are connected in a single flow, allowing users to more easily find and manage the policies they need.

However, if the number of actual users increases, the system will need to continuously improve recommendation accuracy and notification quality. In addition, although the AI chatbot responses are based on public data and policy-related context, the system should be improved to provide responses together with original information from official institutions, considering the legal validity and up-to-date accuracy of policy information. In future work, more user feedback can be collected, and the weights of the recommendation algorithm can be adjusted to improve personalization performance.

IV. CONCLUSIONS

This paper presented ZoopZoop, an integrated personalized public policy information service designed to improve the accessibility and usability of government policy information. The proposed system combines policy search, policy detail management, personalized recommendation, proactive notification, and conversational AI within a unified web-based platform. By integrating these functions, ZoopZoop provides an end-to-end environment in which users can discover, evaluate, save, and receive information about public policies that may be relevant to their individual circumstances.

A key feature of the proposed system is its combination of behavior-based and profile-based recommendation. The behavior-based recommendation component utilizes users' recent search keywords and policy-view histories to identify their interests, while the profile-based component compares user attributes with policy eligibility conditions. This complementary approach enables the system to provide recommendations even when sufficient behavioral history is not available, which is particularly important for newly registered users. The system also incorporates policy data synchronization through public data APIs, allowing newly registered or updated policies to be reflected in the service without requiring continuous manual intervention.

In addition to recommendation, ZoopZoop provides a proactive notification mechanism for approaching application deadlines, newly registered policies, and recommended policies. Duplicate-notification prevention, daily notification limits, and cooldown periods were introduced to reduce unnecessary or repetitive notifications. The system also integrates an AI chatbot that supports natural-language interaction with policy information. Rather than relying solely on free-form generation, the chatbot uses policy information retrieved from the internal database as contextual information for response generation. This design provides a practical mechanism for connecting conversational interaction with structured public policy data.

The prototype was implemented using React, Spring Boot, PostgreSQL, JWT-based authentication, and external public data and AI APIs. During implementation, several practical issues were identified, including duplicate policy records, insufficient behavioral information for new users, the reliability of generative AI responses, and excessive notification generation. These issues were addressed through service-ID-based upsert processing, the combination of behavioral and profile information, retrieval-supported chatbot responses, and notification-control mechanisms. The resulting system successfully integrates the major functions required for personalized public policy information access into a single service environment.

Despite these results, the current study has several limitations. First, the recommendation mechanism uses manually designed scoring factors and has not yet been extensively evaluated using a large-scale real-user dataset. Therefore, the effectiveness of the proposed recommendation approach cannot be fully established from the current prototype implementation alone. Second, the chatbot depends on the accuracy, completeness, and timeliness of the policy information retrieved from the underlying data source. Although retrieval-based contextualization can reduce unsupported responses, additional mechanisms are required to provide stronger traceability to official policy documents. Third, the current system has been implemented primarily as a web-based prototype, and its scalability and long-term performance under a large number of concurrent users require further investigation.

Future research will focus on quantitatively evaluating the recommendation system using real or sufficiently large policy-user interaction datasets. In particular, recommendation accuracy, ranking quality, cold-start performance, and the contribution of individual behavioral and profile features should be systematically evaluated. The recommendation model can also be extended by learning feature weights from accumulated user feedback and interaction data rather than relying primarily on predefined

scoring rules. In addition, the chatbot can be enhanced through stronger retrieval-augmented generation, explicit citation of official policy sources, and mechanisms for detecting outdated or conflicting policy information. Mobile optimization and the integration of additional government data sources are also important directions for future development.

In conclusion, ZoopZoop demonstrates a practical approach to integrating personalized recommendation, proactive notification, and conversational AI into a public policy information service. While further empirical evaluation is required to validate recommendation effectiveness and user benefits, the implemented system provides a foundation for developing more adaptive and user-centered digital public policy services.

ACKNOWLEDGEMENT

"This research was supported by the MSIT(Ministry of Science ICT), Korea, under the National Program for Excellence in SW, supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation) in 2026"(No. 2024-0-00023).

REFERENCES

1. Public Data Portal, "Government24 Public Service Open API." [Online]. Available: https://www.data.go.kr/data/15113968/openapi.do#/tab_layer_detail_function
2. Spring, "Spring Boot Reference Documentation." [Online]. Available: <https://docs.spring.io/spring-boot/>
3. Spring, "Spring Security Reference Documentation." [Online]. Available: <https://docs.spring.io/spring-security/>
4. Meta Open Source, "React Documentation." [Online]. Available: <https://react.dev/>
5. OpenAI, "OpenAI API Documentation." [Online]. Available: <https://platform.openai.com/docs/>
6. PostgreSQL Global Development Group, "PostgreSQL Documentation." [Online]. Available: <https://www.postgresql.org/docs/>
7. Y.-J. Ahn, H.-Y. Cho, and S.-J. Kang, "Customized Recipe Recommendation System Implemented in the Form of a Chatbot," *Journal of the Korea Academia-Industrial Cooperation Society*, vol. 21, no. 5, pp. 543-550, 2020.
8. J.-H. Jung, W.-J. Lee, H.-W. Choi, D.-Y. Seo, J.-H. Lee, H.-B. Park, and H.-Y. Lee, "Development of a Customized Webtoon Recommendation Platform Using the GPT Model," *Proceedings of KIIT Conference*, pp. 1361-1364, 2025.
9. J.-W. Son, H.-B. Kim, D. Yoon, S.-H. Park, K.-I. Lee, and S.-J. Kim, "Development of an AI Chatbot-Based Personalized Restaurant Recommendation System," *Proceedings of KIIT Conference*, pp. 1273-1277, 2025.
10. S.-R. Jung and J. Kim, "Sentiment-Fused Hybrid Recommender System for Movie Domain: Integrating Aspect-Based Emotion Analysis with Collaborative-Content Filtering," *Journal of the Korea Big Data Society*, vol. 10, no. 1, pp. 209-219, 2025.