International Journal of Computer Techniques–IJCT Volume 12 Issue 6, November 2025

Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

CoursePlate: Personalized Restaurant Recommendation System

First Author et al., International Journal of Computer Science and Mobile Computing Vol. 3, December-2025, pg. 1-4

[1]Donggwan Kwon, [2]Sunghoon Jeon, [3]Seungmin Lee, [4]Jaeyoung Yeom, [5]Seungjae Lee

[1,2,3,4,5]Computer Engineering, Sunmoon university, Korea

[1]d0926@naver.com, [2]kkrjbh123456@naver.com, [3]sml0801@sunmoon.ac.kr, [4]duawodud12@naver.com, [4*]leeko@sunmoon.ac.kr(Corresponding Author)

Abstract

This paper presents a personalized restaurant recommendation system, CoursePlate, leveraging open-source tools and APIs, including React Native, Expo, MongoDB, and Naver Maps API. The system provides real-time personalized food recommendations based on user preferences and location, offering an intuitive user interface and seamless navigation experience. The integration of Naver Maps API allows for easy location-based services, while the receipt verification system ensures the credibility of user reviews, preventing fraudulent ones and improving trust in the platform.

Keywords

Open Source, Restaurant Recommendation, Naver Maps API, React Native, MongoDB

I. Introduction

In today's world, personalized recommendation systems have become a cornerstone of modern applications, especially in the food service industry. Our project, CoursePlate, aims to develop a solution that personalizes restaurant recommendations for users based on their preferences and locations. This system integrates several open-source tools and APIs, including Expo, React Native, Naver Maps API, and MongoDB, to deliver a seamless user experience for food exploration and navigation [1, 2, 3].

II. Related Work

Several systems have attempted to personalize restaurant recommendations. For instance, DiningCode and Tripadvisor offer restaurant suggestions based on reviews and ratings[4].

Table 1. Comparison with Existing Services (DININGCODE)

Feature	CoursePlate	DININGCODE
Recommendation Method	Recommendation after user survey analysis	Recommendation based on user reviews and ratings
Recommendation Criteria	Considers user food preferences and time of day	Popular restaurants and visitor ratings

Table 2. Comparison with Existing Services (Tripadvisor)

Feature	CoursePlate	Tripadvisor
Recommendation Method	Recommendation after user survey analysis	Recommendation based on user reviews and ratings
Recommendation Criteria	Considers user food preferences and time of day	Popular restaurants and visitor ratings

However, these systems do not incorporate user preferences or real-time location data to offer truly personalized experiences[5]. Our approach utilizes AI-driven recommendations and location-based navigation, enhanced by real-time data from Naver Maps API.

III. System Architecture

The architecture of the CoursePlate system is composed of the following major components:
- Frontend (Mobile App): Developed using React Native for cross-platform compatibility, leveraging Expo for a streamlined development process.
- Backend: Built using Spring Boot, Flask and MongoDB to handle user data, restaurant information, and recommendation logic[6].
- Open API Integration: Uses Naver Maps API for real-time location services and restaurant navigation[7].

IV. Open Source and Tools Used

1. Expo (Open Source)
- Purpose: We used Expo for mobile app development as it provides a powerful framework for building React Native applications. It simplifies the process by offering tools for rapid development, testing, and deployment.
- Contribution: Expo was essential in setting up the Bare Workflow for integrating Naver Maps API, which is a native module requiring non-managed project settings.

2. React Native (Open Source)
- Purpose: React Native enables us to write native applications using JavaScript and React. It supports building apps for both iOS and Android with a single codebase, making it ideal for our cross-platform application development[8].
- Contribution: With React Native, we developed the user interface and managed state across the app, integrating location services and recommendation algorithms.

3. Naver Maps API (Open API)
- Purpose: The Naver Maps API is used to display real-time maps, restaurant locations, and enable navigation features[9].
- Contribution: By integrating Naver Maps API, we provided users with an interactive map to explore recommended restaurants and access real-time navigation to their destinations.

4. MongoDB (Open Source)
- Purpose: MongoDB was used as the primary database to store user preferences, restaurant data, and recommendation results[10].
- Contribution: With MongoDB, we can efficiently store and retrieve large datasets in a document-oriented structure, ideal for handling diverse restaurant data and user information.

5. CoolSMS (API Service)

- Purpose: CoolSMS was used to implement SMS-based features such as user verification and notification [10].

- Contribution: By integrating CoolSMS, we enabled the system to send verification codes during registrati on and provide users with real-time SMS alerts, enhancing user engagement and security.

6. Spring Boot (Open Source)

-Purpose: Spring Boot was used to build the backend REST API that handles user authentication, data proc essing, and business logic[11].

- Contribution: With Spring Boot, we developed a scalable and efficient server-side architecture that integra tes seamlessly with MongoDB and external APIs such as CoolSMS.

7. Flask (Open Source)

-exposes the /analyzeREST endpoint for sentiment scoring and keyword extraction[12].

V. Research Results of Technology Development

1. App Development Results

● User Authentication and Registration

○ Functionality: We implemented email/password-based user registration and login features. The system uses JWT (JSON Web Tokens) for handling authentication and token renewal, ensuring secure access to the application.

○ State Management: The system stores session data in AsyncStorage. The token is retained during the s ession, and upon app restart, users are prompted to log in again.

2. Expo Bare Workflow Transition & Naver Maps API Integration

● Expo to Bare Workflow Transition

○ Functionality: In order to integrate native modules such as Naver Maps API, the Expo Managed Work flowwas transitioned to a Bare Workflow. This allowed us to fully utilize the native functionality require d for handling map features and geolocation services.

○ Implementation Details: After transitioning to the Bare Workflow, we integrated the '@mj-studio/reac t-native-naver-map' module, which allowed us to display the map and use navigation features directly wi thin the React Native environment.

3. Location-Based Restaurant Recommendation and Mapping

● Real-Time Restaurant Markers on the Map

○ Functionality: The system uses Naver Maps APIto display markers for recommended restaurants base d on user preferences and location. Each restaurant's coordinates (latitude and longitude) are used to plac e markers on the map dynamically.

○ Marker Interaction: When a user selects a restaurant from the list, the map automatically zooms in to t hat restaurant's location, highlighting the marker. This allows for a more engaging user experience, wher e the user can explore food options and navigate easily.

○ Implementation Details: We used the animateCameraTo()function to move the map view to the selecte d restaurant and highlighted the marker in a distinct color to indicate the user's selection.

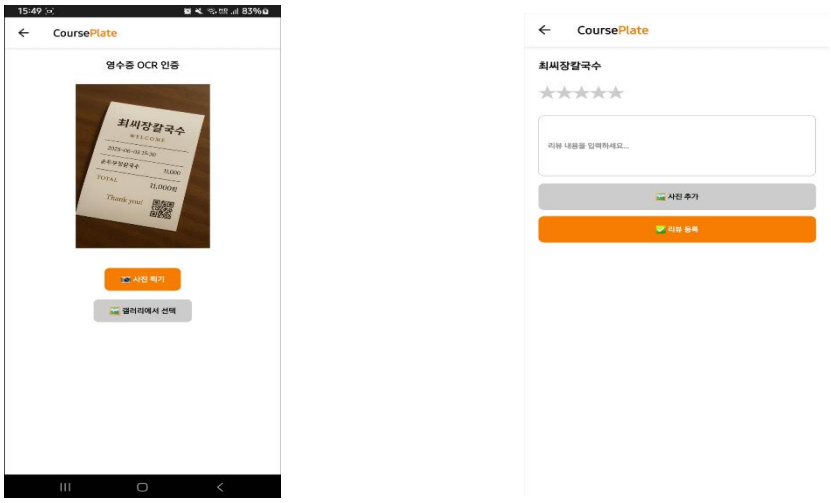
<Fig. 1>: Restaurant markers on the map with highlighted selection.

4. Receipt Verification Review System

● Functionality: We implemented a receipt verification systemto ensure that only legitimate reviews are wr itten. After selecting a restaurant and using the "Get Directions" feature, users have a 48-hour window to w rite a review after providing a valid receipt image.

○ Review Eligibility: The system checks if the user has uploaded the receipt and only allows reviews for users who have valid receipts within the 48-hour period.

○ Backend Integration: The server stores review eligibility data, including the restaurant name, address, and the time of receipt upload. This system ensures that reviews are only written by actual customers, in creasing the reliability and trustworthiness of the information provided on the platform.



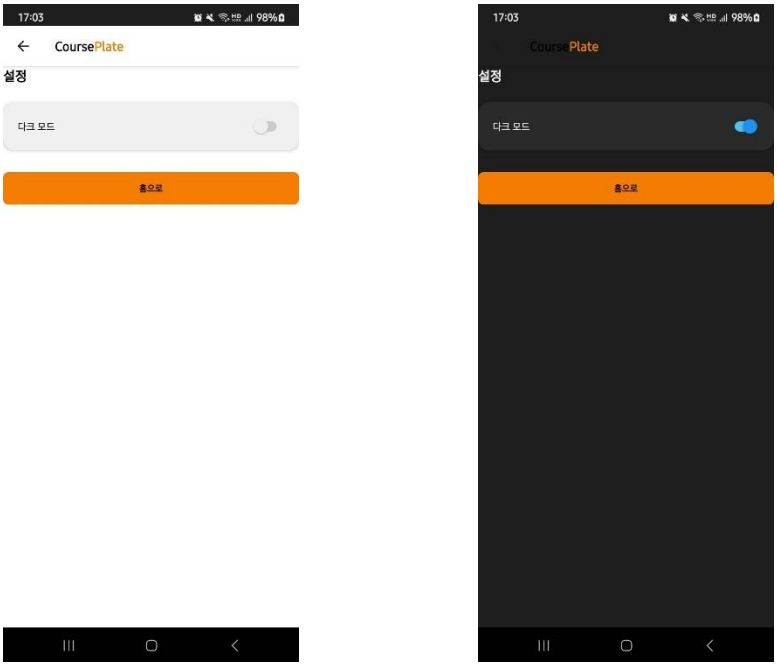
<Fig. 2>: receipt verification

5. Dark/Light Mode and Local Caching

● Dark and Light Mode Management

○ Functionality: We implemented a dynamic theme systemthat allows users to switch between dark and l ight modes based on their preferences. The useTheme()hook manages the theme state, and it is applied gl obally across the app to ensure consistency in UI elements.

○ Local Data Caching: Using AsyncStorage, the app stores essential data like user preferences, region, a nd previously visited locations offline, allowing the system to function without constant network access. When the network is available again, the app synchronizes with the backend to ensure data consistency.



<Fig. 3>: UI showing theme switching between dark and light modes.

6. Explanation of Figures:

Fig. 1: The restaurant map markersimage shows how restaurants are displayed on the map with interactiv e markers that highlight when selected.

Fig. 2: The receipt verification figure depicts how receipts are verified before users can submit a review.

Fig. 3: The theme management UIillustrates how the app supports dark and light modes for improved us er experience.

7. AI Server (Flask) Implementation

Your aiserver/directory provides a complete pipeline for review analysis:

-main.py

Exposes a single POST endpoint (/analyze) that accepts JSON

Delegates analysis to review_analyzer.analyze_text()and returns a JSON response.

-review_analyzer.py

analyze_text(text: str):

1. Vectorizes input via the TF–IDF model.

2. Runs the sentiment classifier to produce positive/negative.

3. Applies simple frequency-based keyword extraction from preprocessed tokens.

4. Returns a tuple (sentiment_label, confidence_score, keyword_list).

-utils.py

Implements preprocessing steps (lowercasing, re-based tokenization, stop-word filtering)

Serializes the tuple into JSON with keys "sentiment", "score", and "keywords".

Outcome:
→ A self-contained Flask service that, given any review text, produces a sentiment label, confidence score, and list of extracted keywords in a single round-trip call.

8. Backend Local Search Service (Spring Boot)

Your NaverLocalSearchServiceclass delivers restaurant data from Naver's Open API:

-Configuration injection

@Value("${naver.client-id}") private String clientId;

@Value("${naver.client-secret}") private String clientSecret;

-searchRestaurants(String query, String location)

1. Constructs the URL

"https://openapi.naver.com/v1/search/local.json?query="

+ URLEncoder.encode(query + " " + location, UTF_8)

+ "&display=10&start=1&sort=random"

2. Opens HttpURLConnection, sets headers:

con.setRequestProperty("X-Naver-Client-Id", clientId);

con.setRequestProperty("X-Naver-Client-Secret", clientSecret);

3. Reads the response stream into a JsonNodevia Jackson's ObjectMapper.

4. Maps each item to your Restaurantentity, populating fields:

_title, address, roadAddress

_telephone, category, link

_Geocoordinates mapx, mapy.

-Persistence & Filtering

(Commented in code) Restaurants can be saved via a RestaurantRepository.

Returned List<Restaurant>is ready for downstream filtering by AI-extracted keywords.

Outcome:
→ A Spring service that reliably fetches up to 10 location-based restaurant entries per query, fully configur ed via properties and ready to integrate with your AI server outputs.

VI. Methodology

We implemented a machine learning-based recommendation system using user survey data and restaurant metadata. The system:
- Collects user preferences through a survey.
- Analyzes these preferences and returns a list of recommended restaurants.
- Integrates with Naver Maps API to provide real-time navigation to the recommended restaurants.
Additionally, we built an authentication and review system that uses receipt verification to ensure the authe nticity of user reviews, preventing fraudulent reviews and increasing the credibility of the system.

VII. Results and Discussion

Our system successfully integrates user preferences with location data to offer personalized restaurant reco mmendations. The Naver Maps API integration allows for seamless navigation to the recommended locatio ns. By implementing receipt verification for reviews, we ensured that only legitimate reviews are presented, boosting the trustworthiness of the information.

VIII. Conclusion

The CoursePlate application, powered by open-source tools like React Native and Naver Maps API, offers a user-friendly and efficient platform for food exploration. The integration of personalized recommendation s, real-time navigation, and reliable review systems has the potential to significantly enhance the dining exp erience. Future work will involve expanding the system's capabilities to include more dynamic data sources and further improve recommendation accuracy.

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 2025"(No. 2024-0-00023).

References

[1] Smith, J., & Johnson, A. (2023). Leveraging Real-time Location Data for Enhanced Personalized Re commendations. Journal of Mobile Applications, 8(2), 112-125.

[2] Chen, L., & Wang, M. (2022). The Role of Open Source Technologies in Developing Smart City Appl

ications. International Conference on Smart Systems and Technologies, 45-50.

[3] Gupta, R., & Sharma, K. (2021). User Experience Design in Mobile Recommendation Systems: A Comprehensive Review. Journal of Human-Computer Interaction, 15(4), 289-302.

[4] Kim, Y., & Lee, S. (2020). Analyzing User Reviews and Ratings for Restaurant Recommendation: A Comparative Study. Journal of Big Data Analytics, 7(1), 55-68.

[5] Brown, P., & Davis, M. (2024). Challenges in Incorporating Real-time User Preferences in Recommendation Algorithms. Conference on Artificial Intelligence and User Behavior, 88-95.

[6] Patel, S., & Singh, V. (2023). Microservices Architecture for Scalable Recommendation Systems using Spring Boot and MongoDB. International Journal of Software Engineering, 10(3), 201-215.

[7] Choi, H., & Park, J. (2022). Integration of Geospatial APIs in Location-Based Services: A Case Study with Naver Maps. Journal of Geographic Information Science, 25(1), 10-23.

[8] Garcia, L., & Rodriguez, F. (2021). Cross-Platform Mobile Development with React Native: Benefits and Best Practices. Mobile Computing and Applications, 18(2), 145-158.

[9] Lee, K., & Kim, M. (2023). Dynamic Map Rendering and Navigation Features in Location-Based Mobile Applications. Journal of Digital Cartography, 6(1), 77-89.

[10] Zhao, X., & Li, H. (2022). NoSQL Databases for Handling Large-Scale User Preference Data in Recommendation Engines. Database Management Journal, 14(4), 310-324.

[11] Wang, Y., & Liu, C. (2024). Developing Robust RESTful APIs for Recommendation Systems using Spring Boot. International Journal of Web Services Research, 21(1), 30-45.

[12] Miller, A., & Taylor, B. (2023). Machine Learning Approaches for Personalized Content Recommendation: A Survey. AI and Data Science Review, 5(3), 160-175.