Engineering and Technology Journal e-ISSN: 2456-3358

Volume 10 Issue 11 November-2025, Page No.-7771-7776

DOI: 10.47191/etj/v10i11.09, I.F. – 8.482

An AI-Powered Mobile Application for Reading Management with a Hybrid Recommendation System

Jaemin Lee[1], Woojin Jang[2], Kyungho Kim[3], Seungjae Lee[4*]

[1,2,3,4]Department of Computer Engineering, Sunmoon University, Korea

ABSTRACT: This paper presents the design and implementation of BookMark, an AI-based mobile reading management application developed to enhance user engagement and provide personalized reading experiences. The system first employs Optical Character Recognition (OCR) to extract text from book covers captured by the user and integrates external APIs to retrieve accurate bibliographic data. It then applies a GPT-based model to generate concise summaries of book descriptions and to create adaptive reading challenges across multiple difficulty levels. Furthermore, a KoBERT-based recommendation engine analyzes user preferences and reading patterns to suggest relevant titles, ensuring a customized reading journey. The overall system is implemented using React Native for mobile development and Firebase for backend services. By integrating these AI-driven functions, BookMark demonstrates an effective approach to intelligent reading support, offering users not only efficient access to information but also motivation to sustain continuous reading habits.

KEYWORDS: OCR, Book Recommendation, Book Information, Text Summarization, Reading Challenge

1. INTRODUCTION

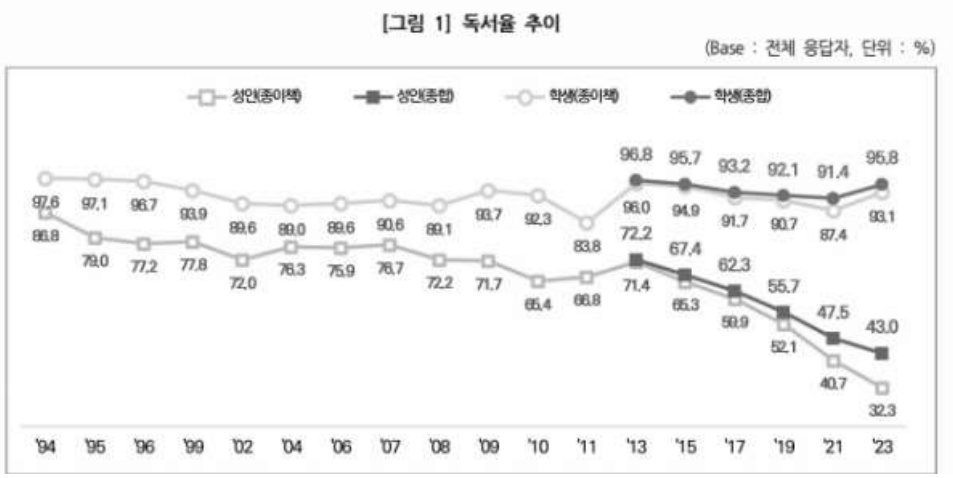

Fig.1. Reading rate trend

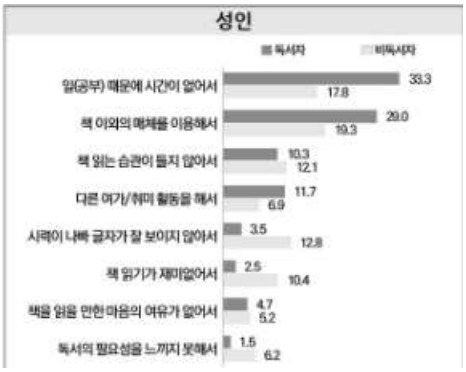

Fig.2. Factors causing reading difficulties in adults

In recent years, the reading rate among adults has shown a steady decline. As illustrated in Fig.1, the adult reading rate

has continuously decreased over the past decade, reaching 43% in 2023, the lowest figure ever recorded. According to the 2023 National Reading Survey conducted by the Ministry of Culture, Sports and Tourism, the primary reasons cited for not reading include lack of time due to work or study (24.4%) and the use of alternative media such as smartphones and digital content (20.6%) . Fig.2 further supports these findings, indicating that insufficient time, absence of consistent reading habits, and the growth of alternative leisure activities are the main factors discouraging adults from reading.

Such a decline in reading rates poses a significant issue, affecting not only individual knowledge acquisition but also the overall development of a reading culture within society. Existing reading-related applications are limited to keyword-based book searches or simple community functions, providing only restricted personalized support. Furthermore, technologies such as Optical Character Recognition (OCR) for book information retrieval or intelligent recommendation systems have been applied only in a few applications.

To address these challenges, this project proposes BookMark, an AI-based reading management application. The system integrates OCR-based book cover recognition, GPT-powered summarization and generation of reading challenges at different difficulty levels, KoBERT-based personalized book recommendation, and a community board for user interaction into a single platform. By combining these functions, the application aims to improve users' reading rates, support the development of sustainable reading habits, and provide a differentiated and intelligent reading experience.

2. SYSTEM ARCHITECTURE AND DEVELOPMENT ENVIRONMENT

2.1 System Architecture

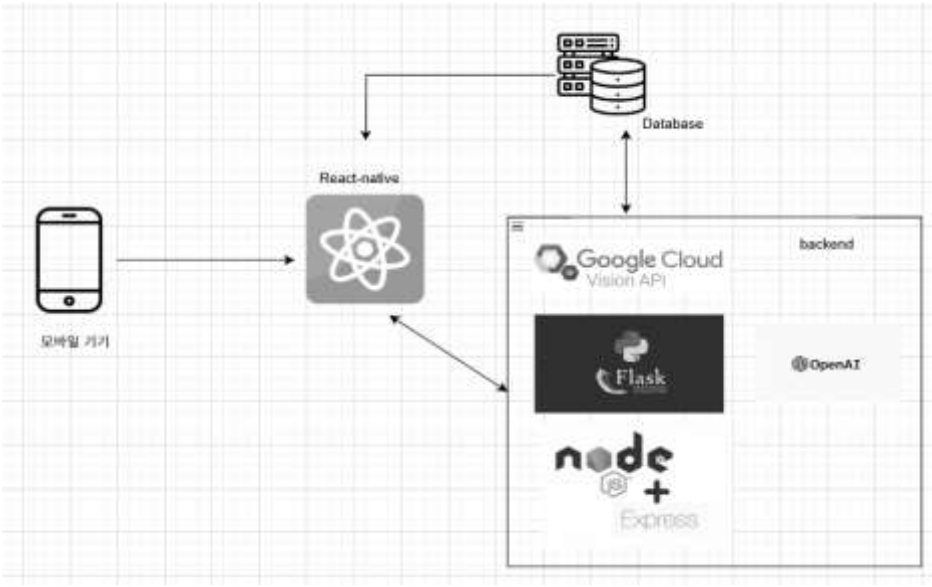

Fig.3. System Architecture

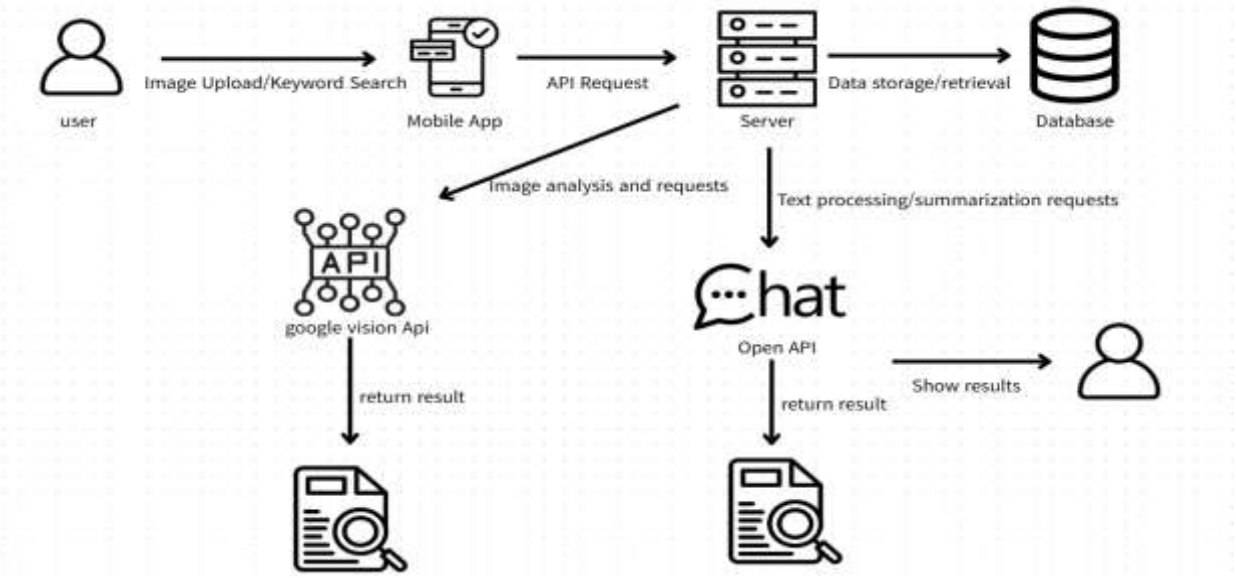

Fig.4. Data Flow

The overall system architecture of the proposed application is illustrated in Fig.3. The system is designed as a client-server model, consisting of a mobile client, a backend server, AI modules, and a database for persistent storage. The mobile client communicates with the backend to utilize various application services, including image recognition, content generation, and data management.

Components Description

Client (Mobile Device):

The client is implemented using React Native, supporting cross-platform deployment on both Android and iOS. Users interact with the application through a graphical user interface (GUI), enabling functionalities such as image capture, text input, participation in reading challenges, community engagement, and personalized book recommendations.

Backend Server:

The backend is responsible for handling client requests, managing data flow, and integrating AI services. It is developed using Flask (Python) and Node.js (Express), providing RESTful APIs for seamless communication between the mobile client and backend services.

AI Modules:

Google Cloud Vision API: Used for image analysis and object recognition on uploaded images.

OpenAI API: Utilized for natural language processing, text generation, summarization, and book recommendation features.

Database:

Based on Firebase, the database stores and manages user accounts, reading preferences, and participation in reading challenges. It is integrated with the backend to ensure persistence and efficient data retrieval.

2.2 Service Flow

Fig.4 illustrates the service flow of the application. Users perform image uploads or keyword searches through the mobile app, and the client transmits this data to the server. Based on the input, the server executes the following processes:

Image Request Processing: Image data is forwarded to the Google Cloud Vision API for analysis and object recognition.

Text Request Processing: Text inputs or recognized results are sent to the OpenAI API for summarization, content generation, and recommendation.

Data Storage and Management: All requests and processed results are stored in the Firebase database, which supports user history management and personalized services.

Result Delivery: The server sends the processed results back to the mobile app, where users can view them via the application interface.

Through this service flow, the system follows a clear sequence of user input → data processing → AI analysis → result delivery, ensuring close cooperation among the client, server, AI modules, and database.

2.3 Development Environment

Frontend: React Native, supporting cross-platform mobile development

Backend: Flask (Python) and Node.js (Express), providing RESTful APIs

AI Services: Google Cloud Vision API (image analysis), OpenAI API (natural language processing and recommendations)

Database: Firebase, integrated with the backend for efficient data management

This architecture ensures modularity, scalability, and efficiency, enabling the application to deliver intelligent and highly responsive services to users.

3. SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

The proposed AI-based reading management application, **Bookmark**, was implemented using **React Native** for cross-platform mobile development, **Firebase** for user authentication and data management, and a **Flask** server for AI model integration.[1] The system features several key modules: OCR-based book recognition, GPT-driven summarization, a hybrid recommendation engine using **KoBERT** and **SBERT**, AI-generated reading challenges, and a community function to foster user interaction.[1]

3.1. OCR functionality

The OCR module utilizes the Google Cloud Vision API to extract text from book covers captured via the in-app camera. The system segments the extracted text into candidate phrases using an

n-gram approach and validates them against external book APIs (**Google Books**, **Aladin**).[1] A string similarity algorithm is then applied to identify the correct book title with high accuracy.[1] The experimental assessment of this module showed consistent recognition accuracy for both Korean and English titles, reliably retrieving the corresponding book information.[1]

Fig.5. Camera Interface

Fig.6. Book Photo OCR Result

3.2. Summary generation function

To reduce cognitive load on the user, book descriptions fetched from external APIs are condensed into 2–3 lines using the OpenAI GPT-3.5 API. The process employs prompt engineering to ensure the generated summaries are concise and contextually relevant. The Express server handles the API calls, ensuring a smooth and efficient process for retrieving the summaries and presenting them to the user. The experimental evaluation confirmed that the function provided readable and coherent summaries that did not require manual editing, allowing users to quickly grasp key information about a book.

Fig.7. Summarized Book Information Output

3.3. Book recommendation function

The hybrid recommendation system generates personalized suggestions by integrating user-selected interest genres with their personal library data. The model was fine-tuned with **KoBERT** to enhance its efficiency and accuracy, particularly for content in the Korean language.[1]

3.3.1. Saved book-based recommendation

The system first analyzes the user's saved books. It uses **K-means clustering** on the **SBERT** embeddings of these books to group similar titles.[1] By calculating the similarity between the cluster centers and a large pool of candidate books, the system recommends a representative book from each cluster.[1] Additionally, a weighted average of the embeddings of the saved books is used to reflect the user's overall reading preferences, generating further recommendations.[1]

3.3.2. Interest genre-based recommendation

The system also recommends books based on the user's selected genres. It extracts frequent keywords from the categories and titles within these genres and uses these keywords to query the **Aladin API** for new candidate books.[1] The system then calculates the **cosine similarity** between the **SBERT** embedding of the user's genre-based query and the embeddings of the candidate books, recommending the top-ranking titles.[1] The results, as shown in Fig.8, demonstrate that users received up to 20 book recommendations.[1] This hybrid approach successfully combined interest-based and history-based methods, resulting in a greater diversity of recommendations.[1]

Fig.8. Book Recommendation Results

3.4. Reading challenge function

The reading challenge module was designed to motivate and engage users by leveraging the generative capabilities of the GPT API to create multi-level reading missions. These challenges, which include measurable objectives such as the number of books to read and a target duration, are stored in **Firebase**.[1] As a user logs completed books, their progress is dynamically updated, providing a clear visual representation of their accomplishments.[1] Prompts for the challenges were designed using external data to prevent the generation of overly simple or generic objectives.[1] As shown in Fig.9 and Fig.10, the system effectively generated challenges that varied by difficulty.[1] The inclusion of clear goals and progress tracking features was successful in motivating users and encouraging consistent reading habits.[1]

Fig.9. Challenge Preview Screen

Fig.10. Challenge After Generation

3.5. Community function

To facilitate social interaction and the sharing of reading experiences, a community feature was implemented as a Firebase-based forum. Users can create posts to share book reviews or recommendations, and they can interact with each other's posts through comment and 'like' functions.1 When a post is created, the image is uploaded to **Firebase Storage** via an **Express** server, and the user's nickname is retrieved based on their User ID (**UID**) to be included in the author's information.[1] The posts created on the community can be seen in the list, as shown in Fig.11.[1] In Fig.12, users were able to write posts and attach images.[1] When a user clicks on a specific post from the list, they can see the content and attached photos, as shown in Fig.13.[1] The comment and 'like' functions were successfully implemented, facilitating smooth communication and contributing to a positive user experience.[1]

Fig.11. Community Post List

Fig.12. Community Post Creation

Fig.13. After Post Creation

3.6. Additional features

The application includes several auxiliary features to enhance the overall user experience:

- **Keyword Search:** A fuzzy search module using **Fuse.js** allows users to search for books by title, author, or description.[1] Results from multiple APIs are merged to improve search accuracy and comprehensiveness.[1]
- **Personal Library:** Users can save books to a personal library, referred to as "My Bookshelf."[1] This saved data is not only for personal organization but also serves as input to refine future recommendations, creating a feedback loop for personalization.[1]
- **Profile and Genre Management:** Users can manage up to three interest genres through their profile settings.[1] These selections are dynamically used to update their recommendation profile, ensuring the suggestions remain aligned with their evolving tastes.[1]

4. CONCLUSION

4.1 Project Summary

This project was developed to move beyond traditional keyword-based search methods by adopting OCR technology, enabling more convenient and efficient book information retrieval. The system was designed to support users in cultivating consistent reading habits and increasing overall reading engagement. Implemented features include book information retrieval through OCR, a personalized recommendation system based on user preferences, a community space for interaction among diverse users, and a reading challenge system tailored to individual difficulty levels.

4.2 Achievements and Contributions

The project demonstrated that OCR-based recognition reduces the inconvenience of keyword-based searches, while the KoBERT-based recommendation system provides personalized book suggestions aligned with user reading patterns. In addition, by utilizing GPT from OpenAI, the system enabled automated summarization of book information and generation of reading challenges, thereby offering users diverse and personalized reading experiences.

4.3 Limitations

The accuracy of OCR is not yet fully reliable, as recognition performance can vary depending on the quality of book cover images. The KoBERT-based recommendation system also remains incomplete, as it combines stored book data and user-preferred genres separately rather than in a fully integrated manner. Furthermore, the system has not yet been sufficiently validated through large-scale experiments with actual users, limiting its evaluation of long-term effectiveness.

4.4 Future Work

Future improvements will focus on enhancing OCR performance to ensure reliable recognition across a variety of book covers, including not only the front but also the back cover. The recommendation system will be refined by integrating user-stored book types and genre preferences into a hybrid model, aiming to deliver more accurate and effective suggestions. Finally, user-centered experiments will be conducted to collect data on recommendation, summarization, and reading challenge systems, providing the basis for continuous improvement and validation of the application's effectiveness.

5. ACKNOWLEDGMENTS

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. Ko, H., Lee, S., Park, Y., & Choi, A. (2022). *A survey of recommendation systems: Recommendation models, techniques, and application fields*. Electronics, 11(1), 141. https://doi.org/10.3390/electronics11010141
2. L. Wu, et al. (2023). *A survey on large language models for recommendation*. arXiv preprint arXiv:2305.19860. https://arxiv.org/abs/2305.19860
3. Ministry of Culture, Sports and Tourism. (2023). *2023 National Reading Survey Report*. Republic of Korea. Retrieved from https://www.mcst.go.kr
4. Nana Ramadijanti & Achmad Basuki. (2017). *Designing mobile application for retrieving book information using optical character recognition*. International Conference on Information & Communication Technology and Systems (ICTS).
5. SKT Brain. (2020). *KoBERT*. GitHub repository. Retrieved from https://github.com/SKTBrain/KoBERT
6. Zhou, H., et al. (2023). *A comprehensive survey of recommender systems with deep learning*. Applied Sciences, 13(20), 11378. https://doi.org/10.3390/app132011378
7. [Author(s) Unknown]. (2017). *Image based book cover recognition and retrieval*. Materials Science and Engineering Conference Series, 263, 042088. https://doi.org/10.1088/1757-899X/263/4/042088