VIETNAM NATIONAL UNIVERSITY – HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

THESISBOARD: A NODE.JS AND REACT WEB-BASED PLATFORM FOR MANAGING PRE-THESIS AND THESIS PROJECTS

By
Tran Thien
ITITIU21318

A thesis submitted to the School of Computer Science and Engineering in partial fulfillment of the requirements for the degree of
Bachelor of Engineering of Information Technology

Ho Chi Minh City, Vietnam
2025

THESISBOARD: A NODE.JS AND REACT WEB-BASED PLATFORM FOR MANAGING PRE-THESIS AND THESIS PROJECTS

APPROVED BY:

______________________________,

____

____

____

____

THESIS COMMITTEE

DECLARATION OF AUTHORSHIP

Author’s declaration

I firmly declare that this thesis, "THESISBOARD: A NODE.JS AND REACT WEB-BASED PLATFORM FOR MANAGING PRE-THESIS AND THESIS PROJECTS", is my own original work and has not been submitted previously for any academic degree at this or any other institution.

All sources of information and data used in this thesis have been properly acknowledged and referenced. I understand the university's policy regarding academic integrity and plagiarism.

Author: Trần Thiện

17.12.2025

ACKNOWLEGMENTS

I would like to express my sincere gratitude to my supervisor, Mr. Le Thanh Son, for his guidance and clear direction throughout this thesis. His valuable insights and consistent support have been essential to my progress.

I am also grateful to Mr. Nguyen Van Sinh for his engaging lectures on *Web Application Development*, which provided a strong foundation and practical knowledge for building the system. My thanks further extended to Mr. Le Trung Nghia for his patience and thorough support during the laboratory sessions, which greatly enhanced my understanding of web development and strengthened my confidence in the field.

TABLE OF CONTENTS

LIST OF FIGURES ix
LIST OF TABLES x
ABSTRACT xi
CHAPTER I 1
INTRODUCTION 1
1.1. **Background and Context** 1
1.2. **Problem Statement** 1
1.3. **Motivation** 2
1.4. **Project Objectives** 2
1.5. **Scope of the Project** 3
1.5.1. **In-Scope** 3
1.5.2. **Out-of-Scope** 4
1.6. **Methodology Overview** 4
1.7. **Contributions of the Project** 5
CHAPTER II 6
LITERATURE REVIEW 6
2.1. **Overview of Related Work** 6
2.2. **Existing Academic Management Systems** 6
2.2.1. **Moodle (Modular Object-Oriented Dynamic Learning Environment)** 7
2.2.2. **Blackboard** 7
2.2.3. **Manual Tools** 8
2.3. **Related Technologies and Frameworks** 8
2.3.1. **Web-based Systems in Education** 8
2.3.2. **Role-Based Access Control (RBAC) in Academic Systems** 9
2.3.3. **Workflow Automation and Document Management** 9
2.3.4. **Authentication and Authorization (Token-based)** 9
2.4. **Comparative Analysis** 10
2.5. **Summary and Research Gap** 11

2.5.1. Key Findings 11
2.5.2. The Research Gap 11
2.5.3. Addressing the Gap 11
CHAPTER III 12
METHODOLOGY 12
3.1. System Overview 12
3.2. Development Technologies 12
3.2.1. Programming Language: TypeScript 12
3.2.2. Frameworks and Libraries 13
3.2.3. Database Management System 14
3.2.4. External Service: Auth0 14
3.3. System Requirements 14
3.3.1. User Roles and Actors 14
3.3.2. Functional Requirements 15
3.3.3. Non-Functional Requirements 15
3.4. System Analysis 16
3.4.1. Use Case Diagram 16
3.4.2. Use Case Analysis 19
3.4.3. Relationship Analysis 21
3.5. System Design 23
3.5.1. System Architecture 23
3.5.2. Class Diagram 25
3.5.3. Sequence Diagrams 34
A. Pre-Theis functionality 34
B. Theis functionality 36
C. Announcement functionality 37
D. System management functionality 37
3.6. Database Design 38
3.6.1. Relational Table Diagram 39
3.6.2. Data Constraints 40

3.6.3. Normal Form Analysis (3NF) 43
3.7. Summary 46
CHAPTER IV 48
IMPLEMENTATION 48
4.1. Core Features 48
4.1.1. User Management 48
4.1.2. Project Registration 48
4.1.3. Project Management 49
4.1.4. Report Exportation 49
4.2. Supporting Features 50
4.2.1. Automated Scheduling and Timeslot Management 50
4.2.2. Announcement Dashboard 50
4.2.3. Statistics Dashboard 51
4.2.4. Notifications 51
CHAPTER V 52
RESULT 52
5.1. System Interface and User Experience 52
5.1.1. Topic Selection Interface 52
5.1.2. Project Detail and Management View 53
5.1.3. Submission and Evaluation Portals 56
5.1.4. Notification System 57
5.2. Discussion 58
5.2.1. Strengths 58
5.2.2. Limitations 59
5.3. Evaluation 60
CHAPTER VI 62
CONCLUSION AND FUTURE WORK 62
5.4. Project Summary 62
5.5. Achievements and Results 62
5.6. Limitations 63

5.7. Future Enhancements..64
5.8. Conclusion 64
REFERENCES.. 65

LIST OF FIGURES

Figure 1. Use Case Diagram of THESISBOARD 17

Figure 2. MVC Model 24

Figure 3. Class Diagram part 1 25

Figure 4. Class Diagram part 2 26

Figure 5. Class Diagram part 3 27

Figure 6. Class Diagram part 4 28

Figure 7. Pre-Thesis Lifecycle Sequence Diagram 35

Figure 8. Thesis Lifecycle Sequence Diagram 36

Figure 9. Announcement System Sequence Diagram 37

Figure 10. Semester Management System Sequence Diagram 38

Figure 11. Relational Table Diagram 39

Figure 12. Topics Page for Student 53

Figure 13. Inside the Project Page (1) 54

Figure 14. Inside the Project Page (2) 55

Figure 15. Thesis Project Management 56

Figure 16. Announcement System 57

Figure 17. The Notification System 58

LIST OF TABLES

Table 1. General Solutions Comparison 10

Table 2. Actor Descriptions and Responsibilities 19

Table 3. Detailed Use Case Descriptions 21

Table 4. Actor-Use Case Associations 23

Table 5. Class Definitions and Relationships 34

Table 6. Data Constraints Analysis 43

Table 7. Normal Form Analysis 46

ABSTRACT

THESISBOARD is a web-based platform developed to modernize and streamline the pre-thesis and thesis management process at the School of Computer Science and Engineering. Currently, the faculty relies on a fragmented workflow of emails and Google Sheets. This manual approach is often inconsistent, leading to delays, miscommunication, and difficulty in tracking where students stand in their progress.

To solve these issues, THESISBOARD brings the entire process under one roof. It centralizes data, improves coordination, and automates many of the administrative tasks that used to be done by hand. The system is designed with specific roles for students, lecturers, moderators, and administrators, ensuring everyone has access to the tools they need. Key features include topic registration, automatic matching of students to supervisors, defense scheduling, document submission, and evaluation modules.

From a technical perspective, the platform is built on a robust stack: React.js and TypeScript for the front end, Node.js with Express.js for the backend, and MySQL with Sequelize ORM for data management. Security and access control are handled via Auth0.

Ultimately, THESISBOARD replaces static, manual procedures with a dynamic and efficient workflow. It offers a practical solution to increase transparency, reduce administrative workload, and enhance the overall management of thesis activities within the faculty.

CHAPTER I

INTRODUCTION

1.1. Background and Context

Managing pre-thesis and thesis activities is a massive undertaking for any academic program, especially in technology-driven environments like the School of Computer Science and Engineering (CSE). Every year, hundreds of students need to register for topics, find supervisors, submit documentation, and prepare for defenses. Simultaneously, faculty and staff are tasked with tracking this progress, managing quotas, and organizing complex schedules.

Despite the critical nature of these tasks, the current workflow at CSE is surprisingly manual. Reliance on fragmented tools primarily email threads and shared spreadsheets create a fragile system. These methods lack consistency and cannot scale effectively. As student cohorts grow, essential tasks like verifying supervisor availability or tracking submission deadlines become increasingly difficult. This situation creates a clear need for a centralized, systematic solution that can support every stakeholder in the thesis lifecycle.

1.2. Problem Statement

The core issue is that the CSE department currently lacks a unified "brain" to handle the thesis process. Relying on outdated, labor-intensive procedures create several bottlenecks:

- Fragile Registration: Topic registration is managed via editable spreadsheets and email, a method that is prone to human error and difficult to manage.

- Lack of Real-Time Data: Students cannot see supervisor availability in real-time, leading to confusion, double-booking, and missed opportunities.
- Communication Breakdowns: Critical information often gets lost in spam filters or buried in long email chains, slowing down coordination between students and lecturers.
- Disorganized Submissions: There is no central repository for document submission, making it hard to track who submitted what and when.
- Administrative Burden: Faculty staff are forced to manually compile reports and update statuses, a process that consumes valuable time and energy.

These limitations don't just annoy users; they actively hinder the faculty’s ability to oversee the program and the students' ability to focus on their research.

1.3. Motivation

The drive to develop THESISBOARD comes from the necessity for a practical workflow that actually works. A unified system does more than just digitize paper processes, it significantly reduces the administrative burden on staff and ensures that students and teachers have access to accurate, real-time information.

Beyond efficiency, there is a strong motivation for fairness. In the current system, a student might miss a registration slot simply because an email was delayed. By automating these processes, we ensure that every student has equal access to information and opportunities, and that supervisors can manage their workload without unintentionally exceeding their limits.

1.4. Project Objectives

The primary goal of this project is to build THESISBOARD, a web-based platform that automates the chaotic elements of the pre-thesis and thesis process. Specific objectives include:

- Centralization: bringing topic registration and supervisor assignment into one hub.
- Transparency: Providing real-time updates on topic availability and supervisor slots.
- Tracking: Enabling systematic document submission and progress tracking for students.
- Management: Giving lecturers tools to manage their student lists and evaluations efficiently.
- Administration: Automating complex tasks for staff, such as committee assignment, defense scheduling, and report generation.
- Communication: Consolidating notifications and announcements so no critical updates are missed.

1.5. Scope of the Project

1.5.1. In-Scope

This project includes the development of the following core functionalities:

- Role-based access (Students, Lecturers, Moderators, Administrators).
- Topic creation, registration, and automatic student-supervisor matching.
- Assignment logic for supervisors, reviewers, and committees.

- Thesis defense scheduling configuration.
- Document submission and status tracking.
- Notification and announcement systems.
- Evaluation and scoring modules.
- Automated administrative reporting.

1.5.2. Out-of-Scope

The following are explicitly not included in this iteration:

- Native mobile application development.
- Direct integration with the university-wide Student Information System (SIS).
- AI-assisted plagiarism detection or automatic grading.
- Predictive performance analytics.

1.6. Methodology Overview

The system is built as a full-stack web application using modern, component-based architecture.

- Frontend: Built with React.js and TypeScript to ensure a responsive and type-safe user interface.
- Backend: Implemented using Node.js and Express.js for scalable API handling.
- Database: MySQL serves as the relational database, managed via Sequelize ORM for structured data handling.

- Security: Authentication and authorization are offloaded to Auth0, ensuring secure, token-based access control.

The development followed an iterative approach, building functional components progressively to allow for testing and feedback throughout the lifecycle.

1.7. Contributions of the Project

THESISBOARD offers a practical, unified solution to the fragmentation currently plaguing the faculty. By replacing scattered spreadsheets and email chains with a cohesive platform, the system improves transparency and drastically reduces the manual workload for staff.

More than just an administrative tool, the project lays a digital foundation for the department, capable of being expanded for future academic management needs. Ultimately, it allows students and lecturers to spend less time on logistics and more time on actual research.

CHAPTER II

LITERATURE REVIEW

2.1. Overview of Related Work

The primary objective of this chapter is to evaluate contemporary solutions and technological frameworks pertinent to the development of academic administration systems. Prior to the architectural design of a novel platform such as THESISBOARD, a critical assessment of incumbent systems is necessary to identify their operational efficacy, constraints, and the specific functional voids they fail to address. The scope of this review encompasses two principal domains: first, an investigation into prevailing Learning Management Systems (LMS) and academic portals utilized by educational establishments; second, an exploration of fundamental web technologies, including security protocols like Role-Based Access Control, that underpin modern educational software. This analysis establishes the theoretical and practical justification for the proposed solution. Numerous digital platforms have been established to facilitate course administration and the coordination between faculty and students. This section scrutinizes the most prevalent commercial and open-source systems, alongside the manual methodologies currently employed within the institution.

2.2. Existing Academic Management Systems

Numerous digital platforms have been established to facilitate course administration and the coordination between faculty and students. This section scrutinizes the most prevalent commercial and open-source systems, alongside the manual methodologies currently employed within the institution.

2.2.1. Moodle (Modular Object-Oriented Dynamic Learning Environment)

Moodle stands as a globally adopted, open-source Learning Management System (LMS) [1]. Renowned for its flexibility, Moodle offers an array of tools for curriculum design, online assessment, discussion forums, and digital submission handling. Its primary advantages lie in its open-source nature, allowing for cost-free deployment, and its extensibility through a vast repository of plugins that cater to diverse user roles, including students, instructors, and administrators. However, despite this adaptability, the platform is frequently cited for its steep learning curve and configuration complexity [1]. More critically, Moodle lacks intrinsic workflows tailored to the specific nuances of thesis management, such as the automated pairing of students with supervisors or the coordination of multi-stage defense schedules. Adapting Moodle to accommodate these specialized departmental rules often necessitates substantial customization or reliance on third-party extensions that may not align perfectly with faculty requirements

2.2.2. Blackboard

In contrast to open-source solutions, Blackboard represents a commercial, enterprise-grade platform, currently employed by International University for broad academic management [2]. It provides a sophisticated interface with powerful tools for grading, data analytics, and institutional communication. Its strengths are rooted in its stability, scalability, and robust security architecture, making it a reliable choice for standard course delivery. Nevertheless, these benefits come with substantial licensing expenditures [2]. Furthermore, its closed-source architecture restricts the ability of individual departments to modify system logic to fit unique requirements. Like Moodle, Blackboard is optimized for general pedagogical activities such as lectures and examinations rather than the distinct, iterative lifecycle of a thesis project.

2.2.3. Manual Tools

Within the School of Computer Science and Engineering, the prevailing workflow largely depends on manual instruments, specifically Google Sheets for registration tasks and email for correspondence. While these tools offer the advantages of ubiquity, zero cost, and immediate familiarity to users, they suffer from significant functional deficits. Specifically, they lack data integrity mechanisms, automation capabilities, and centralized security. The absence of a maintained "system state" renders the process vulnerable to inconsistencies, such as students registering for supervisors who have already reached capacity. Additionally, the reliance on manual data entry for tracking progress creates a workflow that is inherently susceptible to human error. To mitigate the deficiencies of legacy systems, modern web technologies offer specific architectural advantages for constructing a bespoke solution

2.3. Related Technologies and Frameworks

To mitigate the deficiencies of legacy systems, modern web technologies offer specific architectural advantages for constructing a bespoke solution

2.3.1. Web-based Systems in Education

Contemporary academic tools are increasingly migrating from desktop-bound software to web-based architecture [14]. This transition guarantees accessibility across a multitude of devices and operating systems without the prerequisite of local installation. For the proposed platform, a web-based approach ensures that stakeholders can interact with the system remotely, thereby facilitating real-time data synchronization.

2.3.2. Role-Based Access Control (RBAC) in Academic Systems

Given the hierarchical nature of academic administration, Role-Based Access Control (RBAC) serves as a critical security paradigm. RBAC restricts system access based on the specific privileges associated with a user's function [3]. In the context of thesis management, this ensures that a student is limited to viewing their own academic progress, whereas a lecturer can access data pertaining to their assignees, and administrators retain oversight of the entire system. Strict implementation of RBAC is imperative for maintaining data privacy and workflow integrity.

2.3.3. Workflow Automation and Document Management

Unlike static file storage, the management of academic documentation requires dynamic versioning and status tracking distinguishing between states such as "Submitted," "Under Review," and "Approved." Technologies that support status-driven workflows are essential for superseding the disjointed email threads that currently dominate the process.

2.3.4. Authentication and Authorization (Token-based)

Secure digital environments necessitate robust identity management protocols. Token-based authentication, utilizing standards such as JSON Web Tokens (JWT) or services like Auth0, enables stateless authentication [4]. This method is both scalable and secure, ensuring that users maintain valid sessions while navigating the application, a fundamental requirement for modern Single Page Applications (SPAs).

2.4. Comparative Analysis

The following table summarizes the comparison between existing general solutions and the proposed THESISBOARD system.

Feature	Moodle	Blackboard	Google Sheets	THESISBOARD
Primary Focus	General Learning	General Learning	Manual Record Keeping	Thesis Lifecycle Management
Supervisor Matching	Plugin Dependent	Non-Native	Manual/Error-prone	Automated/Native
Customization	High (Complex)	Low (Closed)	High (Uncontrolled)	High (Tailored)
Cost	Low (Hosting only)	High (Licensing)	Low	Low (Development)
User Experience	Generic	Corporate	Fragmented	Streamlined

Table 1. General Solutions Comparison

While Moodle and Blackboard offer extensive utility for standard academic activities like quizzing and content dissemination, they remain general-purpose instruments. They do not intrinsically support the bespoke logic required by the CSE department, such as real-time validation of supervisor quotas or the orchestration of defense committee slots. Conversely, the existing manual framework offers flexibility but lacks the requisite validation and scalability for a growing student body. The proposed system creates a synthesis of these approaches, offering the automation typical of an LMS but integrated with specific business logic tailored to the thesis process.

2.5. Summary and Research Gap

2.5.1. Key Findings

The literature review suggests that although robust educational platforms are readily available, their primary design focus is course delivery rather than the management of thesis lifecycles. Enterprise solutions are often characterized by rigidity and high costs, whereas open-source alternatives demand excessive modification to handle specific workflows like supervisor allocation. Simultaneously, manual methods have proven unsustainable due to scalability limits and error propensity.

2.5.2. The Research Gap

Consequently, a distinct research gap exists for a specialized, lightweight, and process-centric application specifically engineered for the School of Computer Science and Engineering. Existing commercial tools are overly broad and generic, while manual tools are insufficient and prone to risk.

2.5.3. Addressing the Gap

THESISBOARD aims to bridge this divide by deploying a custom web-based solution utilizing modern frameworks (React and Node.js). By incorporating specific logics such as quota management and role-specific dashboards that generic platforms lack, the proposed system ensures functional alignment with the operational requirements of the faculty.

CHAPTER III

METHODOLOGY

3.1. System Overview

The developed platform functions as a comprehensive web-based solution designed to streamline the oversight and organization of pre-thesis and thesis lifecycles. The primary objective of the system is to replace fragmented manual workflows with a centralized digital environment. The application is designed to serve four distinct user groups Students, Teachers, Moderators, and Administrators by providing tailored interfaces that address the specific operational needs of the School of Computer Science and Engineering. Through this centralization, the system aims to enhance transparency, ensure data integrity, and facilitate efficient communication between faculty and candidates.

3.2. Development Technologies

The architectural foundation of the THESISBOARD platform relies on a curated selection of modern technologies, chosen to ensure scalability, type safety, and efficient performance. This section details the specific programming languages, frameworks, database systems, and external services utilized throughout the implementation phase.

3.2.1. Programming Language: TypeScript

To guarantee code reliability and facilitate long-term maintenance, the project employs TypeScript as the primary programming language [15]. Functioning as a statically typed superset of JavaScript, TypeScript enables the detection of syntax and logic errors during the compilation stage rather than at runtime. This characteristic is particularly advantageous for managing the complex business logic required in academic administration, as it enforces strict data structures and clear interfaces between different system components.

3.2.2. Frameworks and Libraries

The application development is supported by a suite of robust libraries and frameworks divided into frontend and backend categories. The client-side interface is constructed using ReactJS, which utilizes component-based architecture to render dynamic and interactive user interfaces. To accelerate the design process and ensure visual consistency, the system integrates AntDesign for a comprehensive set of pre-built UI components [16], while TailwindCSS is employed for utility-first styling, enabling responsive layouts that adapt to various screen sizes [17].

On the server side, the runtime environment is provided by Node.js, selected for its non-blocking I/O model which efficiently handles concurrent network requests. The application logic is organized using the Express.js framework, providing a streamlined structure for routing and middleware integration. Additionally, EJS is utilized as a templating engine to facilitate server-side rendering where necessary [18]. Data interaction is abstracted through Sequelize, an Object-Relational Mapper (ORM) that permits the manipulation of database records using TypeScript objects, thereby simplifying query construction and relationship management.

3.2.3. Database Management System

Data persistence and structural integrity are managed through MySQL. As a relational database management system, MySQL was selected for its proven stability and capacity to handle complex relationships between entities such as students, supervisors, and thesis topics. The system relies on this database to enforce referential integrity and store critical academic records, ensuring that all transaction data regarding registrations and grading remains consistent and secure.

3.2.4. External Service: Auth0

To adhere to rigorous security standards regarding user access, the platform incorporates Auth0 for identity management. Rather than implementing a custom authentication solution, the system delegates the responsibility of verifying user credentials and managing sessions to this external service. This integration ensures the robust application of authentication protocols and facilitates role-based access control, strictly limiting system privileges based on whether the user is an administrator, moderator, teacher, or student.

3.3. System Requirements

3.3.1. User Roles and Actors

The system enforces strict access control mechanisms to ensure that functionalities are accessible only to authorized personnel. The Student role is designed for candidates undertaking their thesis. These users possess permissions to browse available research topics, submit registration applications, upload deliverables such as final reports, and view evaluation results. The Teacher role encompasses faculty members responsible for supervision and assessment. Permissions for this group include the creation and modification of research topics, the validation of student applications, and the submission of grades and qualitative feedback. The Moderator role is tasked with operational oversight, possessing the authority to configure academic deadlines, assign reviewers to specific theses, and monitor the overall progression of topic applications. The Administrator role holds the highest level of privilege, maintaining full control over user account and system-wide management.

3.3.2. Functional Requirements

The functional capabilities of the system are categorized by user role to ensure operational clarity. For Students, the system necessitates a customized dashboard upon login, displaying relevant notifications and project status. Core functions include a topic registration module for viewing supervisor availability, a document submission interface for uploading version-controlled reports, and a grade viewing portal. For Lecturers, the system requires tools for topic lifecycle management, allowing for the definition of titles, descriptions, and quotas. Furthermore, it includes an applicant approval interface for accepting or rejecting student requests and a grading module for assessing submissions in capacities ranging from supervisor to committee member. For Moderators and Administrators, the system demands configuration tools to set critical timeline phases (semester, thesis defense section). Additionally, these roles require functions for assigning committee members based on expertise and generating administrative reports regarding completion rates and grading metrics.

3.3.3. Non-Functional Requirements

To ensure a robust user experience, several non-functional requirements were prioritized. Performance is addressed through asynchronous data handling to ensure rapid response times during file uploads and data retrieval. Security is established via JSON Web Token (JWT) authentication and role-based middleware to prevent unauthorized access. Scalability is supported by modular backend architecture, allowing for future expansion of the user base. Usability is achieved through a responsive interface design that adapts to various device screens. Maintainability is ensured by adhering to a structured codebase that separates controllers, models, and routes.

3.4. System Analysis

The system analysis phase employs Use Case modeling to delineate the functional requirements and behavioral specifications of the THESISBOARD platform. This methodology identifies the interactions between external entities, referred to as actors, and the system itself, thereby establishing a clear scope of functionality. The modeling process is divided into visual representation through diagrams, detailed actor analysis, and a comprehensive breakdown of specific use cases and their interrelationships.

3.4.1. Use Case Diagram

The Use Case Diagram serves as a high-level graphical representation of the system's functionality. It illustrates the boundaries of the application and the interactions between the primary actors—Students, Teachers, Moderators, and Administrators—and the specific processes they initiate within the Pre-Thesis and Thesis modules. The diagram categorizes functionalities into distinct domains, including project registration, academic evaluation, and system administration.

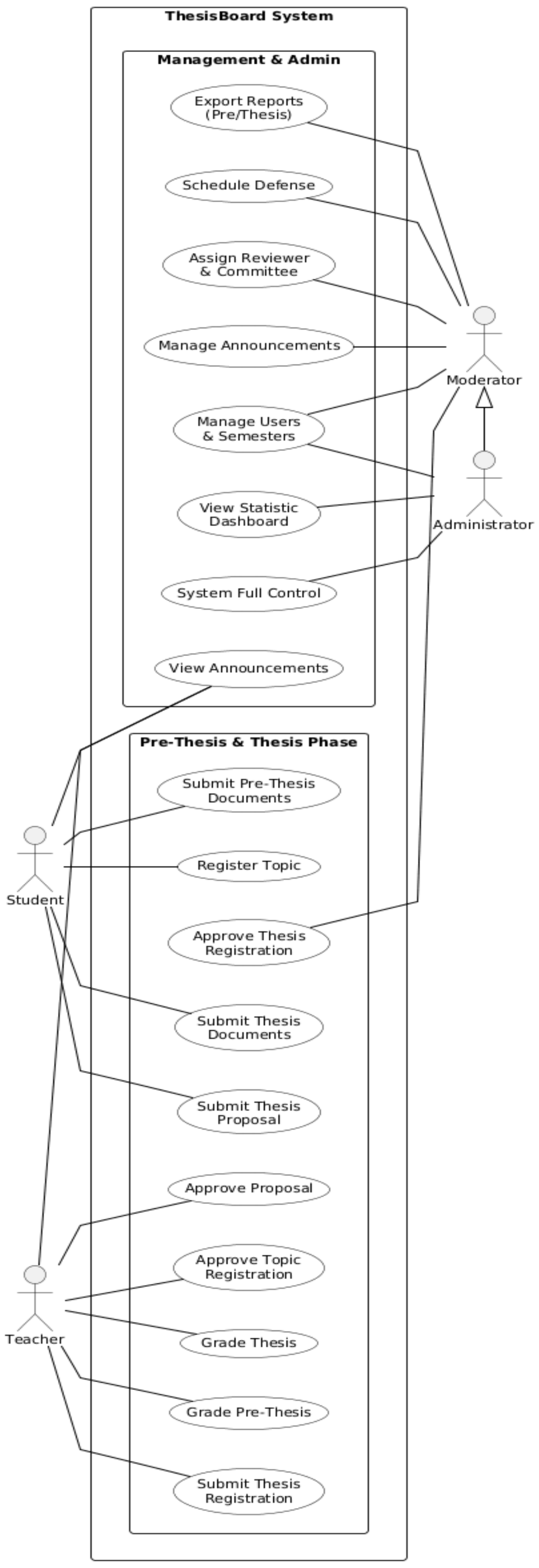


Figure 1. Use Case Diagram of THESISBOARD

The following table defines the distinct roles within the system and their respective responsibilities. Each actor represents a specific class of users with defined privileges and interaction patterns.

Actor	**Description**
Administrator	The Administrator possesses the highest level of privilege within the system. This actor is responsible for the overall health and maintenance of the platform, including access to a comprehensive statistical dashboard. They retain full control over all system configurations and management functions, superseding the permissions of other roles to ensure operational continuity.
Moderator	The Moderator acts as the primary academic staff member managing the operational workflow of the faculty. Their responsibilities include the configuration of semester timelines, user management, and the orchestration of the thesis lifecycle. They are tasked with approving registrations, assigning review committees, scheduling defense sessions, and exporting official academic reports.
Teacher	The Teacher actor interacts with the system in multiple capacities, primarily as a supervisor, reviewer, or committee member. In the pre-thesis phase, they manage topic quotas and assess student projects. During the thesis phase, they are responsible for validating student proposals, submitting registration forms on behalf of students, and providing evaluation grades based on their specific role in the defense council.

Actor	Description
Student	The Student actor represents the undergraduate candidate. Their interactions are centered on the submission of deliverables and the tracking of academic progress. They are authorized to register for pre-thesis topics, submit thesis proposals to supervisors, and upload required project documentation for grading. They also consume system-wide announcements and notifications.

Table 2. Actor Descriptions and Responsibilities

3.4.2. Use Case Analysis

This section details the specific workflows associated with the system's core functionalities. The analysis separates the operational logic into pre-thesis and thesis activities, alongside general system management tasks.

Use Case	Description
Register Topic (Pre-Thesis)	The process wherein a Student views available research topics provided by supervisors and selects a preferred option. This action triggers a validation check against available quotas and initiates a request for supervisor approval.

Use Case	**Description**
Approve Topic Registration	An action performed by the Teacher (Supervisor) to accept a student's request to join a specific research topic. This finalizes the student-supervisor pairing for the pre-thesis semester.
Submit Documents	The functionality allowing Students to upload required deliverables, such as interim reports or final project files, to the centralized system repository for archival and grading purposes.
Grade Project	The evaluation process where the Teacher assigns a numerical score to a submitted project. In the thesis phase, this extends to supervisors, reviewers, and committee members, each contributing to the final assessment based on their specific role.
Submit Thesis Proposal	The initial step in the thesis phase where a Student submits a research abstract and title to a prospective supervisor for review. This occurs prior to the official registration.
Submit Thesis Registration	A formal action taken by the Teacher to officially register a student's thesis topic with the faculty. This form includes the confirmed topic details and must be submitted to the Moderator for final validation.
Approve Thesis Registration	The administrative validation performed by the Moderator to confirm that a thesis registration meets all faculty requirements. Once approved, the thesis moves to the active development phase.

Use Case	Description
Assign Committee & Reviewer	The administrative process where the Moderator allocates specific Teachers to serve as independent reviewers or members of the defense council for a particular thesis project.
Export Result Report	The generation of a formalized document by the Moderator that aggregates grading data and project outcomes for either pre-thesis or thesis projects, utilized for faculty record-keeping.
Manage System Configuration	The set of administrative actions performed by the Moderator or Administrator to define semester dates, manage user accounts, and post official announcements to the system dashboard.

Table 3. Detailed Use Case Descriptions

3.4.3. Relationship Analysis

The relationship analysis clarifies the associations between the identified actors and the specific use cases they execute. This mapping ensures that the system's access control logic aligns with the academic responsibilities of the faculty members and students.

Relationship Context	Involved Actors	Functional Meaning
Topic Registration Association	Student, Teacher	A direct association where the Student initiates a topic selection, and the Teacher concludes the interaction by granting

Relationship Context	**Involved Actors**	**Functional Meaning**
		approval. This establishes the supervisory relationship for the pre-thesis phase.
Proposal & Registration Association	Student, Teacher, Moderator	A multi-stage workflow where the Student submits a proposal to the Teacher. The Teacher then acts as a proxy to submit the formal registration to the system, which is finally validated by the Moderator. This chain ensures a verified hierarchy of approval.
Evaluation Association	Teacher, Student	The relationship governing the grading process. The Teacher (acting as Supervisor, Reviewer, or Committee Member) interacts with the deliverables submitted by the Student to generate an academic score.
Administrative Oversight Association	Moderator, System	The Moderator is associated with high-level management functions such as assigning committees and scheduling defenses. These use cases do not directly involve the student but significantly impact the student's project timeline.

Relationship Context	**Involved Actors**	**Functional Meaning**
System Management Association	Administrator, Moderator	The Administrator and Moderator are associated with global system settings. While the Moderator focuses on academic parameters (semesters, announcements), the Administrator maintains broad access to the statistical dashboard and total system control.

Table 4. Actor-Use Case Associations

3.5. System Design

3.5.1. System Architecture

The architectural design of THESISBOARD follows a standard Client-Server model utilizing the Model-View-Controller (MVC) pattern. This structural choice ensures a clear separation of concerns, thereby enhancing modularity and facilitating future scalability [5]. The presentation layer (frontend) is constructed using React, which manages user interactions and dynamic interface rendering. The application logic layer (backend) is implemented using Node.js and Express.js, functioning as a RESTful API provider that processes client requests and executes business logic. The data persistence layer employs a MySQL for structured relational data.

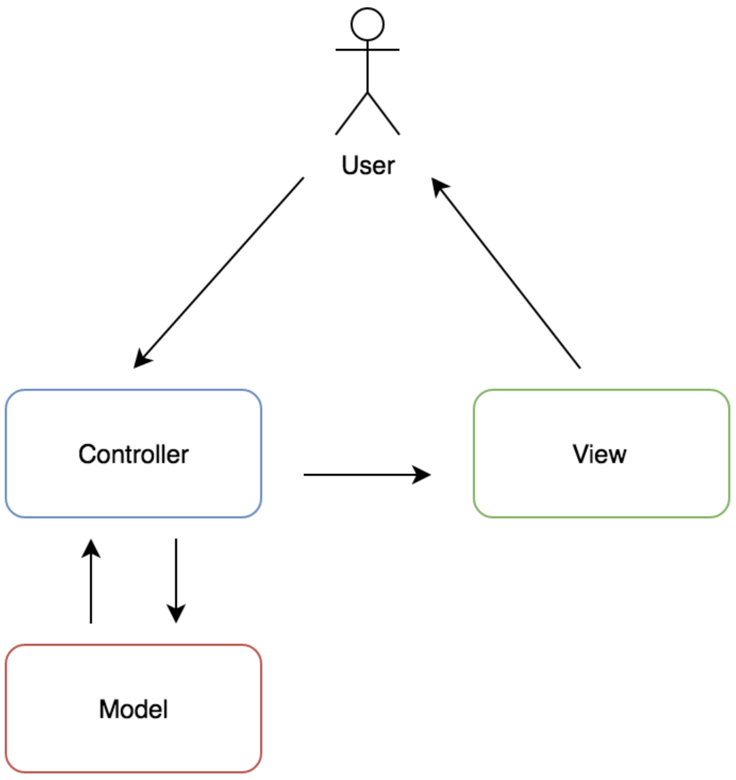


Figure 2. MVC Model

3.5.2. Class Diagram

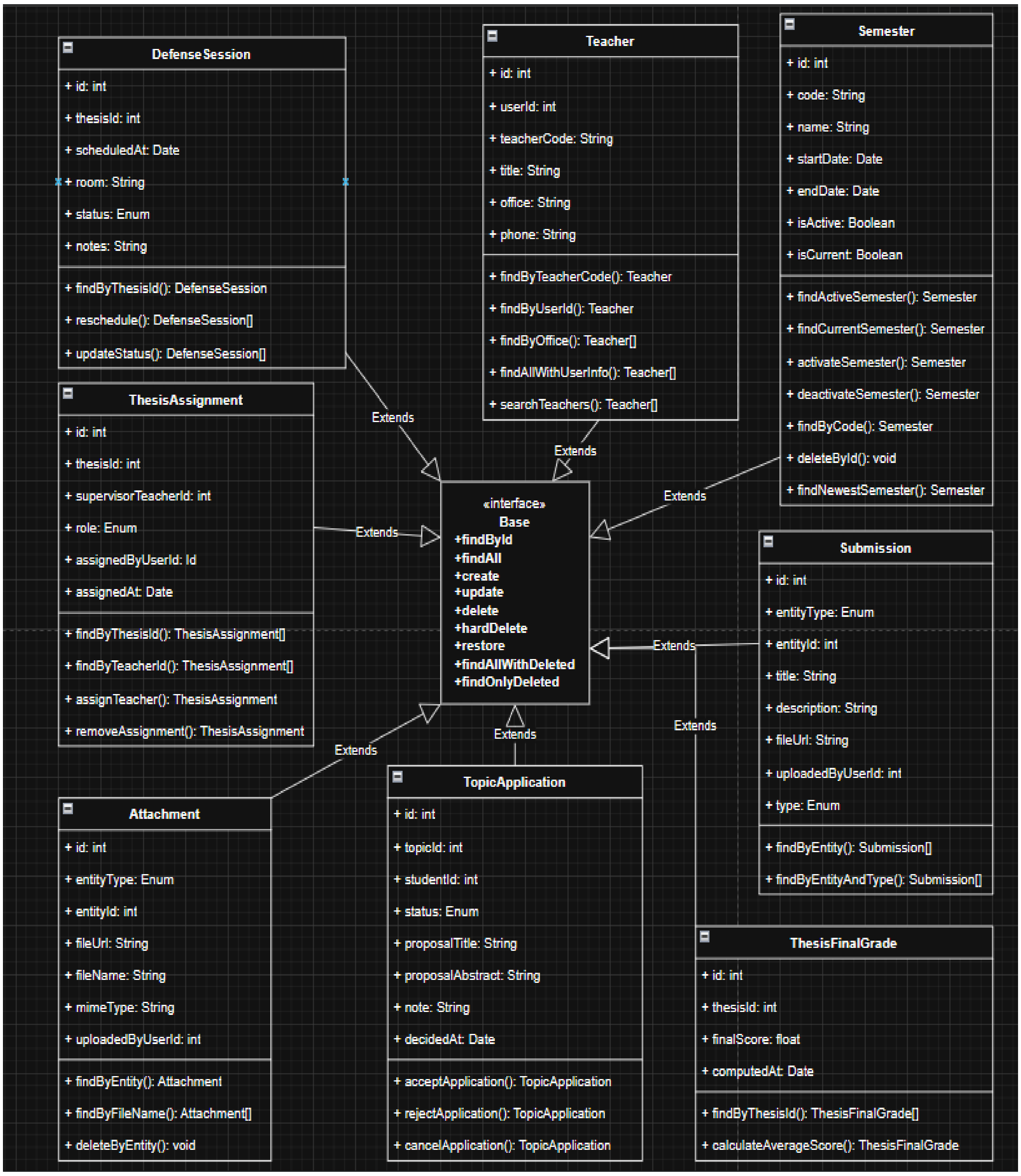


Figure 3. Class Diagram part 1

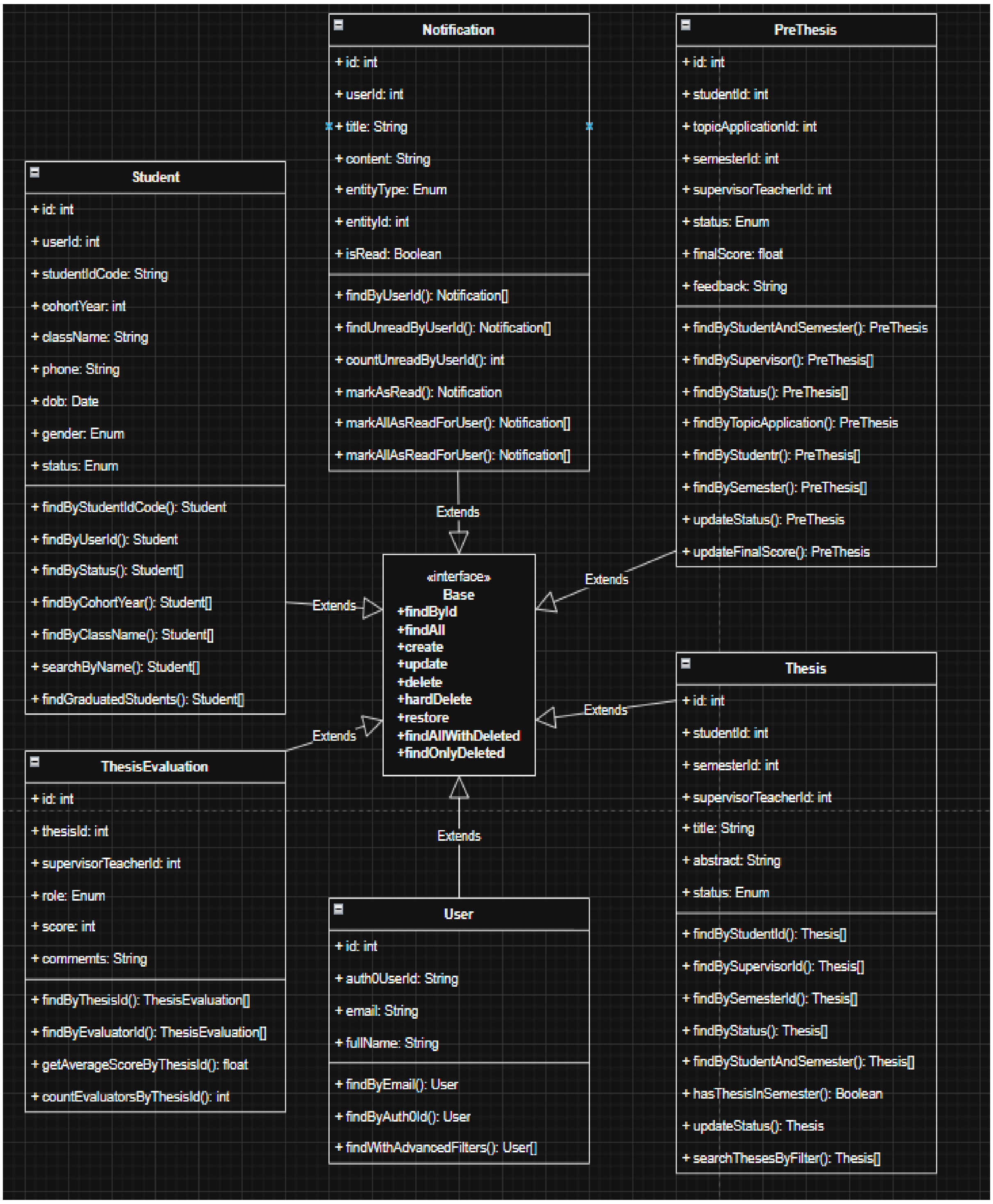

Notification
+ id: int
+ userId: int
+ title: String
+ content: String
+ entityType: Enum
+ entityId: int
+ isRead: Boolean
+ findByUserId(): Notification[]
+ findUnreadByUserId(): Notification[]
+ countUnreadByUserId(): int
+ markAsRead(): Notification
+ markAllAsReadForUser(): Notification[]
+ markAllAsReadForUser(): Notification[]
PreThesis
+ id: int
+ studentId: int
+ topicApplicationId: int
+ semesterId: int
+ supervisorTeacherId: int
+ status: Enum
+ finalScore: float
+ feedback: String
+ findByStudentAndSemester(): PreThesis
+ findBySupervisor(): PreThesis[]
+ findByStatus(): PreThesis[]
+ findByTopicApplication(): PreThesis
+ findByStudentr(): PreThesis[]
+ findBySemester(): PreThesis[]
+ updateStatus(): PreThesis
+ updateFinalScore(): PreThesis
Student
+ id: int
+ userId: int
+ studentIdCode: String
+ cohortYear: int
+ className: String
+ phone: String
+ dob: Date
+ gender: Enum
+ status: Enum
+ findByStudentIdCode(): Student
+ findByUserId(): Student
+ findByStatus(): Student[]
+ findByCohortYear(): Student[]
+ findByClassName(): Student[]
+ searchByName(): Student[]
+ findGraduatedStudents(): Student[]
Extends
Extends
Extends
Extends
Extends
Extends
«interface»
Base
+findById
+findAll
+create
+update
+delete
+hardDelete
+restore
+findAllWithDeleted
+findOnlyDeleted
Thesis
+ id: int
+ studentId: int
+ semesterId: int
+ supervisorTeacherId: int
+ title: String
+ abstract: String
+ status: Enum
+ findByStudentId(): Thesis[]
+ findBySupervisorId(): Thesis[]
+ findBySemesterId(): Thesis[]
+ findByStatus(): Thesis[]
+ findByStudentAndSemester(): Thesis[]
+ hasThesisInSemester(): Boolean
+ updateStatus(): Thesis
+ searchThesesByFilter(): Thesis[]
ThesisEvaluation
+ id: int
+ thesisId: int
+ supervisorTeacherId: int
+ role: Enum
+ score: int
+ commemts: String
+ findByThesisId(): ThesisEvaluation[]
+ findByEvaluatorId(): ThesisEvaluation[]
+ getAverageScoreByThesisId(): float
+ countEvaluatorsByThesisId(): int
User
+ id: int
+ auth0UserId: String
+ email: String
+ fullName: String
+ findByEmail(): User
+ findByAuth0Id(): User
+ findWithAdvancedFilters(): User[]

Figure 4. Class Diagram part 2

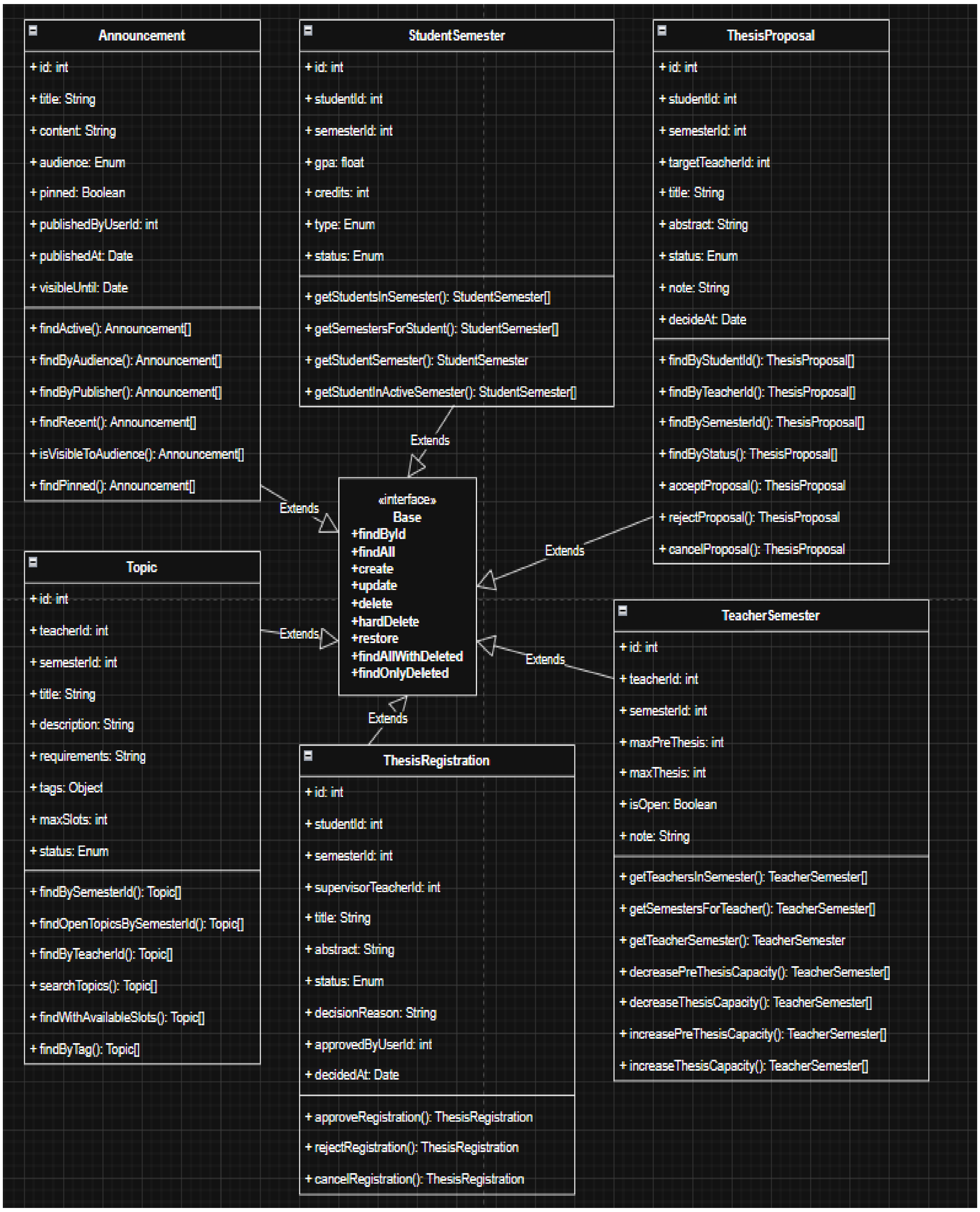

Announcement
+ id: int
+ title: String
+ content: String
+ audience: Enum
+ pinned: Boolean
+ publishedByUserId: int
+ publishedAt: Date
+ visibleUntil: Date
+ findActive(): Announcement[]
+ findByAudience(): Announcement[]
+ findByPublisher(): Announcement[]
+ findRecent(): Announcement[]
+ isVisibleToAudience(): Announcement[]
+ findPinned(): Announcement[]
StudentSemester
+ id: int
+ studentId: int
+ semesterId: int
+ gpa: float
+ credits: int
+ type: Enum
+ status: Enum
+ getStudentsInSemester(): StudentSemester[]
+ getSemestersForStudent(): StudentSemester[]
+ getStudentSemester(): StudentSemester
+ getStudentInActiveSemester(): StudentSemester[]
ThesisProposal
+ id: int
+ studentId: int
+ semesterId: int
+ targetTeacherId: int
+ title: String
+ abstract: String
+ status: Enum
+ note: String
+ decideAt: Date
+ findByStudentId(): ThesisProposal[]
+ findByTeacherId(): ThesisProposal[]
+ findBySemesterId(): ThesisProposal[]
+ findByStatus(): ThesisProposal[]
+ acceptProposal(): ThesisProposal
+ rejectProposal(): ThesisProposal
+ cancelProposal(): ThesisProposal
Extends
Extends
Extends
Extends
Extends
Extends
«interface»
Base
+findById
+findAll
+create
+update
+delete
+hardDelete
+restore
+findAllWithDeleted
+findOnlyDeleted
Topic
+ id: int
+ teacherId: int
+ semesterId: int
+ title: String
+ description: String
+ requirements: String
+ tags: Object
+ maxSlots: int
+ status: Enum
+ findBySemesterId(): Topic[]
+ findOpenTopicsBySemesterId(): Topic[]
+ findByTeacherId(): Topic[]
+ searchTopics(): Topic[]
+ findWithAvailableSlots(): Topic[]
+ findByTag(): Topic[]
TeacherSemester
+ id: int
+ teacherId: int
+ semesterId: int
+ maxPreThesis: int
+ maxThesis: int
+ isOpen: Boolean
+ note: String
+ getTeachersInSemester(): TeacherSemester[]
+ getSemestersForTeacher(): TeacherSemester[]
+ getTeacherSemester(): TeacherSemester
+ decreasePreThesisCapacity(): TeacherSemester[]
+ decreaseThesisCapacity(): TeacherSemester[]
+ increasePreThesisCapacity(): TeacherSemester[]
+ increaseThesisCapacity(): TeacherSemester[]
ThesisRegistration
+ id: int
+ studentId: int
+ semesterId: int
+ supervisorTeacherId: int
+ title: String
+ abstract: String
+ status: Enum
+ decisionReason: String
+ approvedByUserId: int
+ decidedAt: Date
+ approveRegistration(): ThesisRegistration
+ rejectRegistration(): ThesisRegistration
+ cancelRegistration(): ThesisRegistration

Figure 5. Class Diagram part 3

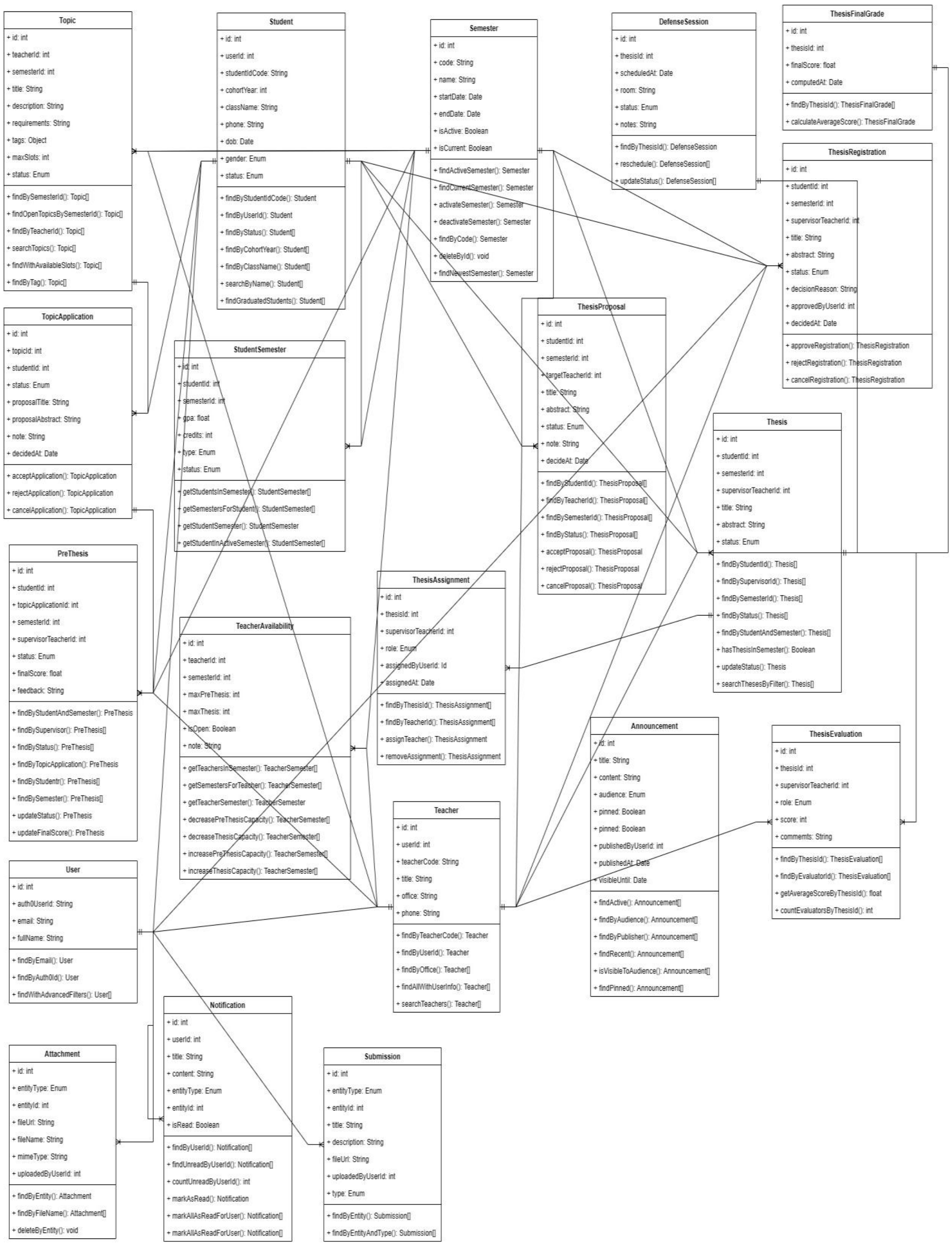

Topic
+ id: int
+ teacherId: int
+ semesterId: int
+ title: String
+ description: String
+ requirements: String
+ tags: Object
+ maxSlots: int
+ status: Enum
+ findBySemesterId(): Topic[]
+ findOpenTopicsBySemesterId(): Topic[]
+ findByTeacherId(): Topic[]
+ searchTopics(): Topic[]
+ findWithAvailableSlots(): Topic[]
+ findByTag(): Topic[]
Student
+ id: int
+ userId: int
+ studentIdCode: String
+ cohortYear: int
+ className: String
+ phone: String
+ dob: Date
+ gender: Enum
+ status: Enum
+ findByStudentIdCode(): Student
+ findByUserId(): Student
+ findByStatus(): Student[]
+ findByCohortYear(): Student[]
+ findByClassName(): Student[]
+ searchByName(): Student[]
+ findGraduatedStudents(): Student[]
Semester
+ id: int
+ code: String
+ name: String
+ startDate: Date
+ endDate: Date
+ isActive: Boolean
+ isCurrent: Boolean
+ findActiveSemester(): Semester
+ findCurrentSemester(): Semester
+ activateSemester(): Semester
+ deactivateSemester(): Semester
+ findByCode(): Semester
+ deleteById(): void
+ findNewestSemester(): Semester
DefenseSession
+ id: int
+ thesisId: int
+ scheduledAt: Date
+ room: String
+ status: Enum
+ notes: String
+ findByThesisId(): DefenseSession
+ reschedule(): DefenseSession[]
+ updateStatus(): DefenseSession[]
ThesisFinalGrade
+ id: int
+ thesisId: int
+ finalScore: float
+ computedAt: Date
+ findByThesisId(): ThesisFinalGrade[]
+ calculateAverageScore(): ThesisFinalGrade
ThesisRegistration
+ id: int
+ studentId: int
+ semesterId: int
+ supervisorTeacherId: int
+ title: String
+ abstract: String
+ status: Enum
+ decisionReason: String
+ approvedByUserId: int
+ decidedAt: Date
+ approveRegistration(): ThesisRegistration
+ rejectRegistration(): ThesisRegistration
+ cancelRegistration(): ThesisRegistration
TopicApplication
+ id: int
+ topicId: int
+ studentId: int
+ status: Enum
+ proposalTitle: String
+ proposalAbstract: String
+ note: String
+ decidedAt: Date
+ acceptApplication(): TopicApplication
+ rejectApplication(): TopicApplication
+ cancelApplication(): TopicApplication
StudentSemester
+ id: int
+ studentId: int
+ semesterId: int
+ gpa: float
+ credits: int
+ type: Enum
+ status: Enum
+ getStudentsInSemester(): StudentSemester[]
+ getSemestersForStudent(): StudentSemester[]
+ getStudentSemester(): StudentSemester
+ getStudentInActiveSemester(): StudentSemester[]
ThesisProposal
+ id: int
+ studentId: int
+ semesterId: int
+ targetTeacherId: int
+ title: String
+ abstract: String
+ status: Enum
+ note: String
+ decideAt: Date
+ findByStudentId(): ThesisProposal[]
+ findByTeacherId(): ThesisProposal[]
+ findBySemesterId(): ThesisProposal[]
+ findByStatus(): ThesisProposal[]
+ acceptProposal(): ThesisProposal
+ rejectProposal(): ThesisProposal
+ cancelProposal(): ThesisProposal
Thesis
+ id: int
+ studentId: int
+ semesterId: int
+ supervisorTeacherId: int
+ title: String
+ abstract: String
+ status: Enum
+ findByStudentId(): Thesis[]
+ findBySupervisorId(): Thesis[]
+ findBySemesterId(): Thesis[]
+ findByStatus(): Thesis[]
+ findByStudentAndSemester(): Thesis[]
+ hasThesisInSemester(): Boolean
+ updateStatus(): Thesis
+ searchThesesByFilter(): Thesis[]
PreThesis
+ id: int
+ studentId: int
+ topicApplicationId: int
+ semesterId: int
+ supervisorTeacherId: int
+ status: Enum
+ finalScore: float
+ feedback: String
+ findByStudentAndSemester(): PreThesis
+ findBySupervisor(): PreThesis[]
+ findByStatus(): PreThesis[]
+ findByTopicApplication(): PreThesis
+ findByStudentr(): PreThesis[]
+ findBySemester(): PreThesis[]
+ updateStatus(): PreThesis
+ updateFinalScore(): PreThesis
TeacherAvailability
+ id: int
+ teacherId: int
+ semesterId: int
+ maxPreThesis: int
+ maxThesis: int
+ isOpen: Boolean
+ note: String
+ getTeachersInSemester(): TeacherSemester[]
+ getSemestersForTeacher(): TeacherSemester[]
+ getTeacherSemester(): TeacherSemester
+ decreasePreThesisCapacity(): TeacherSemester[]
+ decreaseThesisCapacity(): TeacherSemester[]
+ increasePreThesisCapacity(): TeacherSemester[]
+ increaseThesisCapacity(): TeacherSemester[]
ThesisAssignment
+ id: int
+ thesisId: int
+ supervisorTeacherId: int
+ role: Enum
+ assignedByUserId: Id
+ assignedAt: Date
+ findByThesisId(): ThesisAssignment[]
+ findByTeacherId(): ThesisAssignment[]
+ assignTeacher(): ThesisAssignment
+ removeAssignment(): ThesisAssignment
Announcement
+ id: int
+ title: String
+ content: String
+ audience: Enum
+ pinned: Boolean
+ pinned: Boolean
+ publishedByUserId: int
+ publishedAt: Date
+ visibleUntil: Date
+ findActive(): Announcement[]
+ findByAudience(): Announcement[]
+ findByPublisher(): Announcement[]
+ findRecent(): Announcement[]
+ isVisibleToAudience(): Announcement[]
+ findPinned(): Announcement[]
ThesisEvaluation
+ id: int
+ thesisId: int
+ supervisorTeacherId: int
+ role: Enum
+ score: int
+ commemts: String
+ findByThesisId(): ThesisEvaluation[]
+ findByEvaluatorId(): ThesisEvaluation[]
+ getAverageScoreByThesisId(): float
+ countEvaluatorsByThesisId(): int
User
+ id: int
+ auth0UserId: String
+ email: String
+ fullName: String
+ findByEmail(): User
+ findByAuth0Id(): User
+ findWithAdvancedFilters(): User[]
Teacher
+ id: int
+ userId: int
+ teacherCode: String
+ title: String
+ office: String
+ phone: String
+ findByTeacherCode(): Teacher
+ findByUserId(): Teacher
+ findByOffice(): Teacher[]
+ findAllWithUserInfo(): Teacher[]
+ searchTeachers(): Teacher[]
Notification
+ id: int
+ userId: int
+ title: String
+ content: String
+ entityType: Enum
+ entityId: int
+ isRead: Boolean
+ findByUserId(): Notification[]
+ findUnreadByUserId(): Notification[]
+ countUnreadByUserId(): int
+ markAsRead(): Notification
+ markAllAsReadForUser(): Notification[]
+ markAllAsReadForUser(): Notification[]
Attachment
+ id: int
+ entityType: Enum
+ entityId: int
+ fileUrl: String
+ fileName: String
+ mimeType: String
+ uploadedByUserId: int
+ findByEntity(): Attachment
+ findByFileName(): Attachment[]
+ deleteByEntity(): void
Submission
+ id: int
+ entityType: Enum
+ entityId: int
+ title: String
+ description: String
+ fileUrl: String
+ uploadedByUserId: int
+ type: Enum
+ findByEntity(): Submission[]
+ findByEntityAndType(): Submission[]

Figure 6. Class Diagram part 4

The static structural view of the THESISBOARD platform is articulated through a comprehensive Class Diagram, which serves to map the system's data architecture into an object-oriented framework. This diagram delineates the internal structure of the system by defining the classes, their associated attributes, and the relationships that govern the interaction between distinct entities. In the context of this project, the class structure mirrors the database schema defined via the Sequelize Object-Relational Mapper (ORM), ensuring a direct correspondence between the application logic and the underlying relational data.

The system's architecture is centered around the User entity, which serves as the primary identity record linked to the external Auth0 authentication service. This base class branches into two specialized subclasses: Student and Teacher. This separation of concerns ensures that role-specific attributes—such as a student's cohort year or a teacher's office location—are isolated from the authentication credentials. The academic timeline is governed by the Semester class, which imposes temporal constraints on all project activities, while the StudentSemester and TeacherAvailability classes manage the dynamic status of users within a specific academic term, tracking metrics such as accumulated GPA or supervision quotas.

The core operational logic is encapsulated within the project lifecycle classes. The Pre-Thesis phase is managed through the Topic and TopicApplication classes, which facilitate the connection between students and supervisors. Subsequently, the Thesis phase involves a more complex hierarchy, progressing from a ThesisProposal to a formal ThesisRegistration, and culminating in the Thesis entity itself. The evaluation process is handled by distinct classes such as ThesisEvaluation for grading and DefenseSession for scheduling final presentations.

Supporting these core functions are utility classes like Notification, Announcement, and Attachment, which handle communication and file management across the platform.

The following table provides a detailed exposition of the system's key classes, their primary responsibilities, and their architectural relationships.

Class Name	**Description**	**Key Relationships**
User	Represents the fundamental account entity within the system. It stores authentication data derived from Auth0, including the unique Auth0 ID and email address.	1:1 association with **Student** and **Teacher**. 1:N association with **Notification** and **Announcement**.
Student	Encapsulates academic data specific to candidates, including the student ID code, class name, cohort year, and current academic status (e.g., active, graduated).	Belongs to **User**. 1:N association with **TopicApplication**, **PreThesis**, **ThesisProposal**, and **Thesis**.
Teacher	Represents faculty members participating in the system. It includes professional details such as the teacher code, academic title, office location, and contact information.	Belongs to **User**. 1:N association with **Topic**, **PreThesis** (as supervisor), **Thesis** (as supervisor/reviewer), and **DefenseSession**

Class Name	Description	Key Relationships
		(as committee member).
Semester	Defines the academic timeframes (e.g., Semester 1, 2025-2026). It acts as a temporal container for all academic activities, controlling when topics can be registered or defenses scheduled.	1:N association with **Topic**, **PreThesis**, **Thesis**, **StudentSemester**, and **TeacherAvailability**.
Topic	Represents research subjects proposed by supervisors. It contains descriptive data, requirements, tags, and the maximum number of student slots available (quota).	Belongs to **Teacher** and **Semester**. 1:N association with **TopicApplication**.
TopicApplication	Manages the request workflow where a student applies for a specific topic. It tracks the status of the application (pending, accepted, rejected) and supervisor notes.	Belongs to **Student** and **Topic**. 1:1 association with **PreThesis** (upon acceptance).
PreThesis	Represents the active pre-thesis project for a student. It stores the final score, supervisor feedback, and the operational status of the project.	Belongs to **Student**, **Semester**, and **Teacher** (Supervisor).

Class Name	Description	Key Relationships
		Linked to **TopicApplication**.
ThesisProposal	Captures the initial research proposal submitted by a student to a prospective supervisor before official registration. It includes the proposed title and abstract.	Belongs to **Student** and **Teacher** (Target). Limits one active proposal per student per semester.
ThesisRegistration	Represents the formal registration form submitted by a teacher to the faculty. It acts as a request to officially enroll a student in a thesis, requiring moderator approval.	Belongs to **Student** and **Teacher** (Supervisor). Approves the creation of a **Thesis** record.
Thesis	The central entity for the final thesis project. It tracks the overall progress, from research to defense, and links to all subsequent evaluation data.	Belongs to **Student** and **Teacher**. 1:1 association with **DefenseSession** and **ThesisFinalGrade**. 1:N association with **ThesisAssignment**.
ThesisAssignment	Manages the assignment of additional faculty roles to a specific thesis, such	Belongs to **Thesis** and **Teacher**. Tracks the specific role

Class Name	Description	Key Relationships
	as Reviewers or Defense Committee members.	(reviewer/committee) assigned.
DefenseSession	Schedules the final defense event. It stores logistical details such as the date, time, and physical room location for the presentation.	1:1 association with **Thesis**.
ThesisEvaluation	Stores the individual assessment scores and qualitative comments provided by the supervisor, reviewer, and committee members.	Belongs to **Thesis** and **Teacher** (Evaluator).
Announcement	Manages system-wide or role-specific news broadcasts published by administrators or moderators.	Belongs to **User** (Publisher). Can be pinned or targeted to specific audiences.
Notification	Represents alerts generated for specific users regarding system events (e.g., "Topic Approved," "Grade Posted").	Belongs to **User**.
Attachment	A polymorphic entity used to manage file uploads. It links physical file	Belongs to **User** (Uploader). Polymorphic

Class Name	Description	Key Relationships
	URLs to various entities like submissions or announcements.	association with **Thesis**, **PreThesis**, etc.
TeacherAvailability	Tracks a teacher's workload capacity for a specific semester, defining how many pre-thesis or thesis students they can supervise.	Belongs to **Teacher** and **Semester**.

Table 5. Class Definitions and Relationships

3.5.3. Sequence Diagrams

The dynamic behavior of the ThesisBoard system is illustrated through sequence diagrams, which detail the temporal interactions between system actors and object instances. These diagrams explicate the message exchange protocols required to execute complex academic workflows, demonstrating how the architectural components—Controllers, Services, and the Database—collaborate to fulfill the functional requirements identified in the analysis phase.

A. Pre-Theis functionality

The pre-thesis workflow represents the initial phase of student research. The sequence commences with the Supervisor defining available research topics, establishing the constraints for student enrollment. Subsequently, the Student initiates an application process, which requires explicit validation from the Supervisor. Upon acceptance, the system automatically instantiates a PreThesis record, transitioning the workflow into the execution phase where deliverables are submitted and graded.

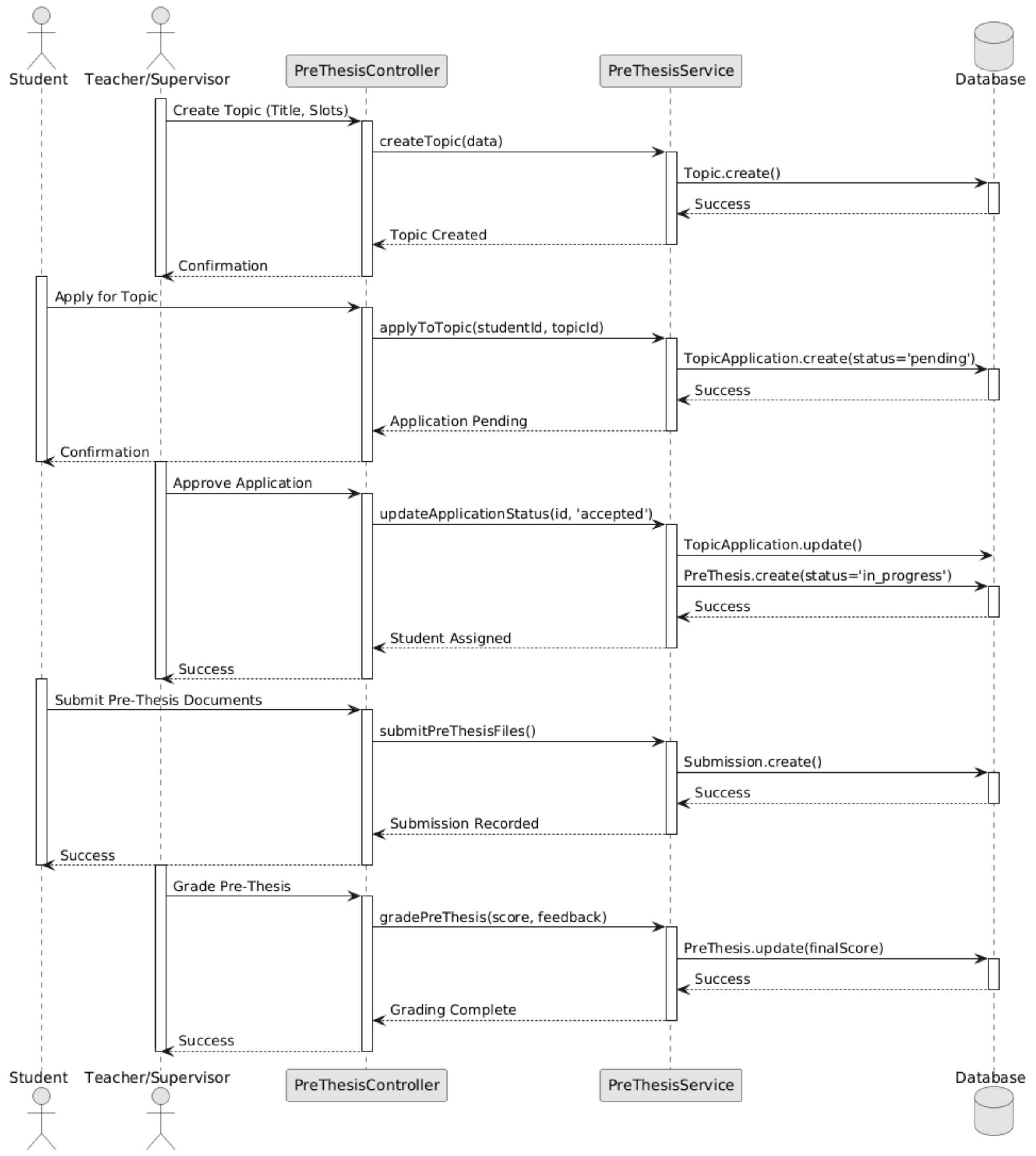


Figure 7. Pre-Thesis Lifecycle Sequence Diagram

B. Theis functionality

The thesis management process entails a rigorous, multi-stage workflow involving Students, Supervisors, Moderators, and the Defense Committee. This sequence begins with a student-initiated proposal, which must be formalized into a registration by the Supervisor and subsequently validated by the Moderator. The process advances through document submission, intermediate evaluation by the Supervisor and assigned Reviewers, and culminates in the organization of a Defense Session by the Moderator.

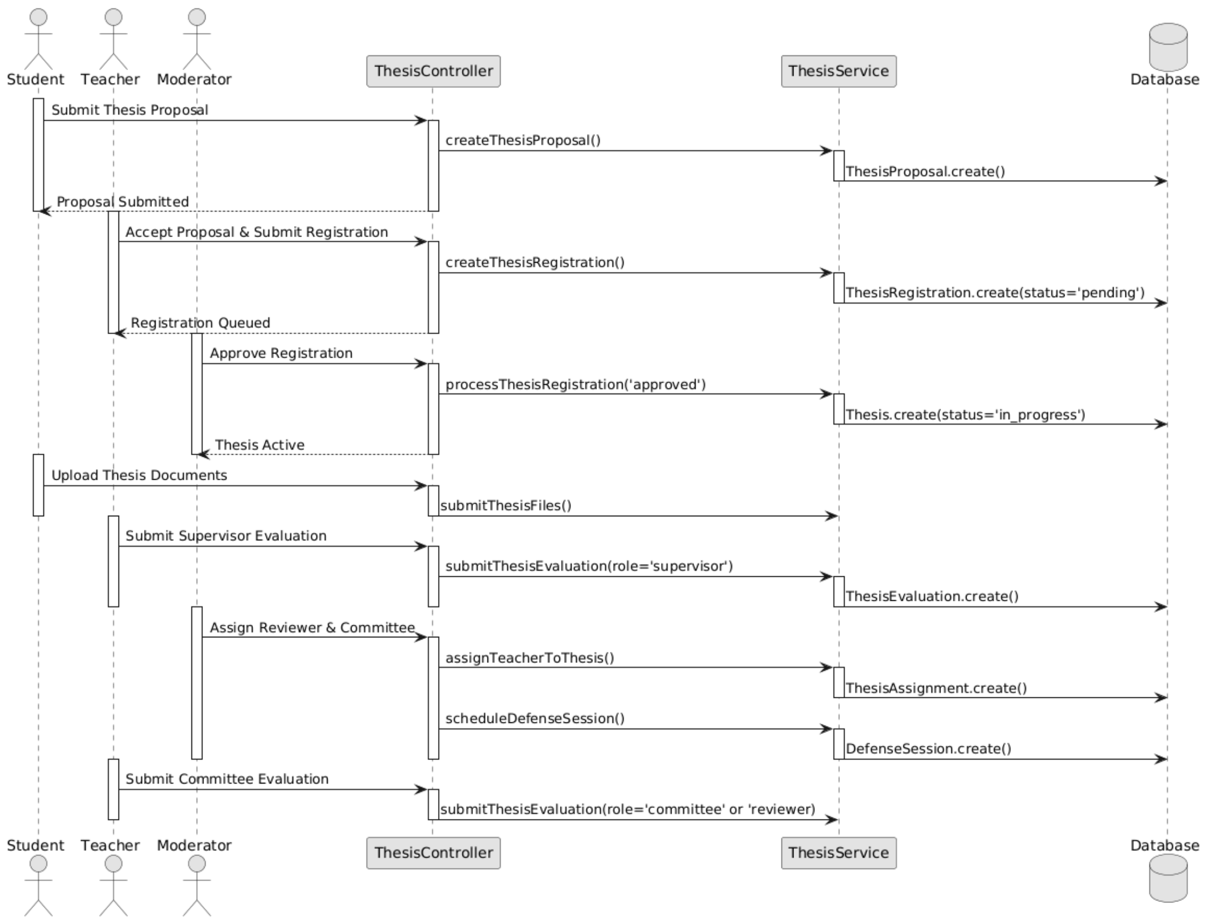

Student
Teacher
Moderator
ThesisController
ThesisService
Database
Submit Thesis Proposal
createThesisProposal()
ThesisProposal.create()
Proposal Submitted
Accept Proposal & Submit Registration
createThesisRegistration()
ThesisRegistration.create(status='pending')
Registration Queued
Approve Registration
processThesisRegistration('approved')
Thesis.create(status='in_progress')
Thesis Active
Upload Thesis Documents
submitThesisFiles()
Submit Supervisor Evaluation
submitThesisEvaluation(role='supervisor')
ThesisEvaluation.create()
Assign Reviewer & Committee
assignTeacherToThesis()
ThesisAssignment.create()
scheduleDefenseSession()
DefenseSession.create()
Submit Committee Evaluation
submitThesisEvaluation(role='committee' or 'reviewer')

Figure 8. Thesis Lifecycle Sequence Diagram

C. Announcement functionality

The announcement subsystem facilitates the dissemination of administrative information. The sequence demonstrates the "Publish-Subscribe" pattern implementation where the Moderator or Administrator broadcasts information, which is then persisted in the database. When users (Students or Teachers) access their dashboards, the client application polls the system to retrieve valid, non-expired announcements based on the user's role authorization.

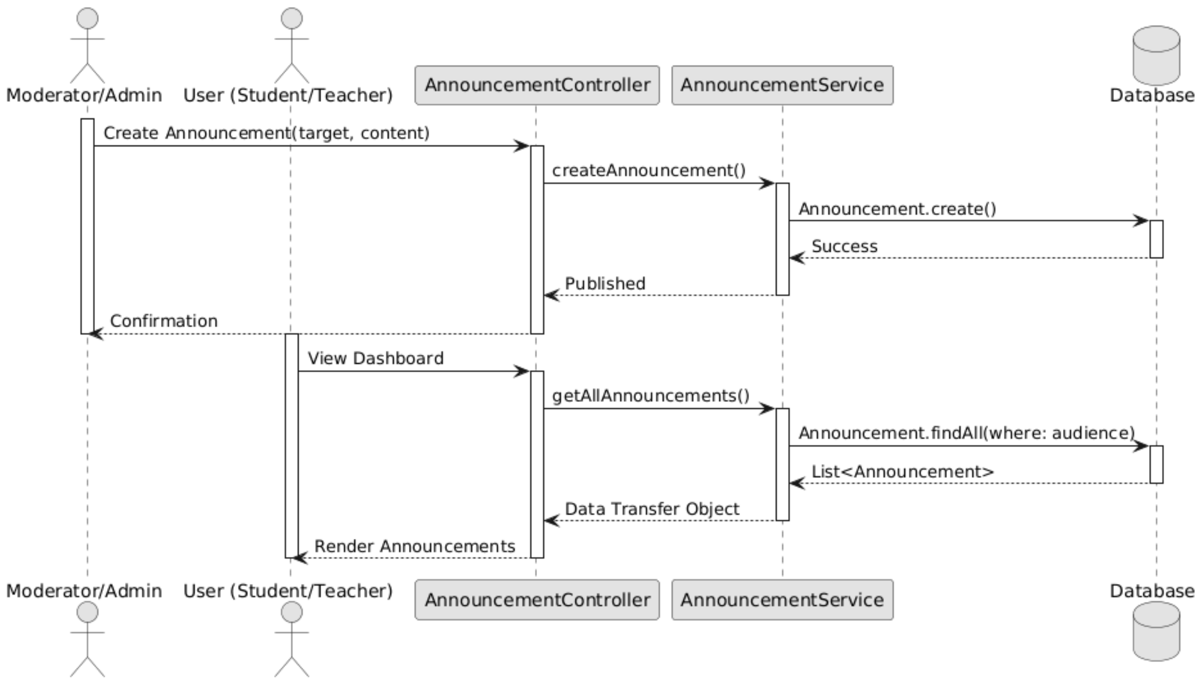


Figure 9. Announcement System Sequence Diagram

D. System management functionality

The administrative management sequence highlights the operational maintenance performed by the Moderator. This includes the configuration of academic semesters and the population of student data. The diagram underscores the strict dependency of academic activities on the existence of an active semester entity, ensuring that all subsequent registrations and topics are correctly associated with the current academic timeline.

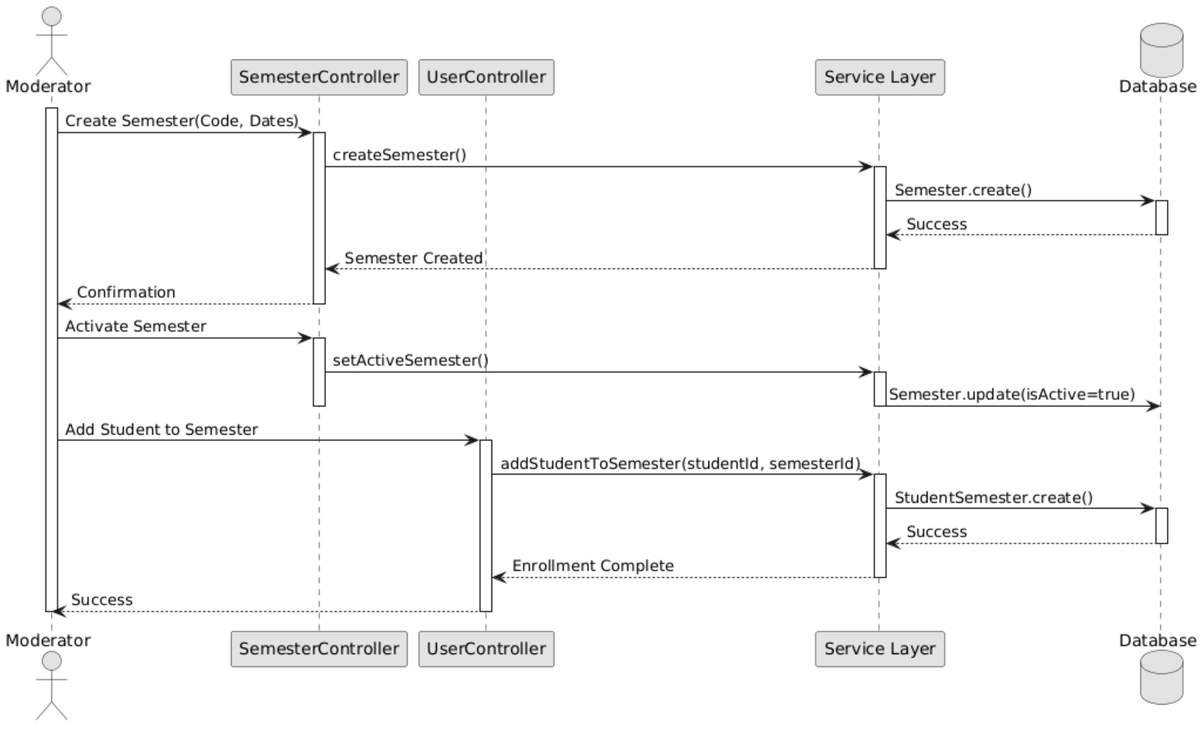


Figure 10. Semester Management System Sequence Diagram

3.6. Database Design

The database architecture for the THESISBOARD platform is constructed to ensure data integrity, scalability, and adherence to relational database principles. This section delineates the structural schema, the enforcement of rigorous data constraints, and the normalization process applied to optimize storage efficiency and minimize redundancy.

3.6.1. Relational Table Diagram

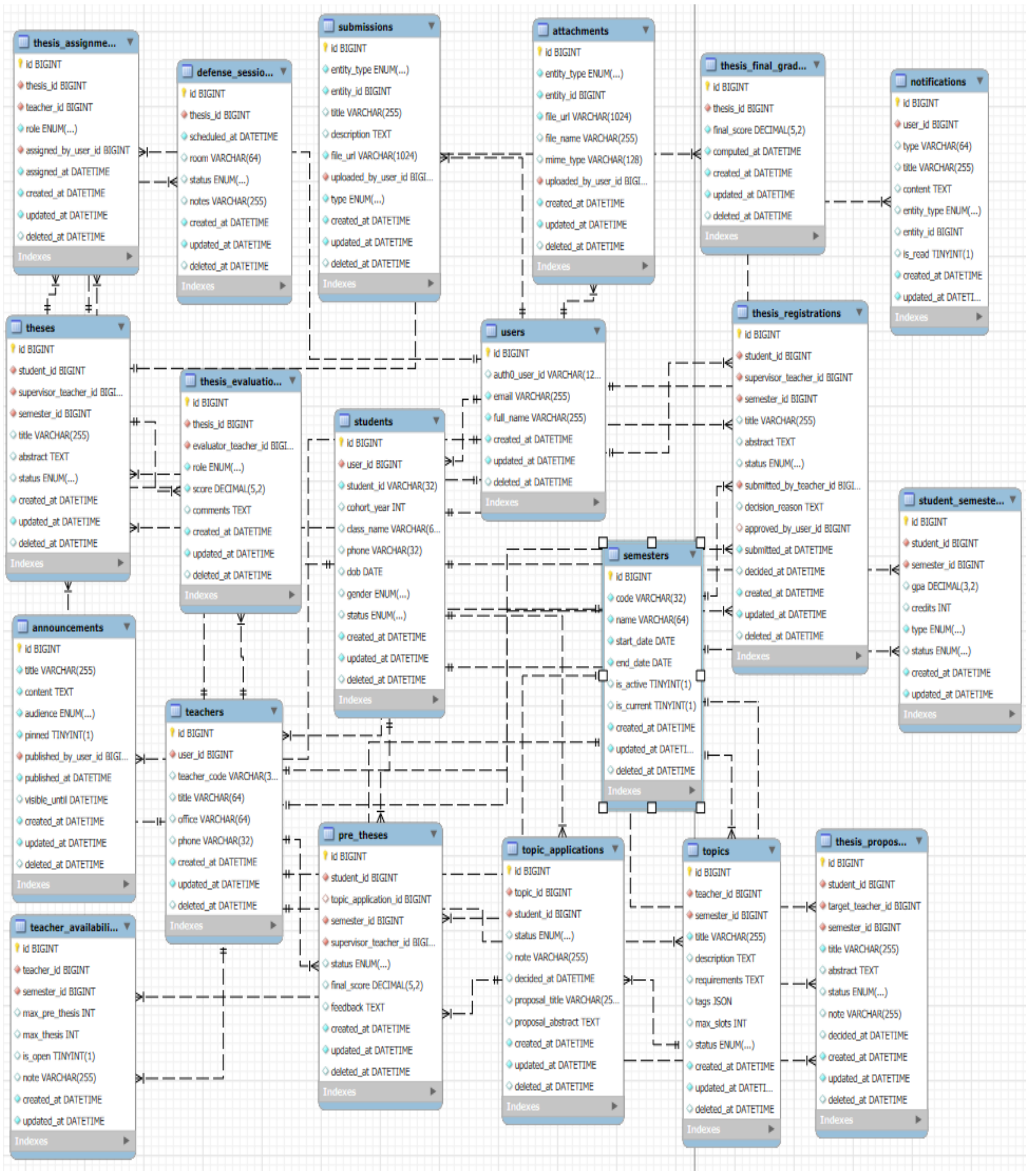


Figure 11. Relational Table Diagram

3.6.2. Data Constraints

To maintain the consistency and reliability of the information stored within the system, a comprehensive set of constraints is enforced at both the database and application levels. These constraints ensure that the data adheres to defined business rules and referential integrity standards.

Constraint Category	Description and Implementation
Primary Key Integrity	**Explanation:** This constraint guarantees that every record within a database table possesses a unique identifier, thereby preventing duplicate entries and ensuring that individual rows can be referenced distinctively. **Examples:** Every entity in the system, including *Users*, *Theses*, and *Announcements*, utilizes a surrogate key named id defined as a BIGINT with an auto-incrementing property. This serves as the unalterable identifier for all relational operations. **Conclusion:** The universal application of surrogate primary keys simplifies complex join operations and maintains entity uniqueness across the entire platform architecture.

Constraint Category	**Description and Implementation**
Referential Integrity	**Explanation:** Referential integrity ensures that relationships between tables remain consistent. It requires that a foreign key field in one table must agree with the primary key in the referenced table, preventing the existence of orphaned records. **Examples:** In the *PreThesis* table, the student_id and supervisor_teacher_id columns are defined as foreign keys referencing the *Students* and *Teachers* tables, respectively. Consequently, a pre-thesis project cannot be assigned to a non-existent student or supervisor. **Conclusion:** This constraint enforces the logical connections between entities, ensuring that academic activities are always correctly attributed to valid system users.
Entity Integrity (Unique constraints)	**Explanation:** Unique constraints are employed to prevent data duplication in columns that are not primary keys but require distinct values for business logic validity. **Examples:** The *User* table enforces a unique constraint on the email and auth0_user_id fields to prevent multiple accounts for a single

Constraint Category	Description and Implementation
	individual. Similarly, the *StudentSemester* table utilizes a composite unique index on [student_id, semester_id], ensuring a student is enrolled only once per academic term. **Conclusion:** These constraints are critical for maintaining the logical validity of the system, preventing scenarios such as double-registration or identity conflicts.
Domain Integrity (Check & Enum)	**Explanation:** Domain integrity restricts the values allowed in a column to a specific format, range, or set of predefined options, ensuring that the data remains within valid operational boundaries. **Examples:** The status field in the *Thesis* table is restricted to an ENUM type containing values such as 'in_progress', 'defense_scheduled', or 'completed'. Furthermore, the gpa field in *StudentSemester* is numerically constrained to a range between 0.00 and 4.00. **Conclusion:** By enforcing these specific data types and ranges, the system prevents the entry of erroneous or nonsensical data, such as a negative grade or an undefined project status.

Constraint Category	Description and Implementation
Business Logic Integrity (Hooks)	**Explanation:** Complex constraints that exceed standard SQL capabilities are implemented via Sequelize hooks (application-level triggers) to enforce dynamic business rules during data transaction events. **Examples:** The *ThesisRegistration* model employs a beforeCreate hook to verify that a student does not already possess an approved registration for the current semester. Additionally, an afterUpdate hook automatically cancels pending registrations if one is approved. **Conclusion:** These application-level constraints automate complex workflow logic, ensuring that the state of the database reflects the intricate academic regulations of the faculty.

Table 6. Data Constraints Analysis

3.6.3. Normal Form Analysis (3NF)

The database schema has been designed to adhere to the Third Normal Form (3NF) to reduce data redundancy and improve data integrity. The normalization process involves decomposing tables to ensure that non-key attributes are dependent solely on the primary key.

Normal Form	Condition and Analysis
First Normal Form (1NF)	**Condition:** A relation is in 1NF if all attributes contain atomic values, and there are no repeating groups or arrays within a single row. **Analysis:** All tables within the THESISBOARD schema, such as *Students*, *Teachers*, and *Theses*, define attributes that hold single, atomic values (e.g., strings, integers, dates). For instance, the *Student* table stores phone and class_name as distinct columns rather than grouped lists. While the *Topic* table contains a tags column stored as JSON, strictly speaking, this is an encapsulation; however, within the context of the relational design, the primary operational attributes remain atomic. There are no repeating groups of columns (e.g., topic1, topic2) within any entity. **Conclusion:** The schema successfully satisfies the requirements of the First Normal Form by ensuring atomicity and eliminating repeating groups.
Second Normal Form (2NF)	**Condition:** A relation is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the primary key, effectively eliminating partial dependencies.

Normal Form	**Condition and Analysis**
	Analysis: The THESISBOARD database utilizes a uniform design pattern where every table possesses a single-column primary key (id). Because there are no composite primary keys utilized for entity identification (composite unique keys exist as constraints but not as identifiers), it is impossible for a non-prime attribute to depend on only a part of the primary key. For example, in the *ThesisAssignment* table, attributes like role and assigned_at depend entirely on the unique id of the assignment record, not partially on the thesis_id or teacher_id. **Conclusion:** Since all entities are identified by a single surrogate key, partial dependencies are inherently impossible, ensuring compliance with the Second Normal Form.
Third Normal Form (3NF)	**Condition:** A relation is in 3NF if it is in 2NF and no non-prime attribute is transitively dependent on the primary key (i.e., no non-key attribute depends on another non-key attribute). **Analysis:** The schema is structured such that all non-key attributes rely directly on the primary key. In the *PreThesis* table, attributes such as final_score and feedback relate directly to the specific pre-thesis record

Normal Form	**Condition and Analysis**
	(id). There are no transitive dependencies; for instance, student information is not stored within the *PreThesis* table but is referenced via the student_id foreign key, and student details (like cohortYear) are stored solely in the *Student* table. Similarly, the *Department* or *Faculty* details (if they existed) would be stored in separate tables rather than repeated in the *Teacher* table. This separation ensures that an update to a student's information in the *Student* table is immediately reflected across all their associated records without data duplication. **Conclusion:** The database design effectively eliminates transitive dependencies, satisfying the criteria for the Third Normal Form and ensuring a robust, normalized structure.

Table 7. Normal Form Analysis

3.7. Summary

This chapter has provided a comprehensive exposition of the methodological framework and architectural design underpinning the THESISBOARD platform. The system analysis phase successfully identified the core actors—Administrators, Moderators, Teachers, and Students—and delineated their functional requirements through detailed use case modeling. These requirements were subsequently translated into a robust technical solution utilizing a modern technology stack, incorporating ReactJS for the user interface, Node.js and Express.js for server-side logic, and MySQL for relational data persistence.

Furthermore, the structural and behavioral aspects of the system were rigorously defined. The static view was articulated through class diagrams that map the object-oriented structure of the application, while the dynamic view was visualized using sequence diagrams to illustrate the complex message exchanges required for workflows such as topic registration and thesis defense scheduling. The database design phase established a normalized schema that enforces strict data integrity and efficient storage. Collectively, these design artifacts constitute a complete engineering blueprint, serving as the definitive guide for the implementation and deployment phases discussed in the subsequent chapters.

CHAPTER IV

IMPLEMENTATION

4.1. Core Features

4.1.1. User Management

The User Management module functions as the foundational pillar of the THESISBOARD system, orchestrating the interaction between the application and the Auth0 identity provider. This component handles the lifecycle of user identities, distinguishing between four primary roles: Students, Teachers, Moderators, and Administrators. Upon initial authentication, the system synchronizes the user's profile with the local database, ensuring that essential academic attributes—such as student cohort years or faculty office locations—are securely linked to their authentication credentials. The implementation enforces strict Role-Based Access Control (RBAC) at the API level, utilizing middleware to intercept requests and verify that the initiating user possesses the requisite privileges for specific actions, such as modifying user accounts or accessing sensitive academic records.

4.1.2. Project Registration

The Project Registration module addresses the critical requirement of pairing students with suitable research supervisors. This feature is realized through a transactional workflow where supervisors define research topics with specific metadata, including titles, descriptions, and maximum slot quotas. To prevent over-subscription, the backend implementation employs atomic database transactions that lock topic rows during the registration process, ensuring that the number of accepted applications never exceeds the predefined capacity. Students are provided with a searchable interface to browse available topics for the current semester, filtered by status and relevance. The application logic further enforces eligibility checks, verifying that a student is enrolled in the appropriate semester module before permitting a topic request.

4.1.3. Project Management

Once a topic is assigned, the Project Management module facilitates the ongoing academic lifecycle of the Pre-Thesis and Thesis projects. This component integrates the submission and evaluation workflows, serving as a central repository for project artifacts. The document submission system, powered by Multer middleware [19], allows students to upload version-controlled deliverables, such as interim reports and source code, directly to the server. Simultaneously, the module provides supervisors and committee members with grading interfaces. These interfaces are secured to ensure that only authorized evaluators can submit scores or qualitative feedback. The system automatically tracks the status of each project, transitioning states from "In Progress" to "Completed" or "Defense Ready" based on the fulfillment of academic requirements.

4.1.4. Report Exportation

To support administrative record-keeping, the system includes a dedicated Report Exportation feature. This module utilizes server-side PDF generation libraries to compile structured academic reports. Administrators and Moderators can trigger the generation of official documents, such as Thesis Registration forms and Final Evaluation reports, which aggregate data regarding student performance, supervisor details, and committee decisions.

These documents are formatted to meet institutional standards, incorporating university branding and layout requirements, thereby streamlining the process of archiving academic results.

4.2. Supporting Features

4.2.1. Automated Scheduling and Timeslot Management

The system reduces the administrative burden associated with organizing defense sessions through an automated scheduling module. This feature assists Moderators in allocating defense timeslots and assigning committee members to specific thesis projects. By analyzing teacher availability and existing schedule constraints, the logic helps prevent scheduling conflicts. The implementation allows for the bulk configuration of defense sessions, linking specific rooms and times to thesis projects, and persisting this data within the DefenseSession entity to ensure all stakeholders are informed of their obligations.

4.2.2. Announcement Dashboard

Communication across the faculty is centralized within the Announcement Dashboard. This feature allows Administrators and Moderators to broadcast critical information, such as semester deadlines or policy updates, to targeted user groups. The backend implementation supports audience filtering, enabling messages to be directed specifically to students, teachers, or the entire faculty. Announcements can be pinned for high visibility, ensuring that time-sensitive information remains accessible at the top of the user’s feed.

4.2.3. Statistics Dashboard

To provide oversight on the faculty's operational health, the Statistics Dashboard aggregates real-time data regarding the academic term. This module queries the database to calculate key performance indicators, such as the total number of registered topics, the distribution of grades, and the ratio of students to supervisors. These metrics are processed on the server and delivered to the frontend, allowing Administrators to visualize completion rates and identify potential bottlenecks in the supervision capacity of the department.

4.2.4. Notifications

The user interface was constructed using React [8], employing component-based architecture to maximize code reusability and clarity. The UI structure is divided into modular components such as the Dashboard, Topic List, and Announcement board. Navigation within the Single Page Application (SPA) is handled by React Router DOM, which manages routing between role-specific views. State management is achieved through React’s built-in hooks, specifically useState for local component state and useEffect for handling side effects like data fetching. Styling was implemented using CSS heavily supplemented by Tailwind CSS to ensure a responsive and aesthetically consistent design.

CHAPTER V

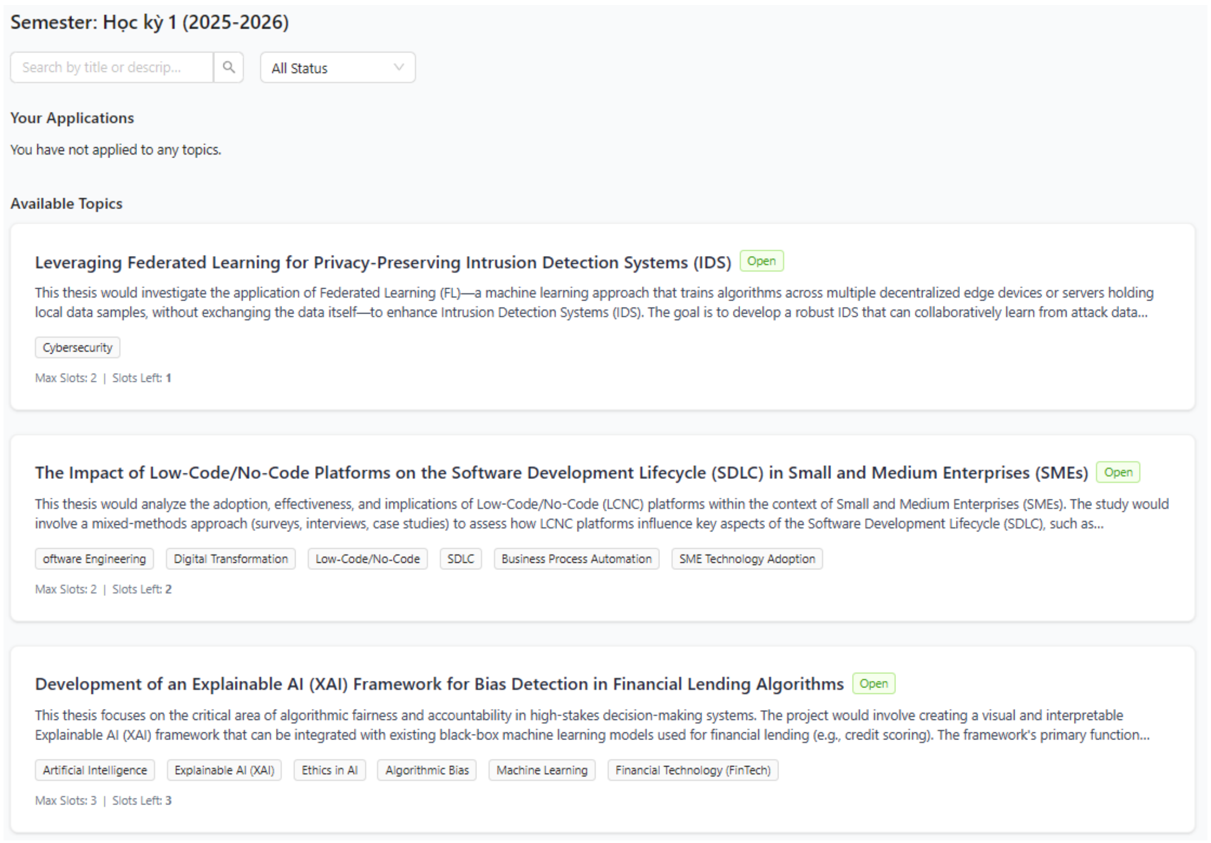
RESULT

5.1. System Interface and User Experience

The realization of the THESISBOARD platform has resulted in a comprehensive suite of user interfaces designed to streamline the complex workflows of thesis management. The following figures illustrate the deployed application, highlighting the key functional areas accessible to students and faculty.

5.1.1. Topic Selection Interface

The student experience begins with the Topic Selection interface. As depicted in the system's design, this page presents a catalog of available research topics for the active semester. Each topic card displays critical information, including the research title, the supervisor's name, and the number of remaining slots. Visual indicators, such as "Open" status tags, allow students to quickly identify viable opportunities. The layout is optimized for readability, enabling students to efficiently browse and filter topics based on their academic interests before initiating an application.


Semester: Học kỳ 1 (2025-2026)

Your Applications

You have not applied to any topics.

Available Topics

Leveraging Federated Learning for Privacy-Preserving Intrusion Detection Systems (IDS) Open

This thesis would investigate the application of Federated Learning (FL)—a machine learning approach that trains algorithms across multiple decentralized edge devices or servers holding local data samples, without exchanging the data itself—to enhance Intrusion Detection Systems (IDS). The goal is to develop a robust IDS that can collaboratively learn from attack data...

Cybersecurity

Max Slots: 2 | Slots Left: 1

The Impact of Low-Code/No-Code Platforms on the Software Development Lifecycle (SDLC) in Small and Medium Enterprises (SMEs) Open

This thesis would analyze the adoption, effectiveness, and implications of Low-Code/No-Code (LCNC) platforms within the context of Small and Medium Enterprises (SMEs). The study would involve a mixed-methods approach (surveys, interviews, case studies) to assess how LCNC platforms influence key aspects of the Software Development Lifecycle (SDLC), such as...

oftware Engineering | Digital Transformation | Low-Code/No-Code | SDLC | Business Process Automation | SME Technology Adoption

Max Slots: 2 | Slots Left: 2

Development of an Explainable AI (XAI) Framework for Bias Detection in Financial Lending Algorithms Open

This thesis focuses on the critical area of algorithmic fairness and accountability in high-stakes decision-making systems. The project would involve creating a visual and interpretable Explainable AI (XAI) framework that can be integrated with existing black-box machine learning models used for financial lending (e.g., credit scoring). The framework's primary function...

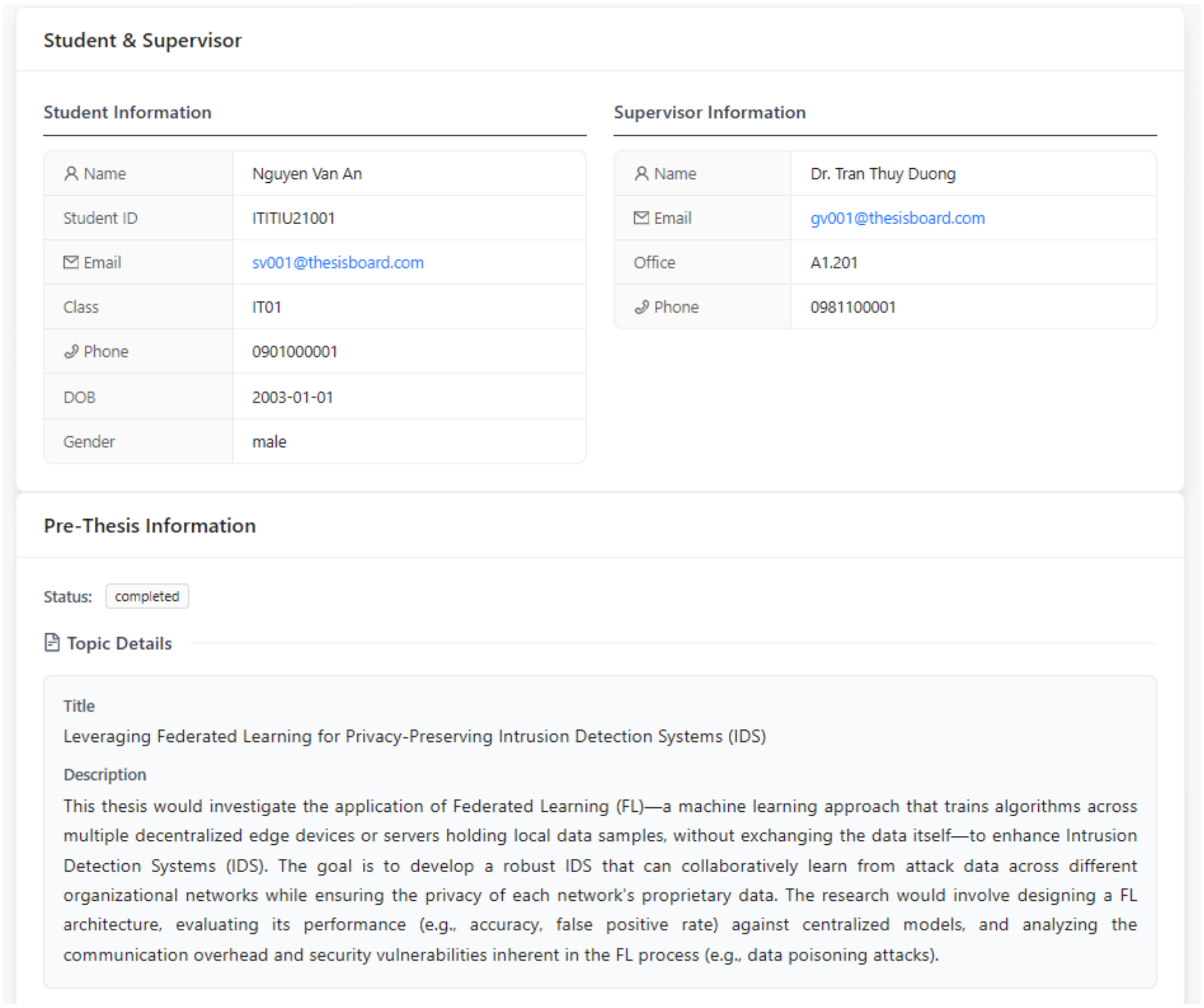
Artificial Intelligence | Explainable AI (XAI) | Ethics in AI | Algorithmic Bias | Machine Learning | Financial Technology (FinTech)

Max Slots: 3 | Slots Left: 3

Figure 12. Topics Page for Student

5.1.2. Project Detail and Management View

Upon successful registration, users interact with the Project Detail interface. This view consolidates all pertinent information regarding a specific thesis project. The interface displays the student's profile alongside the supervisor's contact details, ensuring clear lines of communication. A central feature of this page is the status tracker, which indicates the current phase of the project, such as "In Progress" or "Defense Completed." This centralization of data eliminates the ambiguity often associated with manual tracking methods.



Student & Supervisor

Student Information

Name	Nguyen Van An
Student ID	ITITIU21001
Email	sv001@thesisboard.com
Class	IT01
Phone	0901000001
DOB	2003-01-01
Gender	male

Supervisor Information

Name	Dr. Tran Thuy Duong
Email	gv001@thesisboard.com
Office	A1.201
Phone	0981100001

Pre-Thesis Information

Status: completed

Topic Details

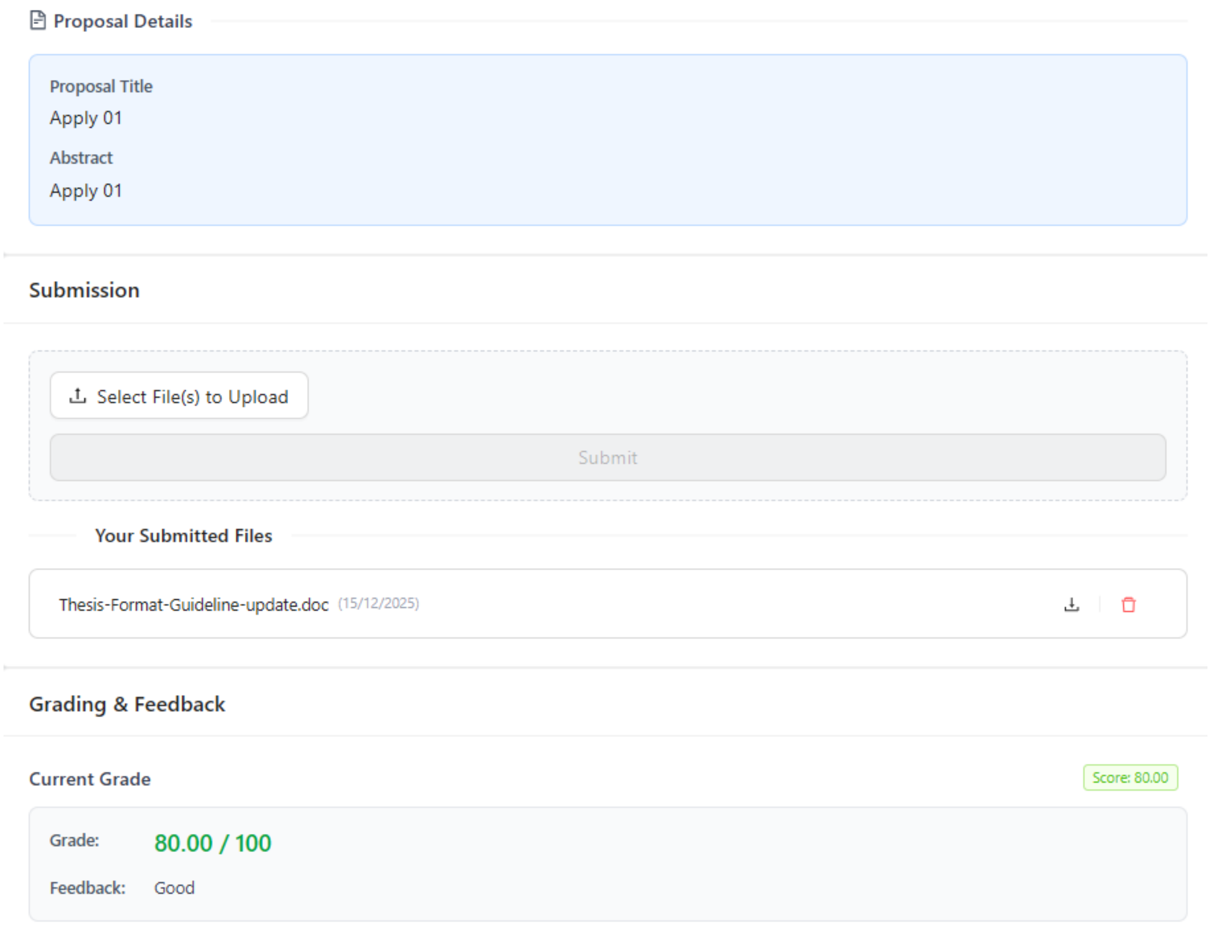
Title

Leveraging Federated Learning for Privacy-Preserving Intrusion Detection Systems (IDS)

Description

This thesis would investigate the application of Federated Learning (FL)—a machine learning approach that trains algorithms across multiple decentralized edge devices or servers holding local data samples, without exchanging the data itself—to enhance Intrusion Detection Systems (IDS). The goal is to develop a robust IDS that can collaboratively learn from attack data across different organizational networks while ensuring the privacy of each network's proprietary data. The research would involve designing a FL architecture, evaluating its performance (e.g., accuracy, false positive rate) against centralized models, and analyzing the communication overhead and security vulnerabilities inherent in the FL process (e.g., data poisoning attacks).

Figure 13. Inside the Project Page (1)


Proposal Details

Proposal Title
Apply 01
Abstract
Apply 01

Submission

Your Submitted Files

Thesis-Format-Guideline-update.doc (15/12/2025)

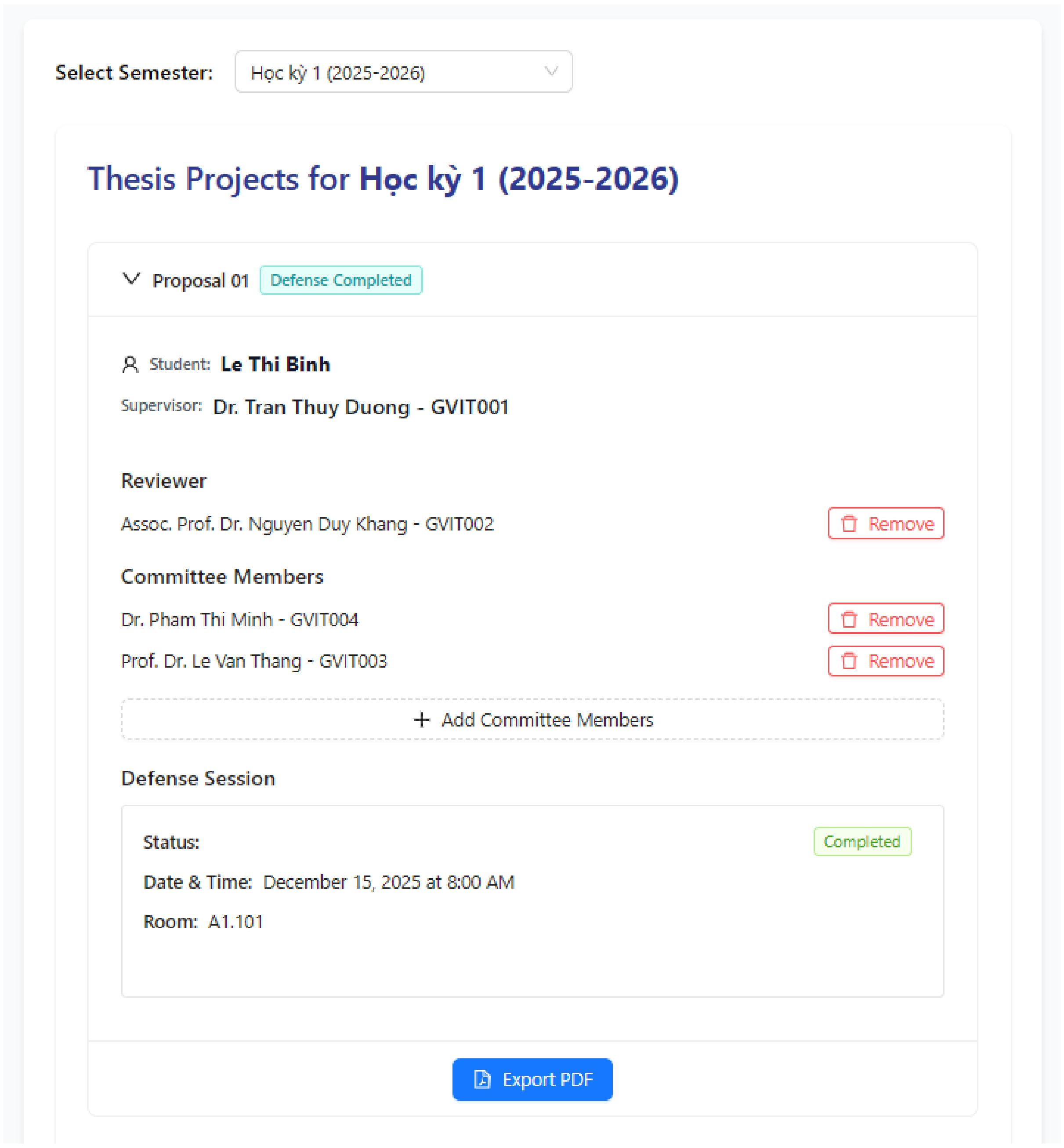
Grading & Feedback

Current Grade

Score: 80.00

Grade: 80.00 / 100
Feedback: Good

Figure 14. Inside the Project Page (2)


Select Semester: Học kỳ 1 (2025-2026)

Thesis Projects for **Học kỳ 1 (2025-2026)**

Proposal 01 Defense Completed

Student: **Le Thi Binh**

Supervisor: **Dr. Tran Thuy Duong - GVIT001**

Reviewer

Assoc. Prof. Dr. Nguyen Duy Khang - GVIT002 Remove

Committee Members

Dr. Pham Thi Minh - GVIT004 Remove

Prof. Dr. Le Van Thang - GVIT003 Remove

+ Add Committee Members

Defense Session

Status: Completed

Date & Time: December 15, 2025 at 8:00 AM

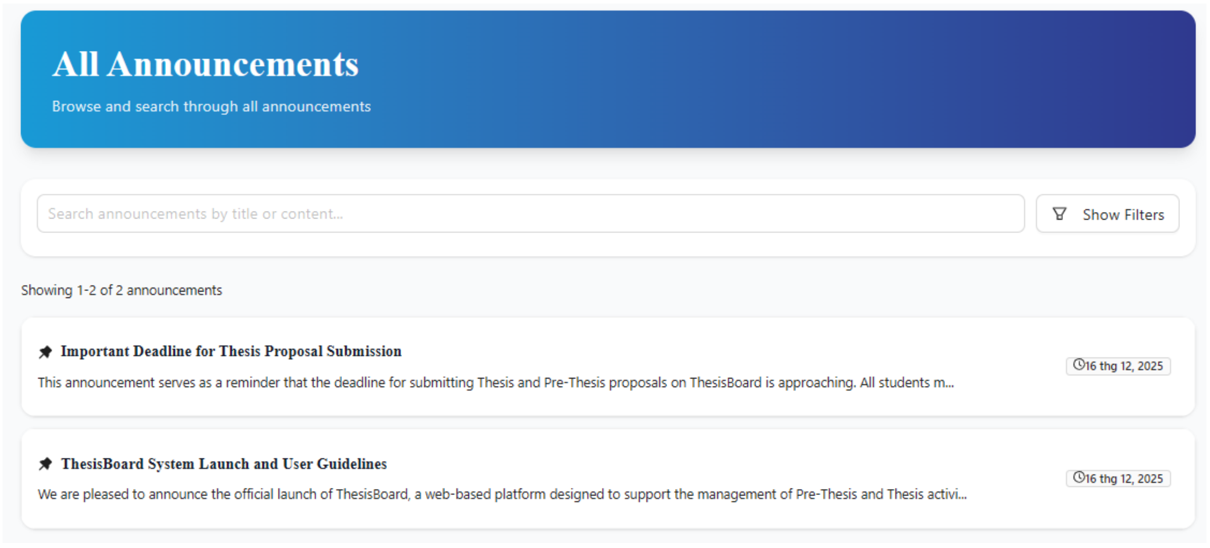
Room: A1.101

Export PDF

Figure 15. Thesis Project Management

5.1.3. Submission and Evaluation Portals

The system includes a dedicated portal for document management and grading. The submission section provides students with a secure upload facility for their thesis materials, maintaining a history of submitted files with timestamps. Below this, the grading and feedback section allows supervisors to input numerical scores and written assessments. The interface displays the current grade prominently, ensuring transparency in the evaluation process. This result demonstrates the system's ability to handle the iterative exchange of academic work and feedback.


All Announcements
Browse and search through all announcements
Search announcements by title or content...
Show Filters

Showing 1-2 of 2 announcements

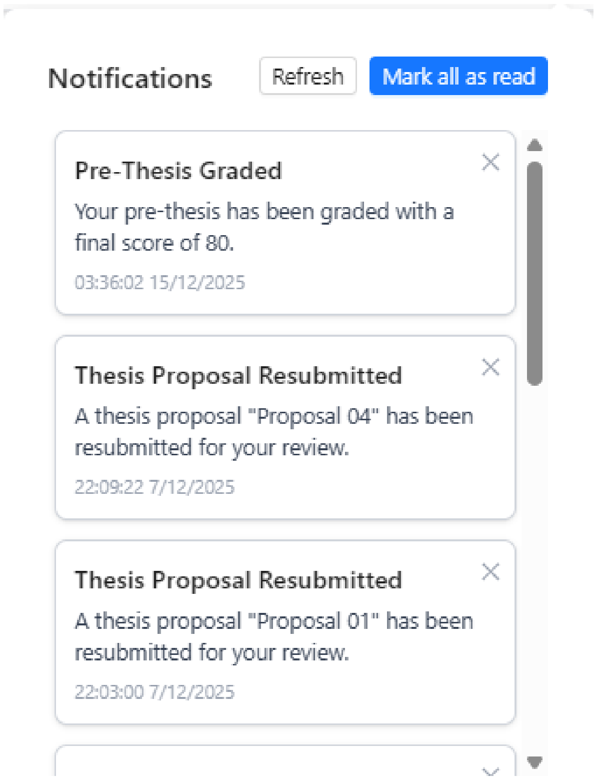
Important Deadline for Thesis Proposal Submission
This announcement serves as a reminder that the deadline for submitting Thesis and Pre-Thesis proposals on ThesisBoard is approaching. All students m...
16 thg 12, 2025

ThesisBoard System Launch and User Guidelines
We are pleased to announce the official launch of ThesisBoard, a web-based platform designed to support the management of Pre-Thesis and Thesis activi...
16 thg 12, 2025

Figure 16. Announcement System

5.1.4. Notification System

The efficacy of the communication features is evidenced by the Notification System interface. This component appears as an overlay or dedicated panel, listing recent alerts such as "Pre-Thesis Graded" or "Thesis Proposal Resubmitted." Each notification is time-stamped and linked to the relevant entity, allowing users to navigate directly to the action item. This result confirms the system's capability to maintain user engagement and ensure timely responses to academic requirements.



Pre-Thesis Graded
Your pre-thesis has been graded with a final score of 80.
03:36:02 15/12/2025

Thesis Proposal Resubmitted
A thesis proposal "Proposal 04" has been resubmitted for your review.
22:09:22 7/12/2025

Thesis Proposal Resubmitted
A thesis proposal "Proposal 01" has been resubmitted for your review.
22:03:00 7/12/2025

Figure 17. The Notification System

5.2. Discussion

5.2.1. Strengths

The deployment of the THESISBOARD platform represents a significant technological advancement over the disjointed administrative methods previously employed by the School of Computer Science and Engineering. The most prominent strength of the system lies in its ability to centralize data management, thereby eliminating the fragmentation inherent in the legacy workflow of spreadsheets and email correspondence. By consolidating user profiles, topic registrations, and evaluation records into a unified relational database, the platform ensures a "single source of truth" for the entire faculty, significantly reducing the likelihood of data discrepancies and administrative errors.

Furthermore, the architectural decision to utilize a modern technology stack—comprising Node.js, React, and TypeScript—facilitates a high degree of system performance and maintainability. The implementation of strict typing through TypeScript enhances code reliability, minimizing runtime errors and streamlining future development efforts. Security is another robust aspect of the application, achieved through the integration of Auth0 for identity management. This approach offloads the complexities of authentication to a specialized provider, ensuring industry-standard security protocols are enforced while enabling precise Role-Based Access Control (RBAC) within the application logic. Consequently, the system effectively protects sensitive academic data by strictly segregating permissions between students, faculty, and administrators.

Operational efficiency is also maximized through the automation of complex workflows. The system’s ability to handle concurrent topic registrations using atomic database transactions ensures fairness and adherence to supervisor quotas without manual intervention. Additionally, the automated scheduling logic for defense sessions significantly reduces the logistical burden on moderators, transforming a task that typically requires days of coordination into a streamlined digital process.

5.2.2. Limitations

Despite these significant achievements, the current iteration of the platform exhibits certain functional constraints that warrant discussion. A primary limitation is the absence of comprehensive visual analytics. While the system successfully captures and stores a vast amount of academic data, it currently lacks the graphical dashboards necessary to visualize this information effectively. Administrators can access raw metrics, but the inability to generate trend lines or distribution charts for grade analysis and completion rates limits the potential for data-driven strategic planning.

Another notable constraint involves the communication infrastructure. The notification system is currently designed as an internal mechanism, restricted solely to the application interface. Consequently, stakeholders must actively log in to the platform to receive updates regarding grading or registration status. The lack of an integrated external notification service, such as email or SMS alerts, creates a potential latency in communication, particularly for users who do not access the dashboard frequently. This isolation of the alerting mechanism could lead to delays in responsiveness during time-sensitive phases of the academic semester.

5.3. Evaluation

The evaluation of the THESISBOARD system demonstrates a high degree of fidelity to the initial project objectives, validating its effectiveness as a solution for academic lifecycle management. Functionally, the application successfully encompasses the entire scope of the thesis process, from the preliminary stage of topic selection to the finalization of defense grading. The modular design of the backend services allows for the seamless execution of distinct workflows for Pre-Thesis and Thesis projects, verifying that the system can accommodate the specific regulatory nuances of each academic module.

From a technical perspective, the system exhibits strong structural integrity. The adherence to the Model-View-Controller (MVC) architectural pattern ensures a clear separation of concerns, which not only enhances code readability but also supports the scalability of the application. The database schema has been normalized to the Third Normal Form (3NF), effectively preventing data redundancy and ensuring the consistency of relational links between entities such as students, semesters, and supervisors.

Security and usability evaluations further confirm the system's robustness. The implementation of JSON Web Token (JWT) verification middleware ensures that all API endpoints are protected against unauthorized access, validating the effectiveness of the RBAC model. Meanwhile, the responsive user interface, constructed with React and Tailwind CSS, provides consistent experience across devices, addressing the accessibility requirements of a diverse user base. Overall, the system not only resolves the immediate logistical challenges of the faculty but also establishes a resilient foundation for future digital transformation initiatives within the department.

CHAPTER VI

CONCLUSION AND FUTURE WORK

5.4. Project Summary

The inception of the THESISBOARD initiative was driven by the imperative to rectify systemic inefficiencies characterizing the administration of pre-thesis and thesis projects within the School of Computer Science and Engineering. Historically, the faculty relied on disparate, manual methodologies—predominantly involving uncoordinated spreadsheet manipulation and electronic correspondence, which engendered significant administrative latency and diminished procedural transparency. The primary objective of this research was to ameliorate these operational bottlenecks through the design and deployment of a centralized, web-based management infrastructure. By organizing the academic lifecycle into a cohesive digital environment, the system provides distinct, functional interfaces tailored to the specific needs of students, faculty, and administrators. Adhering to an iterative development methodology, the project advanced from a rigorous requirements analysis to a comprehensive implementation utilizing modern web technologies, thereby ensuring a robust and structured resolution to the identified administrative challenges.

5.5. Achievements and Results

The implementation phase culminated in the delivery of a fully operational software ecosystem that satisfies the critical functional requirements of the academic department. A principal achievement of this undertaking is the successful deployment of role-specific modules, which facilitate seamless topic registration and document submission for students, whilst simultaneously empowering lecturers with tools for supervision quota management and grading. From a technical vantage point, the system integrates a rigorous Role-Based Access Control (RBAC) mechanism, guaranteeing secure and appropriate data authorization. The backend architecture, established upon Node.js and Express.js, effectively exposes a secure RESTful API capable of processing complex business logic, such as automated slot validation. Furthermore, the frontend implementation offers a responsive user interface that optimizes stakeholder interaction. The strategic utilization of Sequelize ORM for data modeling alongside MySQL for persistence has yielded a system characterized by data integrity and operational stability, representing a substantial advancement over antecedent manual processes.

5.6. Limitations

Notwithstanding the successful realization of core functionalities, the current iteration of the system exhibits specific constraints. A primary limitation is the absence of comprehensive visual analytics; while the platform captures extensive academic data, it currently lacks the graphical dashboards necessary for the high-level analysis of performance metrics, grading distributions, or completion trends. Furthermore, the communication framework remains isolated within the application interface. The absence of an integrated external notification service implies that alerts and announcements do not trigger electronic mail notifications to user accounts, potentially resulting in latency regarding the receipt of time-sensitive information should users fail to access the platform frequently.

5.7. Future Enhancements

To further augment the utility and scope of the platform, several enhancements are proposed for subsequent development cycles. First, the construction of a dedicated mobile application would significantly enhance accessibility, enabling users to receive push notifications and monitor academic progress via portable devices. Second, integration with the university's central Student Information System (SIS) is recommended to automate the synchronization of user enrollment data and final grades, thereby mitigating the need for manual data entry by administrators. Finally, the execution of comprehensive usability testing with a broader cohort of users would provide quantitative data essential for the continued refinement of the user interface and the overall user experience.

5.8. Conclusion

The THESISBOARD system successfully addresses a pivotal administrative void within the faculty by providing a centralized, automated infrastructure for research coordination. By facilitating the transition from error-prone, manual processes to a structured, digital workflow, the project significantly enhances operational efficiency and transparency for all associated stakeholders. The successful deployment of this platform validates the efficacy of modern web technologies in resolving complex academic logistics. Ultimately, the system establishes a resilient foundation for the ongoing digital transformation of departmental processes, offering a scalable solution that actively supports the academic success of the student body.

REFERENCES

[1] Moodle. "About Moodle." Moodle Docs. Accessed: Dec. 16, 2025. [Online]. Available: https://docs.moodle.org/en/About_Moodle

[2] Anthology Inc. "Experience Blackboard (LMS)." Anthology.com. Accessed: Dec. 16, 2025. [Online]. Available: https://www.anthology.com/experience-blackboard

[3] D. F. Ferraiolo and D. R. Kuhn, "Role-based access control," in *Proc. 15th Nat. Comput. Security Conf.*, Baltimore, MD, USA, Oct. 1992, pp. 554–563.

[4] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF, RFC 7519, May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519

[5] A. Al-Hajri, M. A. Rizwan, and M. A. Al-Sharafi, "Evaluating the Performance of Architectural Patterns in Web Application Development," in *Proc. 2024 9th Int. Conf. on Informatics and Computing (ICIC)*, 2024. [Online]. Available: IEEE Xplore.

[6] C.-H. Chen, "Teaching Method for Entity–Relationship Models Based on Semantic Network Theory," *IEEE Access*, vol. 10, pp. 1-14, 2022.

[7] Sequelize. "Sequelize: Feature-rich ORM for modern TypeScript & JavaScript." Sequelize.org. Accessed: Dec. 16, 2025. [Online]. Available: https://sequelize.org/

[8] Meta Platforms, Inc. "React – A JavaScript library for building user interfaces." React.dev. Accessed: Dec. 16, 2025. [Online]. Available: https://react.dev/

[9] OpenJS Foundation. "Node.js v22.12.0 Documentation." Nodejs.org. Accessed: Dec. 16, 2025. [Online]. Available: https://nodejs.org/docs/latest/api/

[10] StrongLoop, IBM, and Contributors. "Express - Node.js web application framework." Expressjs.com. Accessed: Dec. 16, 2025. [Online]. Available: https://expressjs.com/

[11] S. S. N. Challapalli *et al.*, "Web Development and performance comparison of Web Development Technologies in Node.js and Python," in *2021 Int. Conf. on Technological Advancements and Innovations (ICTAI)*, Tashkent, Uzbekistan, Nov. 2021, pp. 1-6.

[12] Okta, Inc. "Auth0 Documentation." Auth0.com. Accessed: Dec. 16, 2025. [Online]. Available: https://auth0.com/docs

[13] Oracle Corporation. "MySQL 8.0 Reference Manual." MySQL.com. Accessed: Dec. 16, 2025. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/

[14] A. Al-Hajri, M. A. Rizwan, and M. A. Al-Sharafi, "Evaluating the Performance of Architectural Patterns in Web Application Development," in Proc. 2024 9th Int. Conf. on Informatics and Computing (ICIC), 2024.

[15] Microsoft, "TypeScript: JavaScript With Syntax For Types," 2025. [Online]. Available: https://www.typescriptlang.org/. [Accessed: Dec. 19, 2025].

[16] Ant Design Team, "Ant Design: An enterprise-class UI design language and React UI library," 2025. [Online]. Available: https://ant.design/. [Accessed: Dec. 19, 2025].

[17] Tailwind Labs, "Tailwind CSS: Rapidly build modern websites without ever leaving your HTML," 2025. [Online]. Available: https://tailwindcss.com/. [Accessed: Dec. 19, 2025].

[18] M. D. E. J. S. Community, "EJS: Embedded JavaScript templates," 2025. [Online]. Available: https://ejs.co/. [Accessed: Dec. 19, 2025].

[19] Express.js Team, "Multer: Node.js middleware for handling multipart/form-data," GitHub, 2025. [Online]. Available: https://github.com/expressjs/multer. [Accessed: Dec. 19, 2025].