

**UNIVERSITY OF TRANSPORT AND COMMUNICATIONS
FACULTY OF INTERNATIONAL EDUCATION**



GRADUATION PROJECT

Topic

**Building a System for UTC ResearchHub
"Project Management System"**

by

L E DINH TU

COURSE 62

CLASS 1

**HIGH-QUALITY PROGRAM OF ENGLISH – VIETNAMESE
INFORMATION TECHNOLOGY**

2021 - 2025 (Course 62)

HANOI - JUNE 2025

**UNIVERSITY OF TRANSPORT AND COMMUNICATIONS
FACULTY OF INTERNATIONAL EDUCATION**



GRADUATION PROJECT

Submitted in partial fulfillment of the requirements for the
degree of engineer

**HIGH-QUALITY PROGRAM OF ENGLISH – VIETNAMESE
INFORMATION TECHNOLOGY**

by

LE DINH TU

Under supervision of Dr(Msc) Luong Thai Le

Examiner of Dr.(Msc)

HA NOI - JUNE 2025

COMMENT OF SUPERVISOR

1. EVALUATION OF CONTENT AND QUALITY OF REPORT:

.....

.....

.....

.....

.....

2. EVALUATION OF CONTENT AND QUALITY OF CODE/PROGRAMS:

.....

.....

.....

.....

.....

3. EVALUATION OF WORKING ATTITUDE OF STUDENT:

.....

.....

.....

.....

.....

.....

4. CONCLUSION:

.....

.....

.....

.....

.....

5. MARK:

...../10.

Hanoi, June 2025
MAIN SUPERVISOR

COMMENT OF REVIEWER

1. EVALUATION OF CONTENT AND QUALITY OF REPORT:

.....

.....

.....

.....

.....

2. EVALUATION OF CONTENT AND QUALITY OF CODE/PROGRAMS:

.....

.....

.....

.....

.....

.....

.....

3 CONCLUSION:

.....

.....

.....

.....

.....

.....

.....

4 MARK:

...../10.

Hanoi, June 2025

REVIEWER

ACKNOWLEDGMENTS

I would like to extend my sincere gratitude to all faculty members at the University of Transport and Communications for their unwavering support and guidance throughout my academic journey. Their dedication to excellence in education has been instrumental in fostering an environment conducive to learning and intellectual growth.

I am particularly grateful to the instructors from the Faculty of International Education and affiliated departments. Their expertise, rigorous teaching, and willingness to share valuable insights have greatly enriched my academic experience and provided a solid foundation for my research endeavors.

My deepest appreciation goes to Ms. Luong Thai Le, my thesis advisor, whose insightful mentorship and meticulous guidance were critical to the successful completion of this project. Her clear direction and thoughtful feedback were invaluable in helping me understand the complex workflows and procedures of the university's graduation thesis defense process. Her continual support ensured that the system developed was well-aligned with the practical requirements of the academic environment.

I also wish to acknowledge the administrative staff and my peers, whose encouragement and assistance contributed positively throughout this process. Their collaboration and support helped create a productive and motivating atmosphere.

Finally, I am sincerely thankful to everyone who has contributed directly or indirectly to the realization of this thesis. This accomplishment reflects the collective efforts and encouragement of a supportive academic community, for which I am profoundly grateful.

Hanoi, June 3, 2025

Student author

Le Dinh Tu

TABLE OF CONTENTS

ACKNOWLEDGMENTS	5
TABLE OF CONTENTS.....	6
TABLE OF FIGURE	8
TABLE OF TABLE	10
INTRODUCTION	11
CHAPTER 1 OVERVIEW OF PROJECT	12
1.1 Introduction of the project	12
1.2 Survey of current solution	13
1.3 Overview of used tools of technologies	13
1.3.1 NextJS.....	13
1.3.2 NestJS	14
1.3.3 TypeScript and VS Code	16
1.3.4 PostgreSQL and Prisma.....	17
1.3.5 Conclusion	18
CHAPTER 2 REQUIREMENTS ANALYSIS AND SYSTEM DESIGN.....	19
2.1. Business Analysis	19
2.1.1 Business Requirements Analysis	19
2.1.2 System Requirements Analysis.....	19
2.1.3 Role-Permission Matrix	20
2.2. System Design	23
2.2.1 Use Case Diagram.....	23
2.2.2 Business Function Diagram	23
2.2.2 System Architecture Diagram.....	24
2.2.3 Sequence Diagram	25
2.3 Design database	27
2.3.1 Database design tables.....	27
2.3.2 Database description.....	30
2.3.3 Conclusion	31

CHAPTER 3 IMPLEMENTATION AND TESTING	32
3.1 Environment, Tools, and Directory Structure	32
3.2 Main Modules Overview.....	32
3.2.1 Authentication.....	32
3.2.2 Selection.....	34
3.2.3 Proposal.....	37
3.2.4 Result	39
3.2.5 Evaluation	39
3.2.6 Dashboard	42
CONCLUSION	43
REFERENCES	45

TABLE OF FIGURE

Figure 1.1 NextJS: Shaping the Future of Web Development [8]	13
Figure 1.2 NestJS Request lifecycle [9]	14
Figure 1.3 The Prisma Data Layer Architecture? [10]	17
Figure 2.1 Use case diagram	23
Figure 2.2 Business Function Diagram	23
Figure 2.3 System Architecture Diagram	24
Figure 2.4 SD – Topic Registration Process	25
Figure 2.5 SD – Student Topic Selection	25
Figure 2.6 SD – Proposal Review and Approval Process	26
Figure 2.7 SD – Submission Process	26
Figure 2.8 SD – Grading	27
Figure 2.9 User management ERD	27
Figure 2.10 Selection management and supervisor assignment ERD	28
Figure 2.11 Result approval ERD	28
Figure 2.12 Research and result management ERD	29
Figure 2.13 Defense and evaluation management ERD	29
Figure 3.1 Login Page Student	32
Figure 3.2 Login Page Faculty	33
Figure 3.3 Login API	33
Figure 3.4 Selection API	34
Figure 3.5 Field Pool Selection Page	35
Figure 3.6 Student Selection Page	35
Figure 3.7 Selection Student Page	36
Figure 3.8 Teacher Selection Page	36
Figure 3.9 Selection Management Page	37
Figure 3.10 Proposal Page for Lecturer	37
Figure 3.11 Proposal API	38
Figure 3.12 Proposal Page for Student	38
Figure 3.13 Submit Project Page	39
Figure 3.14 Evaluation Page for Student	39
Figure 3.15 Evaluation API	40
Figure 3.16 Evaluation Page for Head	40
Figure 3.17 Management Defense Page	41
Figure 3.18 Department Head Dashboard	42
Figure 3.19 Lecturer Portal Dashboard	42

Figure 3.20 Dean Portal Dashboard	42
---	----

TABLE OF TABLE

Table 1 Role-Permission Matrix	21
--------------------------------------	----

INTRODUCTION

In the context of the rapid digital transformation taking place in higher education institutions, the management of graduation theses is still primarily conducted manually. This leads to many drawbacks such as inefficiencies in processes, difficulties in tracking progress, lack of transparency in supervision and evaluation, and fragmented data. Coordination among stakeholders-including students, supervising lecturers, reviewers, department heads, and faculty heads-often faces obstacles due to the absence of a centralized, interactive, and automated system. Therefore, developing a web-based graduation thesis management system is necessary to improve efficiency, increase transparency, support real-time collaboration among parties, and simplify processes such as topic registration, submission of outlines, evaluation, and defense organization.

This system aims to build a secure, user-friendly, and scalable web application integrated with role-based access control to support all stakeholders, including Faculty Heads, Department Heads, Supervising Lecturers, Reviewers, and Students. The goal is to streamline the entire thesis lifecycle-from topic registration, lecturer assignment, outline management, to evaluation, grading, and defense committee organization. The system is designed to meet evolving academic needs, ensuring flexibility and efficiency in management.

The system focuses on lecturers and students at the Faculty of Information Technology, University of Transport and Communications, aiming to automate administrative processes such as topic registration, assignment of supervising lecturers and reviewers, outline management, evaluation, grading, and defense organization. The project only concentrates on managing administrative workflows and does not assess the academic content of the theses, ensuring clarity and focus on management objectives.

CHAPTER 1 OVERVIEW OF PROJECT

1.1 Introduction of the project

Currently, the faculty and departments at the university do not have a comprehensive and unified system for managing graduation projects. The management process is mostly manual, relying on paperwork and direct communication, which leads to difficulties in tracking, assigning, and evaluating projects. This manual approach is time-consuming, prone to errors, and lacks transparency. Therefore, the development of a project management system is essential to improve efficiency and facilitate lecturers, students, and administrative staff throughout the entire process—from registering research topics, assigning supervisors, managing proposals, to evaluating and grading projects.

This project aims to develop a software system that automates and standardizes the graduation project management process, including:

- Defining detailed roles and permissions for Department Heads, Program Chairs, Supervisors, Reviewers, and Students.
- Allowing lecturers to register and manage research directions.
- Enabling Department Heads to assign supervisors to students based on preferences and workload limits.
- Allowing students to create and edit project proposals following provided templates, while supervisors can review and request revisions.
- Managing proposals at the program and department levels, including locking editing rights once proposals are approved.
- Supporting communication and collaboration between supervisors and students.
- Allowing Department Heads to assign reviewers suitable for the research direction.
- Enabling supervisors and reviewers to comment and grade students through digital forms.
- Creating defense committees consisting of lecturers from multiple programs to ensure diversity in research directions.
- Managing grades, revision requests, and generating reports efficiently.

The system will enhance management efficiency, reduce administrative workload, increase transparency, and foster better collaboration among all parties involved in the graduation project process.

1.2 Survey of current solution

The existing thesis management system presents several challenges:

- Manual topic registration causing time inefficiencies.
- Difficulties in tracking student progress in real-time.
- Suboptimal assignment of thesis supervisors.
- Non-standardized processes for thesis reviewing and grading.
- Inefficient document storage and management.

1.3 Overview of used tools of technologies

1.3.1 NextJS

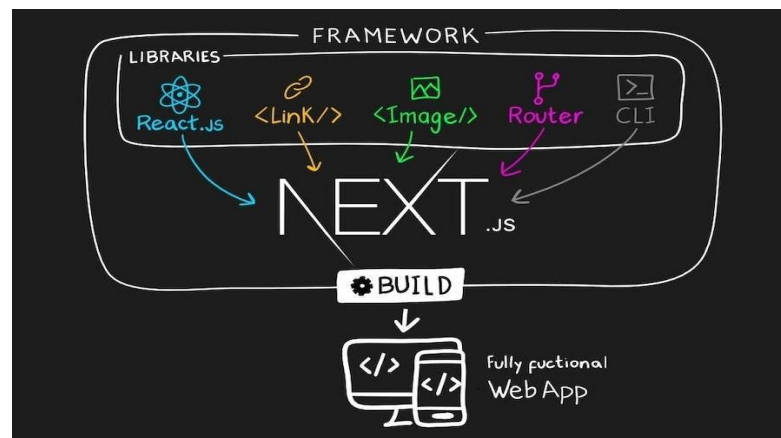


Figure 1.1 NextJS: Shaping the Future of Web Development [8]

NextJS is a robust and versatile React-based framework that excels at building highly performant, SEO-friendly, and scalable web applications. It is particularly well-suited for complex projects like a thesis management system, where responsiveness, reliability, maintainability, flexibility, and user experience are all paramount.

NextJS supports both SSR and SSG out of the box, providing significant advantages in terms of performance and user experience. SSR allows pages to be rendered on the server before being sent to the client, which results in faster initial load times and improved SEO — essential for features such as browsing thesis topics, viewing defense schedules, or accessing student profiles where content freshness and accessibility are critical. Meanwhile, SSG pre-builds static pages at build time, reducing server overhead and improving scalability, especially for pages that do not change frequently. This dual capability ensures the system can dynamically balance between real-time data needs and efficient delivery.

NextJS integrates seamlessly with powerful UI libraries such as MUI (Material-UI) and Tailwind CSS, enabling the creation of polished, intuitive, and accessible interfaces. These technologies support responsive design principles, ensuring the thesis management platform looks and functions flawlessly across a wide range of devices — from desktops and laptops to tablets and smartphones. This cross-device compatibility is crucial for catering to diverse users including students, faculty members, and administrative staff, enhancing overall user satisfaction and engagement.

The NextJS ecosystem is enriched by complementary tools that streamline development and enhance application robustness. Zustand offers an efficient and minimalistic state management solution, reducing boilerplate code and improving developer productivity. React Query simplifies asynchronous data fetching and caching, providing a seamless and reactive user experience even when dealing with complex data workflows like thesis submissions and review processes. Moreover, deploying the application via Vercel, the creators of NextJS, offers optimized hosting with features like automatic scaling, global CDN, and zero-configuration deployment, allowing rapid iterations and quick releases aligned with Agile Scrum methodologies.

NextJS benefits from a large and active developer community along with detailed official documentation. This wealth of resources mitigates technical risks during development by providing best practices, troubleshooting guides, and continuous updates. Consequently, the thesis management system development team can confidently deliver a high-quality product within constrained timelines, ensuring stability, maintainability, and future scalability.

In summary, NextJS not only provides the foundational technologies necessary for building a modern, efficient thesis management system but also fosters a productive development environment that supports quick adaptation, robust performance, and excellent user experience across all facets of the application.

1.3.2 NestJS

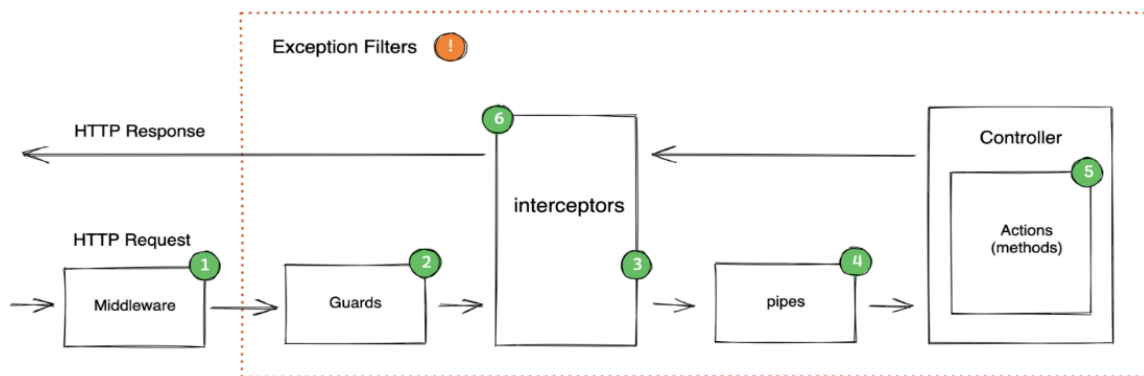


Figure 1.2 NestJS Request lifecycle [9]

NestJS is a cutting-edge Node.js framework built with TypeScript, engineered to facilitate the development of highly efficient, well-structured, and scalable backend applications. Its robust design and advanced features make it an excellent choice for managing the complex and dynamic requirements of a thesis management system.

NestJS adopts a modular architecture that breaks down application logic into discrete, cohesive modules comprising controllers, services, and providers. This modularity allows developers to organize code around specific features — such as thesis topic registration, advisor-student assignments, and outline approvals — making the codebase more maintainable and scalable. Modules can be independently developed, tested, and reused across the system, greatly enhancing developer productivity and ensuring a clean separation of concerns.

Security is paramount in academic management systems, which handle sensitive data like student profiles, grades, and defense schedules. NestJS comes equipped with powerful security mechanisms including middleware, guards, and interceptors. It supports role-based access control (RBAC) to enforce fine-grained permissions, and seamless integration with JSON Web Token (JWT) authentication to ensure secure user sessions and API access. These features collectively protect the integrity and confidentiality of critical academic information, complying with best security practices.

NestJS integrates effortlessly with Prisma, a modern and performant Object-Relational Mapping (ORM) tool. Prisma simplifies database operations by providing a type-safe and declarative query API, reducing boilerplate code and minimizing runtime errors. This integration accelerates the development of reliable RESTful APIs for core system functionalities, including user management, thesis progress tracking, and dynamic reporting. It also improves maintainability by aligning database schema changes with application logic through migrations and schema validations.

Beyond core backend functionalities, NestJS streamlines development workflows by offering built-in support for critical features such as authentication, centralized logging, exception handling, and validation. Its extensible architecture empowers teams to integrate advanced capabilities in the future, such as real-time notifications for defense schedules, AI-powered recommendation systems for advisor assignments, or automated plagiarism detection modules. This flexibility ensures the thesis management system can evolve seamlessly alongside academic needs and technological advancements.

NestJS boasts an active community and a rich ecosystem of plugins, tools, and tutorials. This broad support network provides valuable resources for troubleshooting, optimization, and continuous learning, reducing development risks and enhancing project delivery timelines.

In summary, NestJS offers a powerful, secure, and developer-friendly backend framework perfectly aligned with the demands of a sophisticated thesis management system. Its modular structure, security-first design, and smooth database integration create a solid foundation for building maintainable and scalable academic applications, while its extensibility ensures long-term adaptability.

1.3.3 TypeScript and VS Code

TypeScript, a powerful superset of JavaScript, plays a crucial role in enhancing code reliability, maintainability, and scalability across both the frontend (NextJS) and backend (NestJS) of the thesis management system. By using TypeScript consistently throughout the entire stack, the development team achieves a high level of code uniformity and consistency, which is essential for complex projects that require seamless integration between different layers of the application. This uniformity not only reduces the likelihood of bugs but also makes the codebase easier to understand, maintain, and extend over time.

Complementing TypeScript is Visual Studio Code, a lightweight yet highly extensible and easy-to-integrate code editor that offers a streamlined user experience without compromising on powerful features. Its extensibility allows developers to tailor the environment to their workflow, integrating seamlessly with TypeScript and other development tools. Features such as IntelliSense provide intelligent code completions, real-time error checking, and quick navigation, significantly accelerating development. Visual Studio Code's built-in debugging capabilities also make identifying and fixing issues more efficient, which is particularly valuable during Agile sprints.

TypeScript's static typing system serves as a safety net by catching potential errors early, ensuring smoother interaction between frontend and backend components and minimizing debugging efforts. Strong typing encourages clearer documentation and self-explanatory code, easing onboarding and future maintenance. The synergy between TypeScript and Visual Studio Code results in a highly productive development environment, fostering better collaboration and reducing integration friction. This combination lowers development costs by minimizing bugs and technical debt while supporting scalable, maintainable growth, ultimately leading to a robust and reliable thesis management system.

1.3.4 PostgreSQL and Prisma

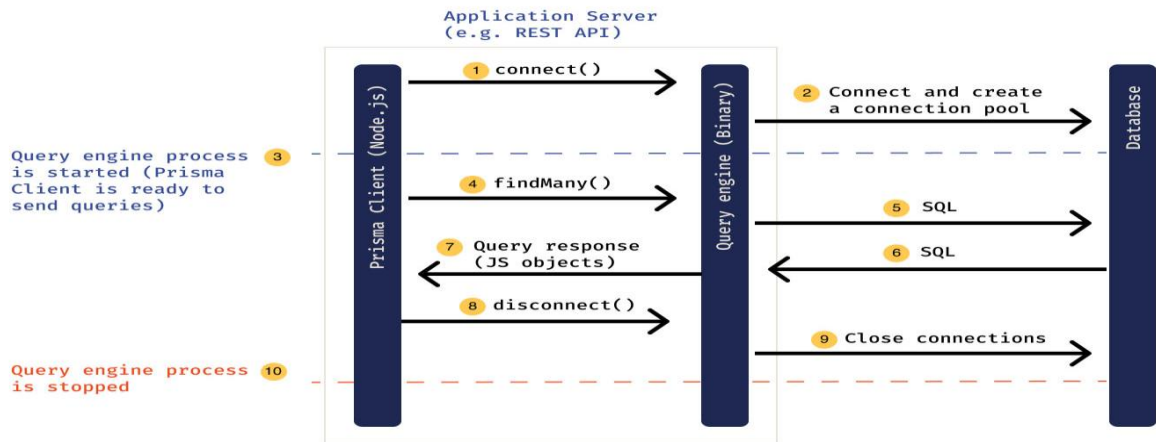


Figure 1.3 The Prisma Data Layer Architecture? [10]

PostgreSQL and **Prisma** together create a robust, reliable, and scalable data layer for the thesis management system. This combination ensures high performance and flexible data handling, supporting long-term development needs effectively.

- **PostgreSQL:** An open-source relational database, PostgreSQL excels in handling complex queries and large datasets, making it ideal for storing information about users, thesis topics, outlines, grades, and defense schedules. Its support for foreign keys and constraints ensures data integrity, accommodating growth as user numbers increase.
- **Prisma:** A modern ORM, Prisma provides an intuitive, type-safe API that simplifies database operations like creating, updating, or querying data. It reduces boilerplate code and errors, while ensuring synchronization between TypeScript code and database schemas.
- **Scalability:** This combination allows the system to handle increasing data volumes, such as when scaling to university-wide use, and supports easy database migrations for future updates.
- **Developer Productivity:** By integrating Prisma with PostgreSQL, developers benefit from automated type generation and seamless database workflows, which speed up development time and reduce the likelihood of runtime errors, resulting in faster feature delivery and easier maintenance.
- **Security:** PostgreSQL offers robust security features, including authentication, authorization, and data encryption, while Prisma helps enforce secure database access patterns through its generated queries, ensuring that sensitive thesis and user data are well protected.

1.3.5 Conclusion

This technology stack facilitates rapid development and seamless integration with external services, which is essential for adapting to the ever-changing needs of academic institutions. The modular architecture of NestJS, combined with the component-based structure of NextJS, enables development teams to work concurrently on different parts of the system without conflict, accelerating feature delivery and boosting overall productivity.

Moreover, employing TypeScript consistently across the entire codebase introduces strong typing and robust tooling support, significantly enhancing code quality and maintainability. Early detection of errors and enforced coding standards help reduce technical debt, while simplifying onboarding for new developers, fostering smoother collaboration and more efficient teamwork.

Together, these technologies establish a solid, scalable foundation that prioritizes security, flexibility, and adaptability. The system is well-equipped to handle growing user demands, integrate new functionalities, and evolve alongside shifting academic requirements. Ultimately, this cohesive stack ensures a reliable, efficient, and future-proof thesis management platform.

In addition, this combination promotes long-term sustainability and simplified maintenance. Clear separation of concerns and well-defined interfaces between the frontend, backend, and database layers allow updates and improvements to be implemented with minimal disruption. Automated testing coupled with strong typing minimizes the risk of bugs during development and deployment, resulting in a stable and resilient system. This thoughtful design not only supports current needs but also paves the way for future innovations, making it an ideal solution for the dynamic environment of academia.

CHAPTER 2 REQUIREMENTS ANALYSIS AND SYSTEM DESIGN

2.1. Business Analysis

2.1.1 Business Requirements Analysis

Research Direction Registration Process:

- Lecturers register primary research directions they can supervise.
- The system enforces maximum student capacity per research direction.
- Department and faculty administrators review and approve research directions.
- Students can view the list of approved research directions.

Supervisor Assignment Process

- Faculty administrators assign supervisors prioritizing lecturer and student preferences.
- Assignments respect maximum supervisory loads and lecturers' expertise areas.

Thesis Outline Management Process

- Students prepare and submit thesis outlines using standardized templates.
- Supervisors review outlines and request modifications when needed.
- Department administrators approve finalized outlines.
- Once approved, outlines are locked from further student edits.

Review and Grading Process

- Faculty administrators assign independent reviewers aligned with thesis topics.
- Reviewers provide evaluations and assign scores.
- The system calculates average scores and generates reports.

Thesis Defense Process

- Defense committees composed of 3–5 faculty members are formed.
- Defense schedules are planned and managed.
- Grades and revision requests are recorded and managed.
- Final defense results and reports are generated.

2.1.2 System Requirements Analysis

User Management

- User registration and authentication.
- Role-based access control.
- Personal information management.
- Management of faculty and student profiles.

Research Direction Management

- Registration and approval of research directions.
- Management of student capacity per direction.
- Search and filter functionality for research directions.

Thesis Topic Management

- Topic registration and supervisor assignment.
- Progress tracking and status monitoring.

Thesis Outline Management

- Creation, editing, and document upload.
- Outline approval workflow and locking mechanism.

Defense Management

- Formation of defense committees.
- Scheduling defenses.
- Grading and reporting functions.

Non-Functional Requirements

- Data security and confidentiality.
- System performance and responsiveness.
- Scalability for increasing user base and data volume.
- Maintainability and ease of updates.
- User-friendly and accessible interface design.
- Reliability and availability to minimize downtime.
- Compliance with relevant legal and institutional regulations.
- Efficient resource utilization for cost-effectiveness.
- Robust error handling and recovery mechanisms.
- Support for multi-device and cross-platform access.

2.1.3 Role-Permission Matrix

This matrix establishes and defines the permissions assigned to each account involved in the thesis defense process. It helps build a clear and secure access framework, ensuring that every user role has the appropriate rights to perform their tasks efficiently throughout the entire workflow.

Table 1 Role-Permission Matrix

Permission / Role	Dean	Department Head / Secretary	Supervising Lecturer	Reviewing Lecturer	Student
Register research topics			✓		
Assign supervisors to students	✓				
Create and upload project proposals					✓
View/edit proposals of supervised students			✓		
Request proposal revisions to supervised students			✓		
Upload finalized/agreed proposal			✓		
Prevent student from editing proposal after final	✓		✓		
Review proposals within department		✓			
Export proposals report (topic, student, supervisor)		✓			
Request proposal revisions at department level		✓			
Allow student to edit proposal after department review		✓			✓
Lock proposal editing rights after finalization	✓				

Permission / Role	Dean	Department Head / Secretary	Supervising Lecturer	Reviewing Lecturer	Student
Export finalized proposal reports	✓				
Assign reviewers to supervisors	✓				
Evaluate and grade students			✓	✓	
Export evaluation and grading reports			✓	✓	
Create thesis defense committees	✓				
Ensure students are not in committees with their supervisors	✓				
Input scores and manage revision requests		✓			
Export committee score sheets		✓			
Send revision requests to students		✓			
Direct interaction between supervisors and students			✓		✓

2.2. System Design

2.2.1 Use Case Diagram

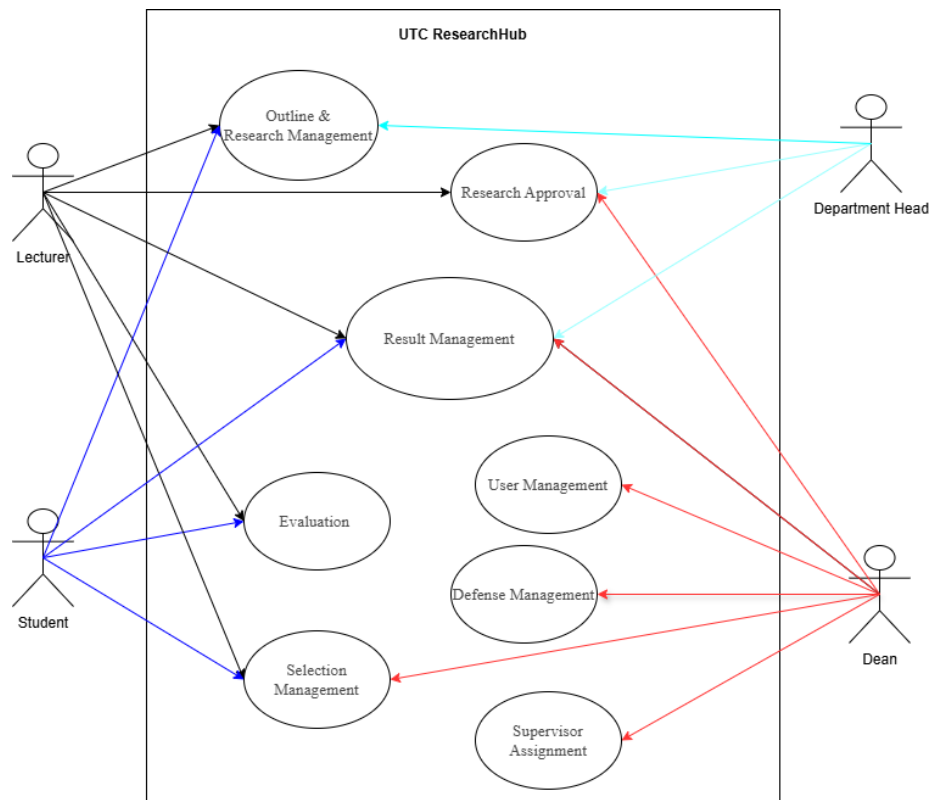


Figure 2.1 Use case diagram

2.2.2 Business Function Diagram

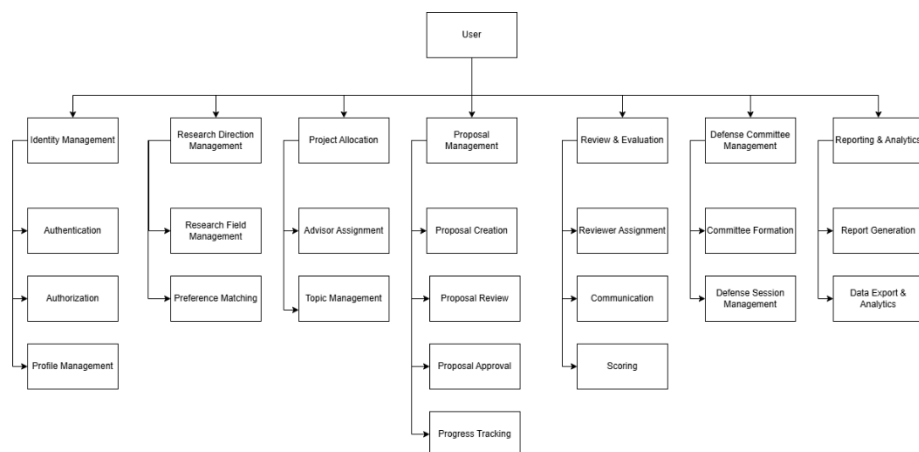


Figure 2.2 Business Function Diagram

2.2.2 System Architecture Diagram

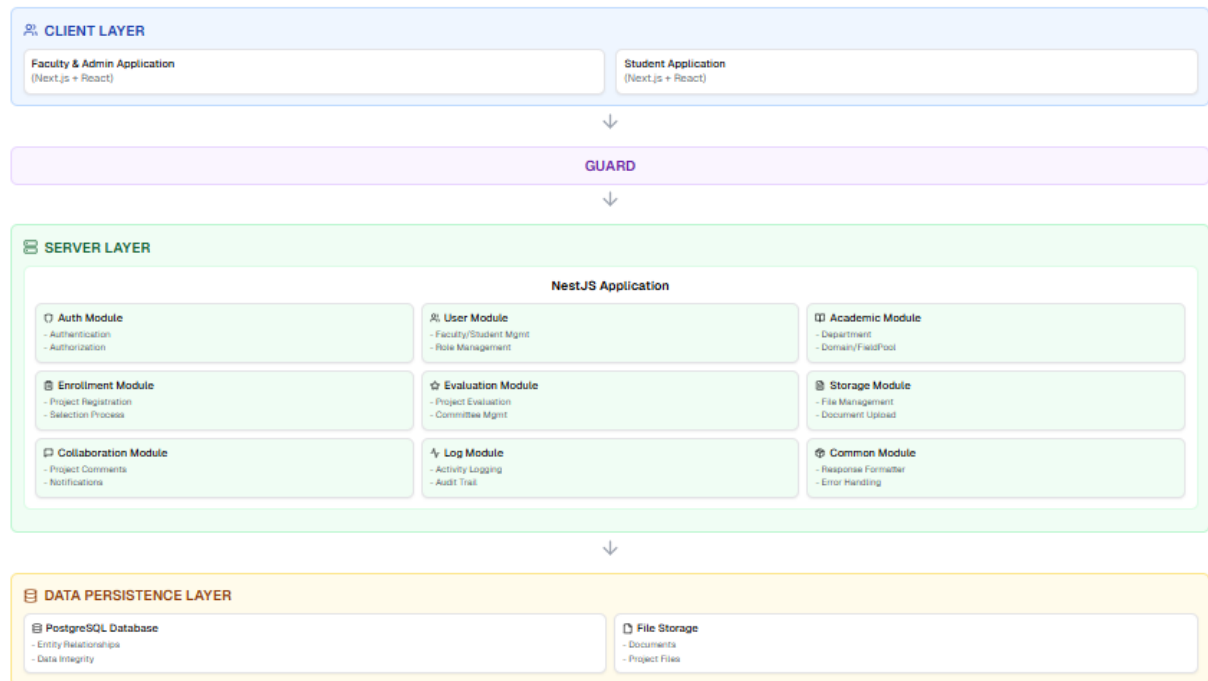


Figure 2.3 System Architecture Diagram

The system is designed with two distinct frontend applications to serve different user groups: one dedicated to students and another tailored for faculty members. Each frontend provides a customized user interface and functionality aligned with the specific needs and workflows of its user base.

Both frontend applications interact with a single, centralized backend server responsible for managing the core business logic, data processing, and communication with the database. The backend handles key operations such as user authentication and authorization, project management workflows, evaluation processes, notification delivery, and file storage management.

This architectural separation enables a more focused and intuitive user experience for students and faculty, while the unified backend ensures consistency, easier maintenance, and scalability. The design supports future expansion and integration without affecting the user interfaces.

2.2.3 Sequence Diagram

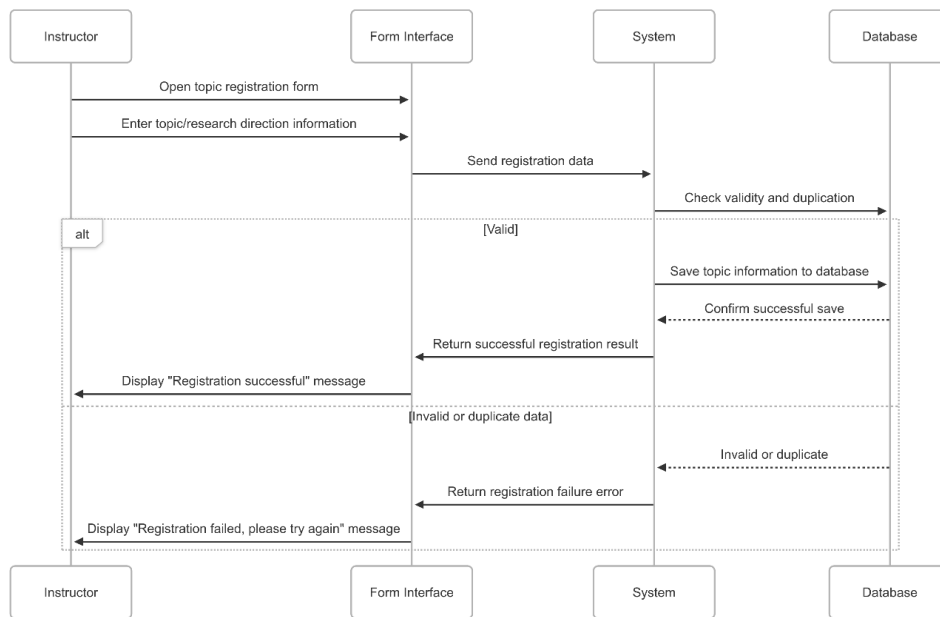


Figure 2.4 SD – Topic Registration Process

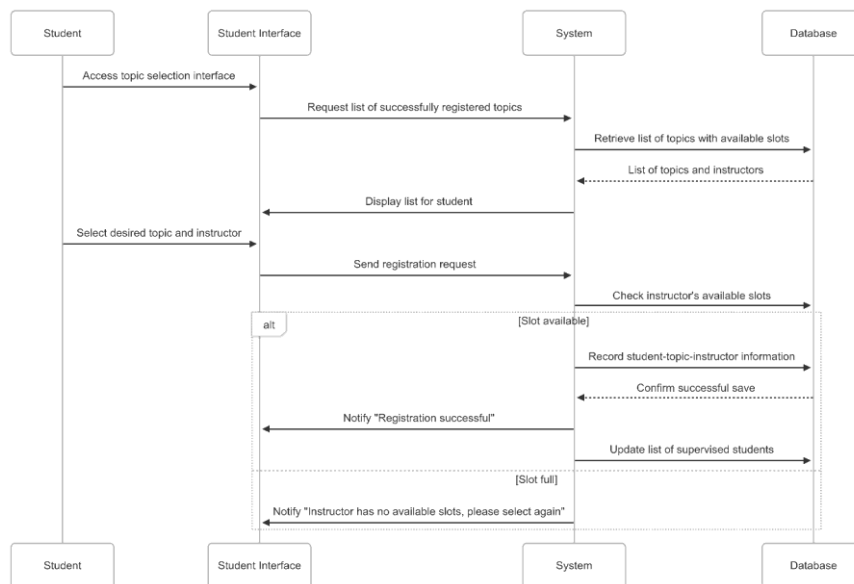


Figure 2.5 SD – Student Topic Selection

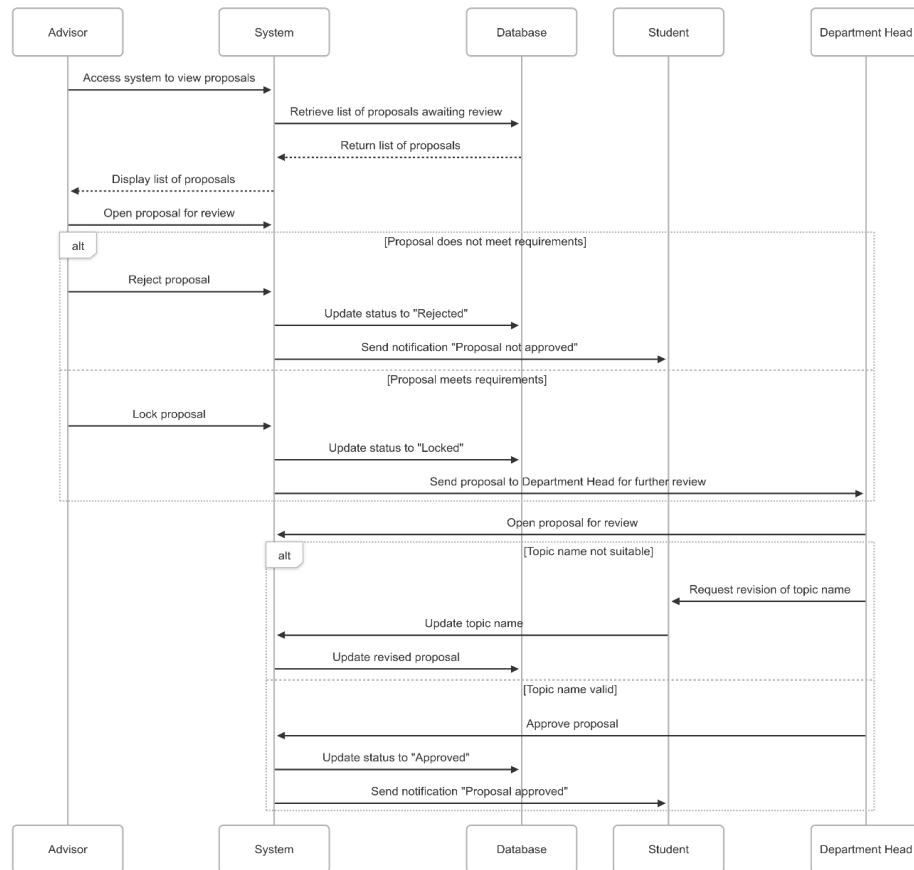


Figure 2.6 SD – Proposal Review and Approval Process

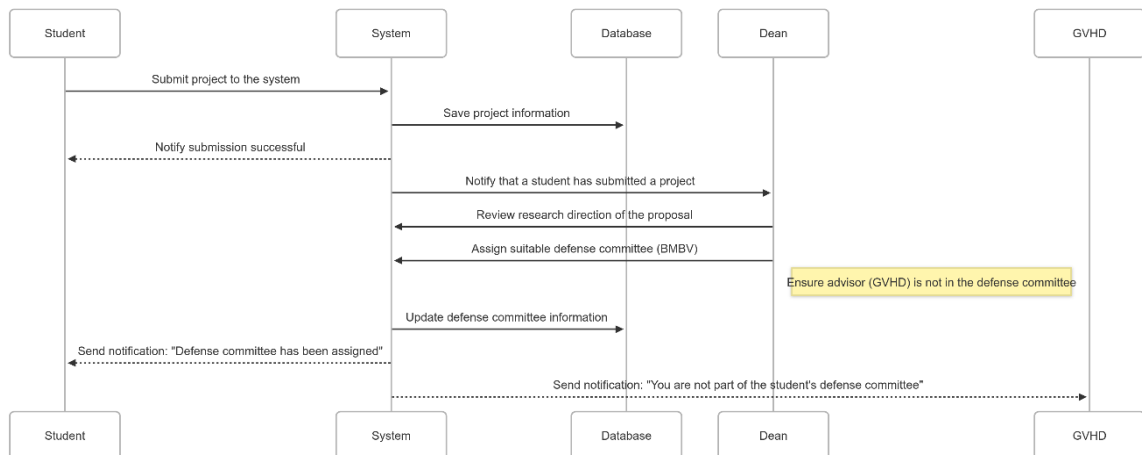


Figure 2.7 SD – Submission Process

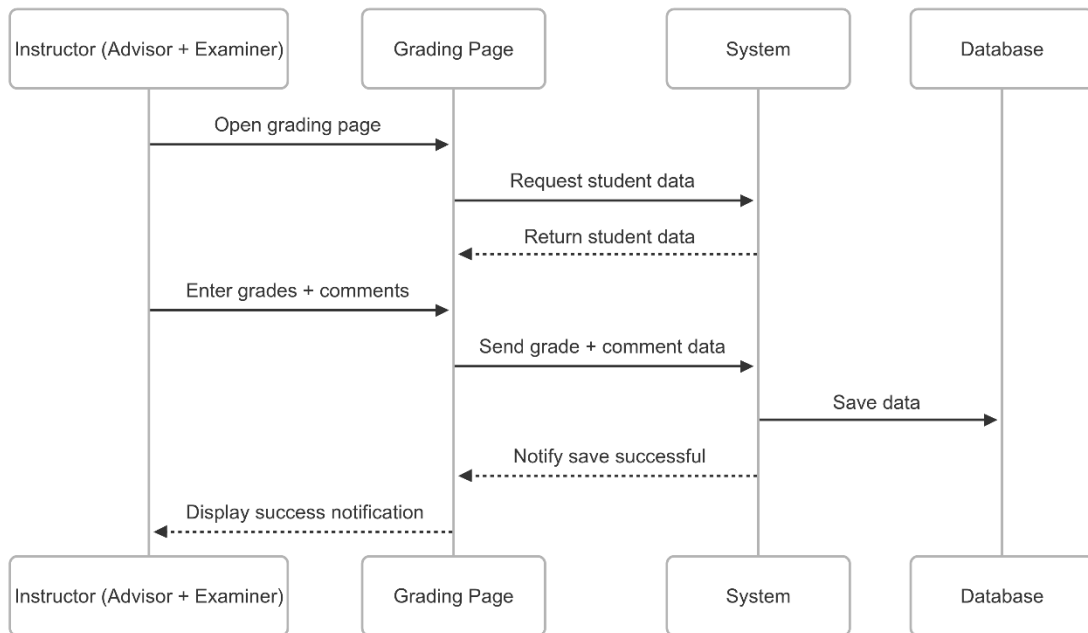


Figure 2.8 SD – Grading

2.3 Design database

2.3.1 Database design tables

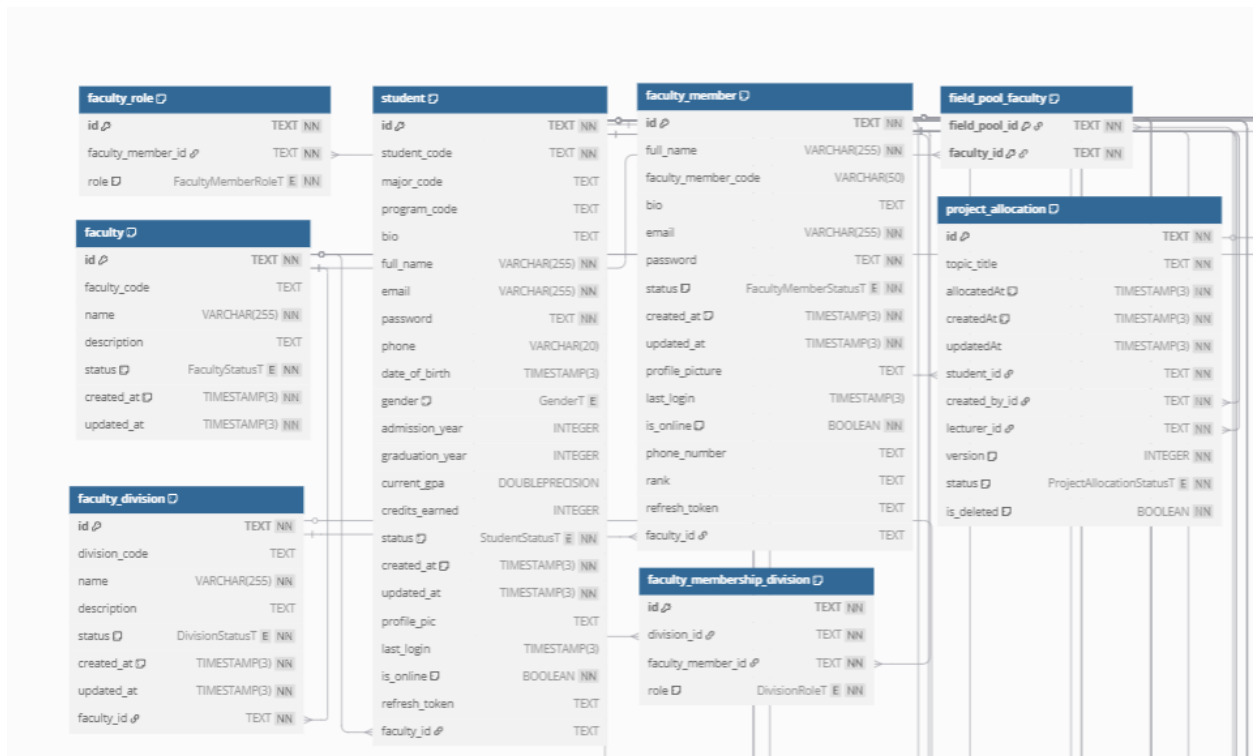


Figure 2.9 User management ERD

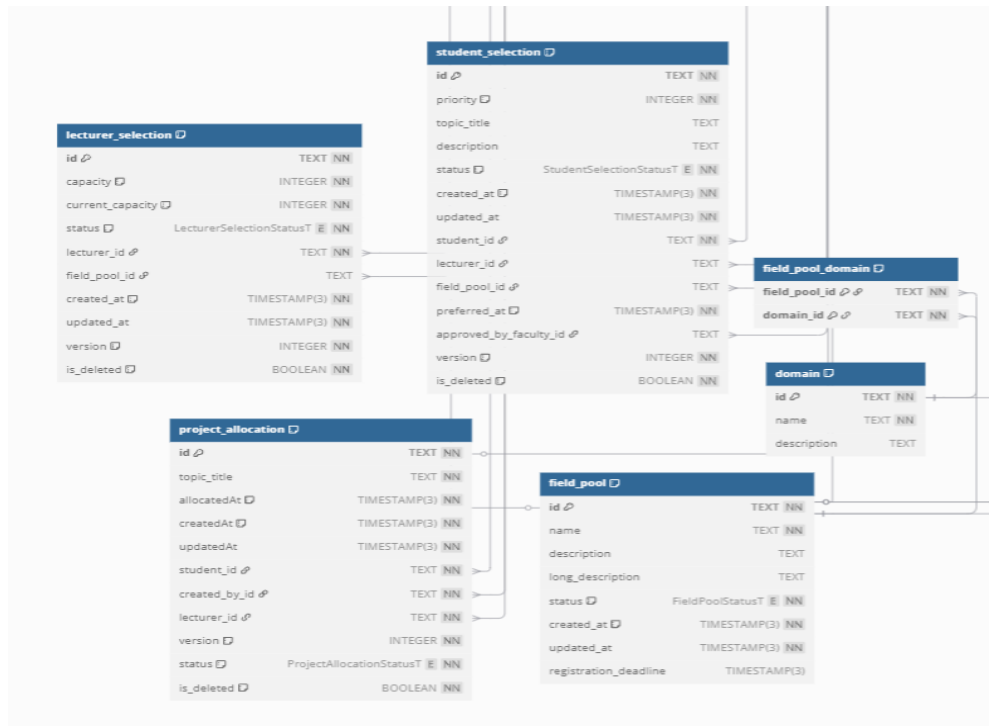


Figure 2.10 Selection management and supervisor assignment ERD



Figure 2.11 Result approval ERD

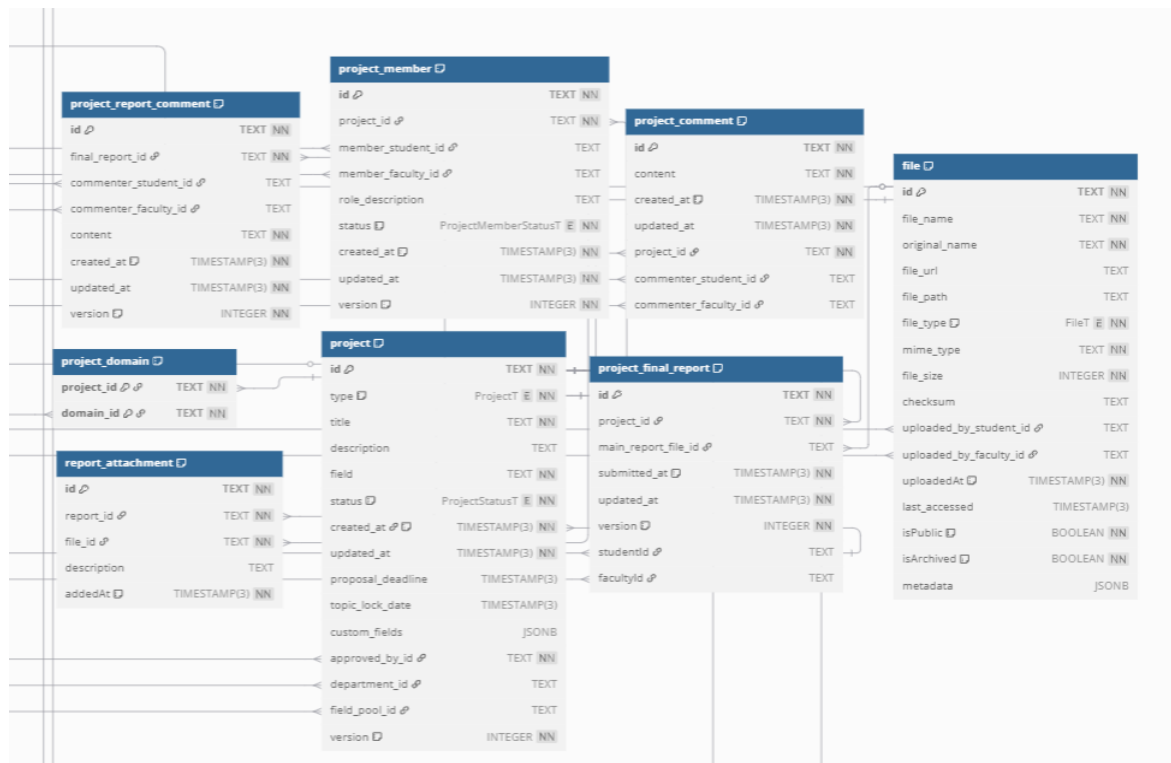


Figure 2.12 Research and result management ERD

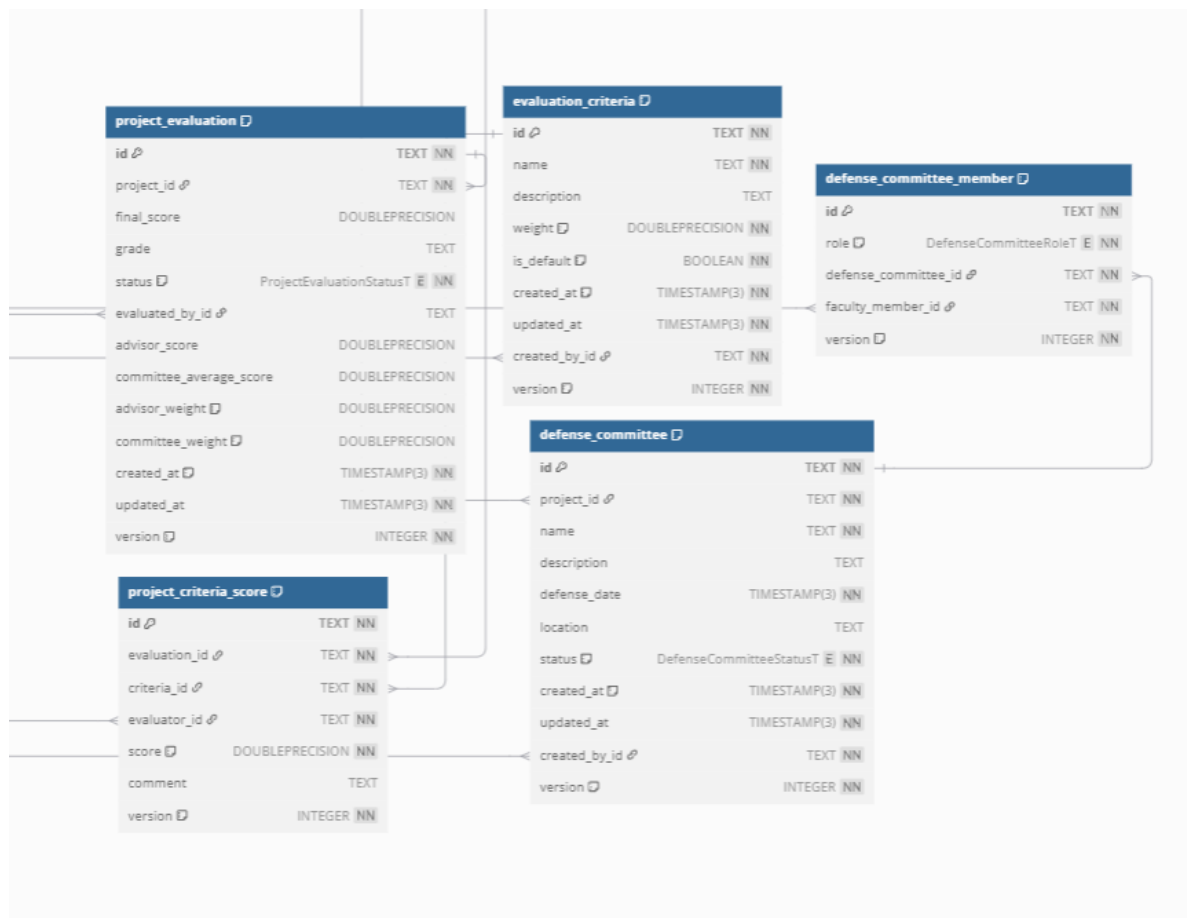


Figure 2.13 Defense and evaluation management ERD

2.3.2 Database description

a) User and Organizational Management

- **student table:** Stores personal and academic information of students, including student codes, full names, email addresses, academic program details, GPA, status, and security credentials for system access.
- **faculty_member table:** Contains data about faculty members, such as identification codes, full names, emails, professional ranks, status, and authentication details.
- **faculty table:** Represents faculties or schools, maintaining essential information and operational status.
- **faculty_division table:** Represents departments or divisions within faculties, including their codes, names, descriptions, and statuses.
- **faculty_role and faculty_membership_division tables:** Manage faculty members' roles within faculties (e.g., Admin, Dean, Lecturer) and departmental memberships with corresponding roles (e.g., Lecturer, Head).

b) Research Fields and Domains

- **field_pool table:** Defines research fields open for student registration to undertake projects.
- **domain table:** Catalogs specialized research domains linked to research fields via the field_pool_domain association table.
- **field_pool_faculty table:** Maps faculties to specific research fields to define their scope of participation.

- Topic Registration and Selection

- **lecturer_selection table:** Records faculty members' declarations of capacity to supervise projects, including maximum capacity and approval status.
- **student_selection table:** Captures students' project topic or supervisor preferences, including priority, status, and approval details.

c) Project and Proposal Management

- **project_allocation table:** Tracks project topic allocations to students and supervisors.
- **proposed_project table:** Manages project topic proposals derived from allocations, tracking detailed workflow statuses from submission to final approval.
- **proposed_project_member table:** Records members involved in a proposed project, including both students and faculty with assigned roles and active status.
- **proposal_outline table:** Stores detailed research proposal outlines, including introduction, objectives, methodology, and approval statuses.
- **project table:** Represents actual ongoing or completed research projects, categorizing project types and tracking statuses, associated fields, and organizational units.

d) Reporting and Feedback

- **project_final_report table:** Stores final project reports linked to projects, including submission timestamps and associated files.
- **report_attachment table:** Manages supporting documents attached to final reports.

- **project_report_comment and proposed_project_comment tables:** Enable commenting and feedback from students and faculty on reports and project proposals.
- e) **Defense Committee and Evaluation**
 - **defense_committee and defense_committee_member tables:** Manage committees responsible for project defenses, including members' roles such as chairperson, secretary, and reviewers.
 - **project_evaluation and project_evaluation_score tables:** Record overall project evaluations and individual scores provided by evaluators in different roles (advisors and committee members).
- f) **File and Security Management**
 - **file table:** Manages all uploaded documents and media files, including metadata such as file type, size, uploader identity, access rights, and archival status.

2.3.3 Conclusion

The database uses ENUM types to standardize statuses and roles, ensuring consistent data. Foreign keys maintain strong relationships between tables. Versioning helps track data changes, and soft deletion preserves historical data without removing it. Security features protect sensitive information and control user access.

To facilitate easier management and future feature expansions, projects that are not yet approved and those that have been approved are separated into two distinct tables. This separation allows for clearer workflows and simpler implementation of new business rules.

Overall, this design covers the full academic research process, enabling smooth collaboration and management between students and faculty. This structure provides a reliable, scalable foundation for a university research project management system.

CHAPTER 3 IMPLEMENTATION AND TESTING

3.1 Environment, Tools, and Directory Structure

Development Tools:

- **Code Editor: Visual Studio Code**, a versatile and efficient code editor, was used for writing and debugging code.
- **Version Control: GitLab** was utilized for version control and collaboration, enabling team members to manage code repositories and track changes effectively throughout the project.
- **Database Management: PGAdmin 4** was used to manage the PostgreSQL database, allowing for easy administration, query execution, and data visualization.
- **API Testing: Swagger** was used for testing and documenting the APIs, enabling easy interaction with the API endpoints and ensuring that they function as expected.

3.2 Main Modules Overview

3.2.1 Authentication

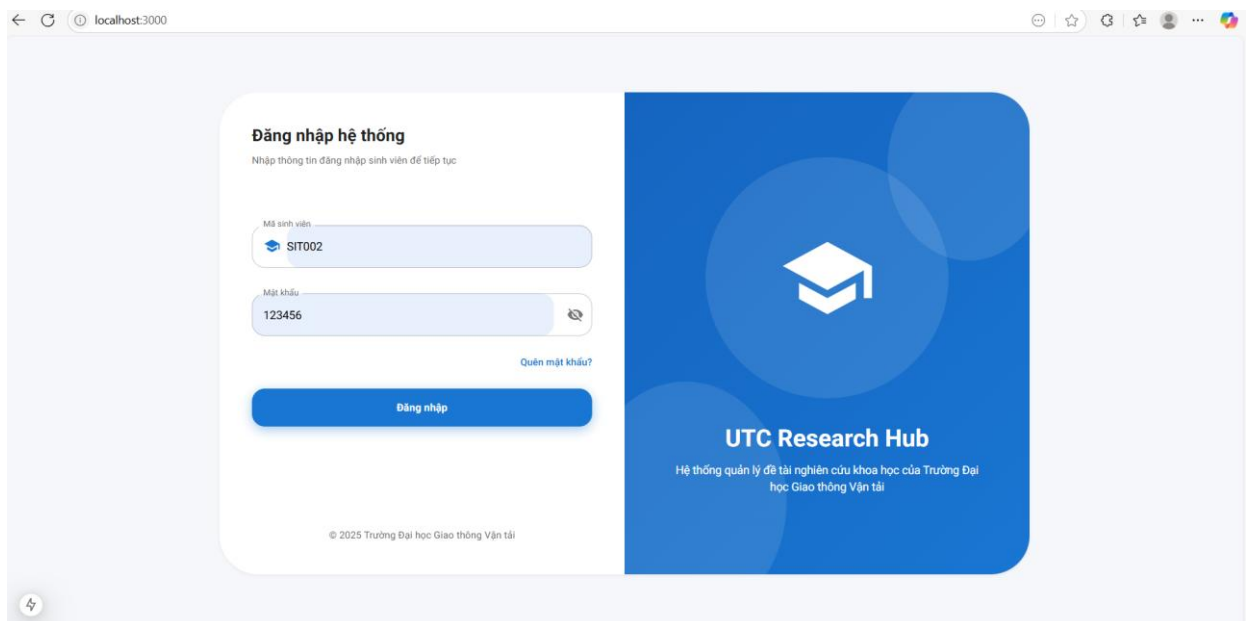


Figure 3.1 Login Page Student

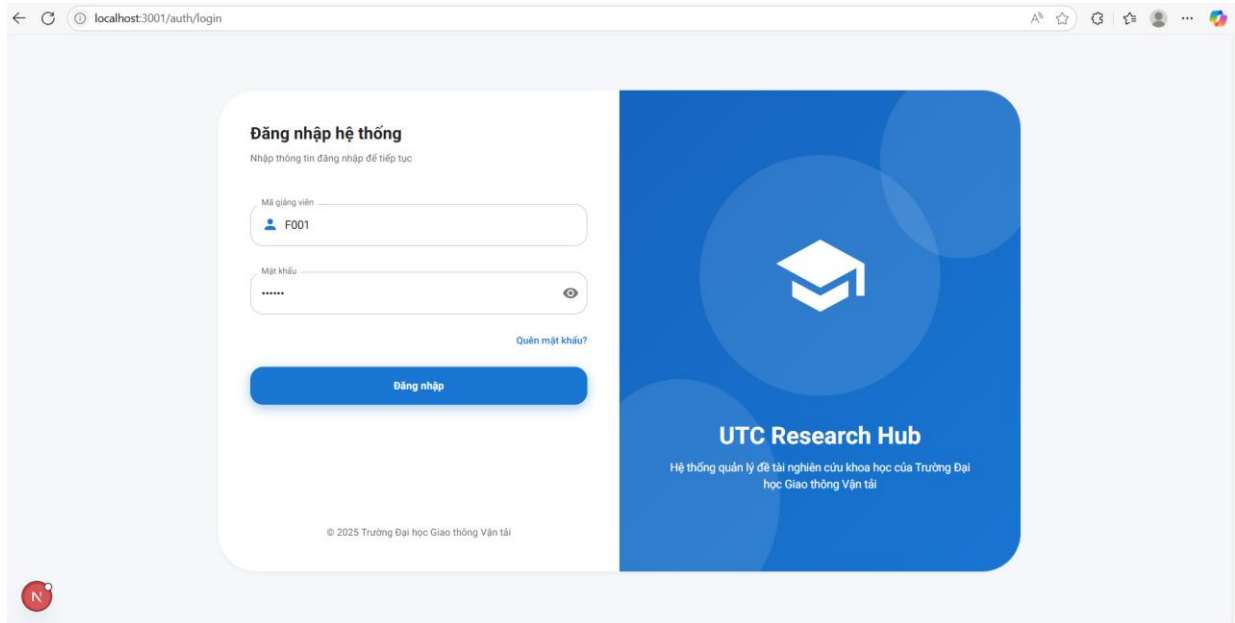


Figure 3.2 Login Page Faculty

Authentication		
GET	/auth/generate-password/{password}	Generate hashed password for testing
POST	/auth/login	Universal login for students and faculty
POST	/auth/refresh	Refresh access token
POST	/auth/logout	Logout the user

Figure 3.3 Login API

3.2.2 Selection

Student Selection				^
POST	/student-selection	[Student] Create a new selection preference	🔒	▼
GET	/student-selection	[All Roles] Find student selections (filtered by role)	🔒	▼
GET	/student-selection/me	[Student] Get my selection preferences	🔒	▼
GET	/student-selection/{id}	[Student, Lecturer, TBM, Dean] Get a student selection by ID	🔒	▼
PATCH	/student-selection/{id}	[Student] Update own selection preference	🔒	▼
DELETE	/student-selection/{id}	[Student] Delete own selection preference	🔒	▼
PATCH	/student-selection/{id}/status/admin	[TBM, Dean] Update student selection status (Admin)	🔒	▼
Lecturer Selections				^
POST	/lecturer-selections	Create a new lecturer selection	🔒	▼
GET	/lecturer-selections	Find all lecturer selections	🔒	▼
GET	/lecturer-selections/{id}	Find one lecturer selection	🔒	▼
PATCH	/lecturer-selections/{id}	Update a lecturer selection	🔒	▼
PATCH	/lecturer-selections/{id}/status	Update selection status by dean	🔒	▼
DELETE	/lecturer-selections/{id}/owner	Soft delete by owner	🔒	▼
DELETE	/lecturer-selections/{id}/dean	Soft delete by dean/admin	🔒	▼
Project Allocation				^
POST	/project-allocations	[TBM, Dean] Create a new project allocation	🔒	▼
GET	/project-allocations	[All Roles] Find project allocations (filtered by role)	🔒	▼
GET	/project-allocations/student/{studentId}	[TBM, Dean, Lecturer] Get allocations by student ID with pagination and search	🔒	▼
GET	/project-allocations/lecturer/{lecturerId}	[TBM, Dean, Lecturer] Get allocations by lecturer ID with pagination and search	🔒	▼
GET	/project-allocations/recommendations	[TBM, Dean] Get recommended allocations as Excel or JSON	🔒	▼
GET	/project-allocations/{id}	[All Roles] Get a project allocation by ID	🔒	▼
PATCH	/project-allocations/{id}	Update project allocation	🔒	▼
DELETE	/project-allocations/{id}	Delete project allocation	🔒	▼
PATCH	/project-allocations/{id}/status	Update project allocation status	🔒	▼
PATCH	/project-allocations/bulk/status	Bulk update project allocation status	🔒	▼
POST	/project-allocations/upload	[TBM, Dean] Import project allocations from Excel	🔒	▼

Figure 3.4 Selection API

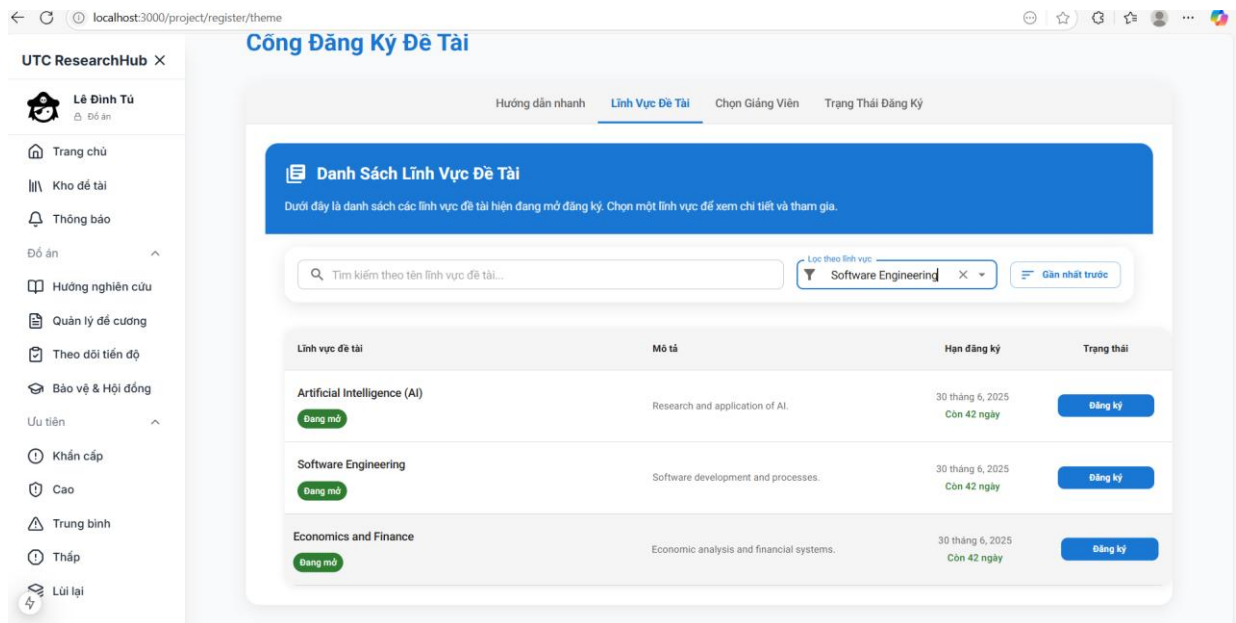


Figure 3.5 Field Pool Selection Page

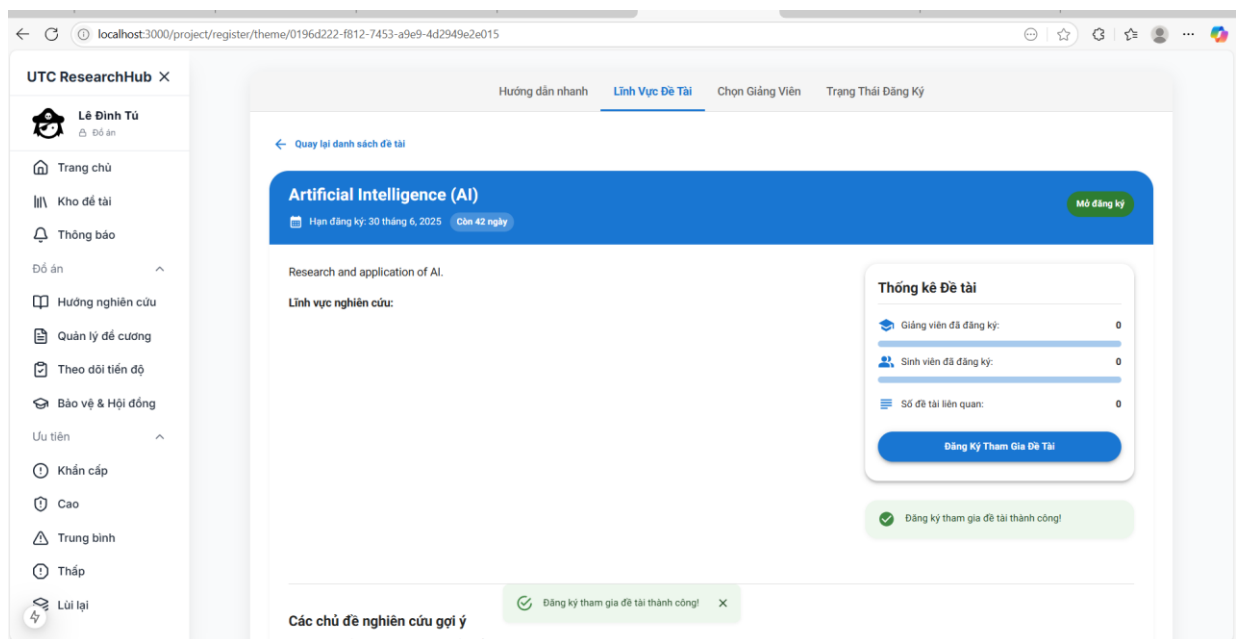


Figure 3.6 Student Selection Page

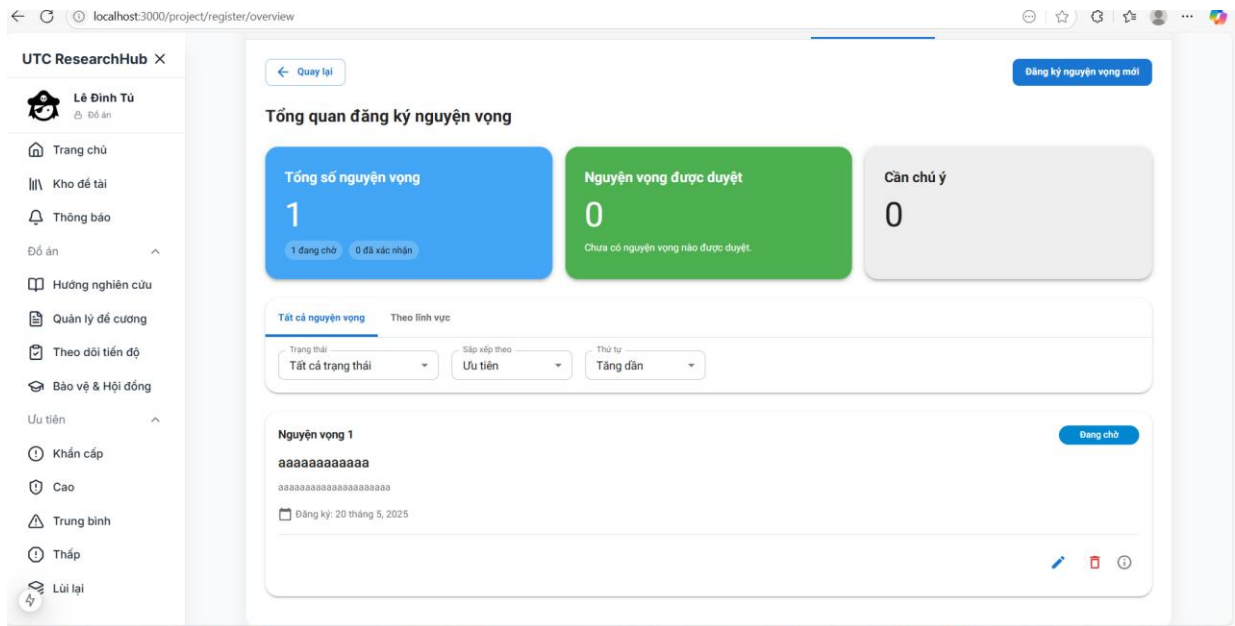


Figure 3.7 Selection Student Page

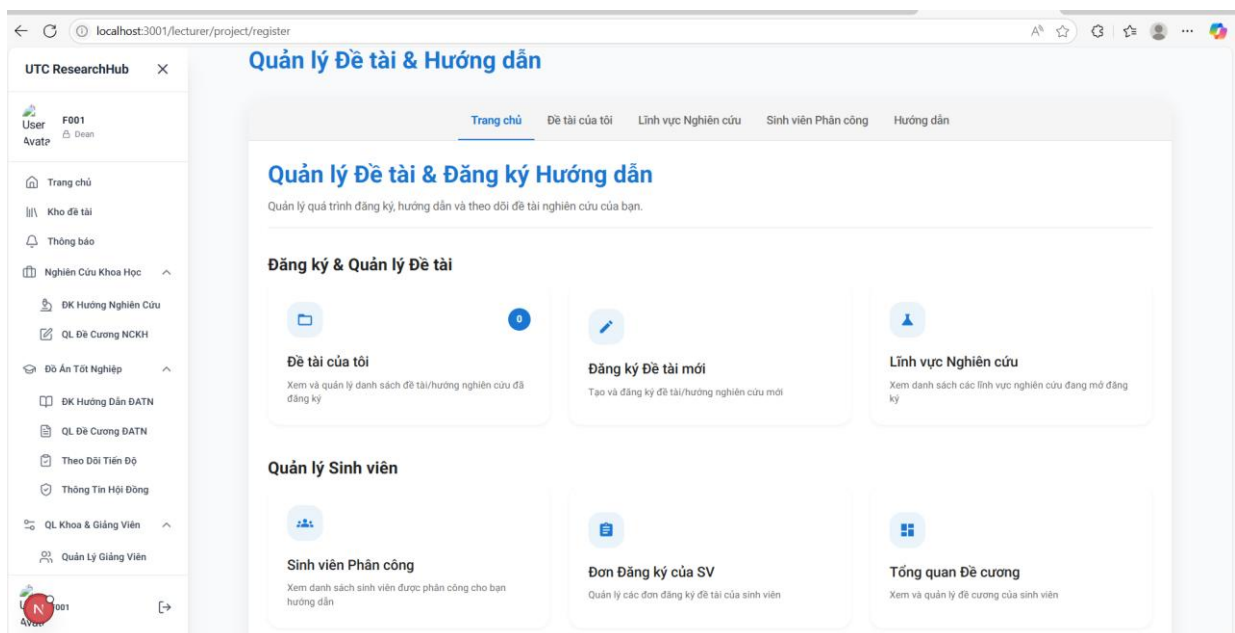


Figure 3.8 Teacher Selection Page

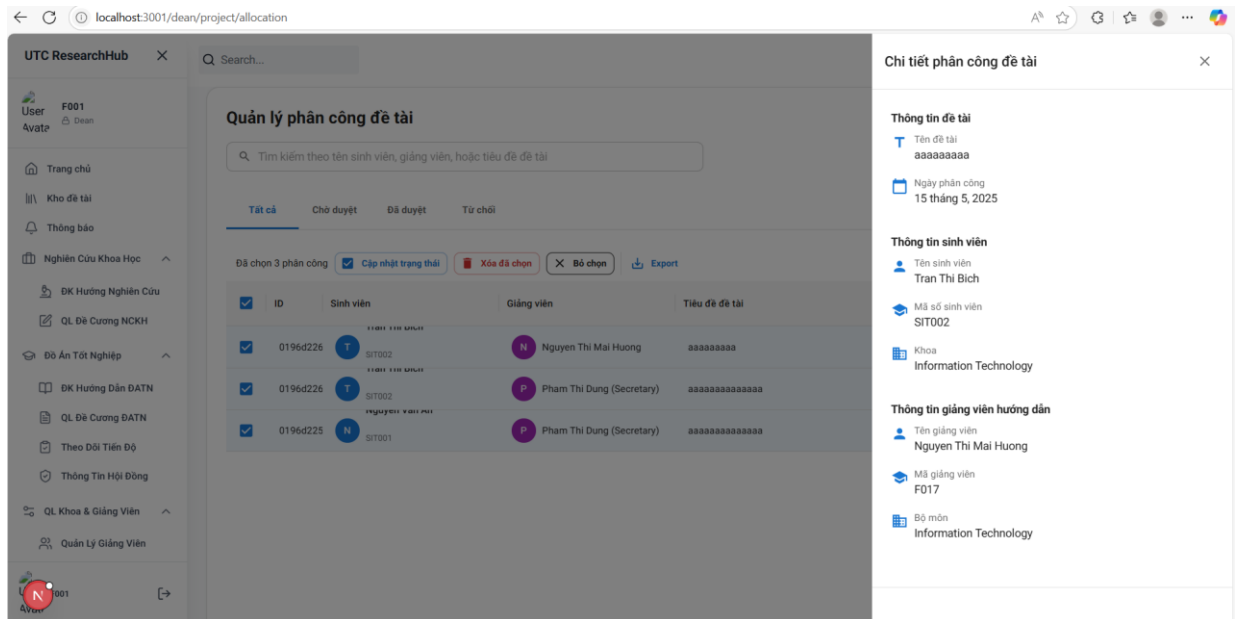


Figure 3.9 Selection Management Page

3.2.3 Proposal

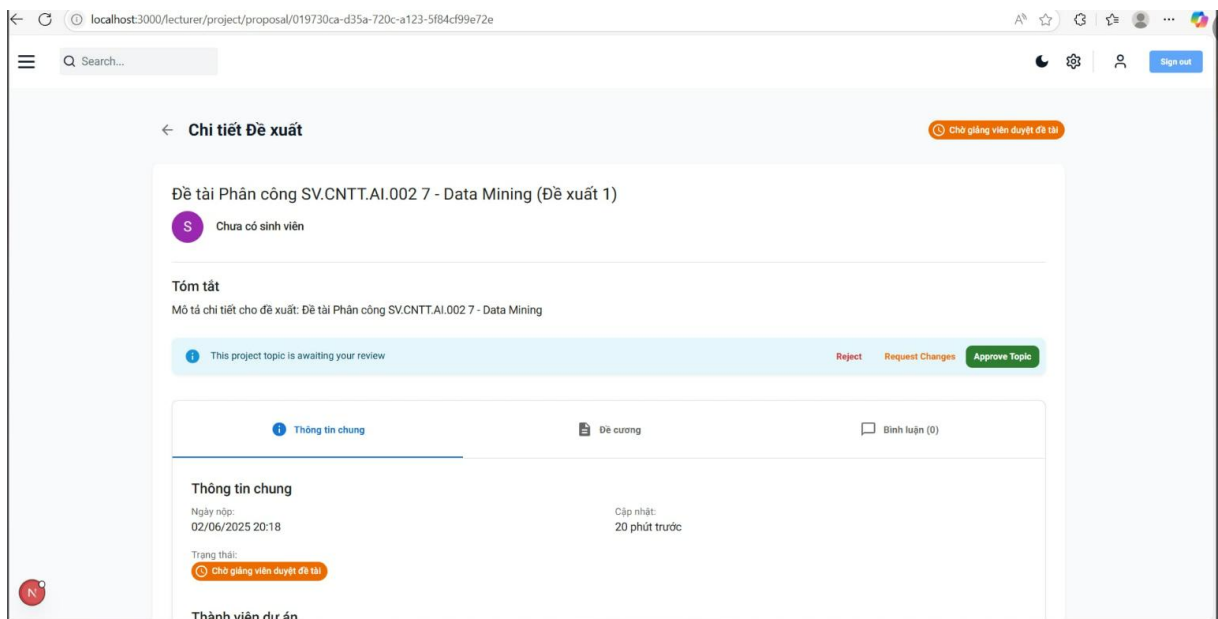


Figure 3.10 Proposal Page for Lecturer

Proposed Projects			^
POST	/proposed-projects/trigger	Initialize project proposal from allocation record	🔒
PUT	/proposed-projects/{id}	Update project proposal information	🔒
GET	/proposed-projects/{id}	Get details of a project proposal	🔒
PUT	/proposed-projects/{id}/advisor-review	Advisor reviews project title	🔒
PUT	/proposed-projects/{id}/head-review	Department head reviews project proposal	🔒
PUT	/proposed-projects/{id}/final-approval	Faculty head gives final approval and creates official project	🔒
PUT	/proposed-projects/{id}/members	Manage members in project proposal	🔒
POST	/proposed-projects/outline	Student submits detailed proposal outline	🔒
PUT	/proposed-projects/outline/{id}/review	Advisor/Department Head reviews proposal outline	🔒
PUT	/proposed-projects/outline/{id}/lock	Faculty Head locks proposal outline	🔒
GET	/proposed-projects	Search for project proposals	🔒
PUT	/proposed-projects/{id}/status	Update project proposal status	🔒
PUT	/proposed-projects/{id}/title	Update proposed project title	🔒
GET	/proposed-projects/export-excel	Export projects to Excel file	🔒
POST	/proposed-projects/bulk-status-update/lecturer	Update status of multiple project proposals as a lecturer	🔒
POST	/proposed-projects/bulk-status-update/department-head	Update status of multiple project proposals as a department head	🔒
POST	/proposed-projects/bulk-status-update/dean	Update status of multiple project proposals as a dean	🔒

Figure 3.11 Proposal API

UTC ResearchHub X

Search...

Đăng xuất

Quay lại Chi tiết đề cương

SV1 Project (REJECTED_BY_HEAD) - Đề tài Phân công Đặc Biệt cho 11 REJECTED_BY_HEAD

Cập nhật: khoảng 1 giờ trước

Đề tài dự kiến Hộp đề cương Xem đề cương Thành viên 2 Bình luận

Thành viên nhóm nghiên cứu (2)

S Sinh viên Web/App 1
Sinh viên • Mã SV: SV.CNTT.WA.001 Sinh viên

K Không xác định
Giảng viên hướng dẫn • Mã SV: N/A Sinh viên

Trang chủ
Kho đề tài
Thông báo
Đồ án
Hướng nghiên cứu
Quản lý đề cương
Theo dõi tiến độ
Bảo vệ & Hội đồng
Ưu tiên
Khẩn cấp
Cao
Trung bình
Thấp
Lùi lại

Figure 3.12 Proposal Page for Student

3.2.4 Result

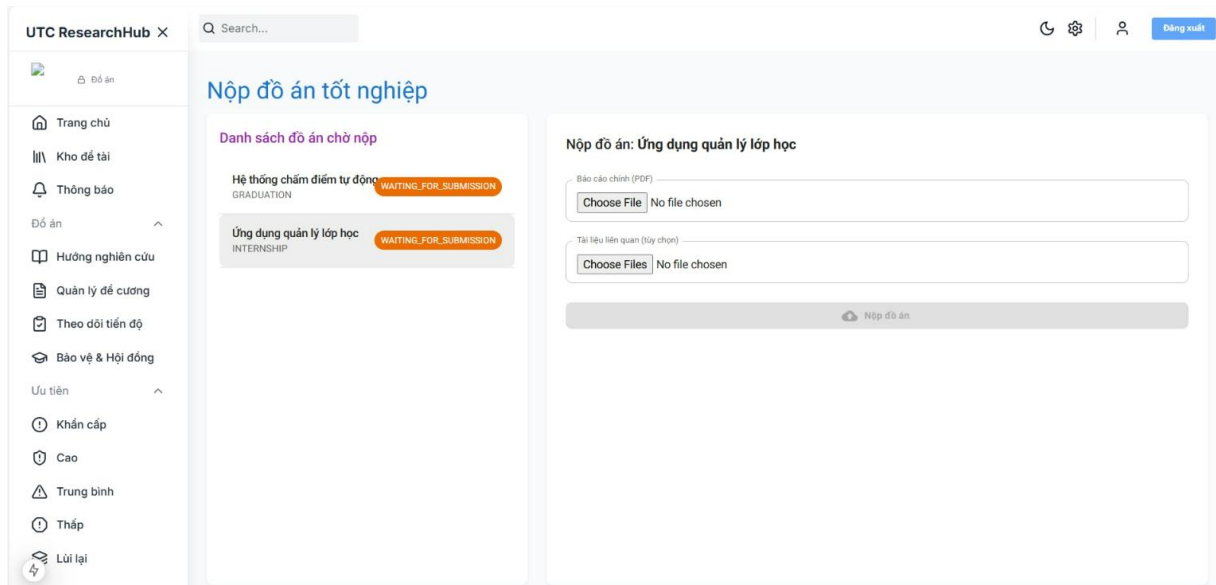


Figure 3.13 Submit Project Page

3.2.5 Evaluation

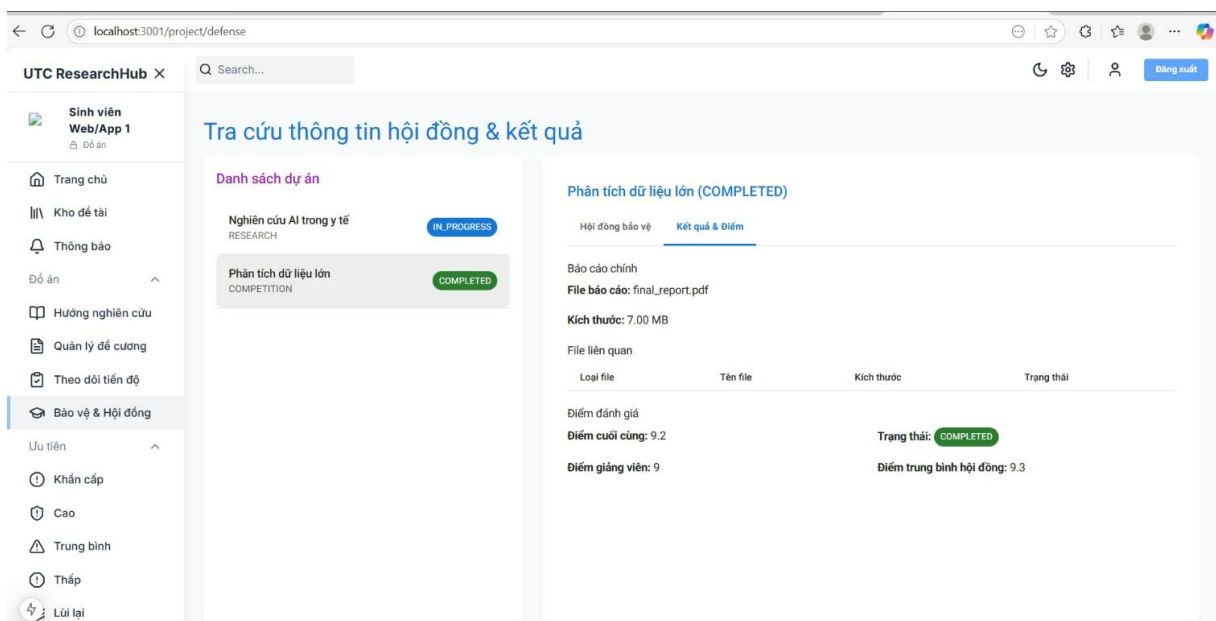


Figure 3.14 Evaluation Page for Student

Evaluation		
GET	/evaluation/projects-to-evaluate	Lấy danh sách dự án cần đánh giá của giảng viên
GET	/evaluation	Lấy danh sách đánh giá dự án
POST	/evaluation	Tạo đánh giá mới cho dự án
GET	/evaluation/{id}	Lấy chi tiết đánh giá dự án theo ID
PUT	/evaluation/{id}	Cập nhật đánh giá dự án
GET	/evaluation/project/{projectId}	Lấy đánh giá dự án theo ID dự án
PUT	/evaluation/{id}/finalize	Chốt điểm cuối cùng cho dự án
GET	/evaluation/{evaluationId}/scores	Lấy danh sách điểm đánh giá theo ID đánh giá
POST	/evaluation/scores	Chấm điểm đánh giá dự án
PUT	/evaluation/scores/{id}	Cập nhật điểm đánh giá dự án
DELETE	/evaluation/scores/{id}	Xóa điểm đánh giá dự án
POST	/evaluation/advisor-score	Giáo viên hướng dẫn chấm điểm dự án
POST	/evaluation/committee-score	Thành viên hội đồng chấm điểm dự án
GET	/evaluation/by-defense-committee/{defenseCommitteeId}	Lấy đánh giá dự án theo ID hội đồng bảo vệ
Defense Committee		
POST	/defense-committees	[Dean] Create a new defense committee
GET	/defense-committees	[All Roles] Find defense committees (filtered by role)
POST	/defense-committees/bulk	[Dean] Create multiple defense committees from waiting projects
POST	/defense-committees/{id}/members	[Dean] Add a member to defense committee
DELETE	/defense-committees/{id}/members/{memberId}	[Dean] Remove a member from defense committee
GET	/defense-committees/waiting-projects	[Dean] Get projects waiting for evaluation (WAITING_FOR_EVALUATION)
GET	/defense-committees/{id}	[All Roles] Get a defense committee by ID
PATCH	/defense-committees/{id}	[Dean] Update defense committee
DELETE	/defense-committees/{id}	[Dean] Delete a defense committee

Figure 3.15 Evaluation API

Duyệt và phê duyệt điểm cuối cùng cho các đồ án của sinh viên

STT	Thông tin đồ án	Điểm GVHD	Điểm hội đồng	Điểm cuối	Trạng thái	Thao tác
1	Dự án 1: Ứng dụng AI trong quản lý chi tiêu Sinh viên: Sinh viên 1 (SV100) GVHD: GV Nguyễn Văn A	7.64	9.45	Chưa phê duyệt	Chờ chấm điểm	Phê duyệt
2	Dự án 2: Phát triển website thương mại điện tử Sinh viên: Sinh viên 2 (SV101) GVHD: GV Trần Thị B	8.23	8.01	Chưa phê duyệt	Sẽ chấm điểm	Phê duyệt
3	Dự án 3: Hệ thống quản lý học sinh Sinh viên: Sinh viên 3 (SV102) GVHD: GV Lê Văn C	7.60	7.26	7.40	Sẽ phê duyệt	✓
4	Dự án 4: Phân tích dữ liệu bán hàng Sinh viên: Sinh viên 4 (SV103) GVHD: GV Nguyễn Văn A	9.11	7.94	Chưa phê duyệt	Chờ chấm điểm	Phê duyệt
5	Dự án 5: Ứng dụng IoT trong nông nghiệp Sinh viên: Sinh viên 5 (SV104) GVHD: GV Trần Thị B	8.01	7.37	Chưa phê duyệt	Sẽ chấm điểm	Phê duyệt

Số hàng mỗi trang: 5 1-5 của 15

Figure 3.16 Evaluation Page for Head

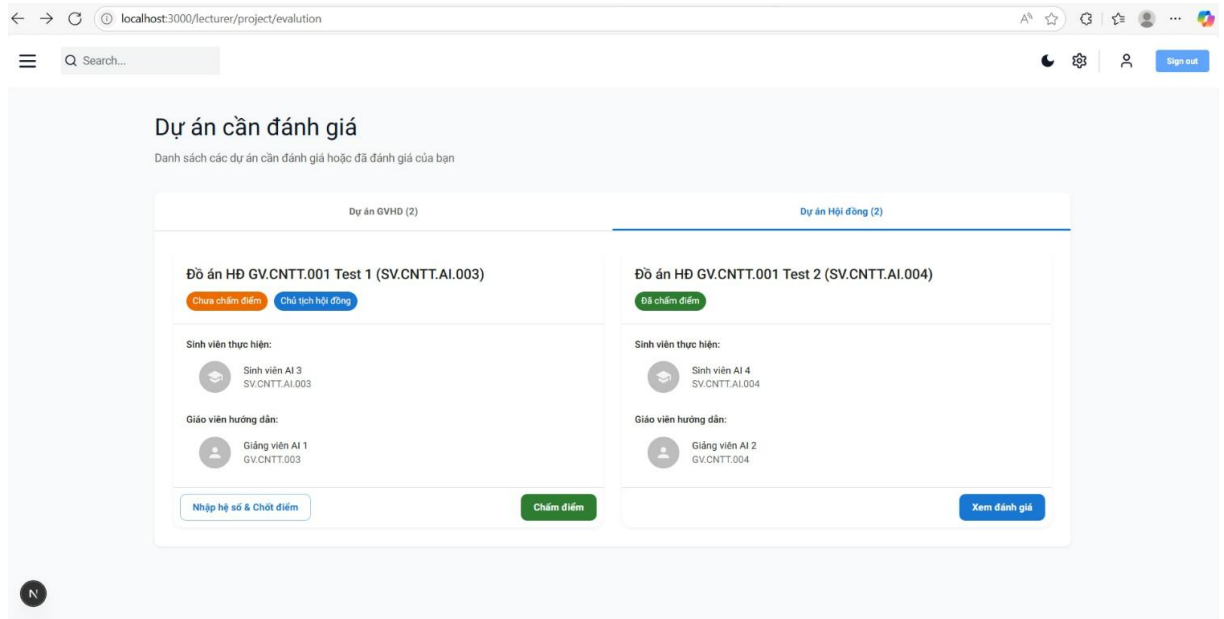


Figure 3.17 Evaluation Page for Defense Member

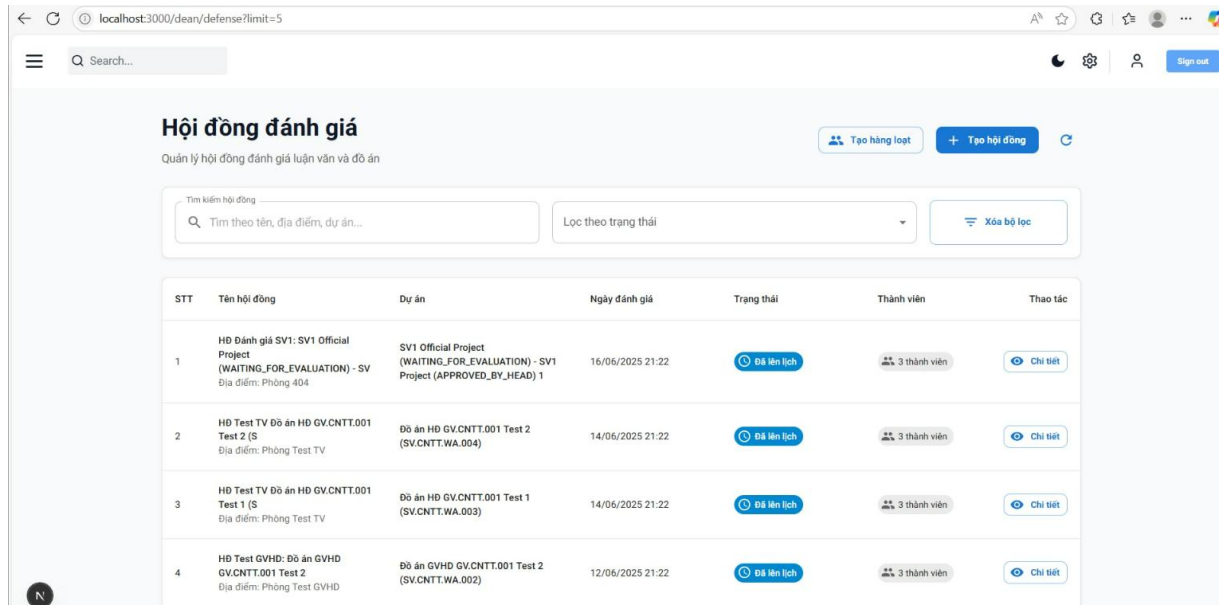


Figure 3.17 Management Defense Page

3.2.6 Dashboard

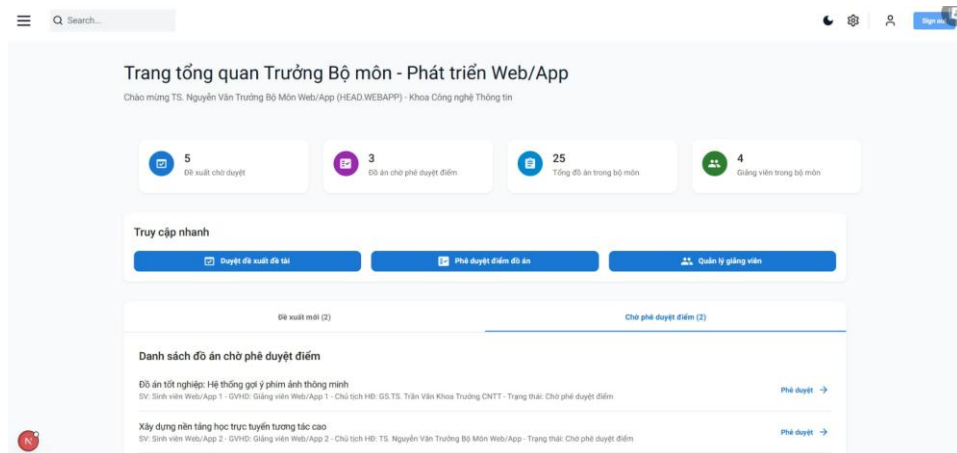


Figure 3.18 Department Head Dashboard

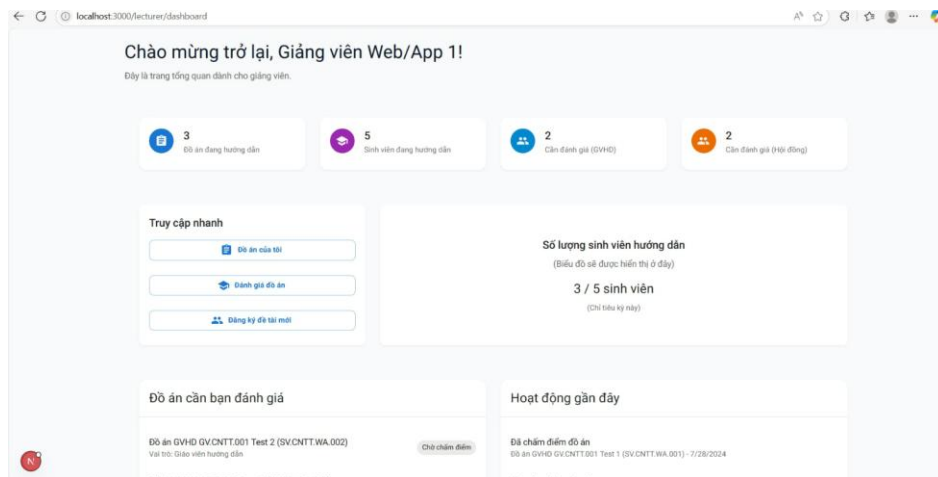


Figure 3.19 Lecturer Portal Dashboard

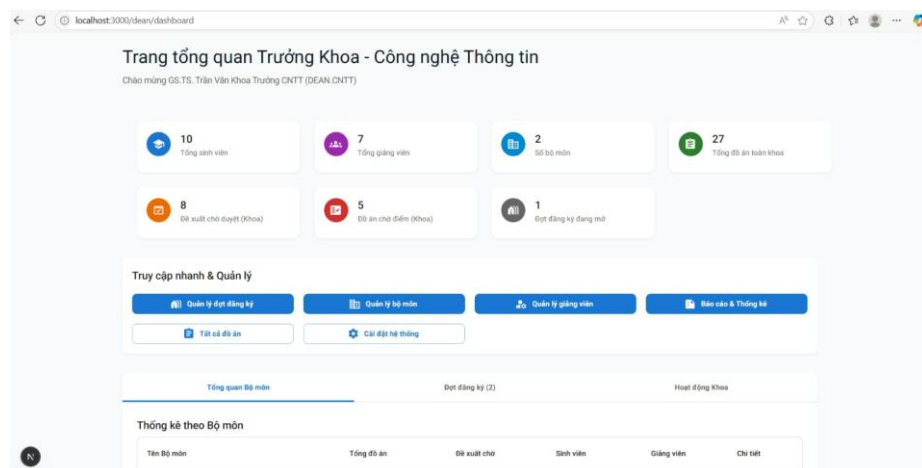


Figure 3.20 Dean Portal Dashboard

CONCLUSION

This thesis management system, developed through this project, effectively addresses the critical needs involved in managing the thesis lifecycle for both students and faculty members in an academic environment. It supports a comprehensive range of core functionalities, including research direction registration, supervisor assignment, thesis outline management, feedback collection, grading, and defense committee coordination. By streamlining these processes, the system facilitates smooth, efficient workflows for topic registration, proposal submission and approval, assignment of supervisors and reviewers, and defense scheduling and management.

A key achievement in the project is the full-stack integration of TypeScript across both the frontend and backend layers. Utilizing NextJS for the user interface and NestJS for the server-side logic has significantly enhanced code quality, maintainability, and scalability. This technological choice reduces development time, minimizes potential errors, and lays a solid foundation for future expansion and feature additions. On the data management front, the employment of PostgreSQL as the relational database and Prisma as the ORM tool has ensured reliable data persistence and seamless database interactions. These technologies provide a robust backbone for complex data relationships inherent in academic project management.

Extensive testing using tools like Swagger and Postman has verified the stability, accuracy, and reliability of the system's key features. Functional testing across core workflows confirmed that the system meets design expectations and operates correctly under various scenarios.

However, the project also encountered several challenges and limitations. Time constraints restricted the scope of development, preventing the inclusion of advanced features such as real-time notifications and intelligent recommendation engines. Moreover, user adoption proved to be a gradual process, as many users were accustomed to traditional manual methods and required time to adjust to the new digital workflows. Additionally, although the system is currently capable of supporting the existing user base, further optimization is necessary to ensure scalability and maintain performance as the number of users and volume of data increase over time.

Looking forward, there are several promising directions for future development that could substantially enhance the system's functionality and user experience. Integrating artificial intelligence could revolutionize the way users interact with the system—for example, by providing automatic topic suggestions tailored to individual students' interests and academic backgrounds, evaluating thesis proposals to assist faculty in decision-making, and offering personalized recommendations throughout the research process. Implementing real-time notifications would significantly improve communication among students, supervisors, and administrative staff by alerting users promptly about outline approvals, feedback submissions, defense scheduling, and other critical events.

To expand accessibility, developing a mobile application would empower users to manage their tasks, submit documents, and track progress anytime and anywhere, accommodating the modern need for flexibility and remote access. Additionally, incorporating advanced reporting and analytics capabilities would provide

administrators with valuable insights into student progress, faculty workload, research trends, and performance metrics—supporting data-driven decision-making and continuous improvement in academic program management.

In summary, the thesis management system developed in this project successfully delivers a structured, automated, and collaborative platform that addresses essential academic workflows. By improving efficiency and streamlining communication between students and faculty, it enhances the overall research experience. The technological stack chosen establishes a scalable and maintainable framework ready for future innovations.

Continued development, including the integration of AI technologies, real-time communication features, mobile support, and enhanced analytics, combined with ongoing user feedback and iterative testing, will ensure that the system evolves to meet the dynamic and growing demands of academic institutions. This project lays a strong foundation upon which future enhancements can build to create a comprehensive, user-friendly, and intelligent research management environment.

REFERENCES

- [1] NextJS Documentation. (n.d.). Retrieved April 2025, from <https://nextjs.org/docs>
- [2] NestJS Documentation. (n.d.). Retrieved April 2025, from <https://docs.nestjs.com/>
- [3] Prisma Documentation. (n.d.). Retrieved April 2025, from <https://www.prisma.io/docs/>
- [4] PostgreSQL Global Development Group. (2023). PostgreSQL Documentation. Retrieved April 2025, from <https://www.postgresql.org/docs/>
- [5] Jsonwebtoken. (n.d.). jsonwebtoken. Retrieved April 2025, from <https://github.com/auth0/node-jsonwebtoken>
- [6] Zustand Documentation. (n.d.). Retrieved April 2025, from <https://docs.pmnd.rs/zustand/getting-started/introduction>
- [7] Material-UI. (n.d.). Material-UI: React components for faster and easier web development. Retrieved April 2025, from <https://mui.com/>
- [8] AppsDevPro. (n.d.). React vs NextJS: A Complete Comparison. Retrieved June 2025, from <https://www.appsdevpro.com/blog/react-vs-next-js/>
- [9] Gilad, Y. (n.d.). NestJS Request Lifecycle [Slide presentation]. Retrieved June 2025, from <https://slides.com/yariv-gilad/nest-js-request-lifecycle>
- [10] NearForm. (2023). Prisma ORM. Retrieved June 2025, from <https://nearform.com/digital-community/prisma-orm/>
- [11] Zod. (n.d.). Zod: TypeScript-first schema declaration and validation. Retrieved June 2025, from <https://github.com/colinhacks/zod>
- [12] ExcelJS. (n.d.). ExcelJS: Excel spreadsheet library for Node.js. Retrieved June 2025, from <https://github.com/exceljs/exceljs>
- [13] bcrypt. (n.d.). bcrypt: A library to help hash passwords. Retrieved June 2025, from <https://github.com/kelektiv/node.bcrypt.js>
- [14] UUID. (n.d.). uuid: Simple, fast generation of RFC4122 UUIDs. Retrieved June 2025, from <https://github.com/uuidjs/uuid>