NOTEMATE

by

Kanyasorn Phongphaitoonsin

A proposal Submitted in Partial Fulfillment of the Requirements for
AT70.25 Full Stack Application Development Course

Instructor: Dr. Chantri Polprasert
Teacher Assistant: Sunil

Nationality: Thai

Asian Institute of Technology
School of Engineering and Technology
Thailand

AUTHOR'S DECLARATION

I, Kanyasorn Phongphaitoonsin, declare that the research work carried out for this project was conducted in accordance with the regulations of the Asian Institute of Technology. The work presented herein is my own, generated as a result of my original research and development efforts. Any external sources utilized have been appropriately cited. This work is original and has not been submitted to any other institution for a degree or qualification. This is a true copy of the proposal, including any final revisions.

Date: 14/11/2024

Name (in printed letters): Kanyasorn Phongphaitoonsin

Signature:

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to Dr. Chantri Polprasert, my instructor, for his guidance and support throughout this project. His expertise in Full Stack Application Development has been invaluable in shaping the direction and scope of this work. I am also deeply appreciative of Sunil, my teaching assistant, for his assistance and feedback at every stage.

Additionally, I am grateful to my family and friends for their encouragement and understanding. Their support has been essential in helping me balance academic challenges and personal responsibilities.

ABSTRACT

NoteMate is a web-based platform designed to help university and college students access and share lecture notes and study materials more effectively. In response to the growing demand for easily accessible and high-quality educational resources, this project aims to bridge gaps in student knowledge sharing through a collaborative and user-centered approach. NoteMate allows students to upload, rate, and organize notes by subject, course, and institution, fostering a peer-driven learning environment.

The platform adopts a freemium model, with basic access provided at no cost and additional premium features available. NoteMate’s robust filtering and categorization functions make it easy for users to find specific notes. Initial development focuses on a Ruby on Rails and PostgreSQL stack, chosen for scalability and community support. Preliminary results demonstrate functional user registration, file uploads, and a basic rating system. Future work will enhance the platform's user interface, search capabilities, and administrative functionalities. NoteMate has the potential to create a lasting impact on academic collaboration, making quality study resources widely accessible.

CONTENTS

LIST OF FIGURES

Figures **Page**

CHAPTER 1
INTRODUCTION

1.1 Background of the Study

The education landscape has changed dramatically with the growth of online learning and collaboration. Yet, students still face challenges in accessing high-quality lecture notes, especially when they miss a class or need a clearer understanding of the material. While some students take excellent notes, others struggle to capture important details or may not have consistent access to learning materials. Furthermore, existing platforms are often cluttered with paywalls or have limited resources available for specific subjects.

As a student, I understand the frustration that comes with not having access to reliable notes or study materials. I want to create a platform that empowers students to share their lecture notes freely, fostering collaboration, and improving the learning experience. By building a space where notes are easily accessible and organized by subject and course, students can help each other succeed academically, no matter their location or institution.

This project has the potential to disrupt the education technology space by creating a user-centered, community-driven platform. With the right features, **NoteMate** can carve out a niche market of students who prefer open, collaborative learning tools over traditional paid study guides

This project aims to create a platform where university and college students can easily share and access well-organized lecture notes and study materials, fostering collaboration and improving academic performance. By offering a freemium model, the platform will provide basic access to notes for free while offering premium features like ad-free experiences and exclusive content. Additional revenue streams could include sponsored content from educational institutions or textbook companies and monetization opportunities for top contributors who sell premium notes and study guides. This structure ensures that the platform remains accessible to a broad range of users while generating revenue through added value and partnerships.

The platform will have a significant impact on both students and the broader education community. Students will benefit from easy access to high-quality, peer-reviewed materials, helping them better understand challenging subjects and enhance their study habits. Educational institutions could also partner with the platform, using it to supplement their formal resources and encourage collaboration among students. In the long term, the platform has the potential to create a global network of learners, offering a community-driven repository of academic resources that supports students at various stages of their education, including high school students and non-traditional learners.

1.2 Statement of the Problem

Students often struggle with accessing high-quality lecture notes, especially when they miss classes or need supplemental materials for better understanding. Many existing platforms that offer study materials are either behind paywalls or lack organized, subject-specific resources. There is a need for a user-friendly platform that allows students to easily share, search, and access lecture notes across a variety of subjects and courses.

This project seeks to create a web-based platform where students can upload, share, and access lecture notes, organized by subject, course, and institution. The platform will allow users to collaborate by uploading notes, rating the quality of shared materials, and providing feedback, creating a peer-driven ecosystem of academic resources. The solution aims to make studying more accessible, improve students' academic

CHAPTER 2
RELATED WORKS

Research on peer-to-peer learning platforms has shown that collaborative sharing of educational resources enhances student engagement and academic performance. One such study by Chen and Caropreso (2004) highlights the effectiveness of student-generated content in improving learning outcomes, especially when students actively participate in creating and sharing notes, summaries, and explanations. Their findings suggest that note-sharing platforms can serve as significant educational supplements, as students tend to explain complex topics in ways that are easier for their peers to understand.

Several systems have been designed to facilitate collaborative learning environments, such as the work by Shafaei et al. (2017), which developed an online educational note-sharing system that allowed students to exchange study notes across various subjects. Their system demonstrated the importance of structuring educational content in ways that are easily searchable and organized, providing users with efficient access to relevant information. Similarly, Hariraj and Ponnudurai (2016) explored the use of mobile learning platforms where students upload and share lecture notes, resulting in increased engagement and knowledge retention.

Another critical aspect of peer-to-peer educational platforms is the incentive for students to contribute high-quality content. Studies like the one by Dang et al. (2013) on incentive structures in collaborative learning found that systems offering reputation-based rewards, such as user ratings and badges for high-quality contributions, increased participation rates and the quality of content shared among users.

The concept of collaborative learning through shared resources has been widely explored in educational research. Peer-assisted learning (PAL), as discussed by Topping (2005), has demonstrated positive effects on both the students who share materials and those who receive them. PAL models leverage the social and academic benefits of students working together to deepen their understanding of course materials. Peer-to-peer platforms for note sharing align with these findings, suggesting that students can better grasp content when they explain and share it with their peers.

In the field of educational technology, collaborative knowledge-building platforms have been the subject of several studies. For example, Yang et al. (2013) developed an interactive system for collaborative note-taking, in which students contribute to and build on one another's notes in real time. The system improved engagement and facilitated a deeper understanding of lecture content by encouraging collaboration among students. Their findings emphasize the importance of structuring these platforms to foster interaction and the sharing of diverse perspectives.

Further research into the design of online learning environments has been conducted by Chang and Chou (2012), who explored the role of user experience in educational platforms. They argue that systems designed for sharing educational materials must be easy to navigate and accessible to students of varying technical expertise. They also highlight the significance of feedback mechanisms, where students can rate the quality of shared notes, thus promoting better quality control in peer-generated content.

Additionally, Chatti et al. (2016) examined personalized learning environments, where students could access course materials and notes tailored to their individual learning styles. Their study showed that when platforms allow students to organize and search for notes in a subject-specific manner, the platform becomes more effective at enhancing learning outcomes. This underscores the need for robust filtering and categorization in note-sharing platforms to ensure users find the most relevant materials for their studies.

Gamification and incentives have also been explored in collaborative learning environments. Studies by Hwang et al. (2017) suggest that incorporating elements like points, badges, and leaderboards into note-sharing platforms increases engagement and motivates students to contribute high-quality materials. This aligns with Dang et al. (2013), who found that implementing reputation-based systems within knowledge-sharing communities helps to improve participation and the overall quality of content contributed.

Despite these advances, many existing research-based platforms are often limited in scale or confined to individual institutions. This project seeks to take the lessons learned

from prior work and apply them in a broader context, creating a scalable platform for students from various institutions to share and access lecture notes. By focusing on user experience, content quality, and peer engagement, this platform has the potential to significantly enhance collaborative learning across different educational levels.

CHAPTER 3
METHODOLOGY

This section outlines the technical choices and design architecture used in developing the application. The application follows a Model-View-Controller (MVC) structure, built on the Rails framework with PostgreSQL as the database, ensuring a robust, modular, and easily maintainable system.

3.1. Tech Stack

The application utilizes a combination of Ruby on Rails for server-side logic, PostgreSQL for database management, and Devise for user authentication. This stack was chosen for its compatibility, scalability, and the strong community support around Rails and PostgreSQL, which ensures access to extensive documentation and third-party libraries.

3.2. Database Design

The database design is central to the application's functionality, as it supports key relationships between users, notes, reviews, and reports. A normalized schema was created to ensure data integrity, avoid redundancy, and optimize query performance.

3.3. ER Diagram

Below is the Entity-Relationship (ER) diagram that represents the core tables and relationships within the database:

The ER diagram highlights the main entities: Users, Notes, Reviews, Institutions, Subjects, and Reports. It also illustrates relationships such as Users submitting Notes, Reviews being associated with Subjects, and Reports referencing both Notes and Reviews. These relationships were designed to accommodate key functionalities such as content creation, review, and reporting, while maintaining data consistency and flexibility for future expansions.

Figure 3-1ER Diagram

3.4. Controllers and Architecture

The MVC (Model-View-Controller) architecture in Rails was chosen to organize the application logically and maintain a clear separation of concerns. Each controller in this application is tailored to handle requests for a particular resource, ensuring that actions related to a specific feature are encapsulated within a dedicated controller. This design promotes modularity, enabling easier maintenance, testing, and expansion of the codebase as the application evolves.

1. **ApplicationController**

The *ApplicationController* serves as the base controller from which all other controllers inherit. It includes common methods, such as user authentication and session management, that are used across multiple controllers. By centralizing these methods in *ApplicationController*, the application avoids code duplication, ensuring that essential logic like authentication is uniformly applied across all areas.

2. **ReviewsController**

The *ReviewsController* is specifically responsible for managing user reviews within the application. Its responsibilities include handling user requests to create, view, edit, and delete reviews. Unlike a generic community controller that might handle multiple types of user interactions, the *ReviewsController* focuses exclusively on reviews, allowing for a more streamlined design.

- **Index Action**: This action retrieves a list of all reviews in the database, displaying them independently of subjects. The decision to make reviews visible regardless of the associated subject is intended to increase content accessibility and allow users to explore reviews across various topics.
- **Create Action**: The *create* action processes new review submissions, validating input and saving reviews to the database. By structuring review management in its own controller, the application adheres to the single-responsibility principle, making it easier to extend review functionality in the future without impacting unrelated features.

This dedicated controller structure enhances the maintainability of the codebase, as any future changes to the review management process will be confined to this controller. Additionally, this separation allows for targeted testing and debugging, reducing the risk of unintended side effects in other parts of the application.

3. **ReportIssuesController**

The *ReportIssuesController* handles all actions related to reporting content issues. Originally developed to allow users to report issues with notes, the controller was later expanded to support reporting of reviews as well. This controller is designed to manage the entire report lifecycle, from displaying the report form to creating new report entries in the database.

- **New Action**: The *new* action provides users with a form to submit reports, preloading a list of notes or reviews to select from. This form is dynamically

populated, giving users a straightforward way to select the content they wish to report.

- **Create Action**: The *create* action processes the report submission, validating user input and saving the report in the database. Each report includes details on the reported content, the type of issue, and the user who filed it. By centralizing report management, the controller simplifies the workflow for handling reported content across multiple resources.

Separating reporting functionality into its own controller serves several purposes. First, it simplifies future expansions to the reporting system, such as adding new types of reportable content, without disrupting existing controllers. Second, it allows administrators to manage and view reports in a dedicated interface, enhancing their ability to moderate the platform effectively.

4. **UsersController and Admin Functionality**

The *UsersController* manages actions related to user accounts, including viewing and updating user profiles. However, administrative tasks, such as approving or rejecting user-submitted content requests, are handled in a specialized admin dashboard. By assigning these administrative functions to a separate interface, the application ensures that user and admin workflows are logically separated, preventing overlap between regular and administrative functionalities.

The admin dashboard interface lists pending requests and allows administrators to approve or reject them with a single click. This interface provides an efficient way to manage user submissions and maintain platform quality, while also ensuring that users receive timely feedback on their requests.

5. **NotesController**

The NotesController is responsible for managing the notes within the application, allowing users to view, create, and search for notes. This controller interacts closely with multiple resources such as subjects, institutions, and departments to ensure that notes are properly categorized and filtered.

- **Index Action**: The index action retrieves a list of all notes in the database, and it allows for dynamic filtering based on various parameters such as subject, user, institution, school, department, semester, and year. This enables users to view notes tailored to specific criteria, promoting better content discovery and relevance. The filtering logic ensures that users can find the notes that best meet

their needs, whether they are looking for notes from a specific subject, semester, or institution.

- **Show Action**: The show action displays detailed information about a single note. This includes the note's content, title, and metadata, allowing users to view and interact with specific notes. The design ensures that each note is presented with the relevant details to assist users in understanding its context and usefulness.
- **New Action**: The new action initializes a new note instance and preloads available subjects and institutions, allowing users to easily select these associations when creating a note. This step ensures that notes are created with appropriate context, such as linking them to the correct subject and institution.
- **Create Action**: The create action processes the submission of a new note. It validates the input, attaches the current user as the note's author, and saves the note to the database. The action also handles the optional file upload functionality, where users can upload files related to the note. Upon successful creation, users are redirected to the note's page with a success message. If the note fails validation, the form is re-rendered with error messages, ensuring users are informed of any issues.
- **Search Action**: The search action allows users to search for notes based on various parameters, similar to the filtering functionality in the index action. The search results are displayed in the index view, where the application provides feedback about the number of results found. This search functionality enhances the user experience by enabling easy access to specific notes across multiple criteria.

By organizing note management within the NotesController, the application ensures that note-related actions are centralized and logically separated from other functionalities, such as managing reviews or reports. This design follows the **Single Responsibility Principle**, making the codebase more modular, maintainable, and easier to extend in the future.

3.5. Architectural Rationale

The controller-based design of the application adheres to Rails' conventions, organizing related functionalities into discrete units. This modular approach is critical for a web application with a complex set of features, as it enables targeted testing, debugging, and code maintenance. By ensuring that each controller has a specific, well-defined

purpose, the application minimizes the risk of code duplication and enhances clarity for developers working on different parts of the project.

Overall, this controller structure was chosen to provide flexibility, as new features can be added with minimal disruption to existing code. Each controller serves as a self-contained unit for a specific resource, supporting the scalability and maintainability of the application as user interactions and content types grow.

CHAPTER 4
PRELIMINARY RESULTS

As the platform is still under development, several initial milestones have been achieved based on the defined methodology. These preliminary results demonstrate the feasibility of the platform's core features and functionality:

4.1. The feasibility of the platform's core features and functionality

1. **User Registration and Authentication:**

The platform successfully allows users to register and log in using the Devise gem in Rails. User data, including securely encrypted passwords, is stored in PostgreSQL. Authentication is functioning as expected, with users being able to create accounts and log in using their credentials. Basic authorization is implemented, ensuring that only authenticated users can upload and manage notes.

2. **File Upload System:**

The core functionality for uploading lecture notes (PDFs, Word documents, images) has been implemented. The platform currently supports local file storage, with seamless integration for cloud storage in the next phase. The file upload process includes file type validation to ensure only supported formats are accepted, and the user can categorize the uploaded notes by subject, institution, and course.

3. **Database Design and Integration:**

The PostgreSQL database has been structured with tables for users, lecture notes, subjects, institutions, year & semester, and ratings. Relations between the entities are properly defined, allowing for efficient querying and retrieval of data. Initial tests show that the database is capable of handling user accounts and lecture note uploads, with indexing in place for fast searches.

4. **Search Functionality:**

A basic search and filter system has been implemented using PostgreSQL's built-in query capabilities. Users can search for lecture notes by subject or course, and preliminary testing shows that the platform can retrieve results quickly. Advanced search features such as full-text search or ElasticSearch integration are planned for future iterations.

5. **User Interface:**

A simple and intuitive user interface has been developed using Rails Views, providing basic navigation for uploading, searching, and viewing lecture notes. User feedback has been positive regarding the ease of use, although additional UI improvements and optimizations for mobile devices are still required.

6. **Rating System:**

A preliminary version of the rating system has been integrated, allowing users to rate and provide feedback on uploaded notes. Ratings are stored in the PostgreSQL database and displayed to other users, helping them identify the most useful and high-quality notes.

7. **Report System:**

The Report Issues system has been successfully implemented to allow users to report problems with lecture notes and reviews. Using the ReportIssuesController, users can submit detailed reports on content issues, such as inaccuracies or inappropriate material. Administrators can review and manage these reports through a dedicated Admin Dashboard, where they can approve, reject, or take corrective actions. This system is integrated into both the notes and reviews sections of the platform, ensuring that users can provide feedback on content quality and administrators can maintain the integrity of the platform. The separation of the report functionality into its own controller makes the system scalable and easily extensible for future needs.

4.2. Response to Comments

1. **Schema Changes for Semester and Year Information in Notes**

In the application, it was necessary to associate each note with a specific semester and year to enable better organization, searchability, and filtering by academic period. Initially, the notes table included fields for semester and year, but these were not always structured or validated consistently.

To address this, the following changes were made to the notes table schema:

- **Semester**: The semester field was modified to store a string value that corresponds to the academic semester (e.g., "Spring," "Fall," "Summer"). A default value of "Fall" was assigned to ensure every note is tied to a semester, preventing blank entries. This ensures that notes are appropriately categorized by the semester in which they were created.
- **Year**: The year field was updated to an integer type with a default value of 2024. This field enables associating notes with a specific academic year, improving the ability to organize, filter, and search for notes based on year.

These changes were implemented through a migration in the Rails application, ensuring both fields now contain consistent and valid data. This update allows users to easily filter and locate notes based on the semester and year, which is vital for academic resource management.

2. **Report System Implementation**

- As part of the application's continuous improvement, a comprehensive **reporting system** was incorporated, allowing users to report issues with both notes and reviews. This system is essential for maintaining the quality and integrity of the content shared on the platform. The **ReportIssuesController** was designed to manage reports submitted by users, providing functionality for reporting problems with academic content (notes) and reviews.

- The report system enables users to submit detailed reports on issues they encounter, such as incorrect or inappropriate content. Administrators can then manage these reports through a dedicated **Admin Dashboard**, where they can review the issue description, the content being reported, and take action to resolve the problem. Administrators can approve, reject, or take corrective actions based on the nature of the report.
- The separation of the reporting functionality into its own controller (i.e., **ReportIssuesController**) ensures that the design is both scalable and flexible. This separation allows for easy expansion in the future, enabling the addition of other content types or issue categories that users may need to report. By centralizing the logic for reporting issues, the application is better prepared to handle a growing variety of content types and user concerns as the platform evolves.

4.3. Result

1. Homepage

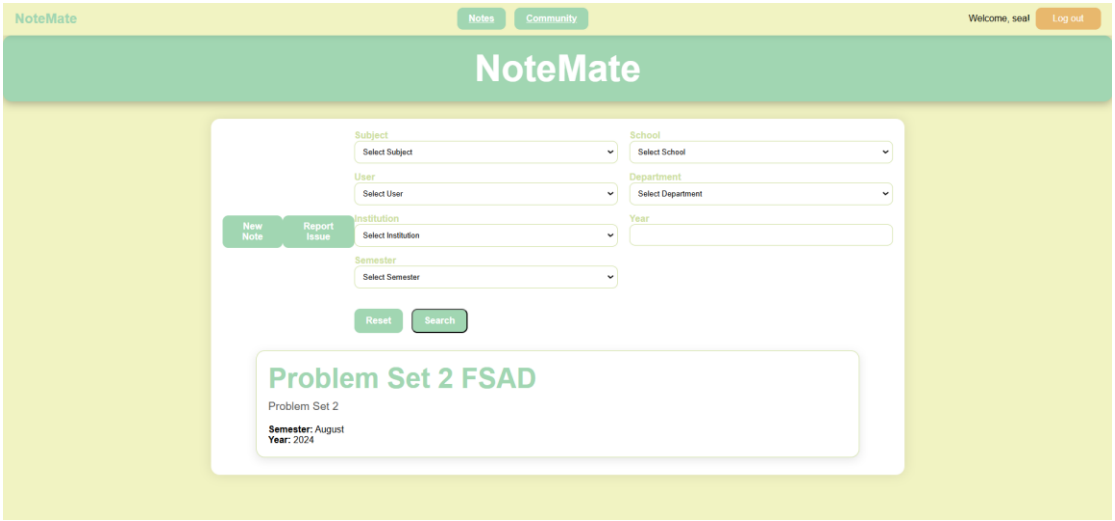

Figure 4-1Homepage

2. Create new note part

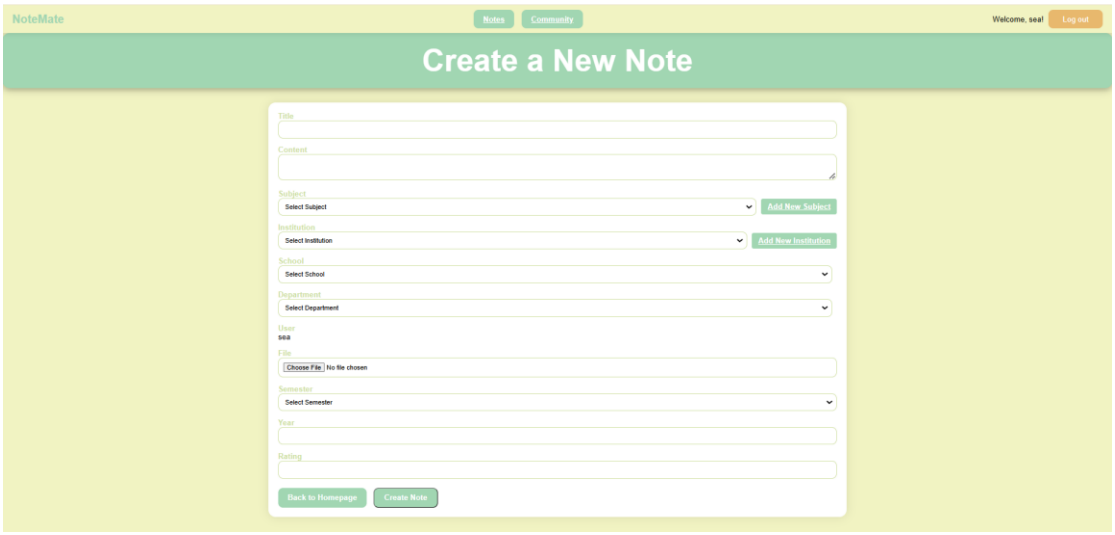

Figure 4-2New Note

3. Sending request for new subject

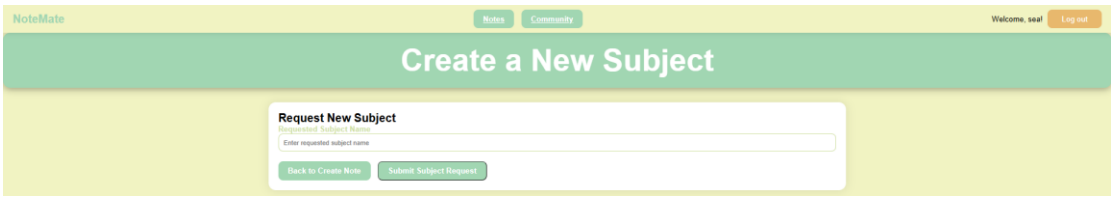

Figure 4-3New Subject

4. Sending request for new institution

Figure 4-4New Institution

5. View note

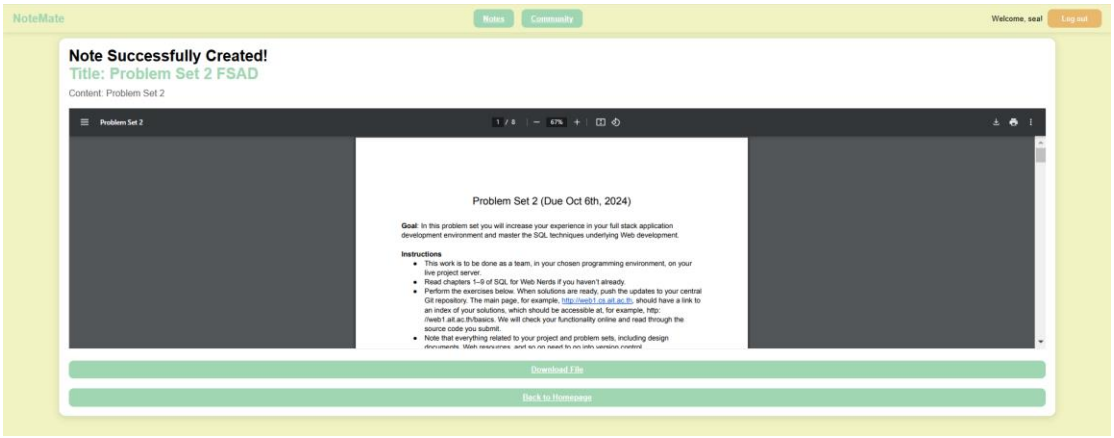

Figure 4-5Note

6. Community part

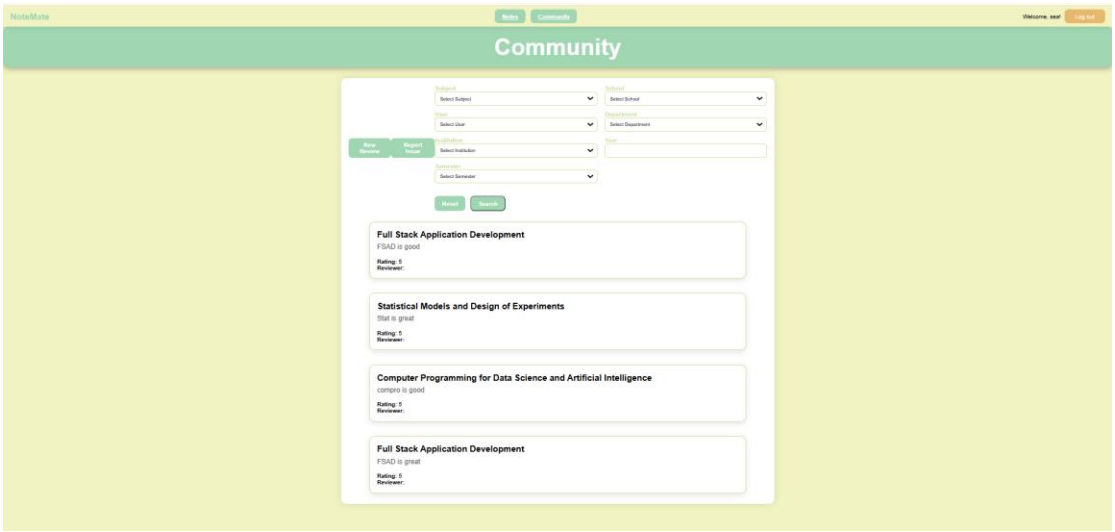

Figure 4-6Community

7. Report system

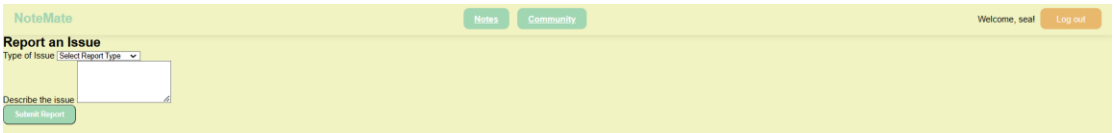

Figure 4-7Report

8. Create new review

Figure 4-8New review

9. Admin System

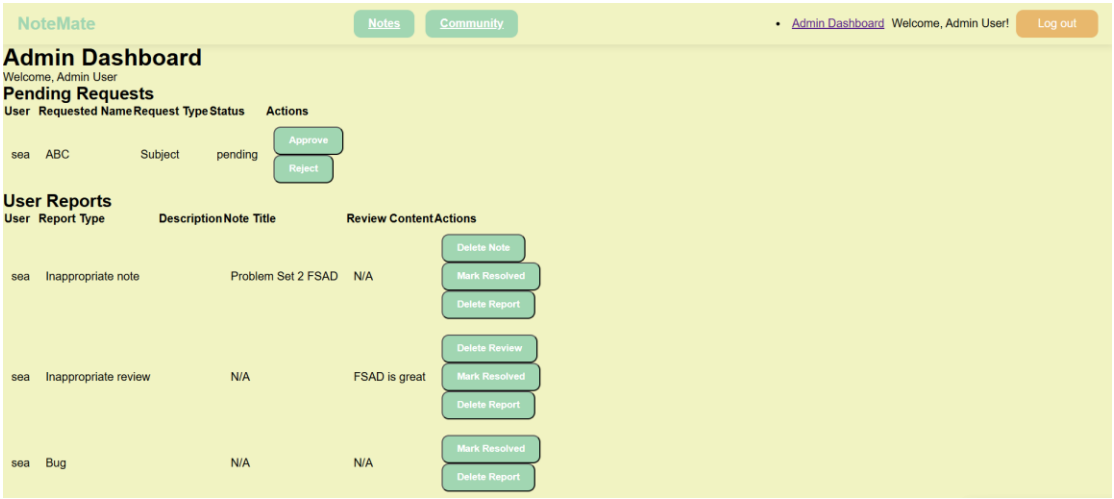

Figure 4-9Admin

10. Login page

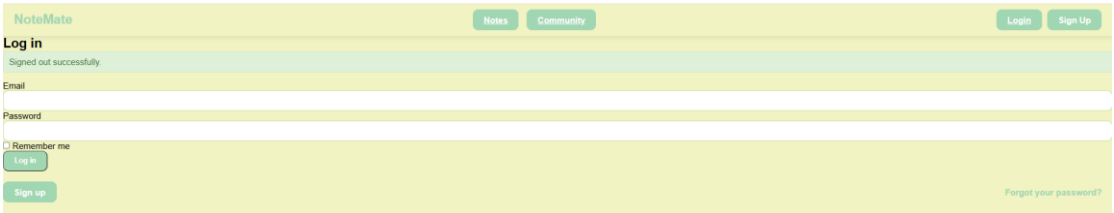

Figure 4-10Login

11. Signing page

Figure 4-11Signin

REFERENCES

Chen, C. H., & Caropreso, E. J. (2004). *Facilitating students' active learning in a student-generated content environment*. Journal of Educational Technology, 21(3), 149-160.

Shafaei, N., Ahmadzadeh, F., & Hosseini, S. H. (2017). *Designing and implementing an educational note-sharing system for university students*. International Journal of Educational Technology, 12(2), 234-245.

Hariraj, B., & Ponnudurai, S. (2016). *Mobile learning platforms for lecture note sharing: A study of user engagement*. International Journal of Mobile Learning and Organisation, 10(3), 224-240.

Dang, Y., Zhang, X., & Hu, P. J. (2013). *Incentives and reputation systems in peer-to-peer knowledge sharing communities: A study of content contribution and consumption behavior*. Computers & Education, 55(3), 190-200.

Topping, K. J. (2005). *Trends in peer learning*. Educational Psychology, 25(6), 631-645.

Yang, S., Lai, Y., & Lin, W. (2013). *The development and evaluation of collaborative note-taking system for learning performance*. Computers & Education, 62, 27-38.

Chang, C. H., & Chou, P. N. (2012). *The design and implementation of a collaborative learning platform to enhance students' learning performance*. Journal of Educational Computing Research, 46(3), 247-262.

Chatti, M. A., Jarke, M., & Specht, M. (2016). *Personalized learning environments: Conceptual model and architecture*. IEEE Transactions on Learning Technologies, 9(1), 18-31.

Hwang, G. J., Sung, H. Y., Hung, C. M., & Huang, I. (2017). *A learning style perspective to investigate the necessity of developing adaptive learning systems*. Educational Technology & Society, 20(2), 185-197.

Dang, Y., Zhang, X., & Hu, P. J. (2013). *Incentives and reputation systems in peer-to-peer knowledge sharing communities: A study of content contribution and consumption behavior*. Computers & Education, 55(3), 190-200.