Capstone Project Document

Group Members	
Supervisor	
Capstone Project code	

- Ho Chi Minh, May 2025 -

Table of Contents

2. Installation Guides
200

3. User Manual
202
172

Acknowledgement

Best regards,
4Bees.

Definition and Acronyms

Acronym	Definition
BA	Business Analysis
BR	Business Rule
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
PM	Project Manager
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirement Specification
UAT	User Acceptance Test

UC	Use Case
API	Application Programming Interface

I. Project Introduction

1. Overview

1.1 Project Information

- **Project name:** FPT Team Matching - Social networking for students project teams
- **Project code:** SP25SE107
- **Group name:** 4Bees
- **Software Type:** Web application

1.2 Project Team

Full Name	Role	Email	Mobile
	Supervisor		
	Leader		
	Member		
	Member		
	Member		

2. Product Background

3. Existing Systems

3.1 Call4Project

Description: The system allows students to join groups, register project topics, and choose their supervising lecturers.

Drawbacks:

- Team matching is not supported for student in system
- Do not have web for council to evaluate topic
- Do not have web for reviewer to review schedule

3.2 Dev.to

Description: Dev.to is a community platform for developers, where users can share articles, discuss, and connect with people who share similar interests in technology. The system provides features for posting, commenting, following other users, and creating groups (organizations) to work together.

Drawbacks:

- UI is not easy to use
- Weak Team Structure
- Not Academically Focused

4. Business Opportunity

By centralizing the management of thesis projects, the system enhances collaboration, improves communication between students and faculty, and optimizes the workflow for both students and lecturers.

5. Software Product Vision

6. Project Scope & Limitations

6.1 Major Features

6.1.1 Major Features for Admin

FE-01: Login/Logout

FE-02: Manage user accounts

6.1.2 Major Features for Manager

FE-01: Login/Logout

FE-02: Manage timeline of semester

FE-03: Manage review and capstone schedule

FE-04: Manage criteria form

FE-05: Evaluate the team's request about updating topics after review 1 and 2 after mentor approval.

FE-06: Manage notification

FE-07: Import student and lecturer

6.1.3 Major Features for Lecturer

FE-01: Login/Logout

FE-02: Mentors can propose an idea.

FE-03: Mentors can respond to a student's request for a topic.

FE-04: Mentors review student's thesis ideas by criteria form before they are submitted to the council.

FE-05: Reviewers can view their review schedule.

FE-06: A council consisting of a number of lecturers evaluates a thesis proposal by criteria form.

FE-07: Provide feedback for team after review 3

FE-08: Communicate with other students or lecturers by chat.

FE-09: Mentors evaluate the team's request about updating topics after review 1 and 2.

6.1.4 Major Features for Student

FE-01: Students create teams, including team details.

FE-02: Team leaders send invitations to members based on their profiles (skills, interests, experience)

FE-03: Team leaders or students can post blog member recruitment or find a team.

FE-04: Students can propose a new idea for approval or request an available mentor's thesis topic.

FE-05: Communicate with other students or lecturers by chat.

FE-06: Manage projects by team progress, rate contribution

6.2 Limitations & Exclusions

LI-1: Verify as FPT students – The current version of FPT Team Matching does not include data about FPTU students, which may prevent the system from accurately verifying whether a user is an FPT University student.

LI-2: LMS Integration: LMS Integration: Limited to basic LMS integration (cannot pull full student academic history).

LI-3: The system currently lacks built-in scheduling functionality for reviews and capstone project timelines.

II. Project Management Plan

1. Overview

1.1 Scope & Estimation

#	WBS Item	Complexity	Est. Effort (man-days)
1	***Initiating***		***20***
1.1	Define project scope	Medium	12
1.2	Collect requirements	Medium	8
2	***Planning***		***14***
2.1	Create kick-off meeting	Medium	7
2.2	Create plan document	Medium	7
3	***Executing***		
3.1	**Analysis**		***18***
3.1.1	Analysis requirements	Medium	8
3.1.2	Feasibility Analysis	Complex	10
3.2	**Design**		***12***
3.2.1	Design conceptual ERD	Medium	4
3.2.2	Design code architecture	Medium	4
3.2.3	Design web application	Medium	4
3.3	**Implementation**		

3.3.1	**Authentication**		***4***
3.3.1.1	Signing with Google	Medium	*2*
3.3.1.2	Get refresh token	Medium	*1*
3.3.1.3	Verify token	Medium	*1*
3.3.2	***User management***		***17***
3.3.2.1	User registration & login	Simple	2
3.3.2.2	User roles & permissions	Medium	4
3.3.2.3	Profile management	Simple	2
3.3.2.4	Scan student CV by AI	Complex	7
3.3.2.5	Manager assign role for lecturer every semester	Simple	2
3.3.3	**Timeline Management**		***2***
3.3.3.1	Manager set timelines for each semester	Simple	*2*
3.3.4	**Criteria Form Management**		***4***
3.3.4.1	Manager set criteria form for each semester	Medium	*4*
3.3.5	**Project Registration & Approval**		***14***
3.3.5.1	Student and lecturer submit thesis proposal	Medium	4
3.3.5.2	Mentor reviews and response ideas proposal	Medium	4
3.3.5.3	Council reviews and response the thesis topic	Medium	6
3.3.6	**Team Formation & Matching**		***20***
3.3.6.1	Create teams	Simple	2
3.3.6.2	AI-powered member suggestions	Complex	7
3.3.6.3	AI-powered team suggestions	Complex	7
3.3.6.4	Leader sends invitations for member	Medium	2
3.3.6.5	Team members and mentor response the invitations	Simple	2
3.3.7	**Team Recruitment Posts**		***5***
3.3.7.1	Create recruitment blog	Simple	*1*
3.3.7.2	Like & comment on blog	Simple	*1*
3.3.7.3	Student apply CV via blog	Medium	*2*

3.3.7.4	Leader view list of cv	Simple	*1*
3.3.8	**Project Progress Management**		***7***
3.3.8.1	Manager import review schedules	Medium	*2*
3.3.8.2	Student submit review checklist	Simple	*1*
3.3.8.3	Student update topic for mentor after review 1, 2	Simple	*1*
3.3.8.4	Mentor review and respond updating topic	Simple	*1*
3.3.8.5	Manager review and respond updating topic after mentor	Simple	*1*
3.3.8.6	Mentor provide feedback after review 3	Simple	*1*
3.3.10	**Team Rate Contribution**		***4***
3.3.10.1	Leader rate contribution	Simple	*4*
3.3.11	**Capstone Management**		***4***
3.3.11.1	Manager import capstone schedules	Medium	*2*
3.3.11.2	Manager update capstone results	Simple	*1*
3.3.11.3	Student view capstone results	Simple	*1*
3.3.12	**Messaging System**		***10***
3.3.12.1	Initiating a Chat	Complex	*7*
3.3.12.2	Sending & Receiving Messages	Complex	*3*
3.3.13	**Notification System**		***9***
3.3.13.1	*Send notification*	Complex	*5*
3.3.13.2	*View notification*	Medium	*2*
3.3.13.3	*Manager create notification for system*	Medium	*2*
3.4	**Testing**		***24***
3.4.1	Unit test	Complex	*8*
3.4.2	Integration Test	Complex	*8*
3.4.3	System Test	Complex	*8*
3.5	**Monitoring and Controlling**		***24***
3.5.1	Control the process	Complex	*12*
3.5.2	Track performance a quality	Complex	*12*

3.6	**Closing**		***20***
3.6.1	Report	Simple	*20*
		Total Estimated Effort (man-days)	***232***

1.2 Project Objectives

- Timelines: The project must be finished before 30 April, 2025
- Allocated Effort (Man-days): 232
- Defect Distributions:

#	Testing Stage	Test Coverage	No. of Defects	% of Defect	Notes
1	Reviewing	**Backend:** Code review (C#, Authentication) **Frontend:** Code review (Next.js, React Components, State Management)	<30	5%	Focus on security, performance, and maintainability
2	Unit Test	**Backend:** Business logic **Frontend:** Component testing, Form validation	<20	3%	Ensure correctness of auth, UI interactions, and state management
3	Integration Test	**Backend:** API integration **Frontend:** API calls, Data binding, UI responsiveness	<5	1%	Verify API interactions, state consistency, and UI behavior

4	System Test	**Backend:** Full system testing with database (PostgreSQL, Neon) **Frontend:** End-to-end navigation, user experience	<5	1%	Ensure smooth UI-Backend communication and DB consistency
5	Acceptance Test	Full workflow	<5	1%	Ensure system meets business requirements

1.3 Project Risks

#	Risk Description	Impact	Possibility	Response Plans
1	Incomplete or inaccurate test cases generated by AI	Incorrect grading results lead to student dissatisfaction and appeals.	Medium	Implement a review process where generated test cases are checked by a human before being used for grading.
2	Data security breaches	Exposure to sensitive student data leads to legal issues and loss of trust.	Medium	Employ strong encryption methods and regularly update security protocols to protect student data.
3	Project management features may not be fully developed or may malfunction.	Students will struggle to track and manage their projects	High	Develop and thoroughly test project management features early in the development cycle.

2. Management Approach

2.1 Project Process

After carefully evaluating different software development models, the project will adopt an Iterative and Incremental Software Development Process. In this model, an initial partial version of the system is developed and delivered early, ensuring that there is always something functional at every stage. The Iterative and Incremental approach is especially beneficial when the project scope is large, major

requirements are already well-defined, but further details will emerge during the development process. This method breaks the system development into smaller, manageable tasks, with each task completed in phases. This allows us to build upon knowledge gained in earlier phases. The reasons for choosing this model include:

- Developing core features based on priority requirements first.
- The flexibility to easily accommodate changes in requirements.
- Continuous testing and debugging during each iteration.
- The ease of managing risks, as risks are identified and addressed during each iteration.
- Clients can provide feedback after each product increment, helping to avoid surprises at the end of the project.
- Important functionality is delivered early, ensuring early value for the clients.

2.2 Quality Management

2.2.1 Defect Prevention: In the event that a defect is identified, the responsible person must be notified immediately. Each defect must be carefully assessed by answering questions such as: "How critical is the defect, and can it potentially damage the system?" and "How long will it take to resolve the defect?" A clear deadline for defect resolution must be established. Additionally, a proactive plan should always be in place to anticipate any possible issues that could arise during development.

2.2.2 Reviewing: The review process must be conducted impartially and without bias towards any project team member. If an error is discovered, the responsible party must be promptly informed. Defects should be recorded in the Bug Tracking software with detailed information, including the defect's priority. The person responsible for addressing the defect must provide a solution and resolve the issue as quickly as possible.

2.2.3 Unit Testing: Test cases must be prepared thoroughly and accurately, ensuring that no possible test scenario is overlooked. The test cases should align with the system's functionalities. Any defects found during testing must be logged in the Bug Tracking software, including relevant details such as priority. The person responsible for the defect must offer a solution to fix it promptly.

2.2.4 Integration Testing: Test cases for integration testing should be prepared carefully and accurately, without omitting any possible scenarios. These test cases should align with the system's functionality. All defects identified during integration testing should be documented in the Bug Tracking software, with

specific details such as priority level. The person responsible for addressing the defect must implement a solution quickly. It's essential that internal modules of the system work seamlessly together.

2.2.5 System Testing: System testing requires that test cases be designed carefully and in full alignment with the system's requirements and architectural design. Any defects encountered during system testing must be logged in the Bug Tracking software, along with important details like priority. The individual responsible for resolving the defect must devise a solution and resolve the issue swiftly. System testing must ensure that all system functionalities are thoroughly covered, including interactions with any external systems still under development.

2.3 Training Plan

Training Area	Participants	When, Duration	Waiver Criteria
.NET (Backend)	All Team member	Week 1 - 7 days	Mandatory
Next.js Typescript (Frontend)	Sơn, Quân, Lộc	Week 1 - 14 days	Mandatory
Git, GitHub	All Team member	Week 1 - 3 days	Mandatory
PostgreSql	All Team member	Week 2 - 4 days	Mandatory

3. Project Deliverables

#	Deliverable	Due Date	Notes
1	Sprint 1	06/01/2025 - 12/01/2025	- Collect user requirements - Training - Report 1
2	Sprint 2	13/01/2025 - 19/01/2025	- Design database - Design UI - Set up the base project, set up CI/CD - Report 2 - Training
3	Sprint 3	03/02/2025 - 09/02/2025	- Implement authentication user module - Implement manage user module
4	Sprint 4	10/02/2025 - 16/02/2025	- Implement manage timeline and schedule module
5	Sprint 5	17/02/2025 - 23/02/2025	- Implement manage topic and evaluation module - Report 3

6	Sprint 6	24/02/2025 - 02/03/2025	- Implement notification and communication module - Set up the project for future release
7	Sprint 7	03/03/2025 - 09/03/2025	- Implement manage team and team collaboration module - Report 4
8	Sprint 8	10/03/2025 - 16/03/2025	- Implement feedback and review result module - Implement validation & constraint value for
9	Sprint 9	17/03/2025 - 23/03/2025	- Continue to implement validation & constraint value for all features - Review system
10	Sprint 10	24/03/2025 - 30/03/2025	- Implement integration test
11	Sprint 11	31/03/2025 - 06/04/2025	- Implement system test - Report 5
12	Sprint 12	07/04/2025 - 13/04/2025	- Release - Report 6
13	Sprint 13	14/04/2025 - 20/04/2025	- Review all features related to projects - Report 7
14	Sprint 14	21/04/2025 - 27/04/2025	- Prepare for final presentation

4. Responsibility Assignments

D~Do; R~Review; S~Support; I~Informed; <blank>- Omitted

Responsibility	**ThuBTT SE171984**	**SonNH SE160611**	**QuanCM SE160599**	**LocLT SE160585**
Project Planning & Tracking	D	D	R	S
Prepare Project Introduction Document	S	D	D	R
Prepare Software Requirement Document	R	S	D	D
Prepare Software Design Document	D	R	S	D
Prepare Test Document	D	D	R	S
Prepare Software User Guides	S	D	D	R
Prepare Final Report	S	R	D	D

5. Project Communications

Communication Item	Who/ Target	Purpose	When, Frequency	Type, Tool, Method(s)
Team weekly meeting	All team members	Review plan, schedule, members work achievements during the week and report the project progress and status	17:30 pm every Friday	Offline at NVH
Weekly Report	Supervisors Team members	- Review work progress, including code and documentation. - Answer requirements and technical questions. - Control project deadlines and ensure projects run on schedule.	1 day / week	Offline Google Meet
Daily Meeting	Team members	Report the progress that members achieved each	Daily	Google Meet
Unscheduled meeting	All team members	When there's a critical problem that needs to be resolved immediately, discuss then resolve that problem	When members find important problems	Google Meet

6. Configuration Management

6.1 Document Management

Document tools: Microsoft Word, Microsoft Excel

File management and others: Google Drive

6.2 Source Code Management

Source code is managed by Git on github.com

6.3 Tools & Infrastructures

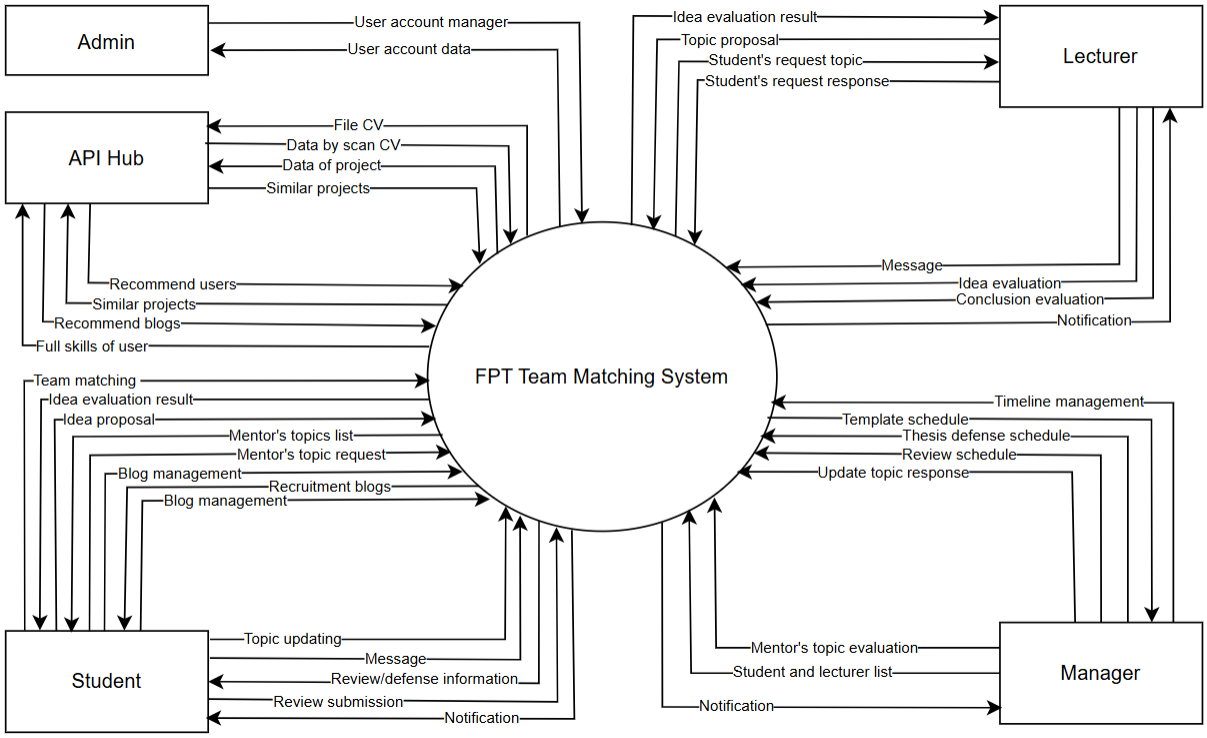
Category	Tools / Infrastructure
Technology	Next.js 14 (FrontEnd), C#/.NET (BackEnd)
Database	PostgreSql, MongoDb
IDEs/Editors	Visual Studio Code, Visual Studio
Diagramming	DrawIO, Visual Paradigm
Documentation	Ms Office, Google Docs/Sheets/Slides
Version Control	GitHub (Source Codes), Google Drive (Documents)
Deployment server	Render, Vercel, Neon, Docker
Project management	Google sheet

III. Software Requirement Specification

1. Product Overview

The FPT Team Matching System is a new software platform that replaces the traditional manual processes for thesis registration, approval, and team coordination at FPT University. The system streamlines the process by allowing students to register thesis topics, select mentors, and undergo structured approvals by mentors and thesis committees. It also facilitates team formation, review scheduling, feedback collection, and thesis defense coordination.

The context diagram below illustrates the external entities and system interfaces for release 1.0. The system is expected to evolve over multiple releases, incorporating additional features such as enhanced automation, improved collaboration tools, and potential integration with university databases for better student eligibility verification.



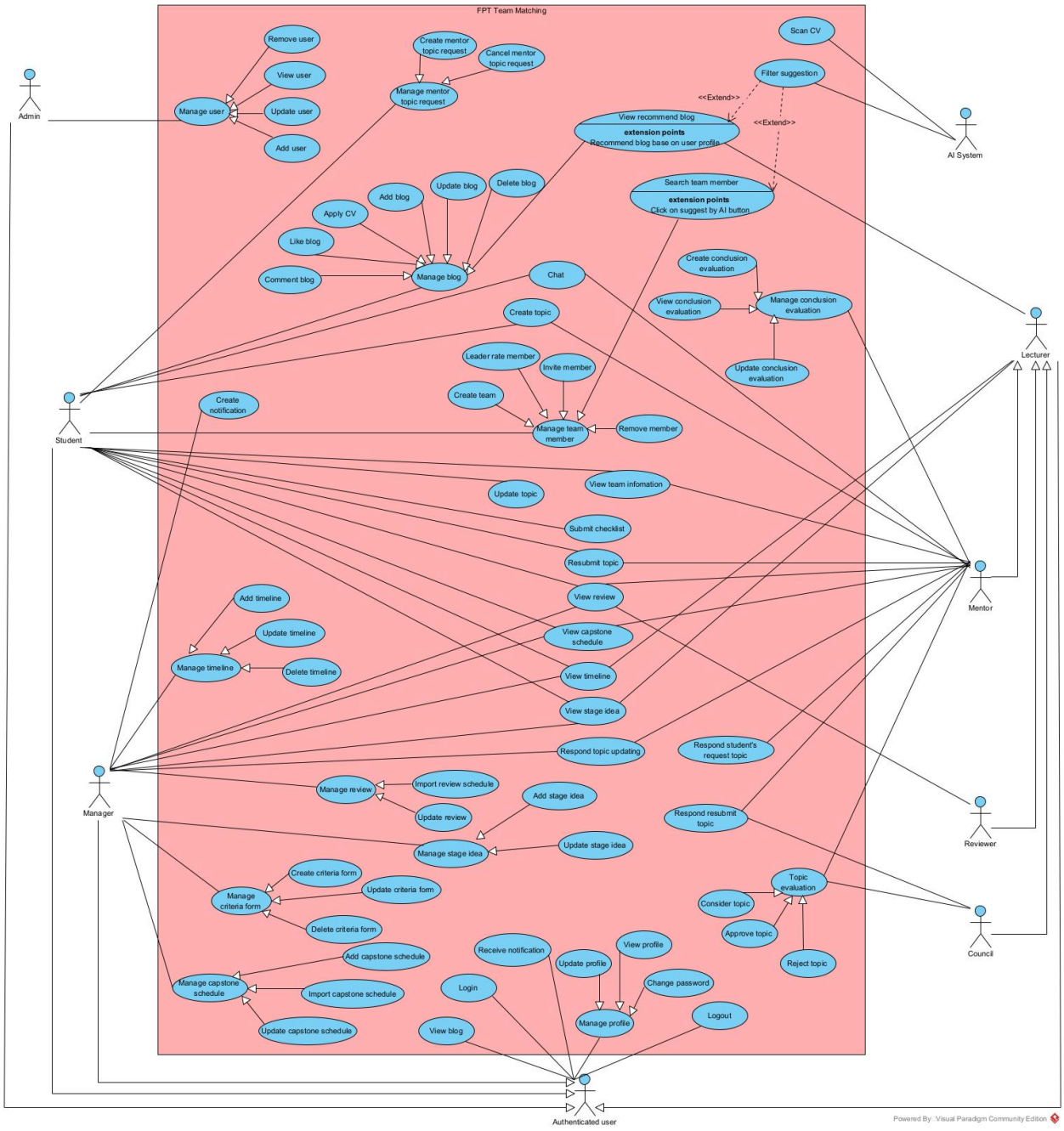
2. User Requirements

2.1 Actors

#	Actor	Description
1	Admin	Responsible for managing the entire platform, including user account approvals, maintaining system security, and ensuring smooth operation.
2	Manager	Oversees the thesis process by creating and managing topic approval periods, assign roles for lecturer in every semester. Managers also schedule thesis defense and review schedules, update capstone results.
3	Lecturer	Mentor: evaluate student ideas, response for topic requests, provide student's conclusion evaluation, evaluate topic updating. Council: evaluate topic proposals submitted by mentors.. Reviewer: view review schedule.
4	Student	The primary user of the system, responsible for forming or joining project teams, submitting project ideas, requesting lecturer-proposed topics and managing project progress. Students can join a team or recruit members through posts, communicate with lecturers for guidance, and receive evaluations based on their performance and contributions.

2.2 Use Cases

2.2.1 Diagram(s)



2.2.2 Descriptions

ID	Use Case	Actors	Use Case Description
01	Login	Authenticated user	User login to FPT Team Matching app
02	Logout	Authenticated user	Users logout
03	Manage profile	Authenticated user	Users can view, update their profile
04	Update profile	Authenticated user	Users edit personal details
05	View profile	Authenticated user	Users view their own profile details
06	Change password	Authenticated user	Users can change their password
07	Receive notification	Authentication user	Receive notification about project, team, schedule.
08	Manage user	Admin	Admin can CRUD users
09	View user	Admin	Admin views the list of all users
10	Add user	Admin	Admin adds a new user to the system
11	Remove user	Admin	Admin delete a user from the system
12	Update user	Admin	Admin updates existing user information
13	Manage timeline	Manager	Manage all timeline related to semester
14	View timeline	Student, Manager, Lecturer	View timeline information
15	Add timeline	Manager	Add a new semester to the system
16	Update timeline	Manager	Update existing timeline details
17	Delete timeline	Manager	Delete existing timeline details
18	Manage capstone schedule	Manager	Manage the overall capstone schedule
19	Import capstone schedule	Manager	Import capstone schedule from a file
20	Add capstone schedule	Manager	Add capstone schedule
21	Update capstone schedule	Manager	Update capstone schedule
22	View capstone schedule	Mentor, Student, Manager	View capstone schedule
23	Manage criteria form	Manager	Manage the evaluation form with criterias.
24	Create criteria form	Manager	Create criteria form
25	Update criteria form	Manager	Update criterias in form

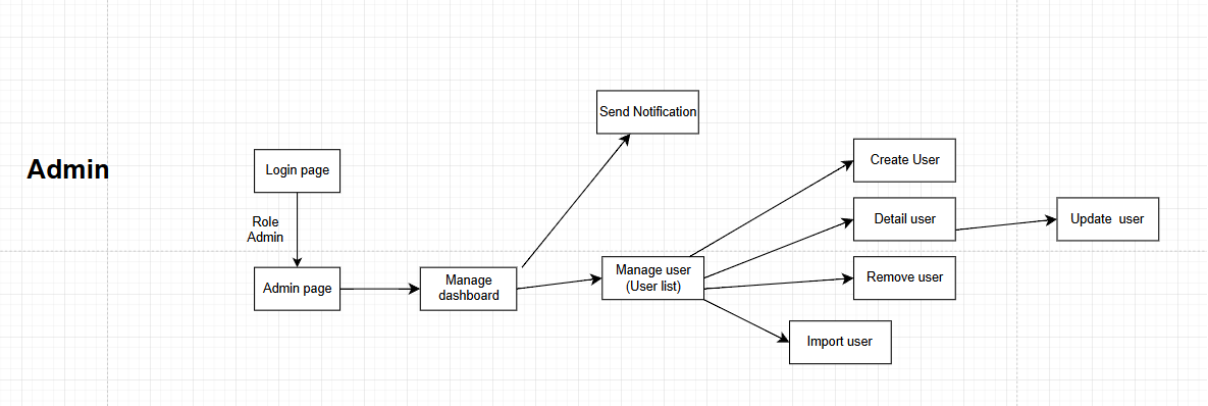
26	Delete criteria form	Manager	Delete criterias or delete form
27	Manage stage idea	Manager	Manage all stage ideas in the system.
28	Add stage idea	Manager	Add a new stage idea.
29	Update stage idea	Manager	Update details of an existing stage idea.
30	View stage idea	Manager, Lecturer, Student	View stage ideas and their progress or details.
31	Manage review	Student, Lecturer, Reviewer, Council	They can add a report, view report, feedback report on this review.
32	View review	Student, Mentor, Manager, Reviewer	View submitted reviews
33	Update review	Manager	Update the review schedule
34	Import review schedule	Manager	Import the review schedule into the system
35	Submit checklist	Student	Upload checklist file
36	Manage blog	Student	Manage blog (CRUD)
37	View blog	Authenticated user	View blog
38	View recommend blog	Student	View recommend blog base on user profile
39	Search team member	Student	Student search for potential team members based on criteria (skills, interests, project requirements)
40	Add blog	Student	Add a new blog post
41	Update blog	Student	Update their blog
42	Delete blog	Student	Delete a blog post
43	Like blog	Lecturer, Student	Like a blog post
44	Comment blog	Lecturer, Student	Comment on a blog post.
45	Apply CV	Student	Submit CV through blog.
46	Topic evaluation	Mentor, Council	They can approve, reject or consider topics.
47	Create topic	Student, Mentor	Both students and lecture can create topic
48	Approve topic	Council, Mentor	They approves submitted topic
49	Reject topic	Council, Mentor	They rejects submitted topic

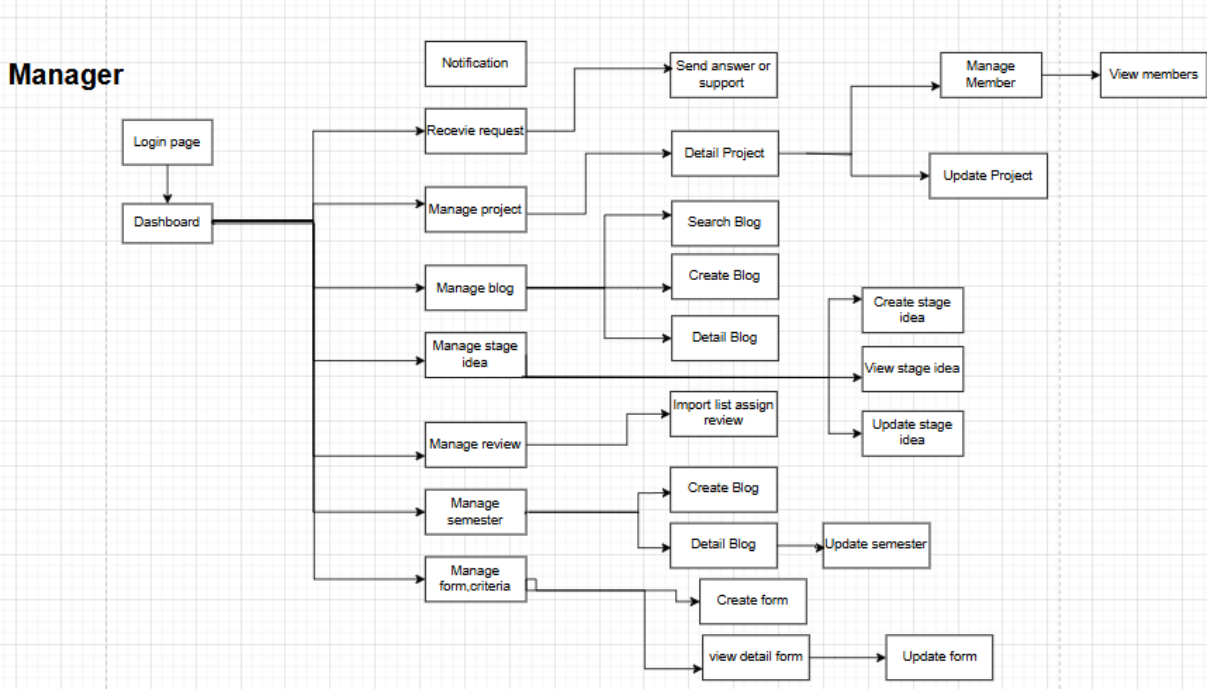
50	Consider topic	Council, Mentor	They consider submitted topic
51	Manage conclusion evaluation	Mentor	Manage feedback provided to students.
52	Create conclusion evaluation	Mentor	Create feedback for a team after review 3
53	View conclusion evaluation	Mentor	View feedback given by mentor
54	Update conclusion evaluation	Mentor	Update feedback
55	Manage mentor topic request	Student	Manage their mentor idea requests.
56	Create mentor topic	Student	Send a request to a mentor to use their idea.
57	Cancel mentor topic request	Student	Cancel a previously submitted mentor topic request.
58	Respond student's request topic	Mentor	Accept or reject mentor topic requests submitted by students.
59	Resubmit topic	Student, Mentor	Resubmit topic when mentor or council consider topic
60	Respond resubmit topic	Mentor, Council	They can approve, reject or consider resubmit form.
61	Update topic	Student	Update topic after review 1, 2
62	Respond topic updating	Mentor, Manager	Accept or reject topic updating by student
63	Chat	Student, Mentor	Students can chat between students and lecture.
64	Create notification	Manager	Create a custom notification.
65	Manage team member	Student	Leader performs actions to manage team members
66	Create team	Student	Student creates a new team
67	Invite member	Student	Leader sends invitations to other users to join their team.
68	Remove member	Student	Leader removes a member from the team.
69	Leader rate member	Student	Leader rates a team member's contribution after review 3.
70	Search team member	Student	Search for team members using AI suggestions.
71	View team information	Student, Mentor	View details of a team
72	Scan CV	AI System	AI scans student CV.
73	Filter suggestion	AI System	AI suggest team members

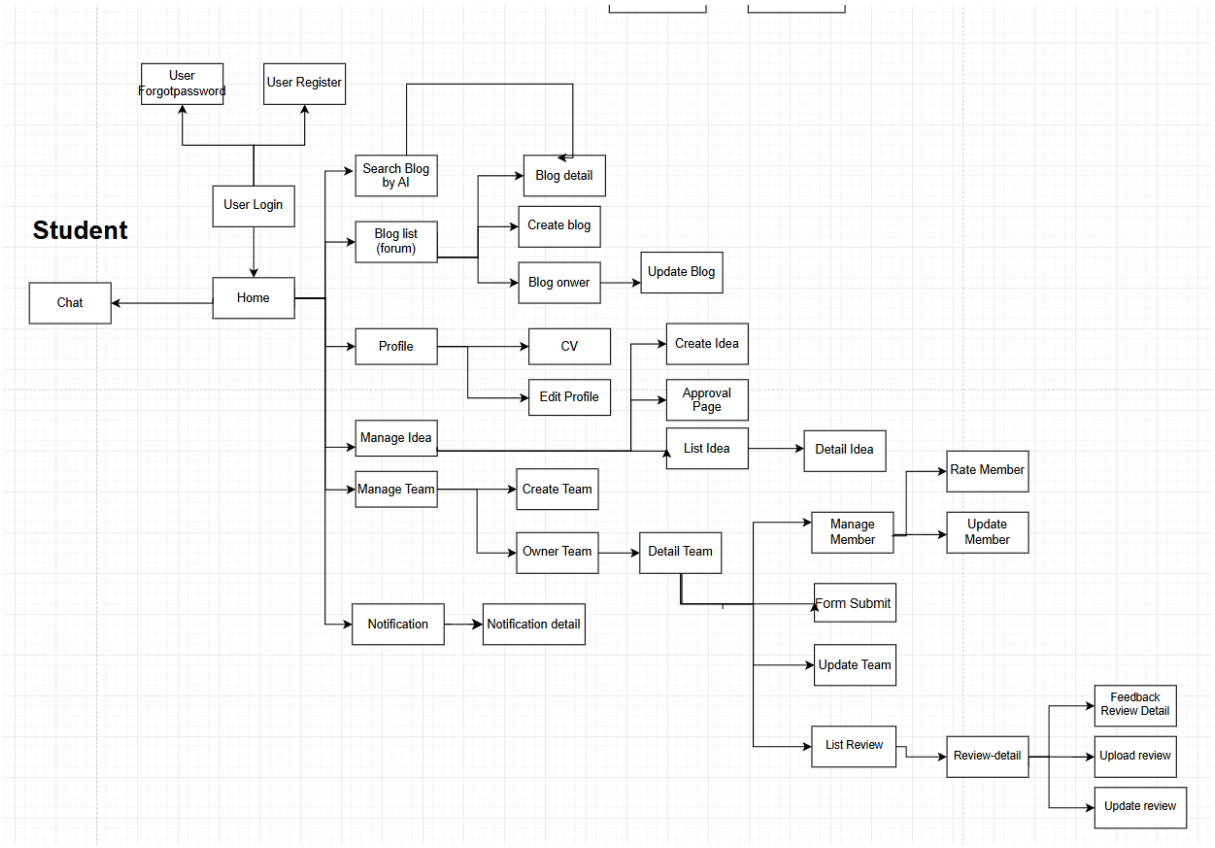
3. Functional Requirements

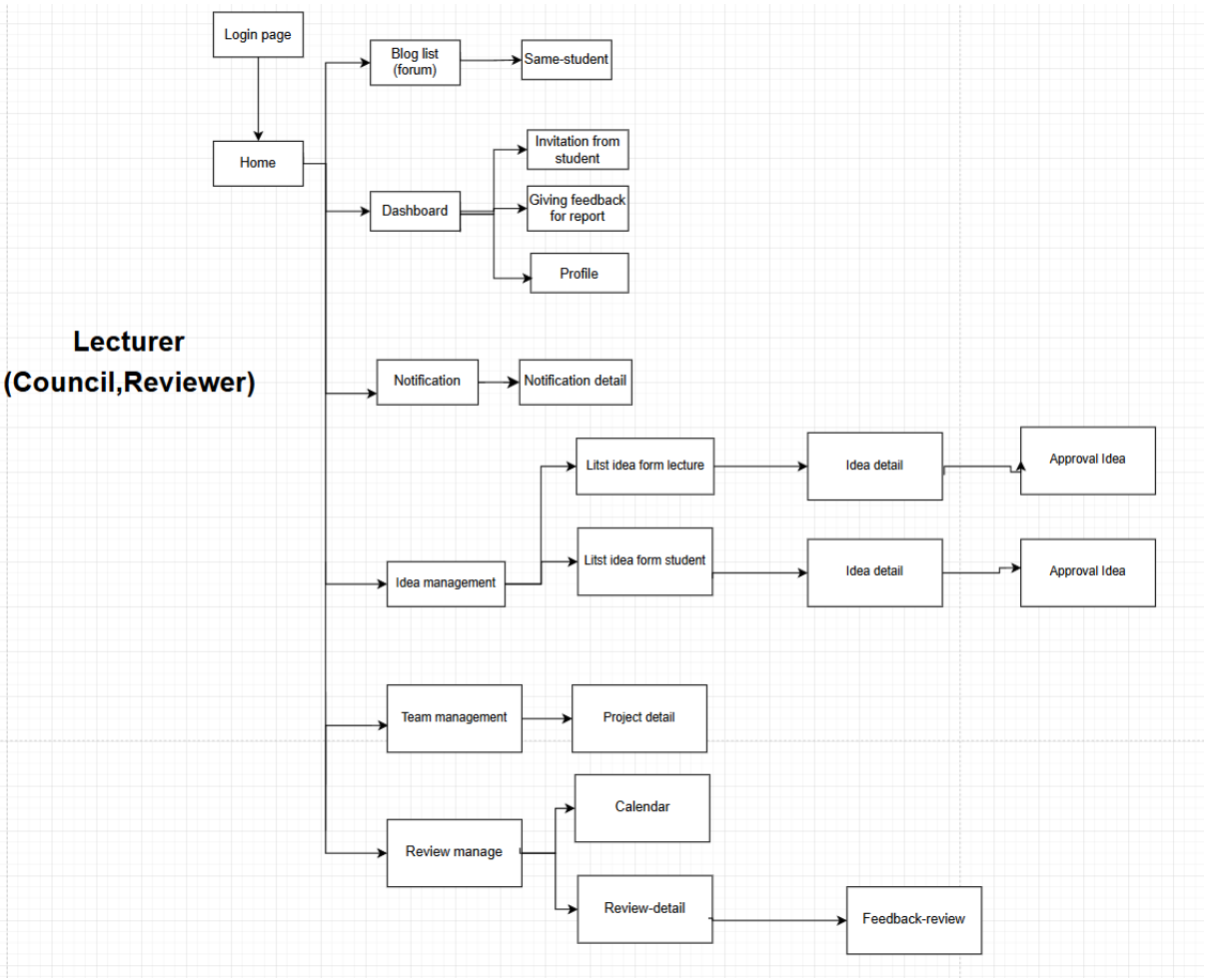
3.1 System Functional Overview

3.1.1 Screens Flow






Student
User Forgotpassword
User Register
User Login
Chat
Home
Search Blog by AI
Blog list (forum)
Profile
Manage Idea
Manage Team
Notification
Blog detail
Create blog
Blog onwer
Update Blog
CV
Edit Profile
Create Idea
Approval Page
List Idea
Detail Idea
Create Team
Owner Team
Detail Team
Notification detail
Rate Member
Manage Member
Update Member
Form Submit
Update Team
List Review
Review-detail
Feedback Review Detail
Upload review
Update review


Lecturer
(Council,Reviewer)
Login page
Home
Blog list (forum)
Same-student
Dashboard
Invitation from student
Giving feedback for report
Profile
Notification
Notification detail
Idea management
Litst idea form lecture
Idea detail
Approval Idea
Litst idea form student
Idea detail
Approval Idea
Team management
Project detail
Review manage
Calendar
Review-detail
Feedback-review

3.1.2 Screen Descriptions

#	Feature	Screen	Description
1	Authentication	Login Page	Allows users (Lecture,Student,Council,Staff) to log into the system using their credentials.
2	Register	Register	Allows students to register accounts into the system .
3	Reset Password	Reset Password	Allows Students to reset passwords .
4	Idea Search	Home	Provides navigation for managing ideas, project, and review requests and research idea on the system
5	Idea Creation	Create Idea	Allows student , lecture to create new idea by specifying details and student can choose the mentor for the idea
6	Update Idea	Update idea	Allow student , lecture to update idea if idea's approved
7	Delete Idea	Delete Idea	Allows users (Lecture,Student) to delete their idea
8	Idea Detail	Idea Detail	Displays in-depth information about a title,spectify,profession,description,max team member
9	List-blog	Social	Allows lecture and student can view all blog
10	Blog-detail	Blog-detail	Allows lecturer and student information about a blog title,description,comment,like,view…
11	Blog Editing	Update Blog	Student and Lecture the option to update various aspects of a blog, such as blog information, title.
12	Blog Removal	Remove blog	Allows lecture,student , staff to delete an existing exam.
13	Comment,like,upload	Blog-detail	The user can contact the blog, and upload cv to invite member
14	Source Code Import	Import Student Source Code	Idea Creation
15	Access Admin,Lecture,Student,Manager	Dashboard	Displays main navigation for all management functionalities.
16	View Dashboard	Manage Dashboard	Displays an overview of management features. Admin can navigate to user management and notification functions from here.
17	Manage Users	Manage User (User List)	Shows a list of all users. From here, admin can create, view details, update, delete, and import users
18	Create a New User	Create User	Provides a form to input and submit new user information to create a new user in the system.
19	View User Details	Detail User	
20	Update User Information	Update User	Allows admin to edit and update information for an existing user from the Detail User screen.

21	Import Users	Import User	Provides an interface to upload a file (e.g., Excel) and bulk import multiple users into the system.
22	Receive Request	Receive Request	Check the list of project requests that require your support or answers
23	Send Answer or Support	Send Answer/ Support Screen	Send your answer or support to the requester for a specific project.
24	Manage Projects	Manage Project	View and manage all projects under your responsibility
25	View Project Details	Detail Project Screen	See detailed information about a specific project you are managing.
26	Manage Members of a Project	Manage Member	View and manage the list of members in a specific project.
27	Update Project	Update Project	Edit and update the information of an existing project
28	Manage Blogs	Manage Blog	View all blogs, and search or manage blog content easily.
29	Search Blogs	Search Blog	Search for blogs by keywords or filters to quickly find what you need.

30	Create a New Blog	Create Blog	Create a new blog post and share important updates or content.
31	View Blog Details	Detail Blog	View full content and detailed information of a selected blog
32	Manage Stage	Manage Stage Idea	Manage all stage ideas related to different projects.
33	Create Stage Idea	Create Stage Idea	Create a new stage idea for a project to define its phases clearly.Create a new stage idea for a project to define its phases clearly.
34	View Stage Idea	View Stage Idea	See all details about a specific stage idea
35	Update Stage Idea	Update Stage Idea	Edit and update the information of a stage idea whenever needed.
36	Manage Reviews	Import List and Assign Reviews	Manage Semester

37	Create Blog	Create Blog	Create a new blog post associated with a specific semester.
38	View Blog Details	View Blog Details	View full details of a blog post
39	Profile Management	Profile Management	View your student profile.
40	Member Evaluation	Rate Member	Give performance ratings to your team members.
41	Review Management	List Review	View the list of reviews related to your work or team.
42	Review Detail	Review Detail	Read the full feedback provided for a review.
43	Upload Review	Upload Review	Submit your own review (e.g., for another team or project).
44	Update Review	Update Review	Edit a review you've already submitted.

45	Notification	Notification	View all system notifications.
46	Communication	Chat	Chat and communicate with other students or team members.
47	Calendar	Calendar	View and manage schedule, deadlines, or presentation date
48	Manage Form,Criteria		View the list of form related to lecture ,council evaluate team
49	Create Form		Create a form with a specific semester.
50	View Detail Form		View full details of a form and criteria
51	Rate Member		Create a rate for evaluation

3.1.3 Screen Authorization

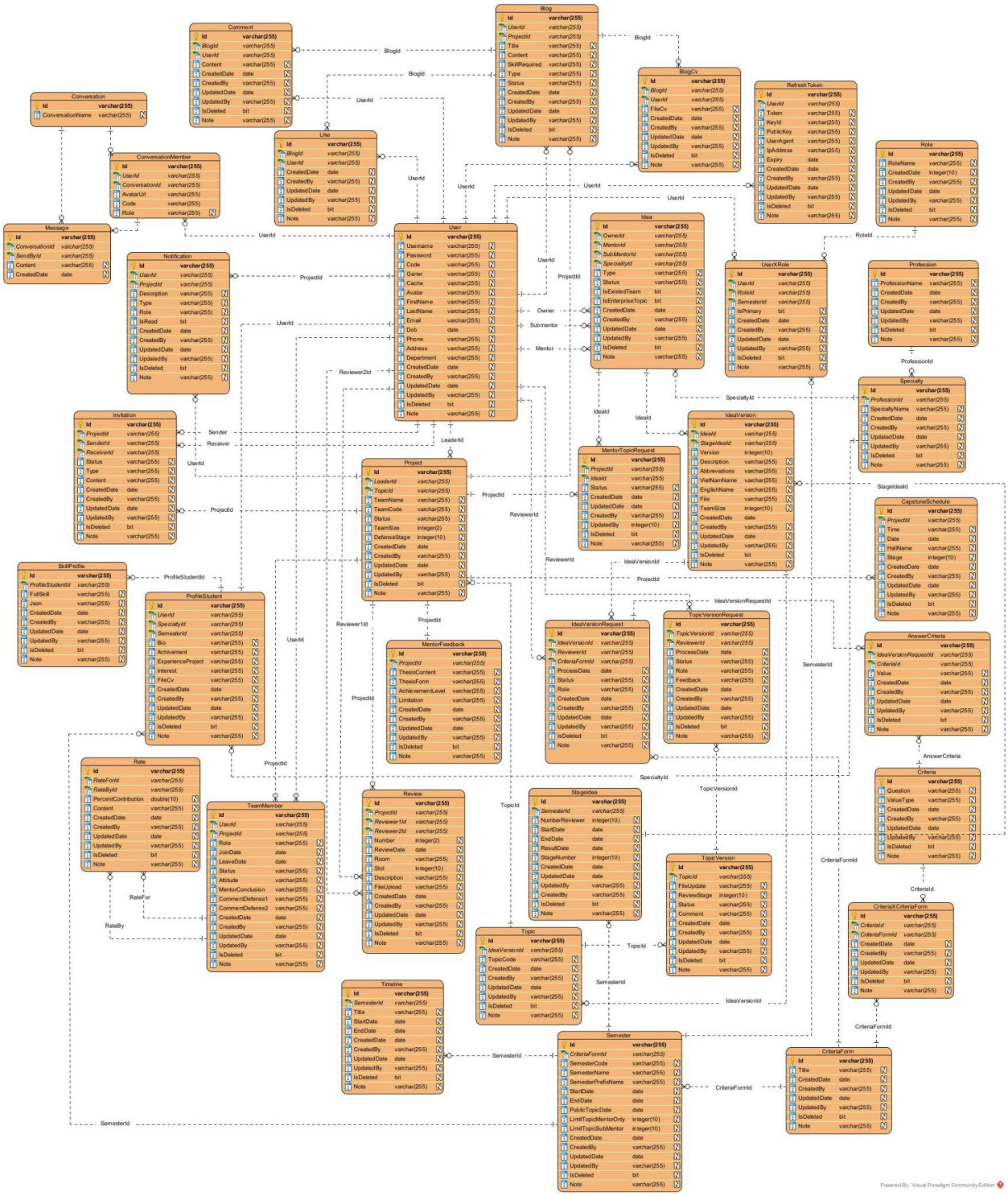
Screen	Admin	Manager	Lecturer	Student
Login Page	x	x	x	x
Register	x			x
Home	x	x	x	x
Create Idea	x		x	x
Update Idea	x		x	x
Idea Details	x	x	x	x
Remove Idea	x		x	x
Social	x	x	x	x
Blog-detail	x	x	x	x
Create Blog	x	x	x	x
Blog Editing	x	x	x	x
Blog Removal	x	x	x	x
Search Blogs	x	x	x	x
Profile	x	x	x	x
Edit profile	x	x	x	x
Manage Users	x	x		
Create a New User	x	x		
View User Details	x	x	x	
Update User Information	x	x		
Import Users	x	x		
Delete User	x	x		
Receive Request	x	x		
Send Answer or Support			x	x
Manage Projects	x	x		
View Project Details	x	x	x	x
Manage Members of a Project		x	x	
Update Project	x	x	x	x

Manage Stage	x	x	x	x
Create Stage Idea	x	x		
View Stage Idea	x	x		
Update Stage Idea	x	x		
Approval Page	x	x	x	x
Manage Reviews	x	x		
Create Review	x	x		
Review Detail	x	x	x	x
Upload Review	x	x		
Update Review	x	x		
Notification	x	x	x	x
Communication	x	x	x	x
Calendar	x	x	x	x
Import calendar	x	x		
Manage Form,Criteria	x	x		
Create Form	x	x		
View Detail Form	x	x	x	x
Submit Form			x	

3.1.4 Non-Screen Functions

#	Feature	System Function	Description
1	Automated Topic Approval	Cron Job	Scheduled job that checks and update final result for topics if all criteria are met.
2	Automated reviews creation	Cron Job	Scheduled job that checks if semester start, create reviews for projects.
3	Automated change status of project	Cron Job	Scheduled job that checks if semester start, change status for project
4	Export template for review schedule	API Endpoint	The system allows manager to export the template file as Excel file for import review schedule for student and reviewers

3.1.5 Entity Relationship Diagram

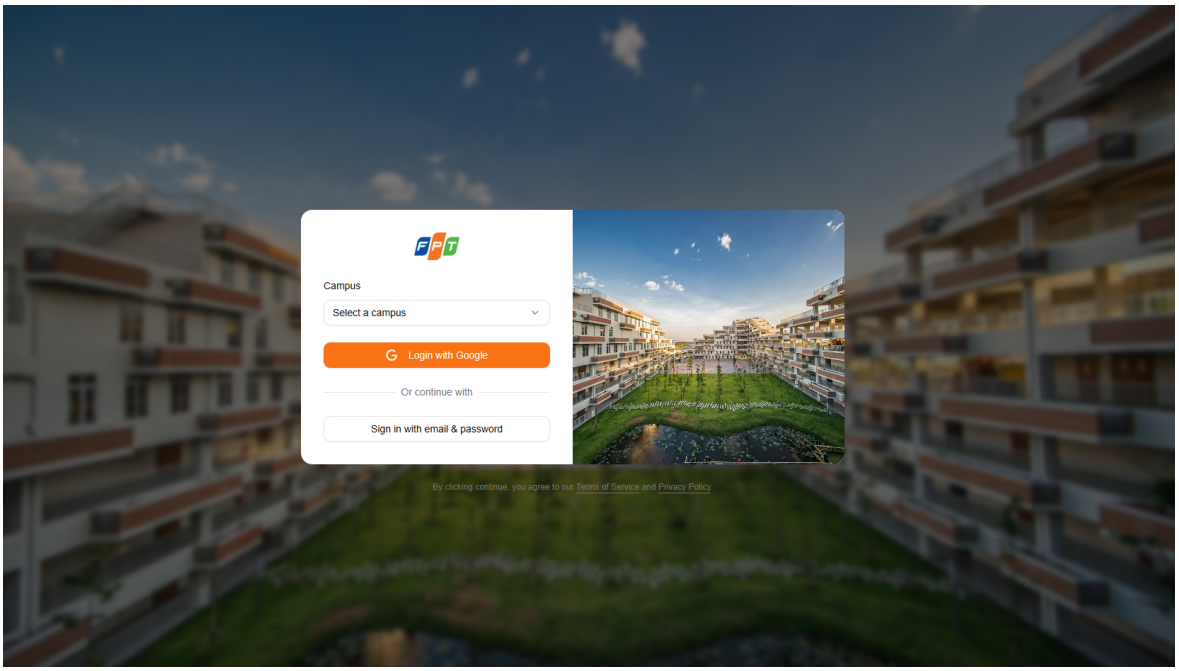


3.2 Authentication

3.2.1 Login

- **Role:** Admin

- **Function trigger:** The admin enters account credentials to log in.
- **Function description:**
    - **Purpose:** System administration.
    - **Data processing:** Authenticate login credentials.
- **Function Details:**
    - If login is successful, redirect to the dashboard.
    - If failed, display an error message.

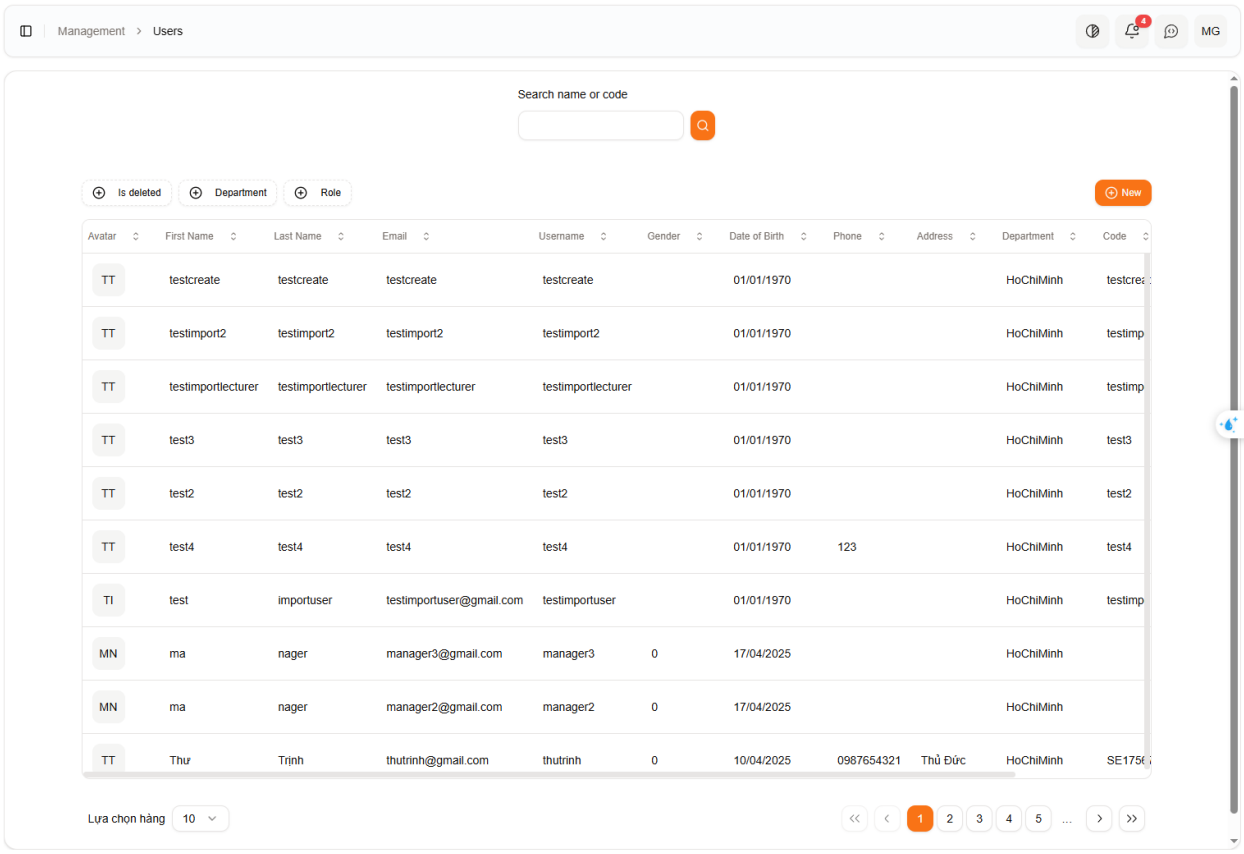


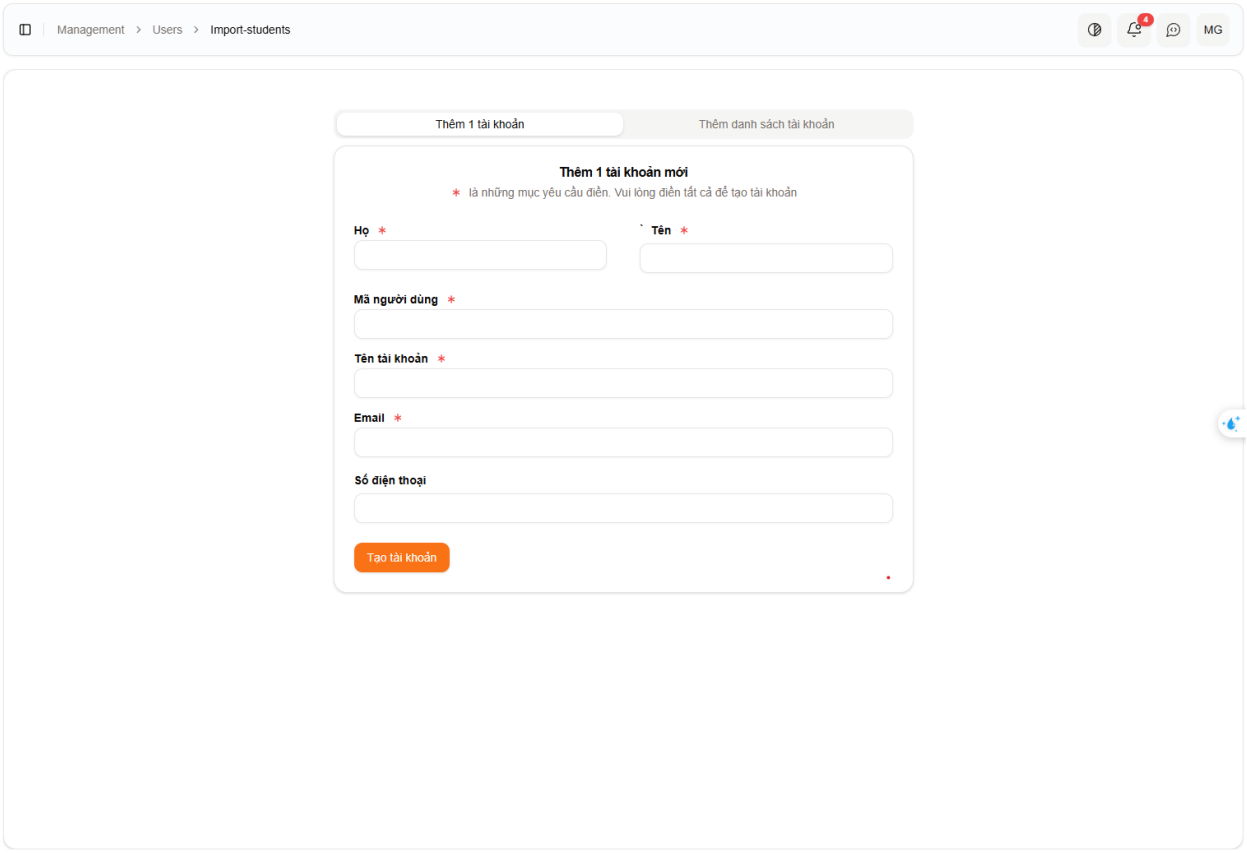
3.3 User Management

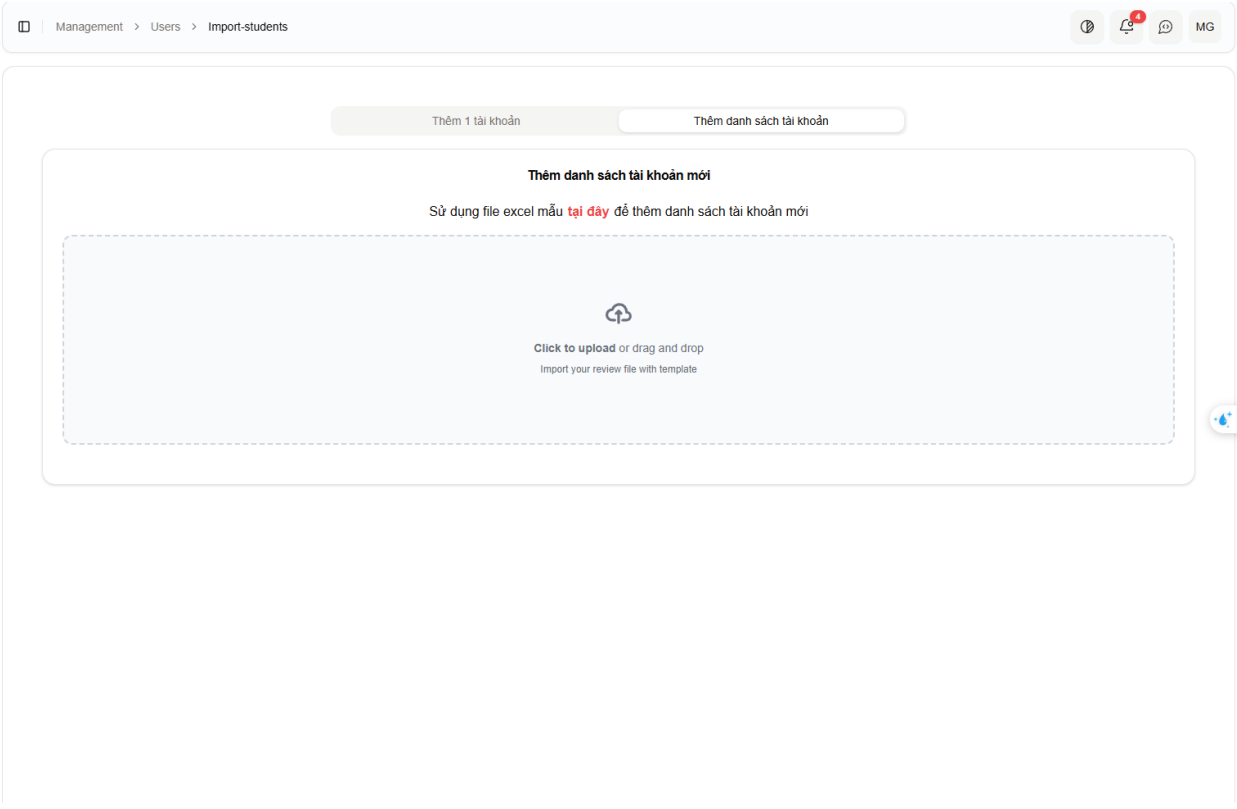
3.3.1 Manage users

- **Role:** Admin
- **Function trigger:** The admin accesses the user management page.
- **Function description:**
    - **Purpose:** Manage user accounts.
    - **Data processing:** Allow creation, editing, and deletion of user accounts.
- **Function Details:**
    - Admin can ban or unban users.

- Search for user by email, name, or ID
- Admin create user, and assign role for user.

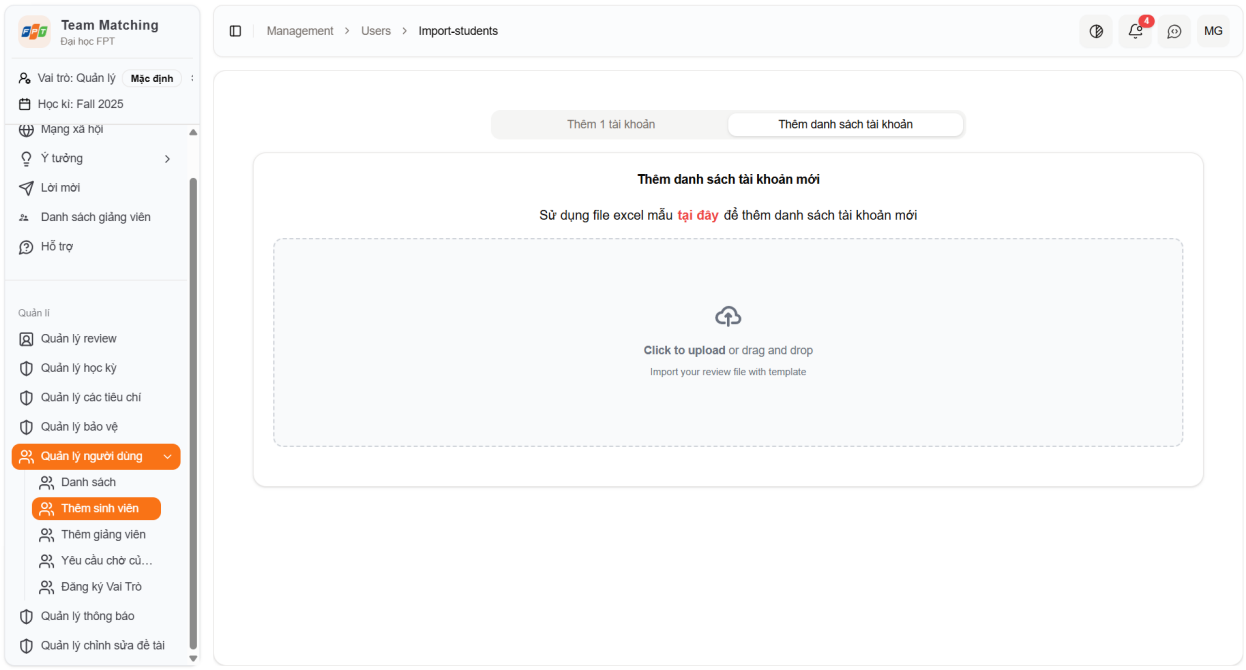



Management > Users > Import-students
MG
Thêm 1 tài khoản
Thêm danh sách tài khoản
Thêm 1 tài khoản mới
* là những mục yêu cầu điền. Vui lòng điền tất cả để tạo tài khoản
Họ *
Tên *
Mã người dùng *
Tên tài khoản *
Email *
Số điện thoại
Tạo tài khoản


Management > Users > Import-students
MG
Thêm 1 tài khoản
Thêm danh sách tài khoản
Thêm danh sách tài khoản mới
Sử dụng file excel mẫu tại đây để thêm danh sách tài khoản mới
Click to upload or drag and drop
Import your review file with template

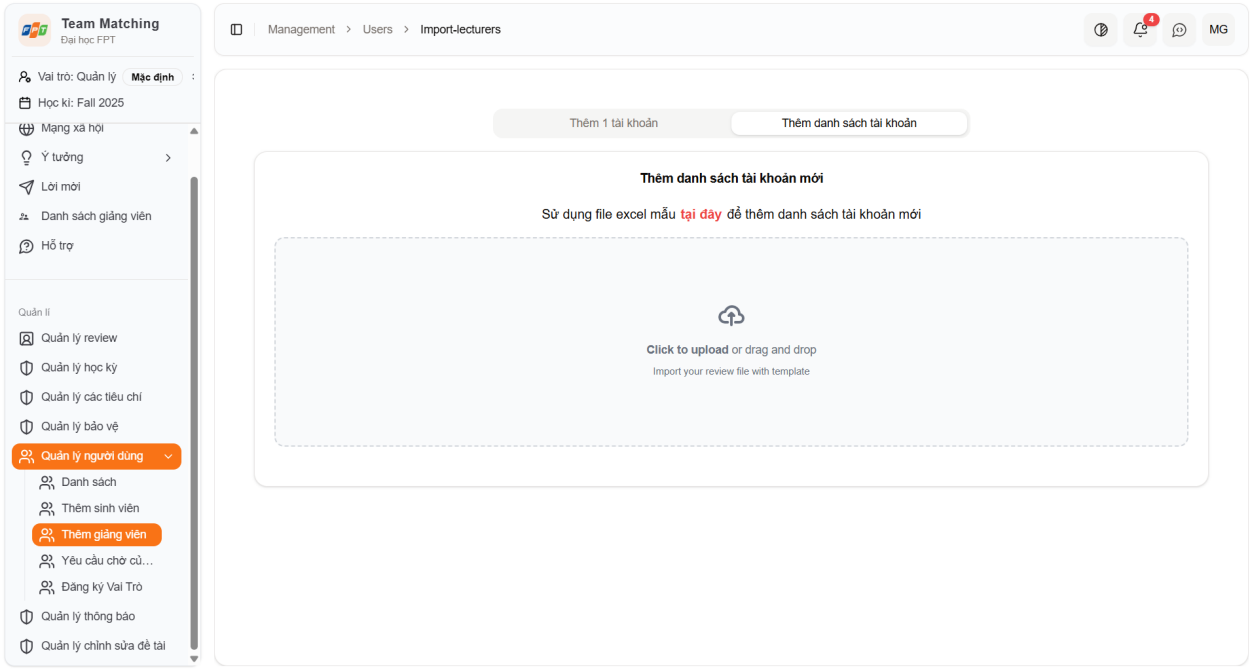
3.3.2 Import students

- **Role:** Manager
- **Function trigger:** Manager logs in → navigates to "Quản lý người dùng" → navigates to "Thêm sinh viên"
- **Function description:**
    - **Purpose:** Manage import students into the system.
    - **Data processing:** Add accounts.
- **Function Details:**
    - Manager import lecturers list into system by excel file



3.3.3 Import lecturers

- **Role:** Manager
- **Function trigger:**
- **Function description:**
    - **Purpose:** Manage import lecturers into the system.
    - **Data processing:** Add accounts.
- **Function Details:**
    - Manager import lecturers list into system by excel file



3.4 Project topic management

3.4.1 Student submit idea proposal

Function Trigger:

Student logs in → navigates to "Tạo ý tưởng"

Function Description:

Actors: Student

Purpose: Allow students to submit their idea proposal.

Interface: A form that includes fields such as abbreviation, description, Vietnamese name, English name, team size, mentor and attached files.

Data Processing: Information is saved to the database with a status of "Pending". A notification is sent to the Mentor.

Function Details:

Validation:

All field : required

Attachment: PDF or DOCX format, max size 10MB

Business Rules:

Student do not have a team or idea approval stage in semester,

Student must be to select the mentor

Each student can only create 1 idea in 1 stage in 1 semester.

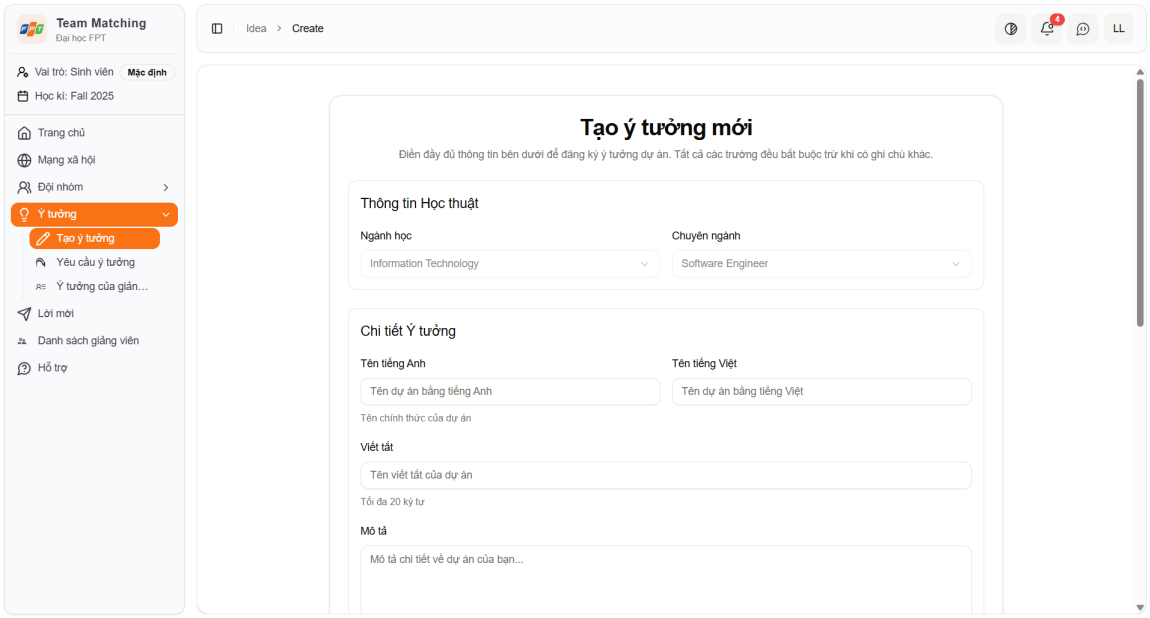
Normal Case:

User fills in all fields → clicks "Gửi ý tưởng" → system saves data and shows a success message.

Abnormal Case:

Missing required fields → show validation messages.

Invalid file → display error.



Tạo ý tưởng mới

Điền đầy đủ thông tin bên dưới để đăng ký ý tưởng dự án. Tất cả các trường đều bắt buộc trừ khi có ghi chú khác.

Thông tin Học thuật

Yêu cầu cập nhật Ngành và Chuyên ngành

* Vui lòng cập nhật cài đặt với thông tin Ngành và Chuyên ngành của bạn trước khi tiếp tục.

Chi tiết Ý tưởng

Tên tiếng Anh

Tên dự án bằng tiếng Anh

Tên tiếng Việt

Tên dự án bằng tiếng Việt

Tên chính thức của dự án

Viết tắt

Tên viết tắt của dự án

Tối đa 20 ký tự

Mô tả

Mô tả chi tiết về dự án của bạn...

Tối thiểu 10 ký tự

Nhóm & Tài liệu

Số lượng thành viên

4 thành viên

Bao gồm cả bạn với vai trò trưởng nhóm

Giảng viên hướng dẫn

Chọn giảng viên

Chọn giảng viên sẽ hướng dẫn dự án của bạn

Giảng viên hướng dẫn 2

Chọn giảng viên 2

Chọn giảng viên sẽ hướng dẫn dự án của bạn

Tài liệu Dự án

Choose File No file chosen

Định dạng chấp nhận: .doc, .docx, .pdf (tối đa 10MB)

Thành viên Nhóm

Bạn sẽ là trưởng nhóm của dự án này

quancmse172093@fpt.edu.vn
(K17 HCM) Cao Minh Quan

Trưởng nhóm

Gửi Ý tưởng

3.4.2 Mentor evaluate student's idea

Function Trigger:

Mentor logs in → navigates to "Duyệt ý tưởng" → selects a pending idea submission.

Function Description:

Actors: Lecturer (Mentor)

Purpose: Allow mentors to review and approve or reject an idea submitted by a student.

Interface: A list of pending ideas with a detail view containing project title, description, attached files, and team members.

Data Processing: The system updates the project's status in the database (Approved, Rejected, Pending, Consider). A notification is sent to the student.

Function Details:

Validation:

All field : required

Attachment: PDF or DOCX format, max size 10MB

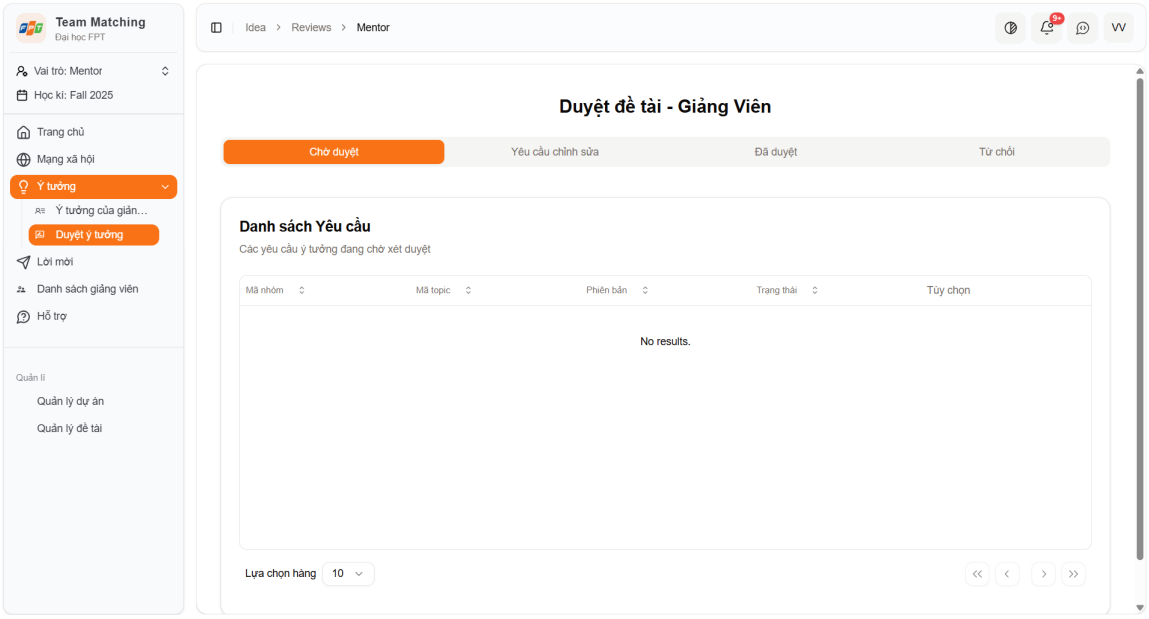
Business Rules:

Only the assigned mentor for a team can evaluate their idea.

Students cannot resubmit a new idea unless the current one is rejected.

Normal Case:

Mentor selects idea → fill form with number of criteria → clicks "Đồng ý" or "Yêu cầu chỉnh sửa " or "Từ chối"→ system processes decision and notifies student.



3.4.3 Mentor submit idea proposal.

Function Trigger:

Mentor logs in → navigates to “Tạo ý tưởng” section.

Mentor logs in → navigates to “Project Topics” → selects a topic (student-submitted) → clicks “Submit to Council” button.

Function Description:

Actors: Lecturerfa

Purpose: Allow a mentor to submit an idea for evaluation and approval by a council of lecturers.

Interface: Mentor selects the project → reviews details → confirms submission. The system will then automatically assign a number of member councils for the topic review.

Data Processing:

Randomly selects a number of council lecturers from the pool of users with Council roles.

Ensures that the mentor who submitted the topic is not included in the council.

Creates review assignments for each council member.

Notifies the assigned council members.

Function Details:

Validation:

All field : required

Attachment: PDF or DOCX format, max size 10MB

Business Rules:

When the lecture generates ideas, if there are more than the specified number of topics, the following topics must have a submentor.

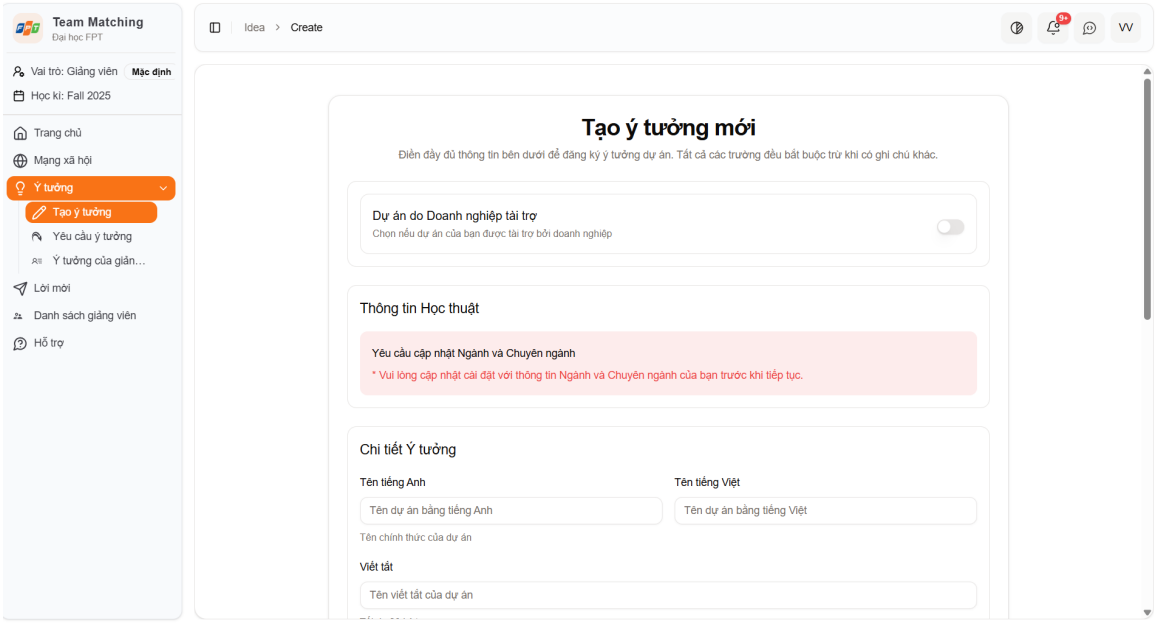
Normal Case:

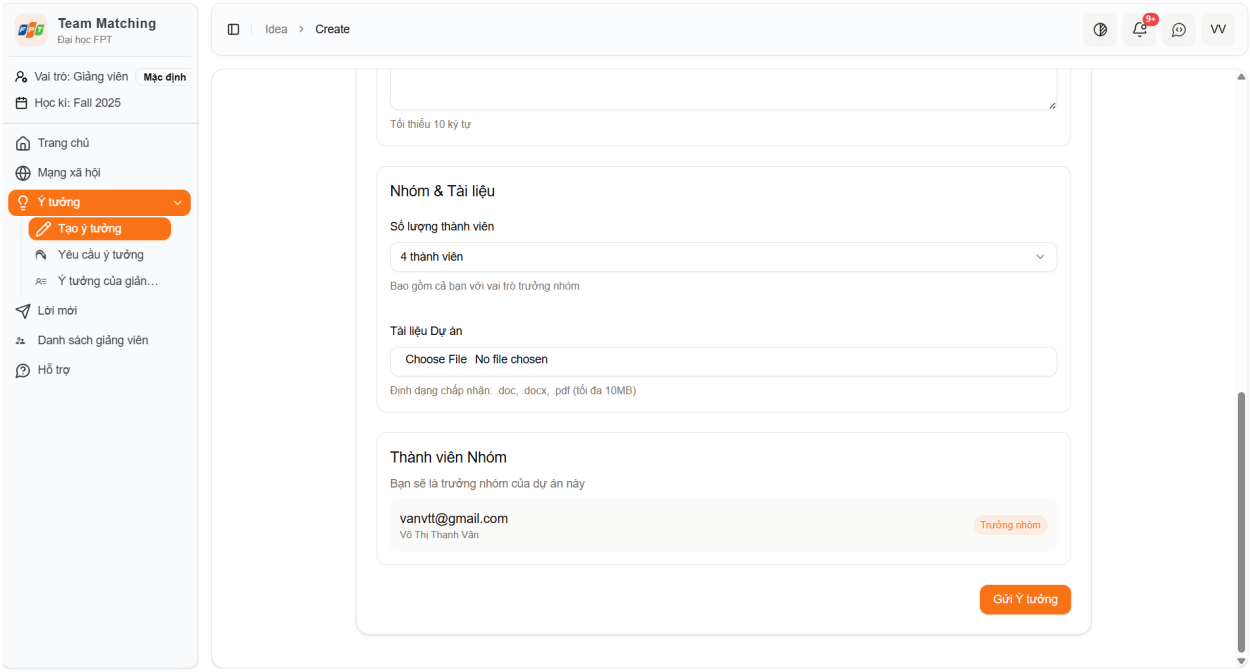
Mentor selects topic → clicks "Nộp cho hội đồng" → system randomly assigns 3 eligible council lecturers → notifications sent.

Abnormal Case:

Not enough eligible council members → error message displayed.

Random assignment fails due to internal error → retry or contact admin.





3.4.4 Council Reviews Submitted Project Topic

Function Trigger:

A lecturer with the role Council logs in → navigates to the "Duyệt ý tưởng" section → views list of topics assigned for review → selects a topic to evaluate.

Function Description:

Actors: Lecturer (Council)

Purpose: Allow assigned council members to review, approve, or reject submitted topic.

Interface: Display a list of topics assigned to the council member, including relevant project details, submitted documents, and evaluation options (e.g., Approve, Request Changes, Reject).

Data Processing:

System ensures only assigned topics are visible to each council lecturer.

Lecturer provides feedback and selects an action (Approve or Reject).

System logs each council member's decision.

Function Details:

Validation:

Only assigned council lecturers can access and evaluate the topic.

Business Rules:

Each topic must be reviewed by exactly 3 council lecturers.

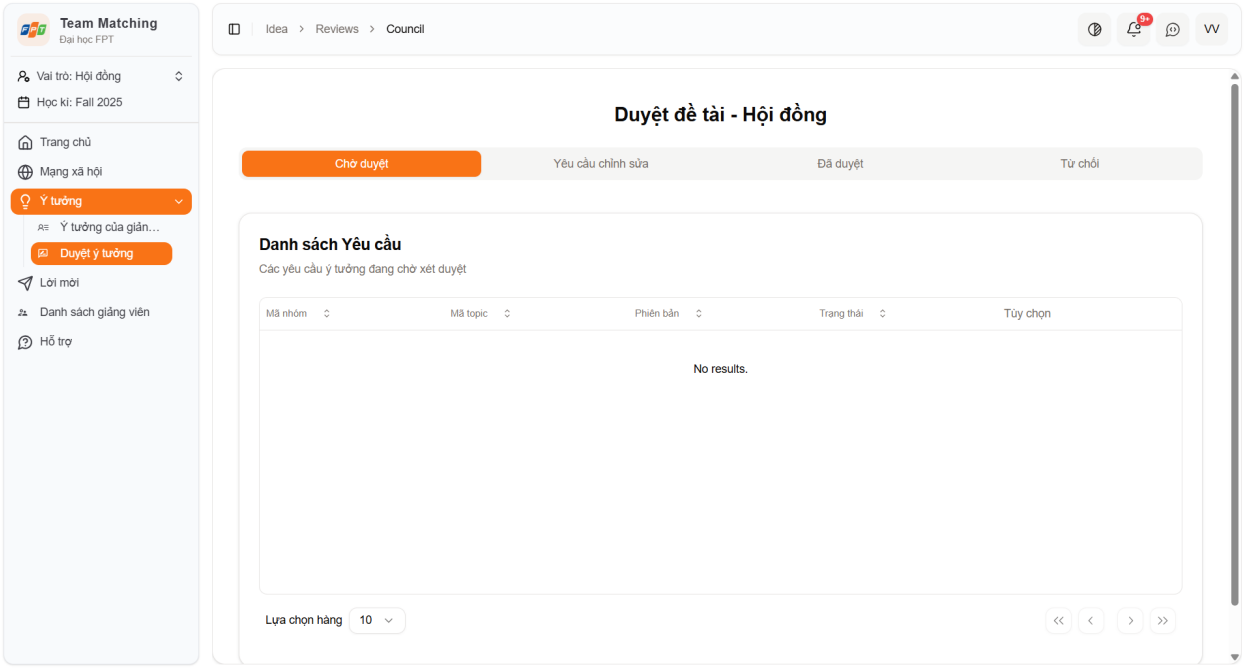
A topic is:

Approved if at least 2 out of 3 council lecturers approve it.

Rejected if 2 or more reject.

Normal Case:

Council lecturer selects a topic → → fill form with number of criteria → clicks “Đồng ý” or “Yêu cầu chỉnh sửa “ or “Từ chối”→ system processes decision and notifies student.



3.5 Team Management

3.5.1 Create Team

Function Trigger:

Student logs in → Navigates to “Đội nhóm” or “Đội của tôi” → Click ”Tạo nhóm”.

Function Description:

Actors: Student

Purpose: To allow a student to create a new team for the project. This team can later be used to register for a project topic and collaborate on the capstone project.

Interface:

A form containing fields such as Team Name.

A “Tạo nhóm” button to submit the form.

Data Processing:

Upon submission, the system validates the provided information and creates a new record in the Teams table.

The student who creates the team automatically becomes the team leader.

A notification is sent to the student confirming team creation.

Function Details:

Validation:

Check that the student is not already a member of any team.

Verify that a pending join request is not already in place for the same team.

Business Rules:

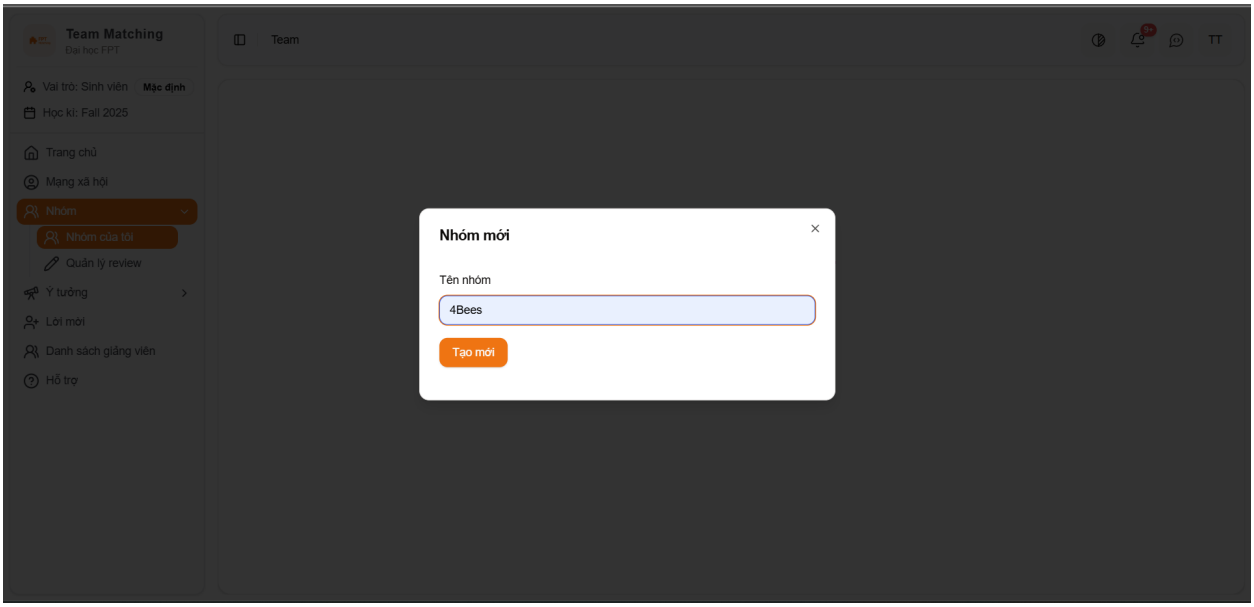
Each student can only create one team per semester but when creating a team, that student has not submitted an idea for approval and has not joined any group.

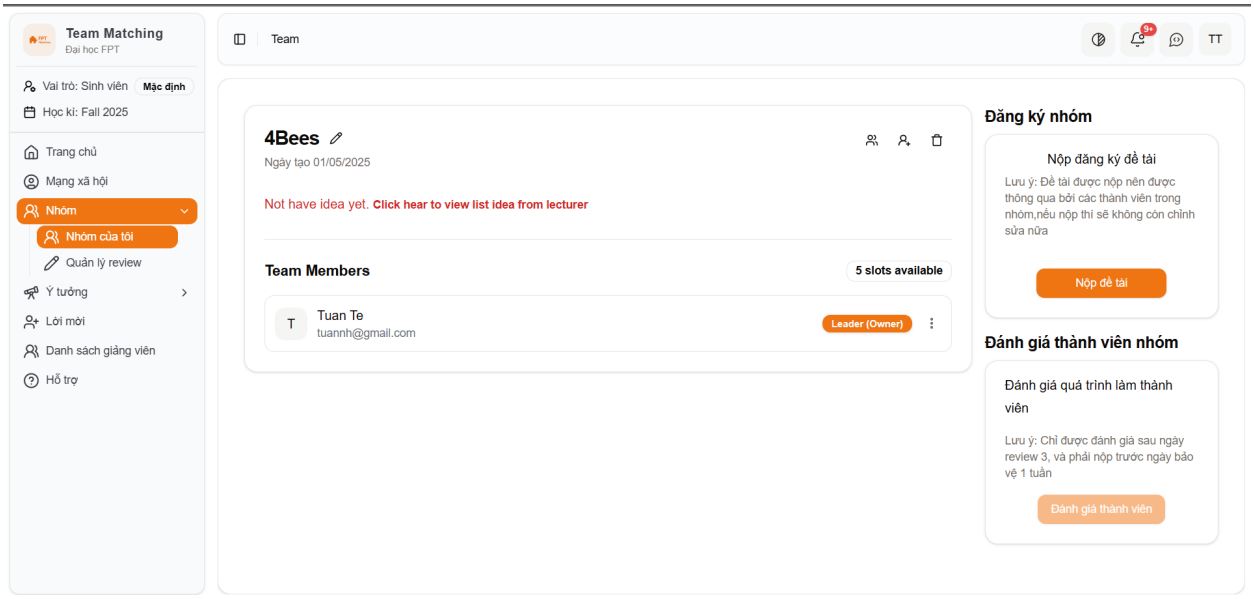
The team leader has the exclusive ability to invite members or approve join requests from other students.

All teams must adhere to the team size limits as defined by the system (e.g., minimum of 4 and maximum of 5 members).

Normal Case:

The student fills in the form with valid data → clicks “Tạo nhóm” → the system validates input, creates a new team record, assigns the student as team leader, and displays a success message (e.g., “Team created successfully.”).





3.5.2 Student Request to Join Team

Function Trigger:

Student views Suggested Team → Clicks “Tham gia”.

Function Description:

Actors: Student

Purpose: Allow the student to send a join request to a selected team.

Interface: A confirmation dialog or popup prompting the student to confirm the join request.

Data Processing:

Record the join request in the Invitation table with status “Pending”.

Notify the team leader of the received request.

Function Details:

Validation:

Check that the student is not already a member of any team.

Verify that a pending join request is not already in place for the same team.

Business Rules:

Each team must adhere to the maximum number of members.

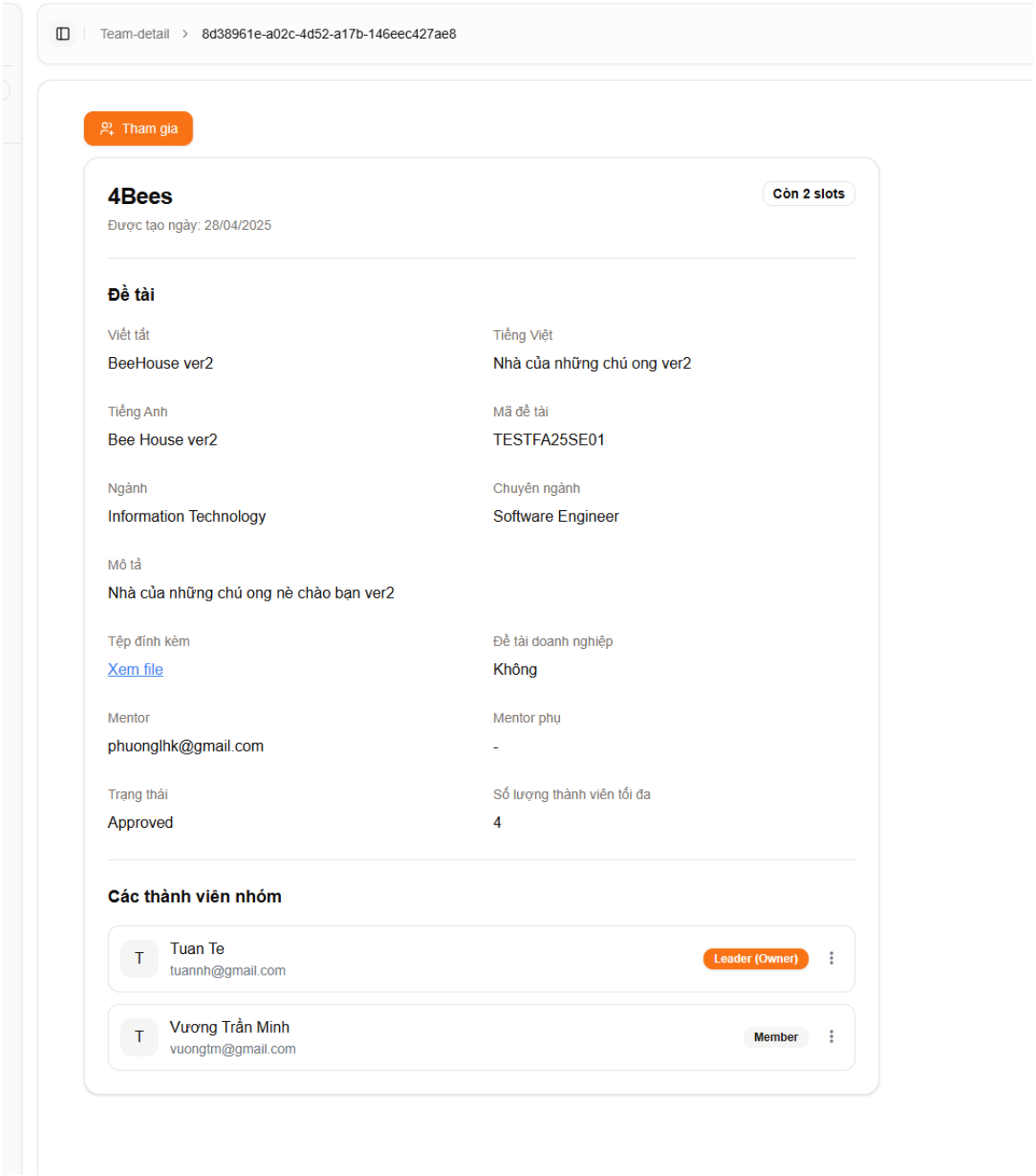
Student must not have a group and have not posted idea approval

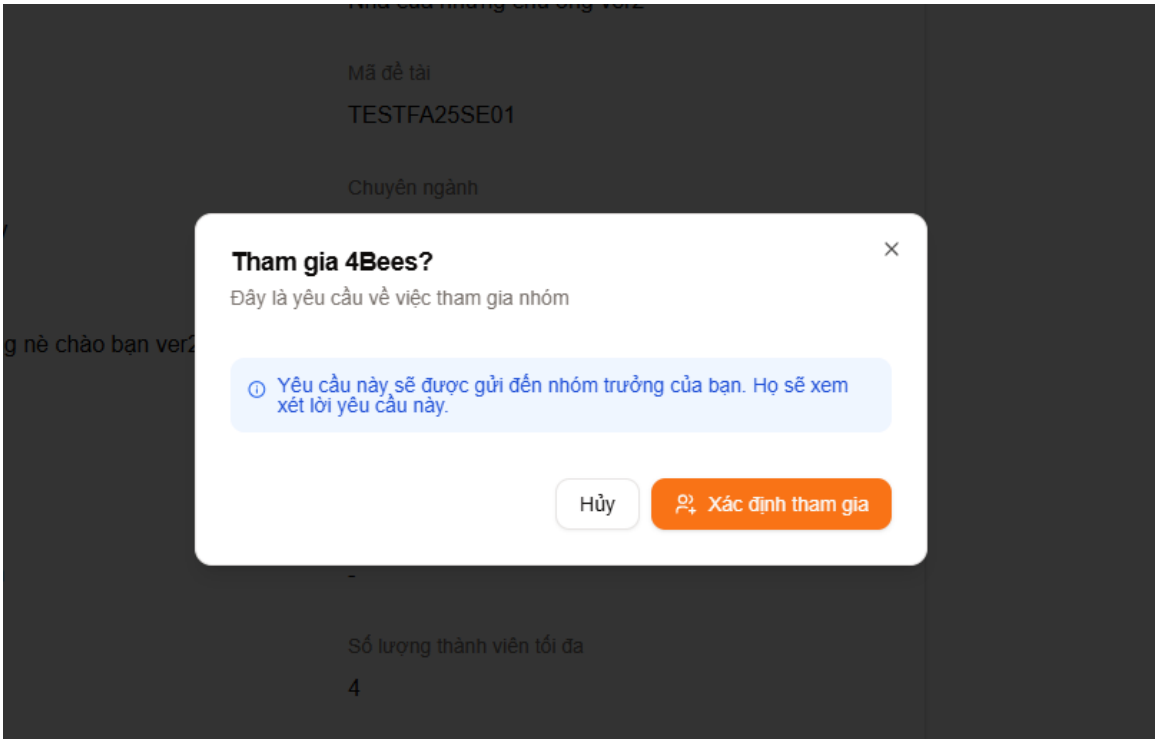
Normal Case:

Request is successfully submitted → pending approval.

Abnormal Case:

Duplicate request or team is full → error message is displayed.


Team-detail > 8d38961e-a02c-4d52-a17b-146eec427ae8
Tham gia
4Bees
Còn 2 slots
Được tạo ngày: 28/04/2025
Đề tài
Viết tắt
BeeHouse ver2
Tiếng Việt
Nhà của những chú ong ver2
Tiếng Anh
Bee House ver2
Mã đề tài
TESTFA25SE01
Ngành
Information Technology
Chuyên ngành
Software Engineer
Mô tả
Nhà của những chú ong nè chào bạn ver2
Tệp đính kèm
Xem file
Đề tài doanh nghiệp
Không
Mentor
phuonglhk@gmail.com
Mentor phụ
-
Trạng thái
Approved
Số lượng thành viên tối đa
4
Các thành viên nhóm
T
Tuan Te
tuannh@gmail.com
Leader (Owner)
T
Vương Trần Minh
vuongtm@gmail.com
Member



3.5.3 Leader Respond to Team Invitation

Function Trigger:

Team Leader logs in → Navigates to icon Student with red dot (within the "Đội của tôi" section) → Views pending join requests.

Function Description:

Actors: Student (Team Leader)

Purpose: To review and respond to join requests from students who wish to join the team.

Interface:

A list of join requests with details such as the applicant's name, profile, and a message (if any).

Action buttons for "Approve" or "Cancel" next to each request.

Data Processing:

Upon approval, the system updates the JoinRequest record to "Approved" and adds the student to the GroupMembers table.

Upon rejection, the record status is updated to "Rejected," and the student is notified accordingly.

Function Details:

Validation:

Only the team leader can access and respond to join requests.

Ensure the team has not reached the maximum number of members before approval.

Business Rules:

Each join request must trigger a notification to the applicant after a decision is made.

Once a join request is approved, the student is added to the team, and no further requests from that student for the same team are processed.

Normal Case:

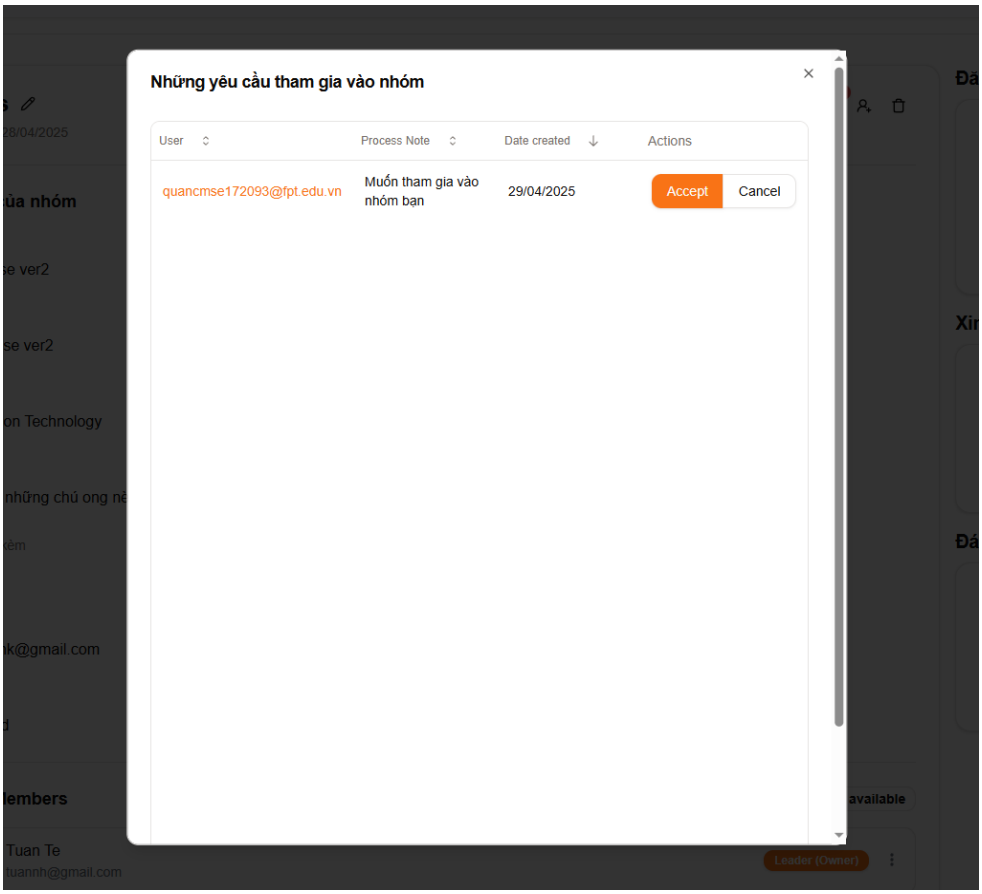
Team leader approves a request → student is added to the team → a confirmation notification is sent.

Or, the team leader rejects a request → the request is marked as “Rejected” → a notification is sent.

Abnormal Case:

If the team is already full at the time of approval → display an error message and prevent approval.

If a decision is attempted on an already processed request → display an appropriate error message.



3.5.4 Team Leader Invites a Student

Function Trigger:

Team Leader logs in → Navigates to student icon with plus (within the "Đội của tôi" section) → Click" Thêm vào nhóm."

Function Description:

Actors: Student (Team Leader)

Purpose: To allow the team leader to proactively invite a student to join the team.

Interface: A form or list displaying potential student profiles (or a simple input field for entering the student's email/ID) along with an "Thêm vào nhóm" button.

Data Processing:

Upon sending an invitation, the system creates a record in the TeamInvitations table with details such as TeamId, InvitedStudentId, InvitationDate, and a status of "Pending."

Function Details:

Validation:

Verify that the student is not already a member of any team.

Ensure that the same student is not invited multiple times concurrently for the same team.

Business Rules:

Each invitation must be unique per team and student combination.

Team have a slot member for student,

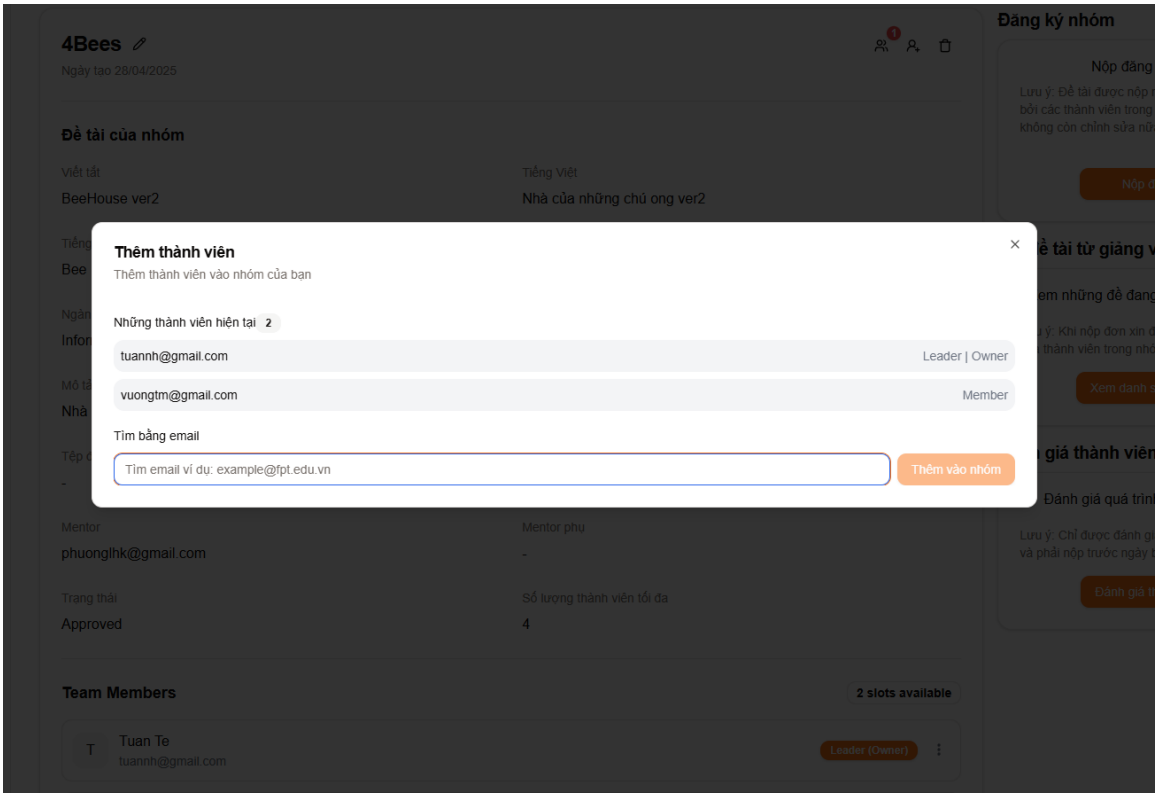
The invitation should trigger a notification to the invited student.

Normal Case:

Team leader selects or enters a student → clicks “Thêm vào nhóm” → system creates the invitation record → notification is sent to the student.

Abnormal Case:

If the student is already part of a team or has an existing pending invitation → the system displays an error message (e.g., “This student has already been invited or is part of another team.”).



3.5.5 Student Respond to Team Invitation

Function Trigger:

Invited student logs in → Navigates to “Lời mời” → Views the pending invitation → Clicks “Accept” or “Cancel.”

Function Description:

Actors: Student

Purpose: To allow the student to accept or decline a team invitation.

Interface:

A list of invitations showing details like Team Name, Inviter’s Name, and any invitation message.

“Accept” and “Cancel” buttons for each invitation.

Data Processing:

If accepted, the system updates the invitation record to “Accepted” and adds the student to the GroupMembers table.

If rejected, the record status is updated to “Cancel,” and the inviter is notified.

Function Details:

Validation:

Verify that the invitation is still valid (not expired or already processed).

Ensure the student is not already part of another team before accepting.

Business Rules:

The response must trigger an immediate notification back to the team leader.

Once an invitation is accepted, all other pending invitations for the student should be invalidated (if the business rule prohibits multi-team membership).

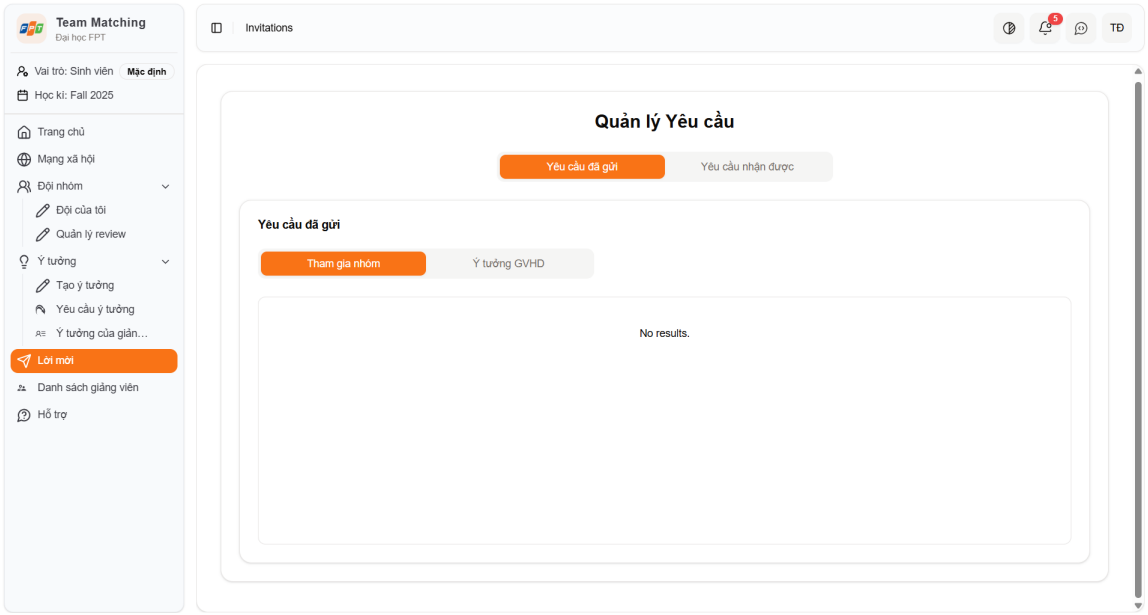
Normal Case:

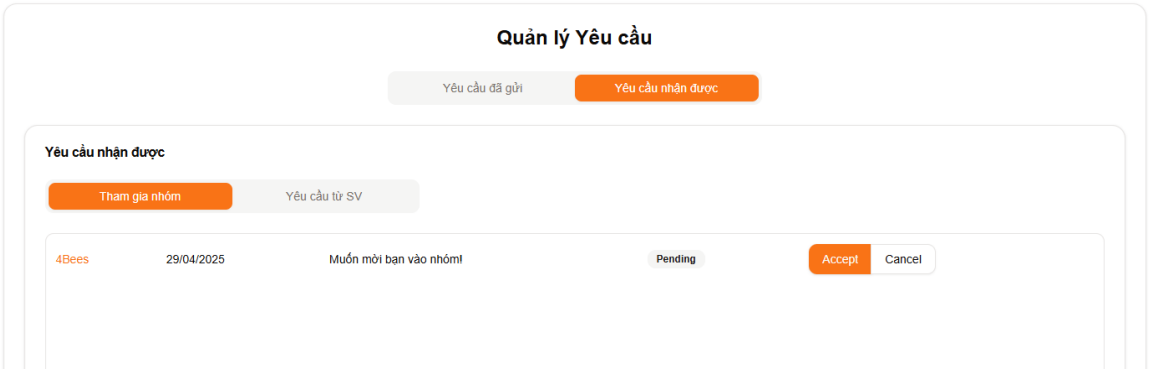
Student clicks “Accept” → system updates the invitation record, adds the student to the team, and sends a notification to the team leader.

Alternatively, when “Cancel” is clicked, the system marks the invitation as “Cancel” and notifies the team leader.

Abnormal Case:

If the invitation has expired or the student is already in a team → an error message is displayed





3.6 Review Management

3.6.1 Export Template Review Schedule and Reviewers

Function Trigger:

Manager logs in → Navigates to the “Quản lí review” menu → Selects “Tải template tại đây”.

Function Description:

Actors: Manager

Purpose: To download an Excel template file used for assigning review schedules and reviewers, which includes pre-filled group information.

Interface: A button labeled “Tải template tại đây” that triggers file download.

Data Processing:

Generate an Excel file containing all student groups with columns such as: Idea code, Team code, Abbreviation, Vietnamese name, English name.

Slot, Room, Date columns are left empty for the Manager to fill in later.

Function Details:

Validation:

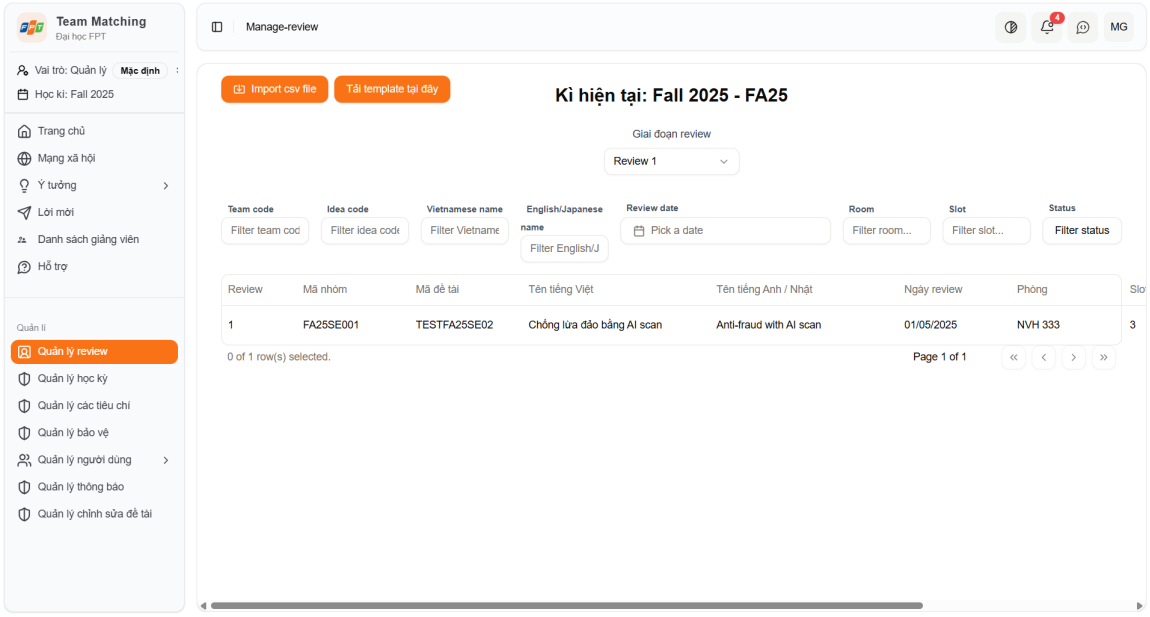
Business Rules:

Only groups that are eligible for review (i.e., submitted an idea or passed previous steps) are included in the exported template.

The exported file must follow the same structure as the import format to ensure compatibility.

Normal Case:

Manager clicks “Tải template tại đây” → System generates the file → Download is triggered successfully



3.6.2 Import Review Schedule and Reviewers

Function Trigger:

Manager logs in → Navigates to the “Quản lí review” menu → Selects “Import csv file”.

Function Description:

Actors: Manager

Purpose: To import an Excel file containing the review schedule and list of reviewers.

Interface: File upload button and a preview table displaying the imported data.

Data Processing:

Parse the uploaded Excel file, validate the format, and store the data into the ReviewSchedule table

Function Details:

Validation:

The uploaded file must be in .xlsx format.

Required columns: as the export file

Business Rules:

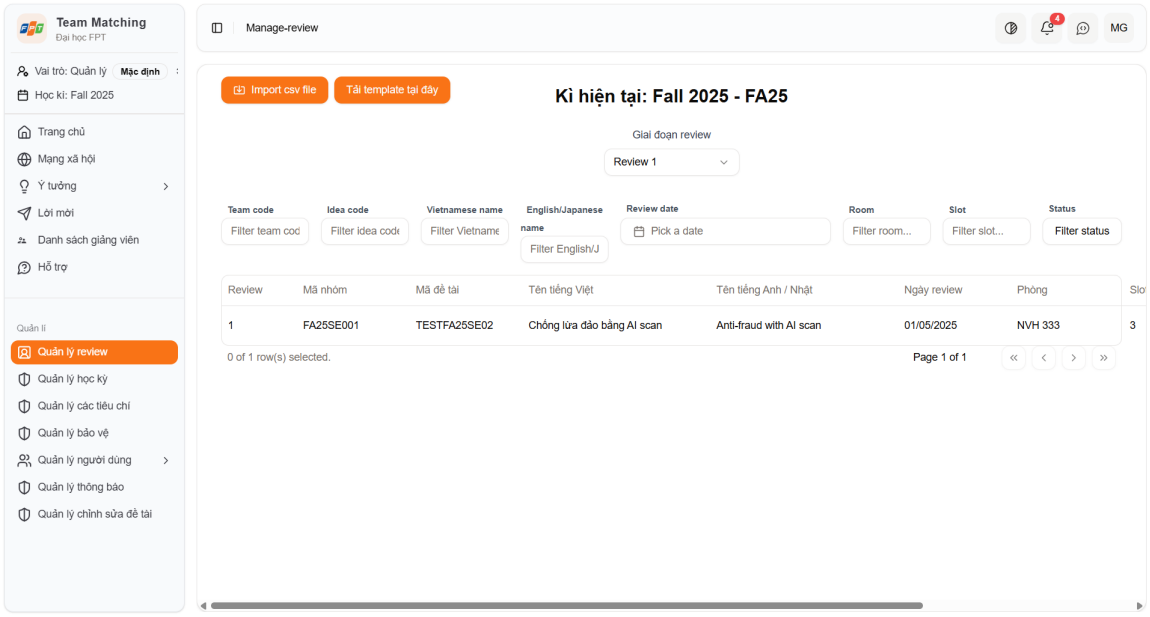
Information of some first columns must match with database

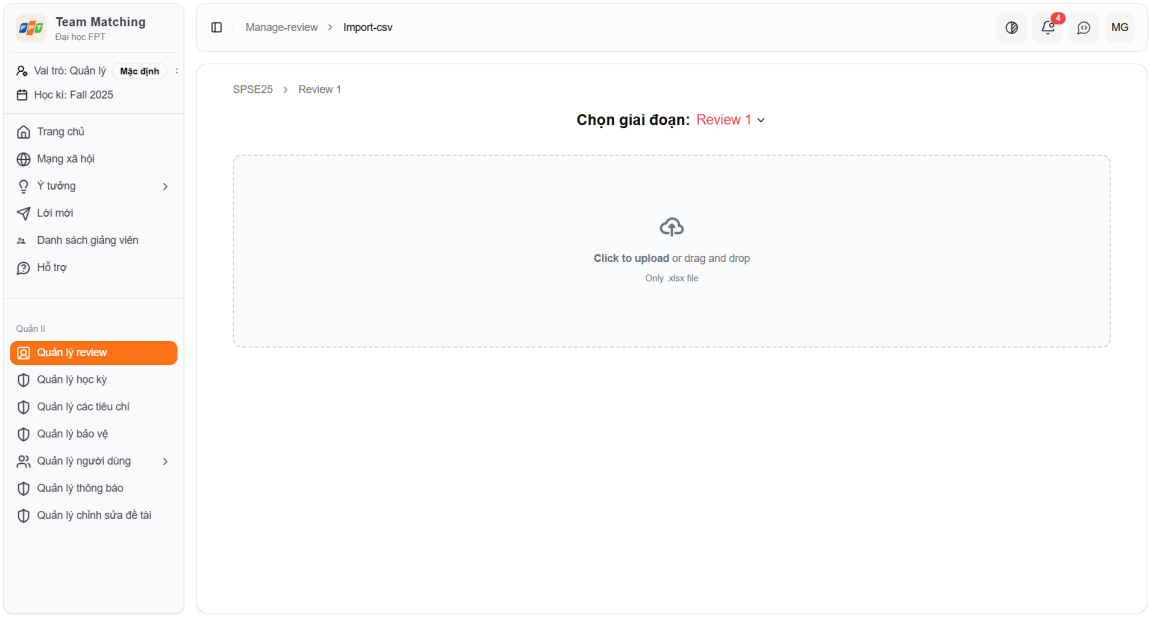
Normal Case:

Valid file → Data preview is displayed → User confirms import → Data is saved successfully → Success notification shown.

Abnormal Case:

Missing columns or invalid format → Detailed error message is shown → Data is not saved.





3.6.3 Student Submit Checklist File

Function Trigger:

Student logs in → Navigates to “My Reviews” → Clicks on “Submit Revision”.

Function Description:

Actors: Student

Purpose: To submit updated documents based on reviewer’s verbal feedback after a review session.

Interface: File upload area and comment box.

Data Processing:

Upload files to server and link to Review table

Function Details:

Validation:

File size & type restrictions (e.g., .pdf, .docx, max 10MB)

Business Rules:

Each review allows only one submission per group.

Normal Case:

Upload file → Confirm → Success message shown.

Abnormal Case:

Wrong file format → Error message.

3.7 Defense Management

3.7.1 Export Template Defense Schedule

Function Trigger:

Manager logs in → Navigates to the “Quản lý bảo vệ” → Selects “Import danh sách bảo vệ”→ Selects “Tải mẫu bảo vệ lần 1 tại đây” .

Function Description:

Actors: Manager

Purpose: To export an Excel template containing student groups that are eligible for final defense, where the manager can fill in the defense time and assign council members.

Interface: A button labeled “Tải mẫu bảo vệ lần 1 tại đây” triggers the file download.

Data Processing:

Generate an Excel file containing all student groups with columns such as: Mã đề tài, Tên đề tài Tiếng Anh, Date, Time, Hall name columns are left empty for the Manager to fill in later.

Function Details:

Validation:

Business Rules:

Only groups marked as eligible for defense are included.

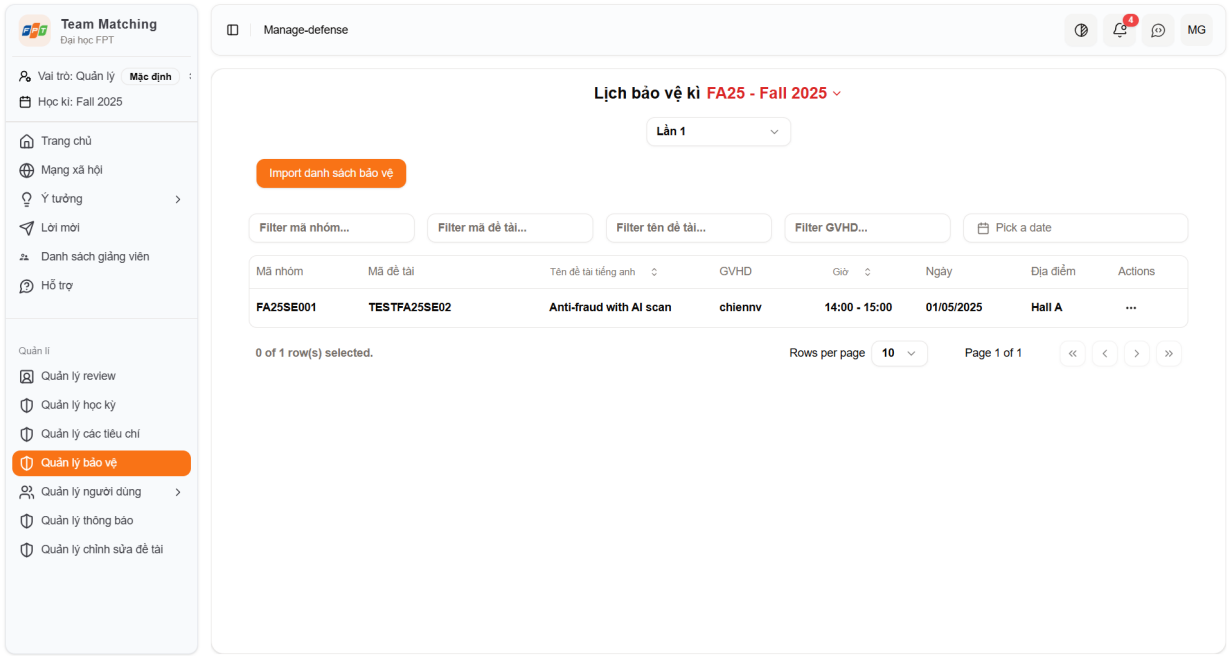
Exported file must match the required format for importing later.

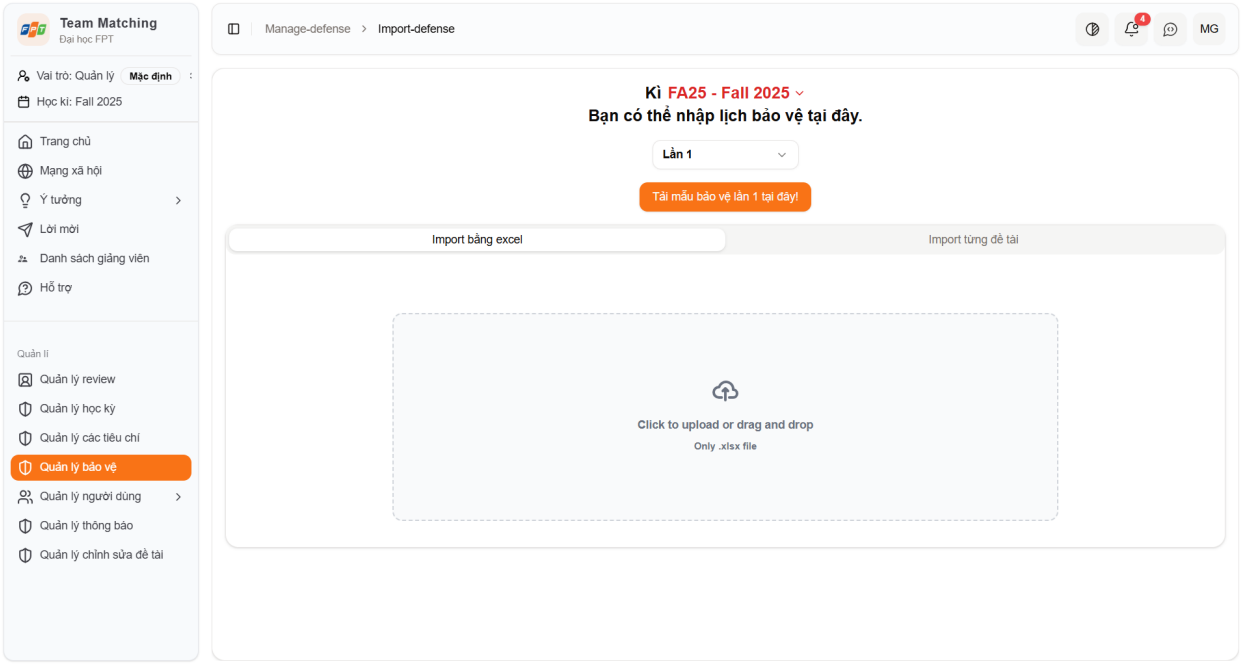
Normal Case:

Manager clicks “Import danh sách bảo vệ” → System generates the file → Download starts automatically.

Abnormal Case:

If no eligible groups are found → Still download file but with no group.





3.7.2 Import Capstone Schedule

Function Trigger:

Manager logs in → Navigates to “Quản lý bảo vệ” → Selects “Import danh sách bảo vệ” → Selects “Click to upload or drag and ”.

Function Description:

Actors: Manager

Purpose: To upload an Excel file that contains defense schedules

Interface: File input for uploading .xlsx file, table preview of parsed data, and a confirmation button.

Data Processing:

Parse the uploaded Excel file, validate the format, and store the data into the CapstoneSchedule table.

Function Details:

Validation:

The uploaded file must be in .xlsx format.

Required columns: as the export file

Business Rules:

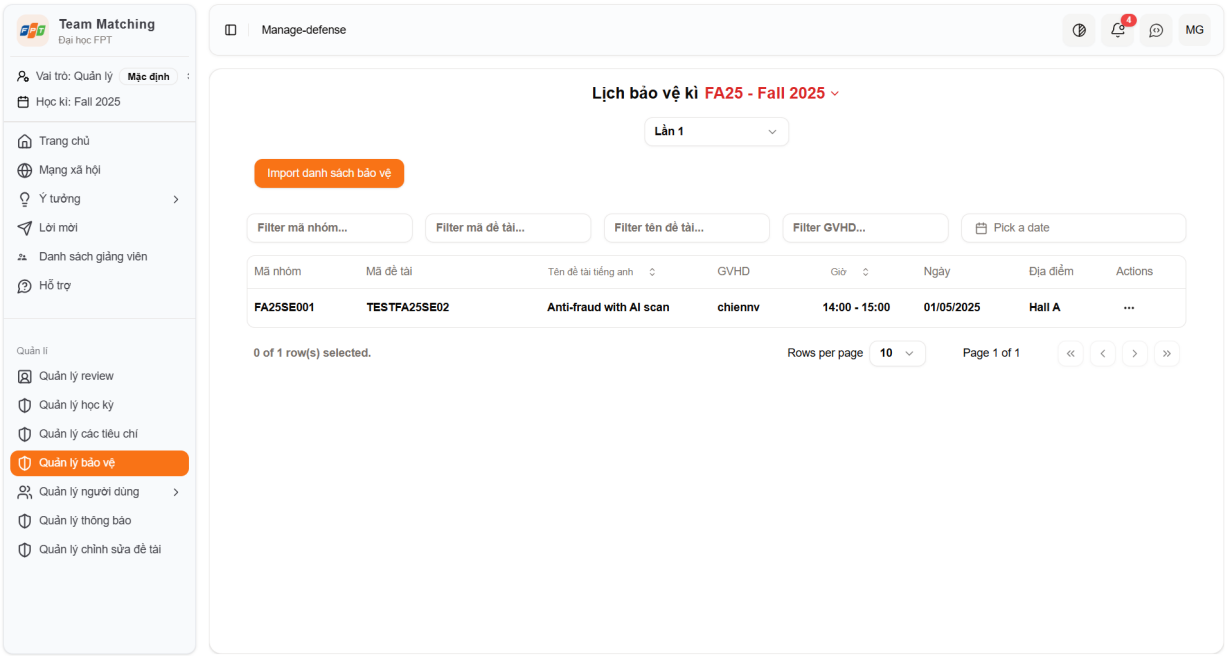
Information of some first columns must match with database

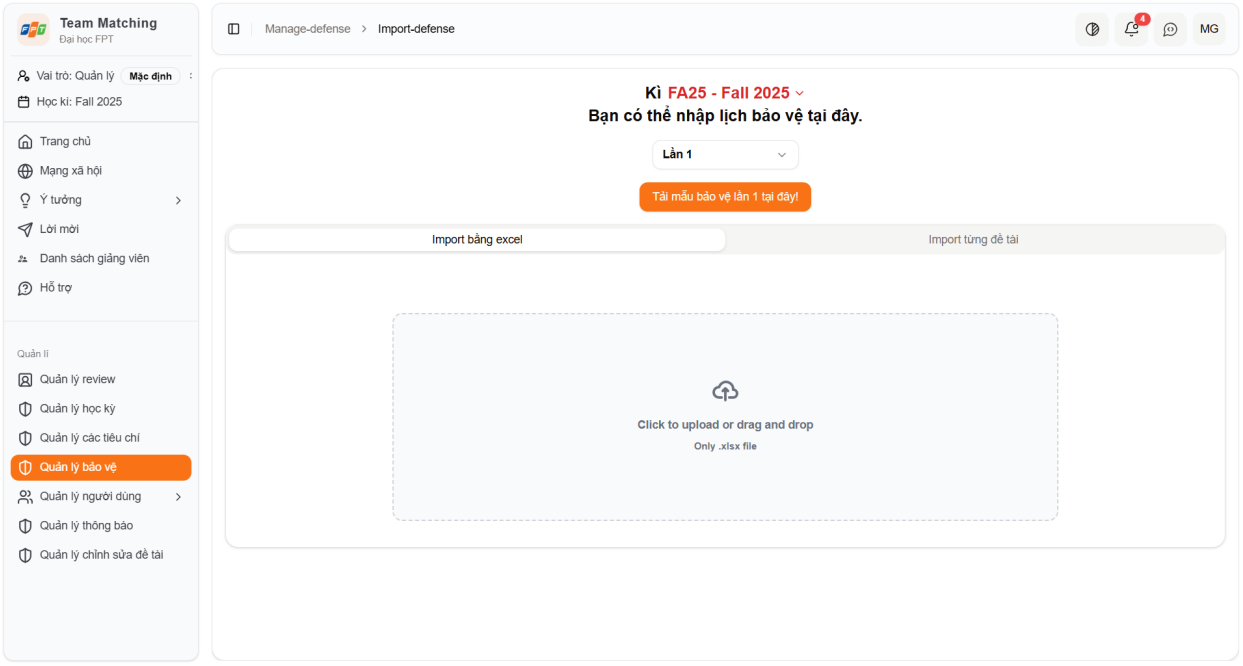
Normal Case:

Valid file → Data preview is displayed → User confirms import → Data is saved successfully → Success notification shown.

Abnormal Case:

Missing columns or invalid format → Detailed error message is shown → Data is not saved.





3.8 Timeline Management

3.8.1 Manage Timeline for Reviews and Defenses

Function Trigger:

Manager logs in → Navigates to “Quản lí học kì” → Selects which semester user want to

Function Description:

Actors: Manager

Purpose: To allow the manager to view, create, and adjust the schedule for 3 review sessions and 2 defense sessions for the project.

Interface:

A dashboard that displays a calendar or timeline view with pre-defined time slots for Review 1, Review 2, Review 3, Defense 1, and Defense 2.

Input fields or drag-and-drop controls for updating dates and times for each session.

Data Processing:

Retrieve existing session schedules from Timeline table.

Allow manager modifications and validate the changes before saving updates.

Function Details:

Validation:

Ensure that the input dates and times are valid (e.g., future dates, correct time format).

No overlapping of review and defense session times.

Each review and defense session is unique and must be scheduled exactly once per project cycle.

Business Rules:

There must be exactly 3 review sessions and 2 defense sessions defined per semester.

The timeline for reviews should precede the timeline for defenses (e.g., the first defense must be scheduled after the third review).

Changes must be logged, and notifications might be sent to concerned actors (students, mentors) upon timeline updates.

Normal Case:

Manager adjusts the timeline (either by typing new dates or using a drag-and-drop interface) → clicks “Save” → the system validates the input, updates the schedule in the database, and displays a confirmation message ("Timeline updated successfully").

Abnormal Case:

If an entered date/time is invalid (e.g., in the past, wrong format) → system shows a validation error next to the affected field.

If there is an overlap or a session is attempted to be scheduled out of the logical order (e.g., a defense scheduled before the final review) → system displays an error message (e.g., “Defense sessions must be scheduled after the final review session”).

System errors during saving → error message is shown and changes are not applied.

3.9 Blog Management

3.9.1 Create Blog Post

Function Trigger:

User (Student or Lecturer) logs in → Navigates to “Blog” → Clicks “Create New Post” button.

Function Description:

Actors: Student

Purpose: To allow authorized users to write and publish blog posts related to projects, academic topics.

Interface:

A rich text editor screen with fields for blog title, content

Buttons for “Post bài”

Data Processing:

The system saves the post as either a draft or a published entry in the blog database.

It updates the blog feed and optionally triggers notifications to relevant users (followers, team members, or interested parties).

Function Details:

Validation:

Title and content fields are mandatory.

Business Rules:

Only authorized users can create posts (e.g., students and lecturers with verified accounts).

Posts may be subject to moderation if flagged by the system or an admin.

Normal Case:

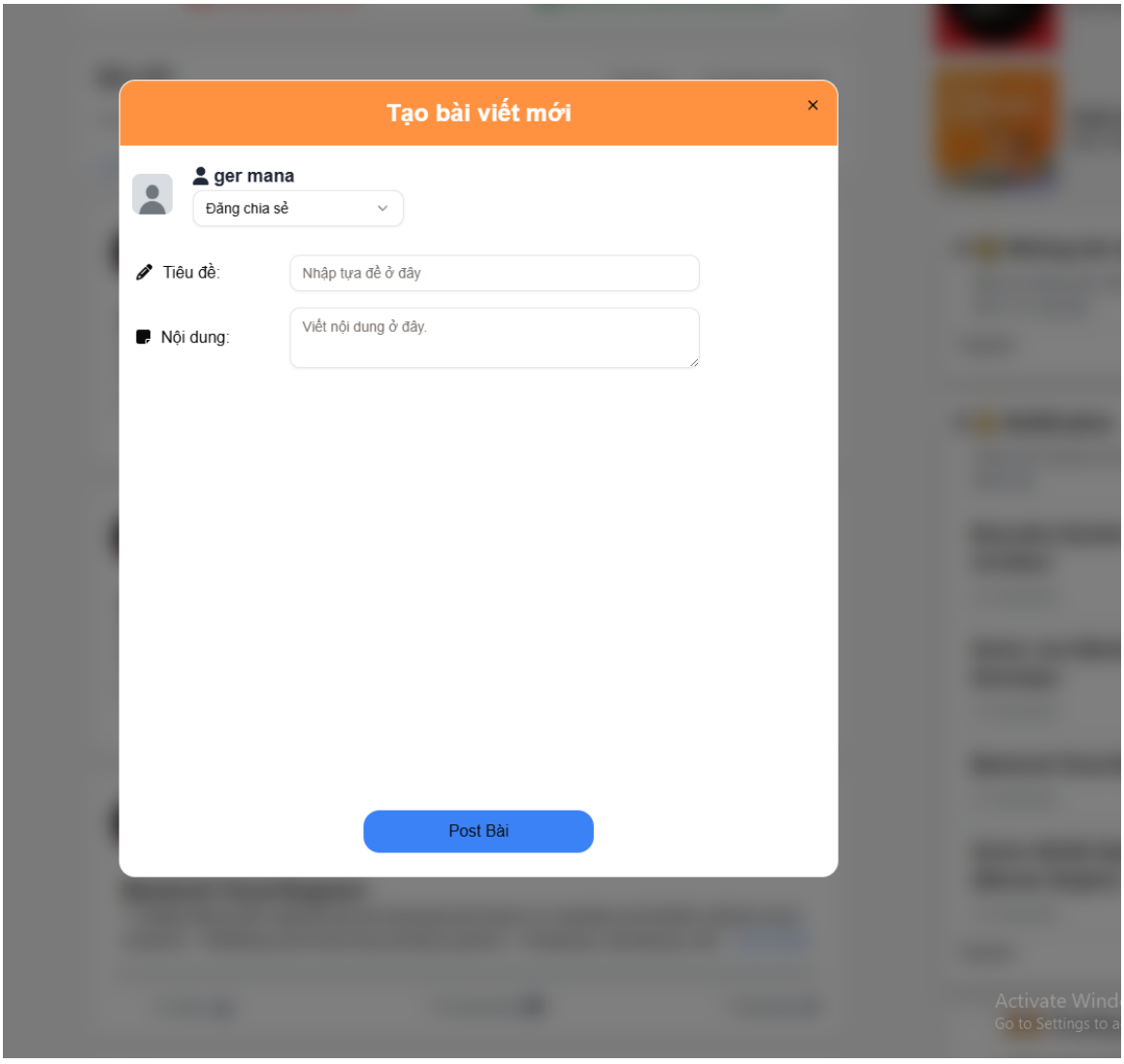
User enters valid information → clicks “Post bài” → Blog is stored in the system and appears on the blog feed.

Abnormal Case:

Missing title or content → display inline error messages.

Image format/size invalid → prompt user to select a different file.

System error during saving → display an error message and suggest retrying later.



3.9.2 Interact with Blog Post (Like and Comment)

Function Trigger:

User logs in → Navigates to the blog feed → Selects a blog post to view details → Clicks on “Like” button or enters a comment.

Function Description:

Actors: Student

Purpose:To allow users to engage with blog posts by liking, commenting, and possibly sharing posts.

Interface:

A detailed view of a blog post with a “Like” button (showing a count), a comment section with a text input, and a “Submit Comment” button.

Data Processing:

When a like is registered, the system increases the like count in the database.

When a comment is submitted, the system stores the comment linked to the specific post and displays it below the post.

Function Details:

Validation:

For comments: the comment text must not be empty.

Ensure that each like or comment is linked to a valid blog post ID.

Business Rules:

Each user can like a post only once (toggle like if clicked again).

Comments should be moderated for inappropriate content by a profanity filter or admin oversight.

Normal Case:

User clicks “Like” → system records the like and updates the count.

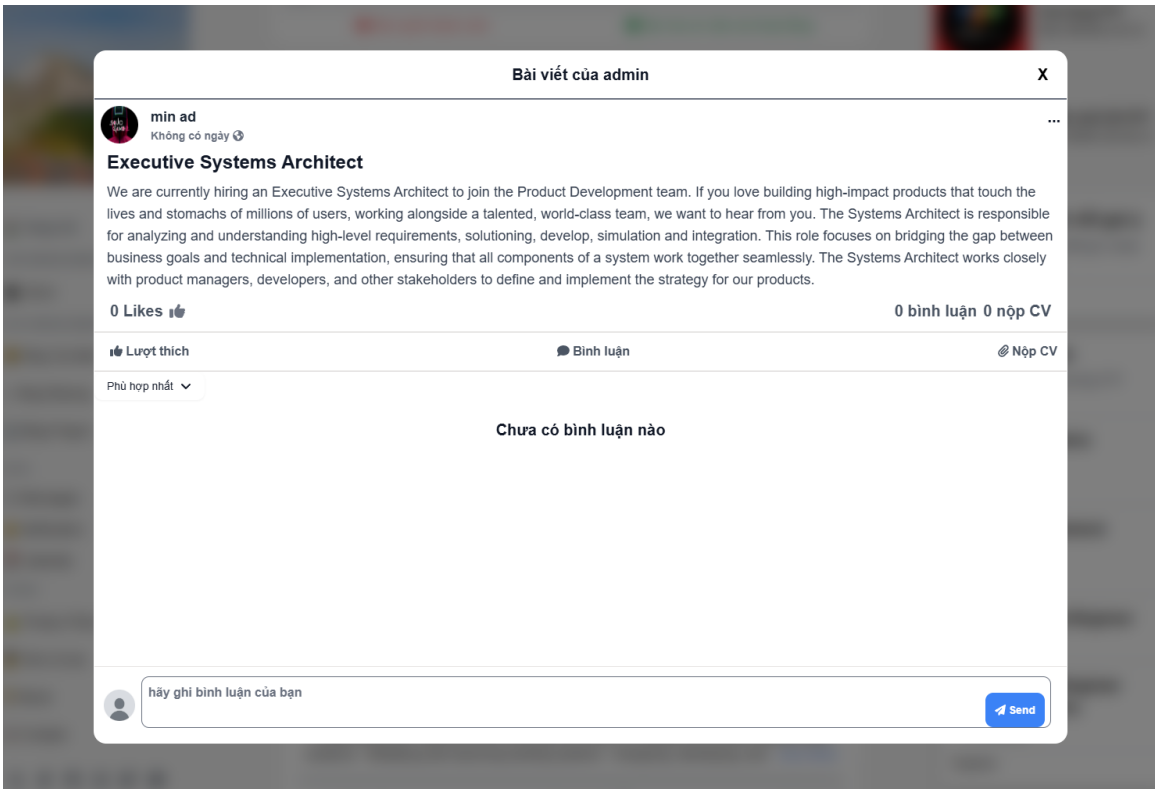
User enters a comment → clicks “Submit Comment” → comment appears immediately on the post.

Abnormal Case:

Like action fails (e.g., network error) → display an error message prompting retry.

Comment submission with empty text → prompt user with “Please enter a comment.”

If the comment contains prohibited words → system displays a validation error and does not save the comment.



3.9.3 Apply CV to Team Recruitment Blog Post

Function Trigger:

A student views a team recruitment blog post → Clicks the “Apply CV” button/option on the post.

Function Description:

Actors: Student

Purpose: To allow a student interested in joining a team to apply by attaching their CV to the recruiting blog post.

Interface:

A form on the blog post detail page, including an upload field for the CV file, an optional cover letter/text area, and an “Apply” button.

Display of the blog post with recruitment details alongside the application form.

Data Processing:

Upon submission, the system validates and uploads the CV file.

It then creates an application record linking the student, the specific blog post, and any provided cover letter.

The system notifies the team leader or recruiting party of the new application.

Function Details:

Validation:

The uploaded CV file must be in an accepted format (e.g., PDF or DOCX) and should not exceed the maximum file size limit (e.g., 5MB).

The form must ensure that a file is attached.

Optional cover letter input may be validated for a maximum character limit (e.g., 1000 characters).

Business Rules:

A student can only apply once to a specific team recruitment blog post.

The application must include a valid CV file; otherwise, the submission will be rejected.

Once an application is submitted, the student cannot modify it unless the application has been rejected or the recruitment period is still open for updates.

Students must not have a team and have not posted for idea approval

Normal Case:

The student selects a valid CV file, optionally enters additional information, and clicks “Apply”.

The system successfully uploads the file, creates a corresponding application record, and shows a confirmation message (e.g., “Your application has been submitted successfully.”)

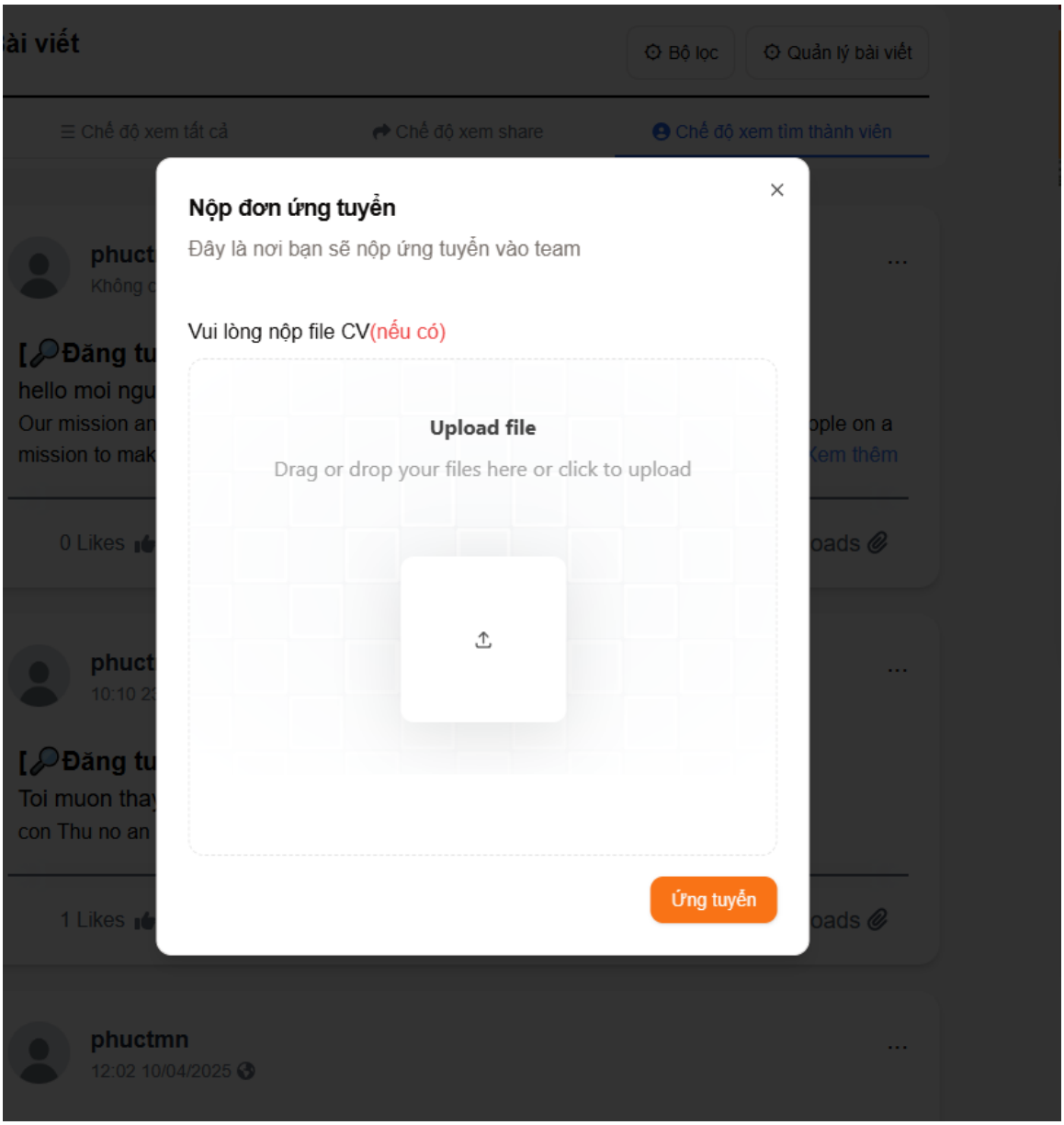
A notification is sent to the recruiting team leader.

Abnormal Case:

If no file is attached or the file format/size is invalid, the system displays an inline error message (e.g., “Please attach a valid CV in PDF or DOCX format, not exceeding 5MB.”)

If the student has already applied to the same blog post, the system will block duplicate applications and display an error message.

In case of system errors during file upload or record creation, the system displays a generic error message (e.g., “An error occurred. Please try again later.”).



3.10 Semester Management

3.10.1 Manage Semester

Function Trigger:

Manager or Admin logs in → Navigates to “Semester Management” → Clicks “Manage Semesters” or “Add/Edit Semester”.

Function Description:

Actors: Manager

Purpose: To allow authorized users to create, update, and manage semester details for the project. This includes setting the start and end dates, review session timelines, defense session dates, and other key milestones related to each semester.

Interface:

A dashboard listing all existing semesters with options to “Add New”, “Edit”, or “Delete”.

A form for inputting semester details, including fields such as Semester Name, Start Date, End Date, Review Dates (Review 1, Review 2, Review 3), Defense Dates (Defense 1, Defense 2), and optionally additional notes.

Data Processing:

When creating or editing a semester, the system validates the input, updates the Semester table in the database, and automatically links associated schedule details for reviews and defenses.

The system may recalculate or adjust dependent timelines in other modules based on semester changes.

Function Details:

Validation:

All fields (Semester Name, Start Date, End Date, etc.) are required.

Date fields must follow the proper format (e.g., YYYY-MM-DD) and logically be set in the future (or as applicable).

Review and defense dates must fall within the semester’s start and end dates.

No overlapping semester periods are allowed.

Business Rules:

Normal Case:

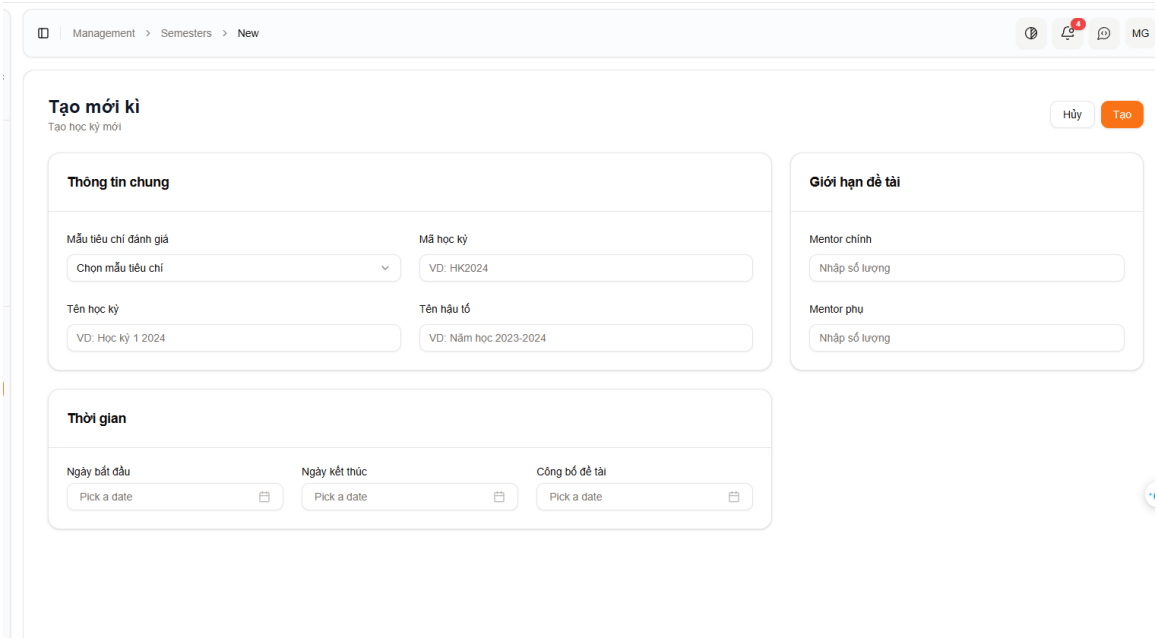
The user fills in all fields with valid data → clicks “Save” → the system validates the input, updates the semester schedule in the database, and displays a success message (e.g., “Semester details updated successfully.”).

Abnormal Case:

If any required field is missing or incorrectly formatted → the system highlights the error and displays an inline validation message.

If review or defense dates are set outside of the semester period or in an incorrect order → the system displays an error message indicating the logical conflict (e.g., “Defense sessions must be scheduled after the final review session and within the semester duration.”).

If the new semester period overlaps with an existing semester → the system blocks the action and alerts the user.



3.11 Criteria Form Management

3.11.1 Manage Criteria Form

Function Trigger:

Manager logs in → Navigates to “Form Management” → Clicks “Add/Edit Form,Criteria”.

Function Description:

Actors: Manager

Purpose: To allow authorized users to create, update, and manage form details for the project. This includes setting the start and end dates, and some criteria in the form , and set form each semester

Interface:

A dashboard listing all existing semesters with options to “Add New”, “Edit”, or “Delete”.

A form for inputting form details, including fields such as title, and criteria including field , question,value type,...

Data Processing:

When creating or editing a form and criteria, the system validates the input, updates the Form , Criteria table in the database, and automatically links associated schedule details for reviews and defenses.

The system may recalculate or adjust dependent timelines in other modules based on semester changes.

Function Details:

Validation:

All fields (Title, Start Date, End Date, etc.) are required.

Date fields must follow the proper format (e.g., YYYY-MM-DD) and logically be set in the future (or as applicable).

Review and defense dates must fall within the semester’s start and end dates.

No overlapping semester periods are allowed.

Form can reused each semester

Business Rules:

Only managers have the right to create forms and create criteria in the form, mandatory criteria must be filled in.

Only lecturers have the right to evaluate students.

Only councils have the right to evaluate lectures.

Normal Case:

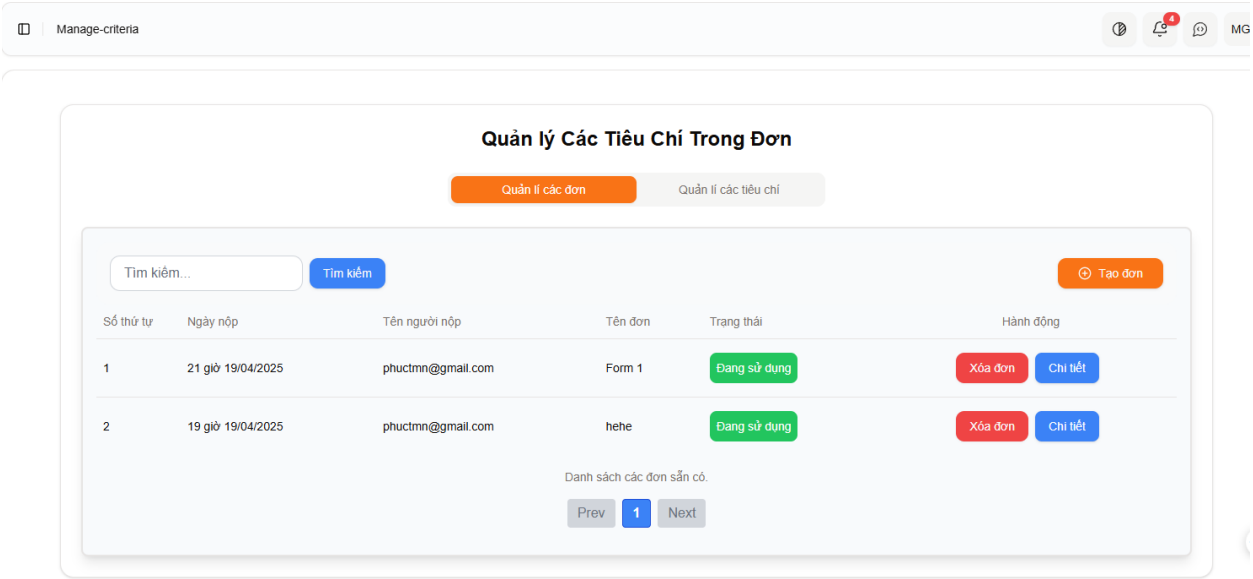
The user fills in all fields with valid data → clicks “Save” → the system validates the input, updates the semester schedule in the database, and displays a success message (e.g., “Semester details updated successfully.”).

Abnormal Case:

If any required field is missing or incorrectly formatted → the system highlights the error and displays an inline validation message.

If review or defense dates are set outside of the semester period or in an incorrect order → the system displays an error message indicating the logical conflict (e.g., “Defense sessions must be scheduled after the final review session and within the semester duration.”).

If the new semester period overlaps with an existing semester → the system blocks the action and alerts the user.



4. Non-Functional Requirements

4.1 External Interfaces

To ensure that the system communicates effectively with both Customers and external hardware/ software elements, the following interfaces must be clearly defined:

- **Customer Interface:**
    - The interface must be Customer-friendly, intuitive, and easy to use for all groups (customers, administrators, etc.).
    - It should adhere to popular GUI design standards (e.g., Microsoft GUI standards, IBM CUA) so that Customers can quickly adapt and reduce training time.
    - It must support multiple platforms, ensuring compatibility on desktop computers, tablets, and smartphones.
- **Payment Gateway Interface:**
    - Integrate online payment gateways such as VNPAY, payOS, etc., ensuring secure data exchange, encryption, and proper Customer authentication during financial transactions.
- **Shipping and Delivery System Interface:**
    - Integrate APIs to track order statuses and update shipping information in real time.
    - Ensure data synchronization between the sales system and the delivery partner's system.

4.2 Quality Attributes

The system must meet various quality attributes regarding usability, reliability, and performance as outlined below:

4.2.1 Usability

Customer-friendly and intuitive interface:

- Design the interface based on popular GUI design standards (Microsoft, IBM CUA) to ensure a smooth Customer experience.
- Ensure a clear, well-organized layout that is accessible for both novice and advanced Customers.

Training and Familiarization Time:

- For basic Customers: training should not exceed 1 hour to master key operations (placing orders, making payments, tracking orders).
- For advanced Customers (administrators, warehouse managers): training should last approximately 2-3 hours, accompanied by detailed documentation.

Usability Metrics:

- Establish measurable indicators, such as the time taken to complete critical tasks (for example, the time from placing an order to completing the transaction should not exceed 2 minutes).
- Gather Customer feedback via post-use surveys to assess and further refine the system's usability.

4.2.2 Reliability

Availability:

- The system should maintain a minimum uptime of 99.9% annually, ensuring it is always available to serve Customers.
- Schedule regular maintenance during low-traffic periods to minimize disruptions.

MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair):

- Set a target MTBF of over 10,000 continuous operating hours for critical components.
- Ensure that the MTTR for serious issues does not exceed 30 minutes to minimize business impact.

Accuracy and Error Control:

- Guarantee that transaction data and order information are updated accurately in accordance with internal standards and industry norms.
- Clearly define error levels (minor, significant, critical), with "critical" errors referring to issues that completely disable key functionalities such as payment processing or order management.

4.2.3 Performance

- **Response Time:**
    - Each transaction (e.g., placing an order, making a payment) should have an average response time of under 2 seconds, with a maximum of 5 seconds under high load conditions.
- **Throughput:**
    - The system must be capable of processing at least 1,000 transactions per minute during periods of high simultaneous Customer activity.
- **Scalability and Concurrent Customers:**

- Design the system architecture to support up to 10,000 concurrent Customers without degradation in performance.
- Consider employing load balancing and database optimization strategies (using PostgreSQL) to maintain stable performance.

- **Resource Management:**
  - Optimize system resource usage (CPU, memory, network bandwidth, disk storage) to ensure efficient operation even under high load conditions.

Implement monitoring and automatic alert mechanisms to detect and respond to resource usage that exceeds established thresholds

5. Requirement Appendix

5.1 Business Rules

ID	Rule Definition
BR-01	A student can be in one project at a time.
BR-02	A student can register one topic at a time.
BR-03	A project must have at least 4 and at most 6 students in a team.
BR-04	Each idea must have at least one mentor
BR-05	Once a student accepts an invitation, they are officially added to the project team.
BR-06	All major project updates will trigger notifications for relevant users.
BR-07	Each review stage will have two assigned reviewers to provide comments.
BR-08	Student has idea is being pending or approved can not apply in other project
BR-09	Students and lecturers can submit project proposals only during the designated proposal submission periods.
BR-10	From 5th topic, mentor’s topic needs submentor
BR-11	When request mentor’s topic, team size of team must be equal to team size of topic
BR-12	Manager has to give feedback for students after review 3 and 1 week before the defense 1 start

BR-13	Manager has to import defense 1 schedule after review 3 and 1 week before the defense 1 start
BR-14	Manager has to import defense 2 schedule after review 3 and 1 week before the defense 2 start
BR-15	When the team size of team equal to the team size of topic, project will block
BR-16	
BR-17	
BR-18	
BR-19	
BR-20	

5.2 Common Requirements

Authentication and Authorization: Role-based access control (RBAC) must be enforced to ensure that users only access the functionalities relevant to their role.

Data Privacy: The system must adhere to data privacy guidelines, ensuring that student data is handled securely and is not exposed to unauthorized users.

Scalability: The system must be scalable to handle increased numbers of students, grading processes, and exam submissions without performance degradation.

5.3 Application Messages List

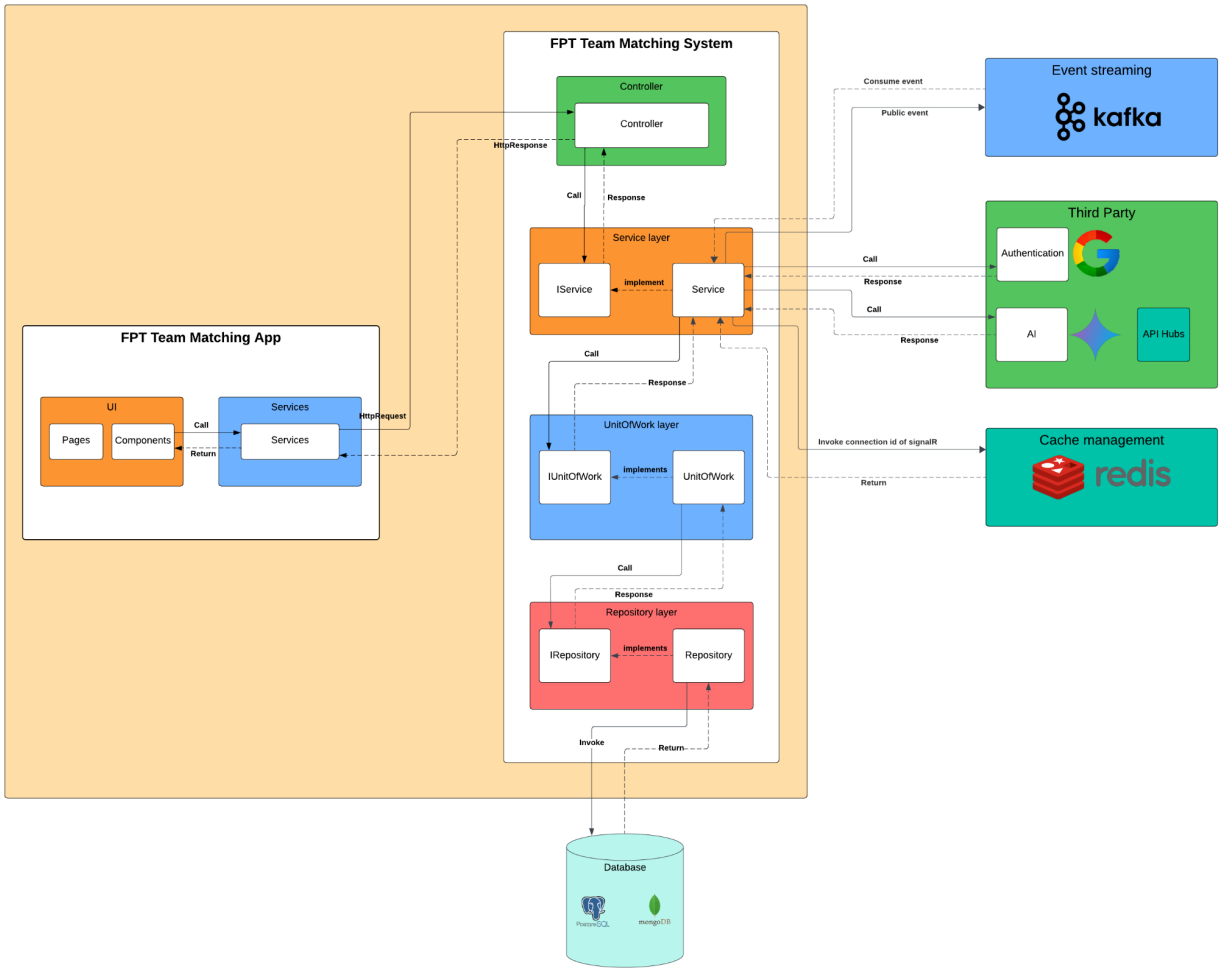
#	Message code	Message Type	Context	Content
1	MSG01	Error	Submission Failure	*"Failed to upload the source code. Please try again."*
2	MSG02	Success	Submission	*"All Student's source code has been successfully uploaded."*
3	MSG03	Error	Invalid File Format	*"Invalid file format. Please upload a .zip file."*
4	MSG04	Info	Grading Status	*"The grading process is currently in progress."*
5	MSG05	Success	Grading Completed	*"Your exam has been graded. Please check your results."*
6	MSG06	Error	Appeal Submission Failure	*"Failed to submit your appeal. Please try again later."*
7	MSG07	Success	Appeal Submission	*"Your appeal has been successfully submitted for review."*

5.4 Other Requirements…

IV. Software Design Description

1. System Design

1.1 System Architecture



1.1.1 Frontend:

- Google cloud: The platform provides a cloud-based frontend deployment environment, ensuring application performance and scalability.
- TypeScript(TS): Core technology for the frontend, helping to build user interfaces with extensibility, strong data type checking and good integration with modern libraries.

1.1.2 Backend:

- Azure(A): Used as a backend deployment platform, providing cloud services such as storage, data processing and APIs.

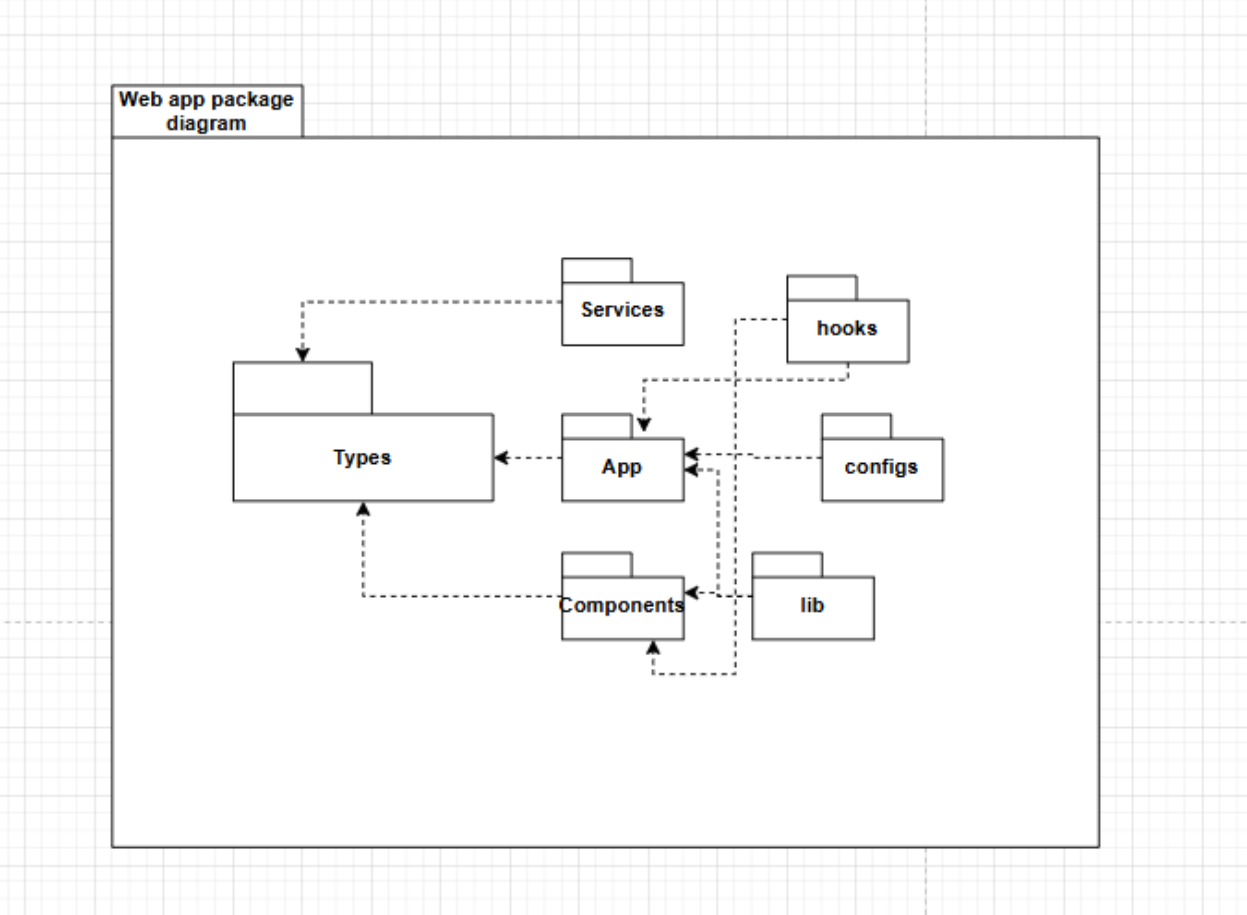
- Spring Boot: The main framework for the backend, helps develop applications quickly, supports features such as dependency injection, REST API management, security, and connecting to databases.
- MySQL: Relational database management system, stores system data such as user information, exam questions, student sources.
- Docker: A containerization tool is used by the project to serve the grading purpose, by running the exam unit tests to test the students source code.
- 7-zip: A file processing tool, used to decompress student source code files sent to the server.
- SQL Server: Another database management system is used for scoring purposes, supporting the analysis and storage of data from the exam database.
- Gherkin(Cucumber): It is a simple language used to write test scripts that are easy to understand for non-programmers. The tool uses Gherkin as a language to write test scripts and uses these scripts to generate scripts in Postman with the help of AIchat.

1.1.3 Third-party:

- Postman: API testing tool, helps test backend APIs. AIchat generates Postman scripts based on the Gherkin format received from AIchat.
- Gemini: A default AI chat tool is used for exam paper creation.

1.2 Package Diagram

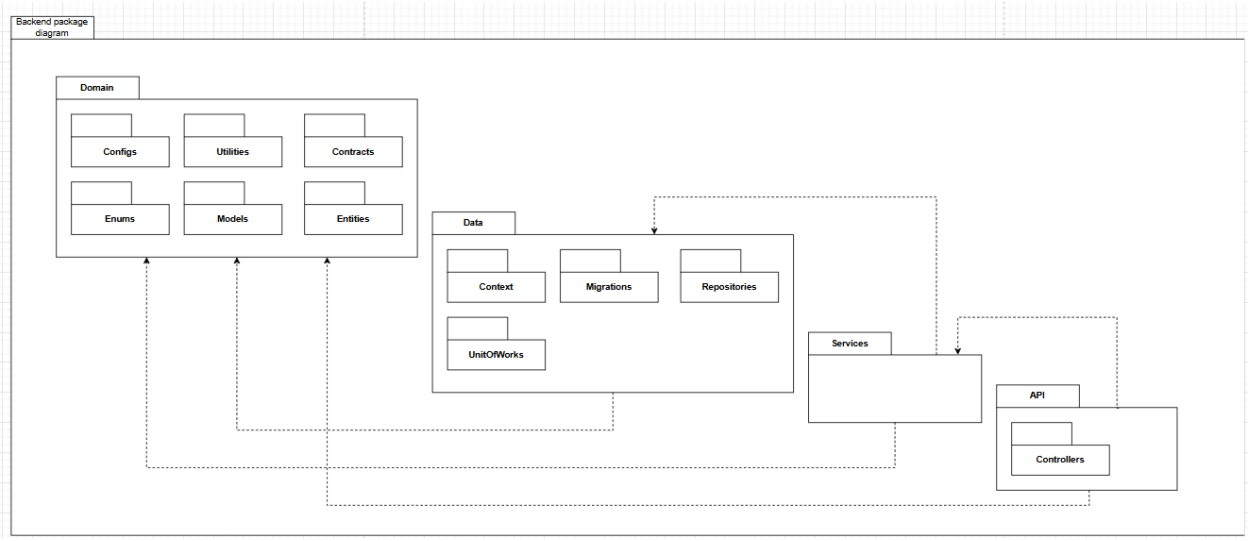
1.2.1 Frontend



No	Package	Description
1	src	The root directory contains the entire source code of the application. This is where the modules, UI components, processing logic, and project configuration are located.
2	public	Contains static resources such as images, icons (favicons), or files that need to be served directly from the server without going through the build process (eg HTML, JSON).

3	src/app	Contains the core components or main structure of the application. This is where main modules such as pages, features, or important contexts are managed.
4	src/context	Contains the core components or main structure of the application. This is where main modules such as pages, features, or important contexts are managed.
5	src/component	Contains common UI components such as buttons, modals, or input forms. Often combined with TailwindCSS for consistent styling.
6	src/assets	Store static resources related to the interface such as images, icons, fonts or files that do not change frequently.
7	src/hooks	Includes custom React Hooks for reusing complex logic, such as API management, dynamic state, or synchronous actions.
8	src/config	Contains the main configuration of the application, including environment variables definition file, API endpoint configuration or theme configuration with TailwindCSS.
9	src/lib	Contains libraries or generic logic source code, for example utility functions or logic processing classes that are not dependent on any specific component.
10	src/layouts	Defines the general layout for pages (such as pages with navigation bars, sidebars, or footers). Helps ensure UI consistency across the application.
11	src/routes	Manage the application's routing system. Contains files that define URL paths as well as the components that will render for each path.

1.2.2 Backend

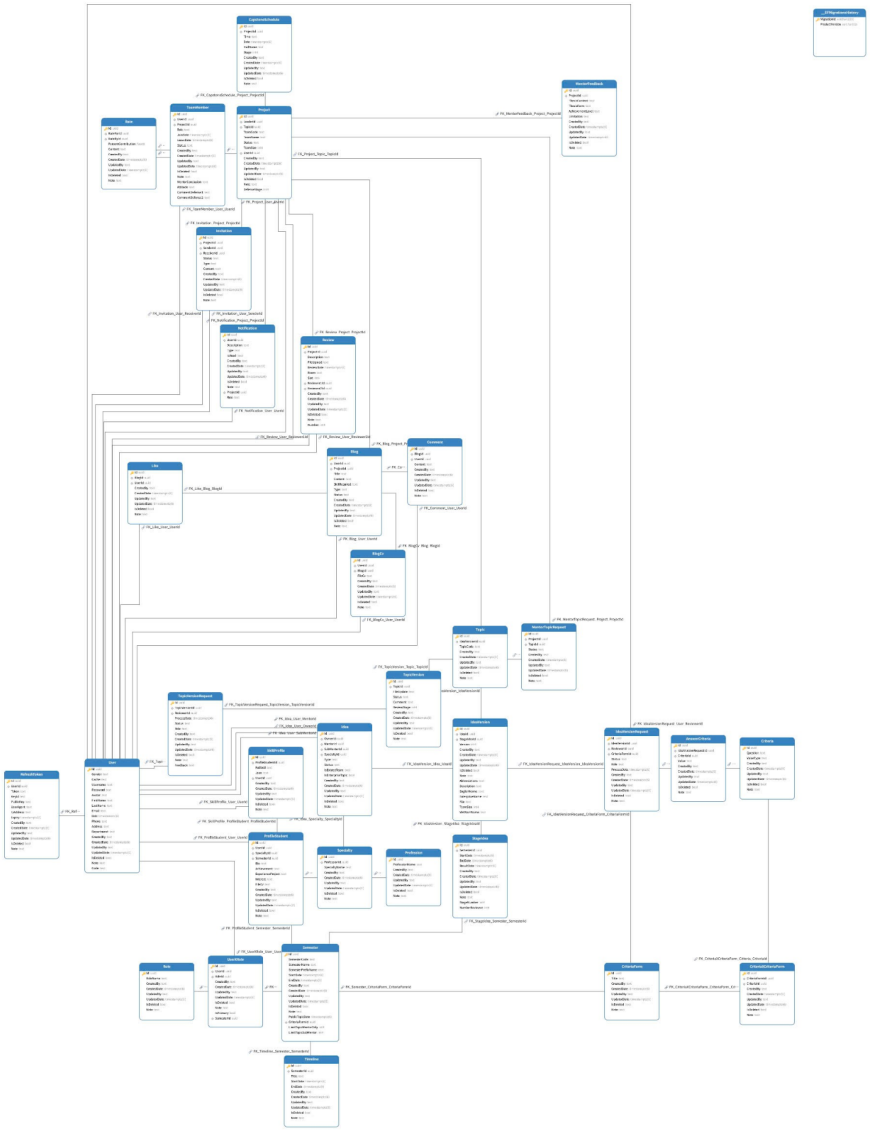


No	Package	Description
1	src/configuration	Holds configuration files required for application setup, such as environment variables, database configurations, and application-level settings.
2	src/controllers	Manages HTTP requests and responses. Controllers expose endpoints for client applications, process incoming requests, and invoke relevant services. Implements authentication using JWT (JSON Web Token) to ensure secure access to protected routes.
3	src/security	Handles security components such as JWT token creation, validation, and user authorization. Provides methods for secure password management and role-based access control (RBAC).

4	src/services	Contains the business logic of the application. Implements methods for processing and managing data across different modules. Features include: JWT Authentication logic integration. A method to send requests to Gemini AI for AI-powered responses. Support for file operations, including extracting compressed files using 7-Zip technology.
5	src/utils	Utility classes and helper functions for general-purpose tasks such as string manipulation, date formatting, and file management. Contains a 7-Zip extraction utility for handling and decompressing archived files.
6	src/mappers	Provides mapping logic to convert data between different layers (e.g., entity-to-DTO conversion). Ensures clean and structured data flow between components.
7	src/exceptions	Manages custom exceptions and error handling for the application. Ensures clear and consistent error responses for both HTTP and service-level operations.
8	src/repositories	Acts as the Data Access Layer (DAL). Provides CRUD operations for interacting with databases.
9	src/specification	Defines query specifications and search filters to simplify database queries and enhance maintainability.
10	src/models	Houses domain models and data structures that represent entities and objects used across the application.

11	src/resources	Contains static files or resource files required by the application, such as configurations, templates, or shared assets.

2. Database Design



Entities Description

2.1 User Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
Username	varchar(255)	Login username	yes	no	no
Password	varchar(255)	Encrypted password	no	no	no
Code	varchar(255)	User code	yes	no	no
Cache	varchar(255)	Latest role of user	no	no	no
Gender	varchar(255)	Gender (Male/Female/ Other)	no	no	no
Avatar	varchar(255)	Profile picture URL	no	no	no
FirstName	varchar(255)	User's first name	no	no	no
LastName	varchar(255)	User's last name	no	no	no
Email	varchar(255)	Email address	yes	no	no
Dob	date	Date of birth	no	no	no

Phone	varchar(255)	Phone number	no	no	no
Address	varchar(255)	Physical address	no	no	no
Department	varchar(255)	Department name	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.2 Role Table

Field Name	Type	Description	Unique	Not null	PK,FK

Id	varchar(255)	Unique identifier for role	yes	yes	PK
RoleName	BIGINT	Identifier linking to the account table	no	yes	FK
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.3 RefreshToken Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for the refresh token	yes	yes	PK
UserId	varchar(255)	User reference	no	no	FK
Token	varchar(255)	User refresh token	no	no	no
KeyId	varchar(255)	User key id	no	no	no
PublicKey	varchar(255)	User key to create access token	no	no	no
UserAgent	varchar(255)	User browser	no	no	no
IpAddress	varchar(255)	User local ip address	no	no	no
Expiry	date	Expiration date	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.4 Semester Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
CriteriaFormId	varchar(255)	Reference to Criteria	no	no	FK
SemesterCode	varchar(255)	Semester code	no	no	no
SemesterName	varchar(255)	Semester name	no	no	no
SemesterPrefixName	varchar(255)	Semester’s prefix string for topic code	no	no	no
StartDate	date	Date semester start	no	no	no
EndDate	date	Date semester end	no	no	no
PublicTopicDate	date	Date public topic list of semester	no	no	no
LimitTopicOnlyMentor	integer(10)	Limit topic that lecturer can be mentor alone in this semester	no	no	no
LimitTopicSubMentor	integer(10)	Limit topic that lecturer can be submentor in this semester	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no

UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.5 UserXRole Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
UserId	varchar(255)	Reference to user	no	no	FK
RoleId	varchar(255)	Reference to role	no	no	FK
SemesterId	varchar(255)	Reference to semester	no	no	FK
IsPrimary	bit	Primary role of user	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.6 Timeline Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
SemesterId	varchar(255)	Reference to semester	no	no	FK
Title	varchar(255)	Title of timeline	no	no	no
StartDate	date	Date time line start	no	no	no
EndDate	date	Date time line end	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.7 StageIdea Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
SemesterId	varchar(255)	Reference to semester	no	no	FK
NumberReviewer	integer(10)	Number of reviewer in this stage idea	no	no	no
StartDate	date	Date stage idea start	no	no	no
EndDate	date	Date stage idea end	no	no	no
ResultDate	date	Date stage idea show result	no	no	no
StageNumber	integer(10)	Number of this stage idea	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.8 Profession Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for profession	yes	yes	PK
ProfessionName	varchar(225)	Name of profession	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.9 Specialty Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for specialty	yes	yes	PK
ProfessionId	varchar(255)	Reference to profession	no	no	FK
SpecialtyName	varchar(255)	Name of specialty	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.10 Criteria Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for criteria	yes	yes	PK
Question	varchar(255)	Question of criteria	no	no	no
ValueType	varchar(255)	Value type of the answer of question	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.11 CriteriaForm Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for criteria form	yes	yes	PK
Title	varchar(255)	Title of criteria form	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.12 CriteriaXCriteriaForm Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for criteriaXcriteriaform	yes	yes	PK
CriteriaId	varchar(255)	Reference to Criteria	no	yes	FK
CriteriaFormId	varchar(255)	Reference to CriteriaForm	no	yes	FK
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.13 Idea Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for idea	yes	yes	PK
OwnerId	varchar(255)	Reference to user	no	no	FK
MentorId	varchar(255)	Reference to user	no	no	FK
SubMentorId	varchar(255)	Reference to user	no	no	FK
SpecialtyId	varchar(255)	Reference to specialty	no	no	FK
Type	varchar(255)	Type of idea	no	no	no
Status	varchar(255)	Status of idea (Pending/ Approved/Rejected/ ConsiderByMentor/ ConsiderByCouncil)	no	no	no
IsExistedTeam	bit	Idea has team or not	no	no	no
IsEnterpriseTopic	bit	Idea is enterprise topic or not	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no

UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.14 IdeaVersion Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for idea version	yes	yes	PK
IdeaId	varchar(255)	Reference to idea	no	no	FK
StageIdeaId	varchar(255)	Reference to stage idea	no	no	FK
Version	integer(10)	Number version of idea	no	no	no
Description	varchar(255)	Description of idea version	no	no	no
Abbreviations	varchar(255)	Abbreviations of idea version	no	no	no
VietNamName	varchar(255)	Viet nam name of idea version	no	no	no
EnglishName	varchar(255)	English name of idea version	no	no	no
File	varchar(255)	File url of idea version	no	no	no
TeamSize	integer(10)	Team szie idea version	no	no	no

CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.15 IdeaVersionRequest Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for idea version request	yes	yes	PK
IdeaVersionId	varchar(255)	Reference to idea version	no	no	FK
ReviewerId	varchar(255)	Reference to idea user	no	no	FK
CriteriaFormId	varchar(255)	Reference to idea criteria form	no	no	FK
ProcessDate	date	Date process idea version request	no	no	no
Status	varchar(255)	Status of idea version request	no	no	no
Role	varchar(255)	Role of who respond request (Mentor/ Council)	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no

Note	varchar(255)	Additional notes	no	no	no

2.16 AnswerCriteria Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for answer criteria	yes	yes	PK
IdeaVersionRequestId	varchar(255)	Reference to idea version request	no	no	FK
CriteriaId	varchar(255)	Reference to criteria	no	no	FK
Value	varchar(255)	Value of criteria	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.17 Topic Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for topic	yes	yes	PK
IdeaVersionId	varchar(255)	Reference to idea version	no	no	FK
TopicCode	varchar(255)	Code of topic	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.18 TopicVersion Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for topic version	yes	yes	PK
TopicId	varchar(255)	Reference to topic	yes	yes	FK
FileUpdate	varchar(255)	File url of topic version	no	no	no
ReviewStage	integer(10)	Indicates which evaluation round (1st or 2nd review) this topic version was submitted for	no	no	no
Status	varchar(255)	Status of topic version (Pending/Approved/ Rejected)	no	no	no
Comment	varchar(255)	Comment carries the meaning of evaluating the topic	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no

Note	varchar(255)	Additional notes	no	no	no

2.19 TopicVersionRequest Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
TopicVersionId	varchar(255)	Reference to topic version	no	no	FK
ReviewerId	varchar(255)	Reference to user with role reviewer	no	no	FK
ProcessDate	date	Date of process	no	no	no
Status	varchar(255)	Status of topic version request	no	no	no
Role	varchar(255)	Role of reviewer	no	no	no
Feedback	varchar(255)	Opinion of reviewer	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no

Note	varchar(255)	Additional notes	no	no	no

2.20 Project Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for project	yes	yes	PK
LeaderId	varchar	Reference to user with role student who is leader of team	no	no	FK
TopicId	varchar(255)	Reference to topic	no	no	FK
TeamName	varchar(255)	Name of a team	no	no	no
TeamCode	varchar(255)	Code of a team	no	no	no
Status	varchar(255)	Status of a team	no	no	no
TeamSize	integer(2)	Size of a team	no	no	no
DefenseStage	integer(10)	Stage of defense	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no

IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.21 TeamMember Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for team member	yes	yes	PK
UserId	varchar(255)	Reference to user	no	no	FK
ProjectId	varchar(255)	Reference to project	no	no	FK
Role	varchar(255)	Role of each member in a team	no	no	no
JoinDate	date	Join date of a member	no	no	no
LeaveDate	date	Leave date of a member	no	no	no
Status	varchar(255)	Status of team member	no	no	no
Attitude	varchar(255)	Attitude of a member in team	no	no	no
MentorConclusion	varchar(255)	Conclusion of mentor for member in team	no	no	no
CommentDefense1	varchar(255)	Comment of mentor for member in defense stage 1	no	no	no

CommentDefense2	varchar(255)	Comment of mentor for member in defense stage 2	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.22 Review Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for review	yes	yes	PK
ProjectId	varchar(255)	Reference to project	no	no	FK
Reviewer1Id	varchar(255)	Reference to user with role Reviewer	no	no	FK
Reviewer2Id	varchar(255)	Reference to user with role Reviewer	no	no	FK
Number	integer(10)	Number of a review	no	no	no
ReviewDate	date	Date of a review	no	no	no
Room	varchar(255)	Room number for a review (location)	no	no	no
Slot	varchar(255)	Slot of the review	no	no	no
Description	varchar(255)	Description of a review	no	no	no
FileUpload	varchar(255)	File upload for a review	no	no	no
CreatedDate	date	Creation date	no	no	no

CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.23 Invitation Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
ProjectId	varchar(255)	Reference to project	no	no	FK
SenderId	varchar(255)	Reference to user who send the invitation	no	no	FK
ReceiverId	varchar(255)	Reference to user who receive the invitation	no	no	FK
Status	varchar(255)	Status of an invitation	no	no	no
Type	varchar(255)	Type of an invitation	no	no	no
Content	varchar(255)	Content of an invitation	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no

UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.24 MentorFeedback Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for mentor feedback	yes	yes	PK
ProjectId	varchar(255)	Reference to project	yes	yes	FK
ThesisContent	varchar(255)	Feedback of mentor about the thesis content of team compare to research objective	no	no	no
ThesisForm	varchar(255)	Feedback of mentor about the layout, presentation methods, English, citation	no	no	no
AchievementLevel	varchar(255)	Level of achievement compare to the target (compare to the plan)	no	no	no
Limitation	varchar(255)	Limitation of the project	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no

IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.25 ProfileStudent Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for profile student	yes	yes	PK
UserId	varchar(255)	Reference to user with role student	yes	yes	FK
SpecialityId	varchar(255)	Reference to speciality	yes	yes	FK
SemesterId	varchar(255)	Reference to semester	yes	yes	FK
Bio	varchar(255)	Bio of a student	no	no	no
Achievement	varchar(255)	Achievement of a student	no	no	no
ExperienceProject	varchar(255)	Experience of a student for project	no	no	no
Interest	varchar(255)	Interest of student	no	no	no
FileCv	varchar(255)	File of cv of a student	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no

UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.26 SkillProfile Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for user	yes	yes	PK
ProfileStudentId	varchar(255)	Reference to profile student	yes	yes	FK
FullSkill	varchar(255)	Full skill of a student	no	no	no
Json	varchar(255)		no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.27 Notification Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for notification	yes	yes	PK
UserId	varchar(255)	Reference to user who receive the notification	no	no	FK
ProjectId	varchar(255)	Reference to project of which team receive the notification	no	no	FK
Description	varchar(255)	Content of a notification	no	no	no
Type	varchar(255)	Type of a notification	no	no	no
Role	varchar(255)	Serving the purpose of notification by role	no	no	no
IsRead	bit	Status of a notification is read or not	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no

Note	varchar(255)	Additional notes	no	no	no

2.28 Conversation Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for conversation	yes	yes	PK
ConversationName	varchar(255)	Name of conservation	no	no	no

2.29 ConversationMember Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for conservation member	yes	yes	PK
UserId	varchar(255)	Reference to user	yes	yes	FK
ConversationId	varchar(255)	Reference to conservation	yes	yes	FK

2.30 Message Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for message	yes	yes	PK
ConversationId	varchar(255)	Reference to conservation	yes	yes	FK
SendById	varchar(255)	Reference to user who send message	yes	yes	FK
Content	varchar(255)	Content of a message	no	no	no
CreatedDate	date	Creation date	no	no	no

2.31 Blog Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for blog	yes	yes	PK
UserId	varchar(255)	Reference to user who create blog	no	no	FK
ProjectId	varchar(255)	Reference to project related to blog	no	no	FK
Title	varchar(255)	Title of blog	no	no	no
Content	varchar(255)	Content of a blog	no	no	no
SkillRequired	varchar(255)	Require of skill in a blog	no	no	no
Type	varchar(255)	Type of a blog	no	no	no
Status	varchar(255)	Status of a blog	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.32 BlogCv Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for blog cv	yes	yes	PK
BlogId	varchar(255)	Reference to blog	no	no	FK
UserId	varchar(255)	Reference to user who apply cv to blog	no	no	FK
FileCv	varchar(255)	File CV of a blog	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.33 Like Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for like	yes	yes	PK
BlogId	varchar(255)	Reference to blog	yes	yes	FK
UserId	varchar(255)	Reference to user who like the blog	yes	yes	FK
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.34 Comment Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for comment	yes	yes	PK
BlogId	varchar(255)	Reference to blog	no	no	FK
UserId	varchar(255)	Reference to user who comment in blog	no	no	FK
Content	varchar(255)	The content of the comment within a blog	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.35 Rate Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for rate	yes	yes	PK
RateForId	varchar(255)	Reference to user being rated	no	no	FK
RateById	varchar(255)	Reference to user who rates	no	no	FK
PercentContribution	double(10)	Contribution percentage of a member in team	no	no	no
Content	varchar(255)	Evaluation comments	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.36 CapstoneSchedule Table

Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for capstone schedule	yes	yes	PK
ProjectId	varchar(255)	Reference to associated project	no	no	FK
Time	varchar(255)	Presentation time	no	no	no
Date	date	Presentation date	no	no	no
HallName	varchar(255)	Physical/Virtual location (name of hall)	no	no	no
Stage	integer(10)	Stand for which stage the thesis defense will be presented of a teateam	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

2.37 MentorTopicRequest Table

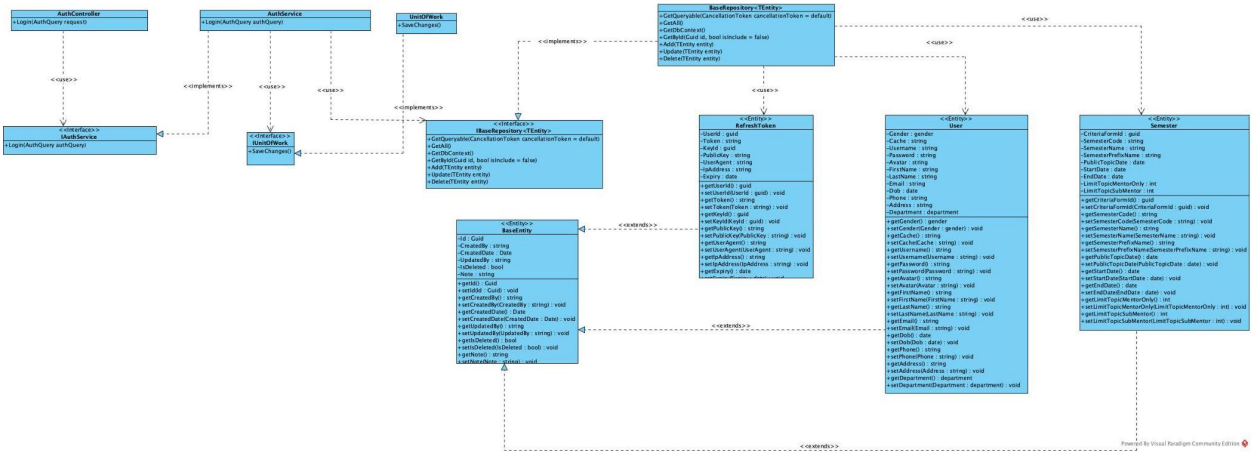
Field Name	Type	Description	Unique	Not null	PK,FK
Id	varchar(255)	Unique identifier for mentor topic request	yes	yes	PK
ProjectId	varchar(255)	Reference to associated project	no	no	FK
IdeaId	varchar(255)	Reference to original idea submission	no	no	FK
Status	varchar(255)	Request status(Pending/ Approved/Rejected)	no	no	no
CreatedDate	date	Creation date	no	no	no
CreatedBy	varchar(255)	Who created	no	no	no
UpdatedDate	date	Last modification date	no	no	no
UpdatedBy	varchar(255)	Who last updated	no	no	no
IsDeleted	bit	Delete flag (0=active, 1=deleted)	no	no	no
Note	varchar(255)	Additional notes	no	no	no

3. Detailed Design

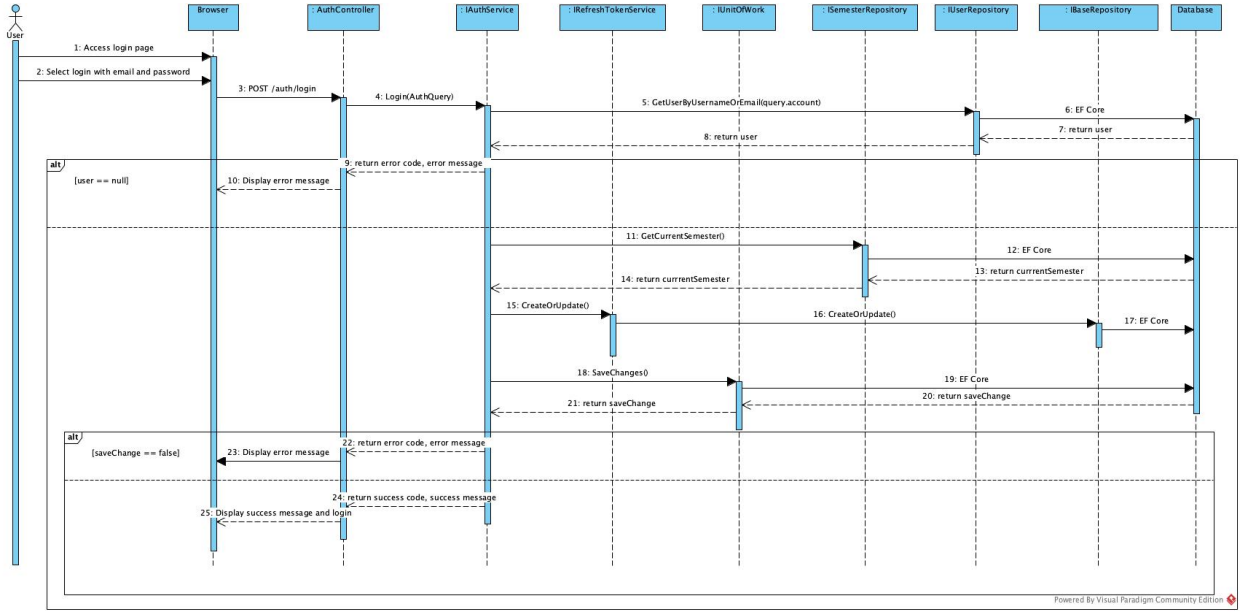
3.1 Authentication

3.1.1 Login

3.1.1.1 Class Diagram



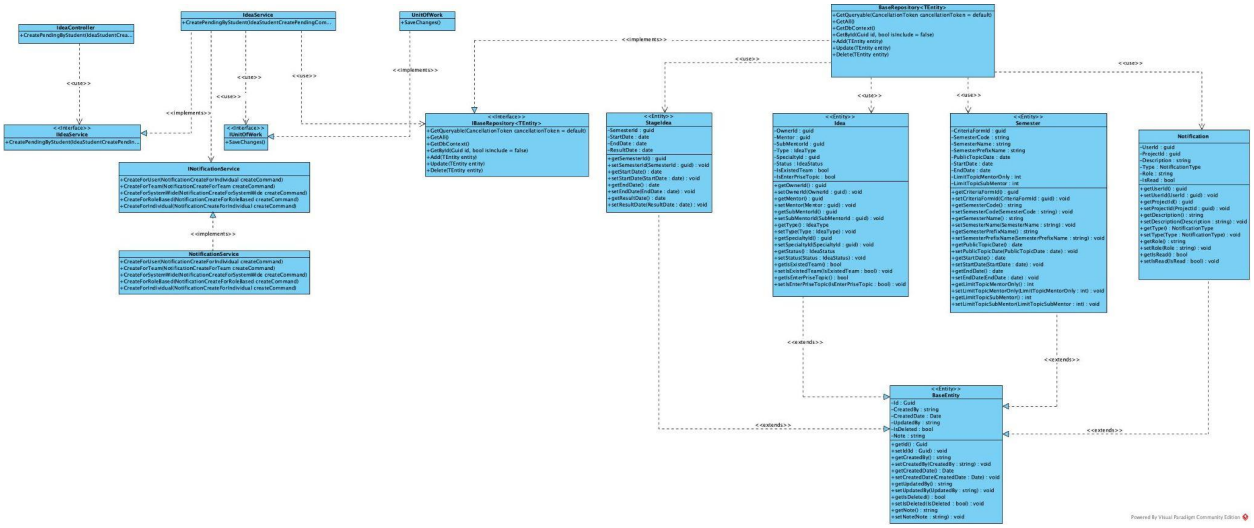
3.1.1.2 Sequence Diagram



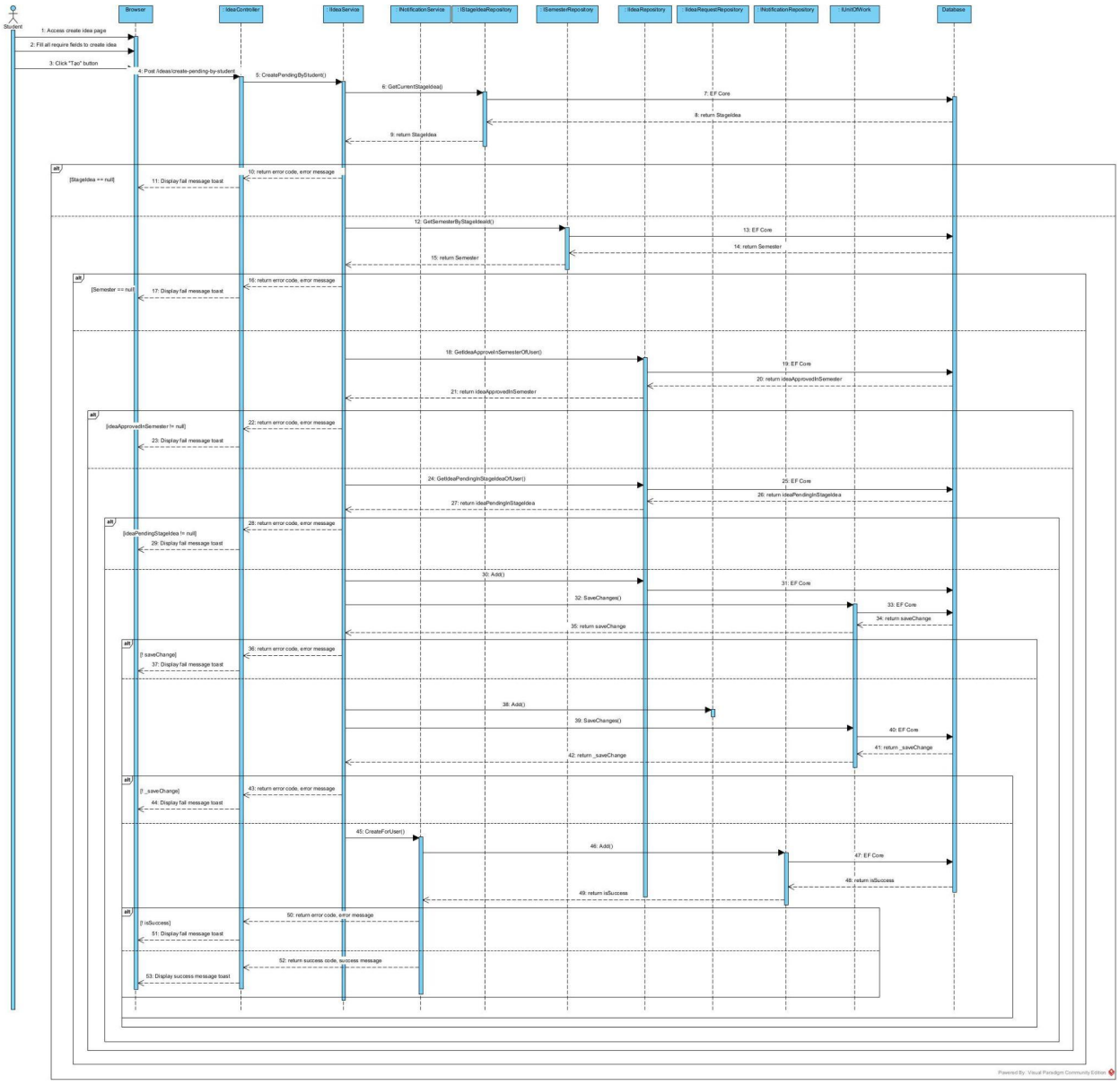
3.2 Idea Management

3.2.1 Students create ideas.

3.2.1.1 Class Diagram

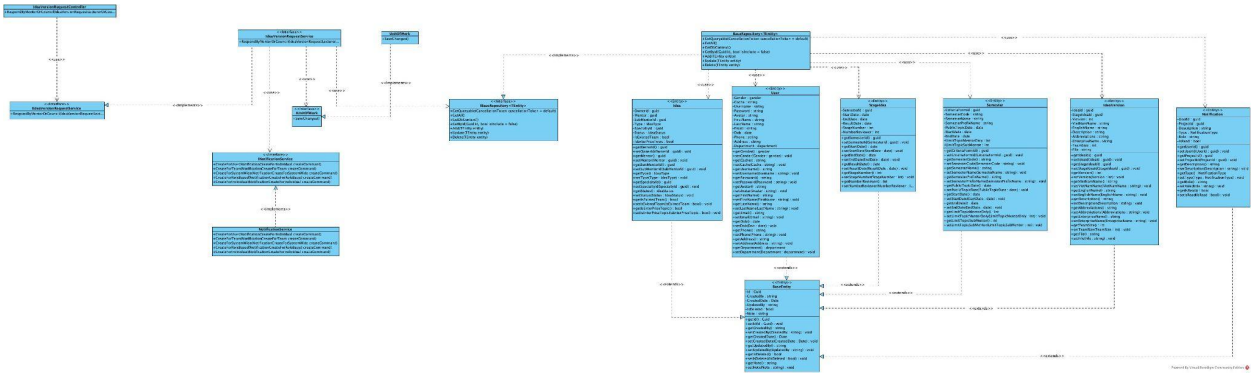


3.2.1.2 Sequence Diagram

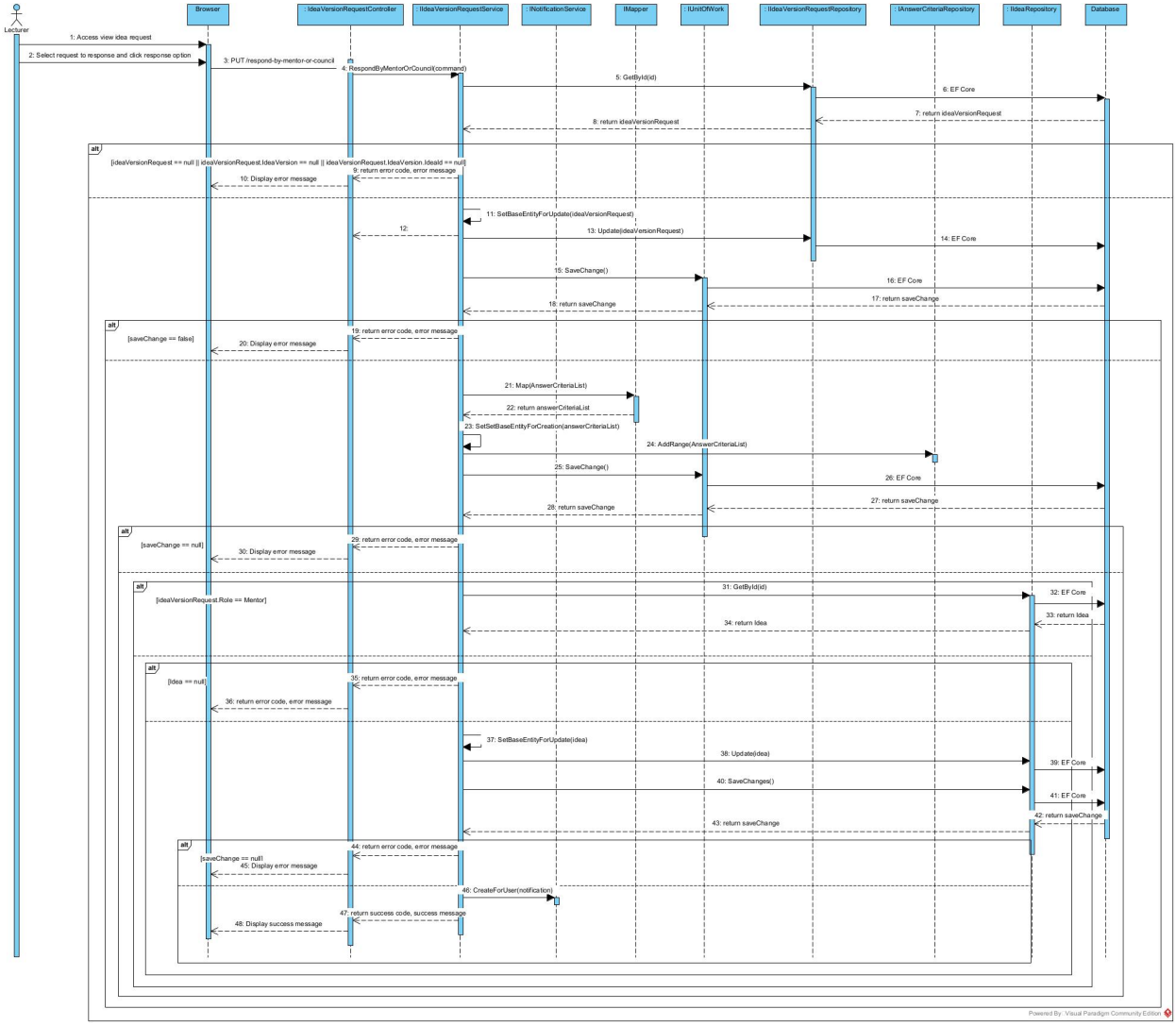


3.2.2 Lecturer evaluate idea

3.2.2.1 Class Diagram

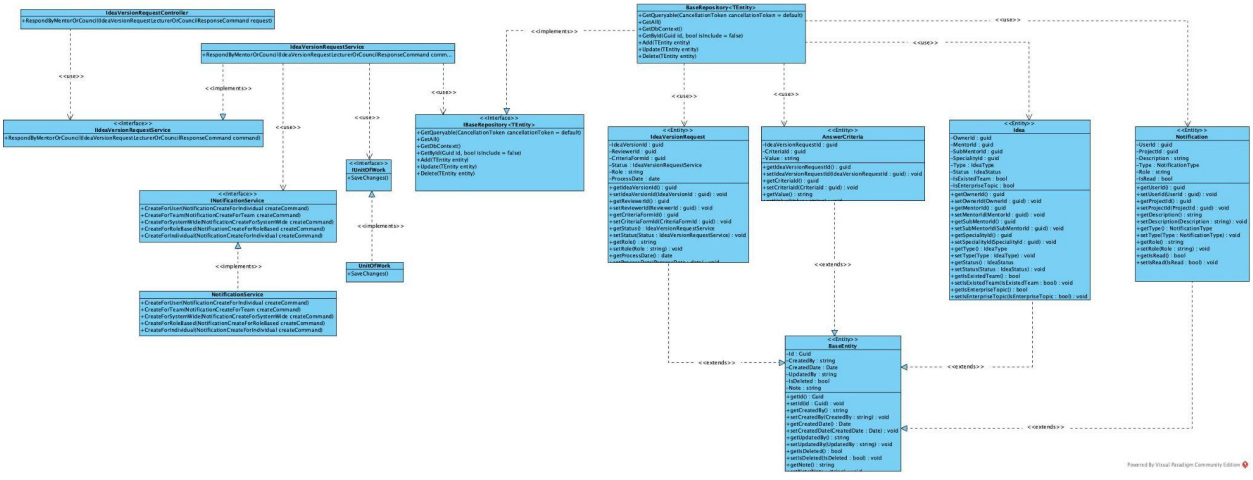


3.2.2.2 Sequence Diagram

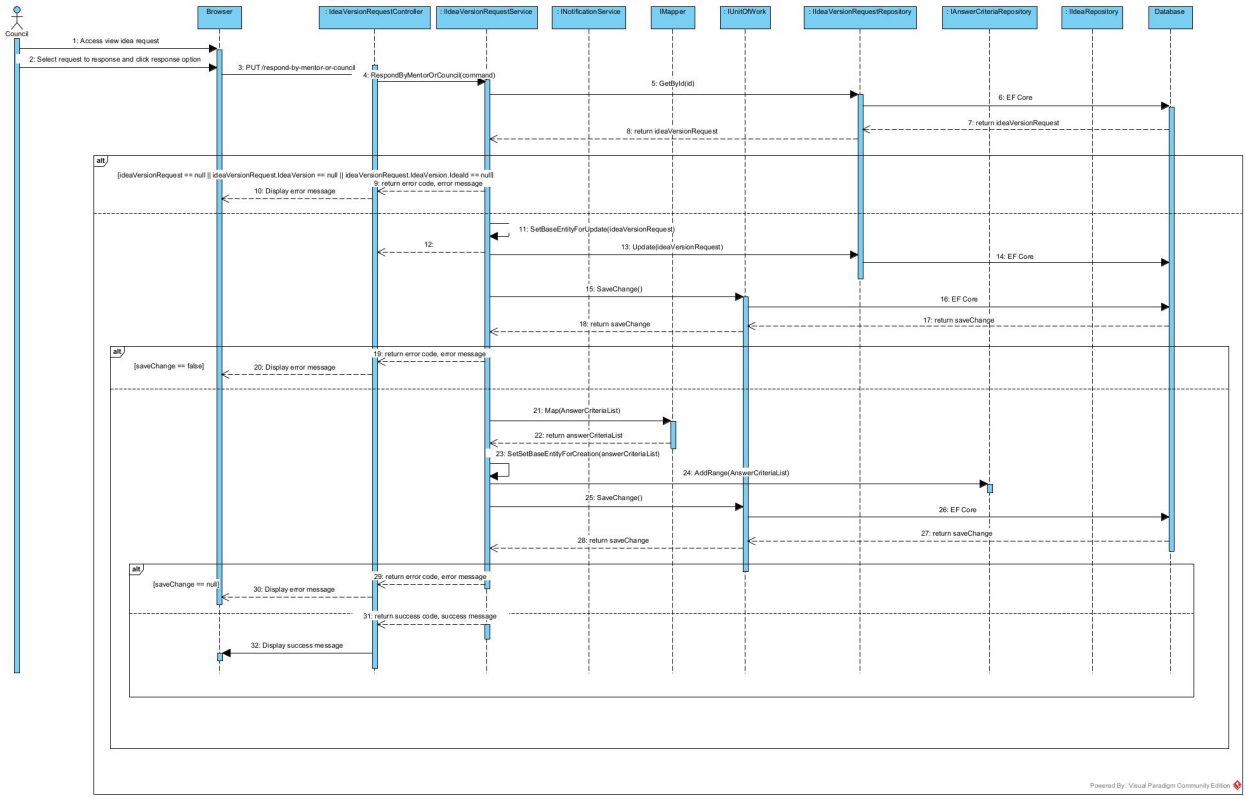


3.2.3 Council evaluates ideas.

3.2.3.1 Class Diagram

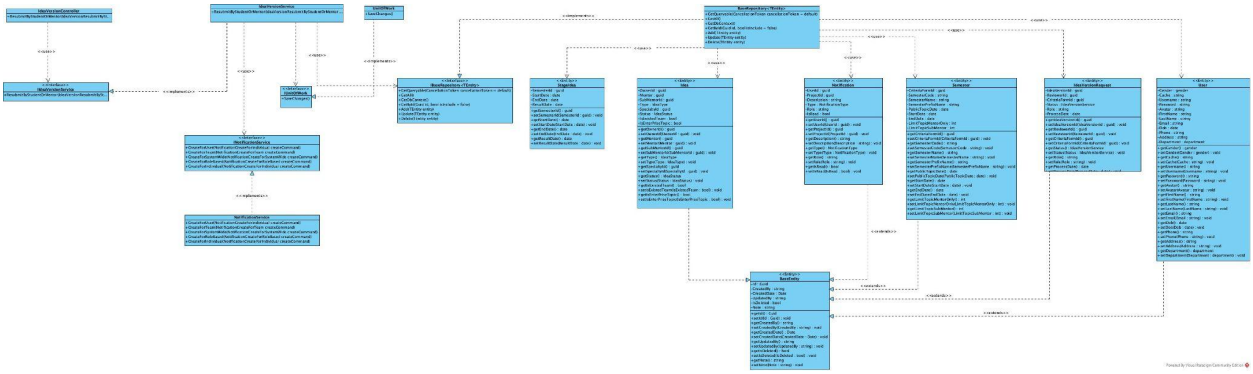


3.2.3.2 Sequence Diagram

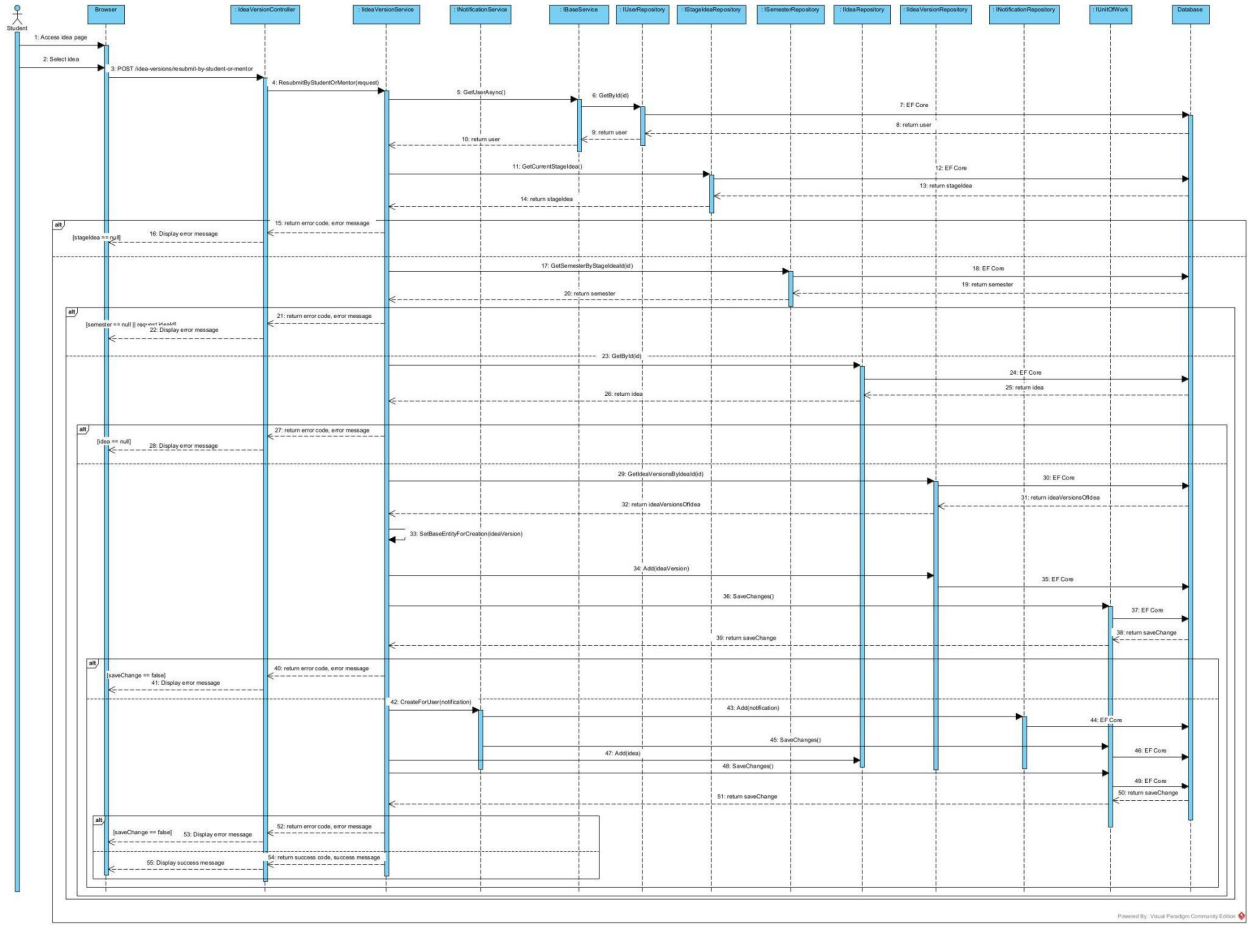


3.2.4 Student resubmit idea.

3.2.4.1 Class Diagram

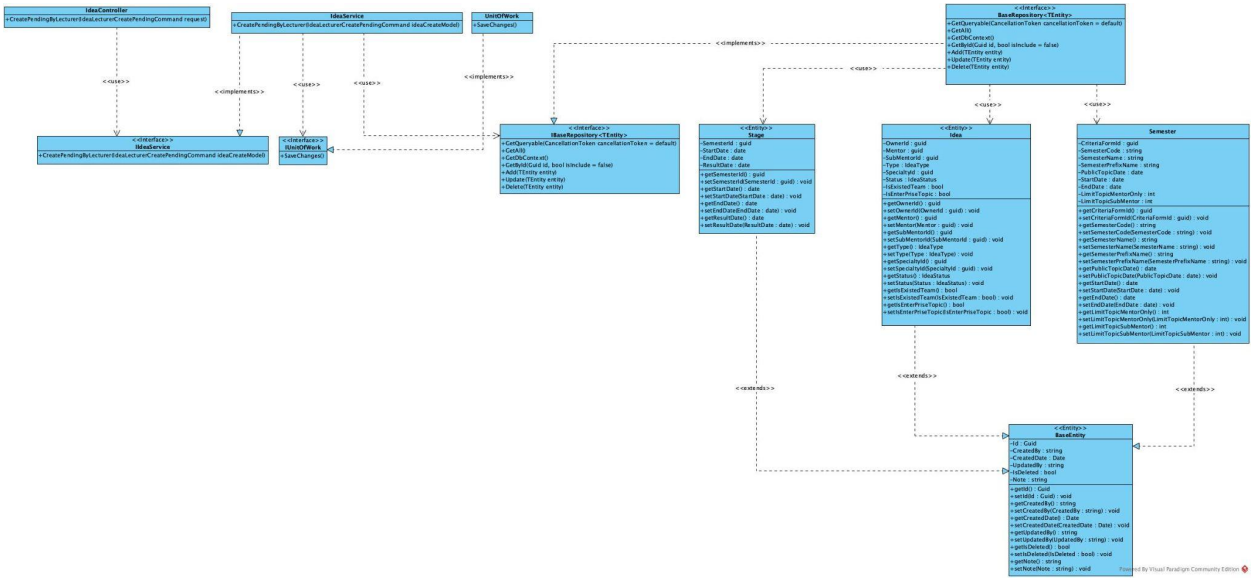


3.2.4.2 Sequence Diagram

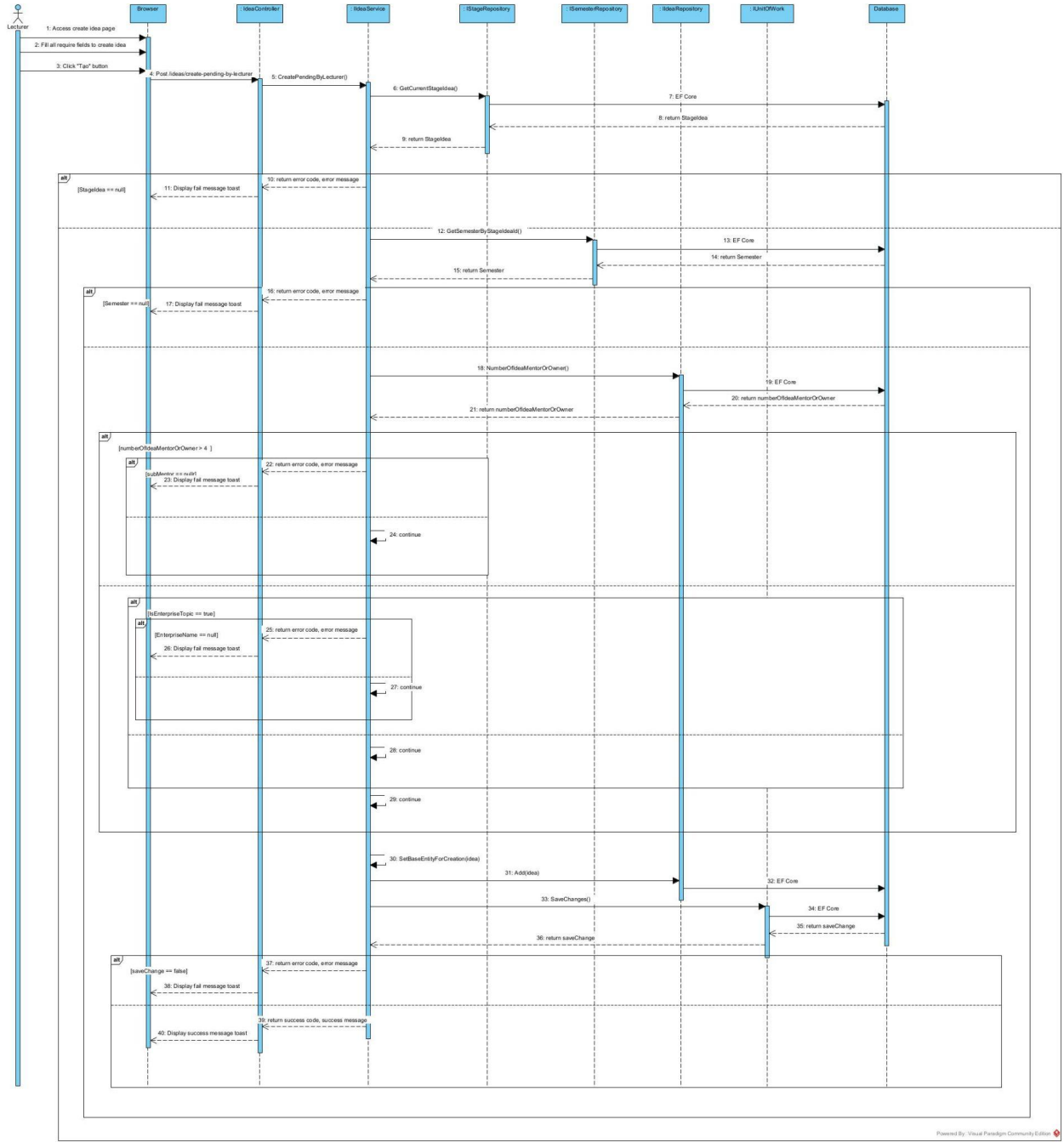


3.2.5 Lecturer create idea.

3.2.5.1 Class Diagram

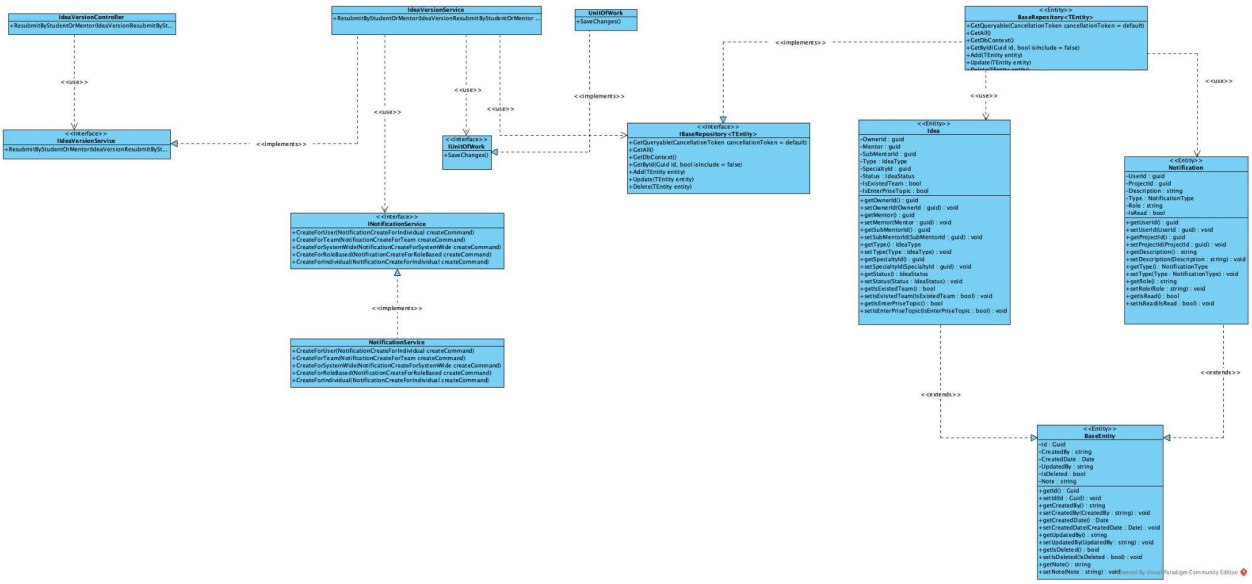


3.2.5.2 Sequence Diagram

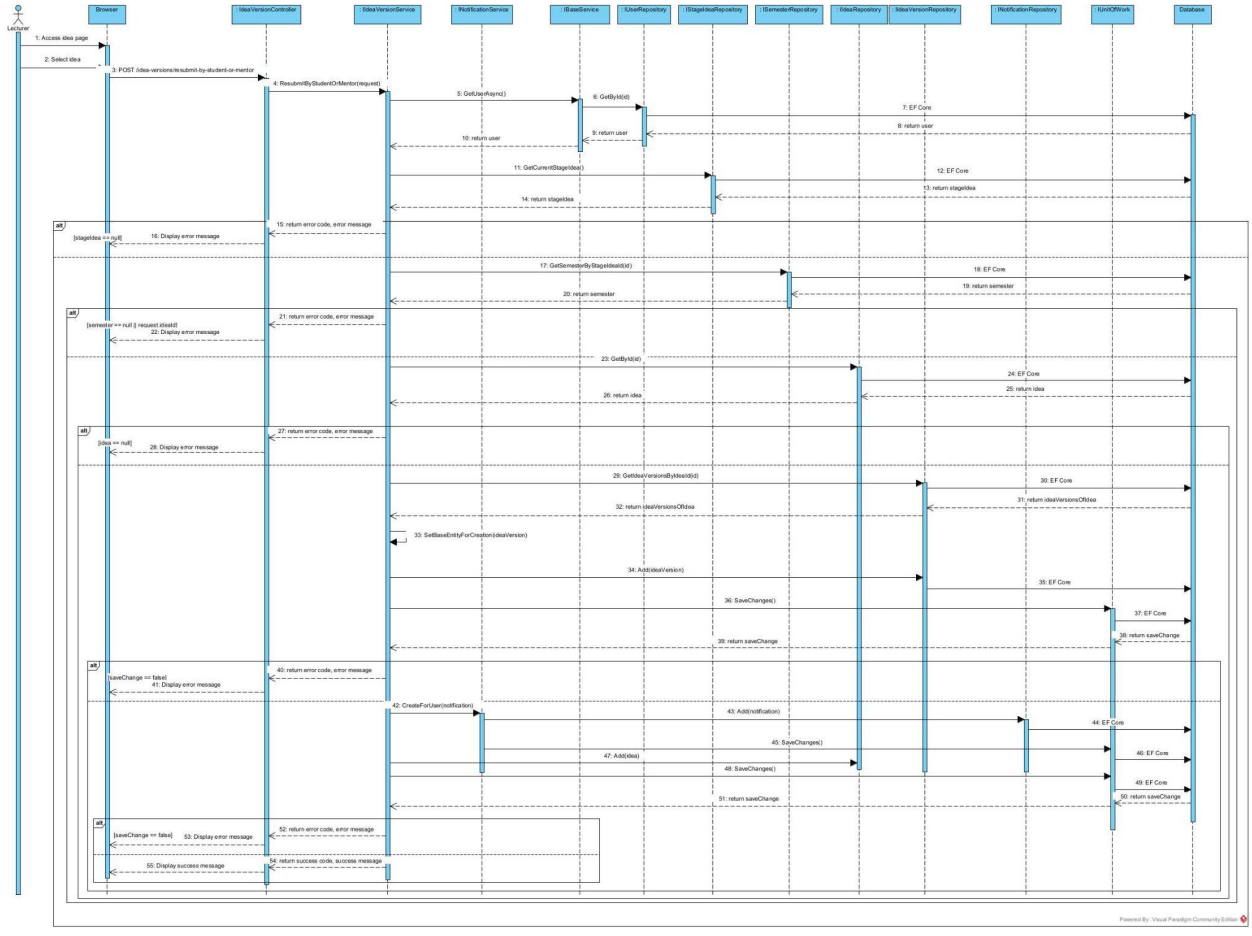


3.2.4 Lecturer resubmit idea.

3.2.4.1 Class Diagram



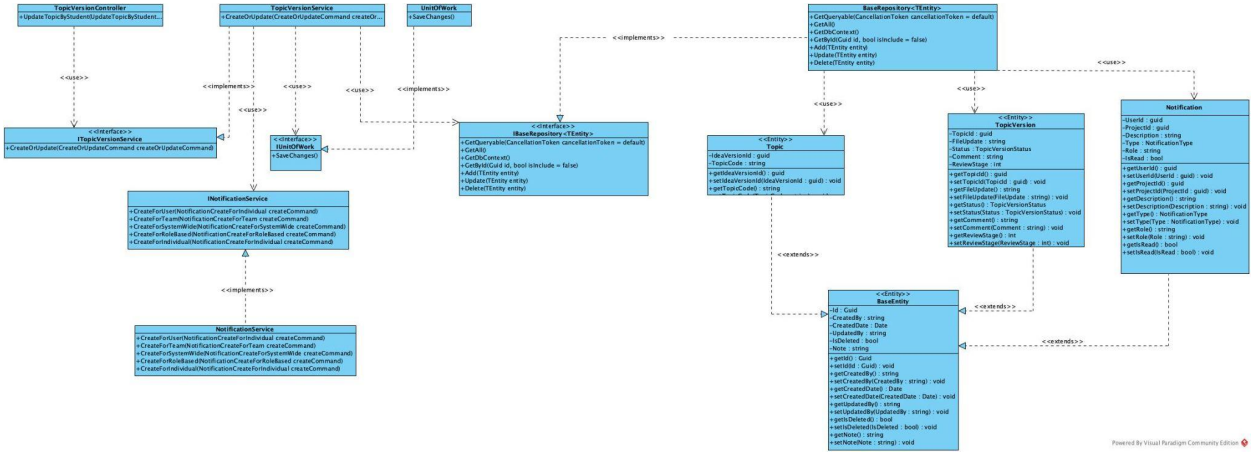
3.2.4.2 Sequence Diagram



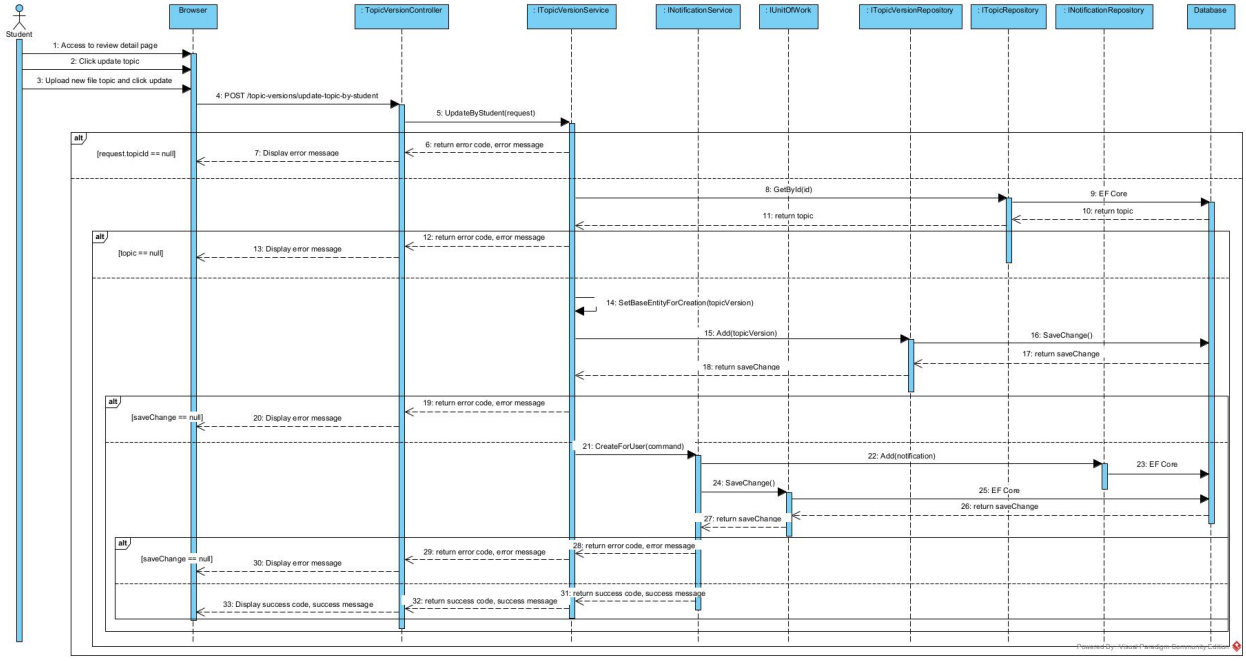
3.3 Topic Management

3.3.1 Student update topic

3.3.1.1 Class Diagram

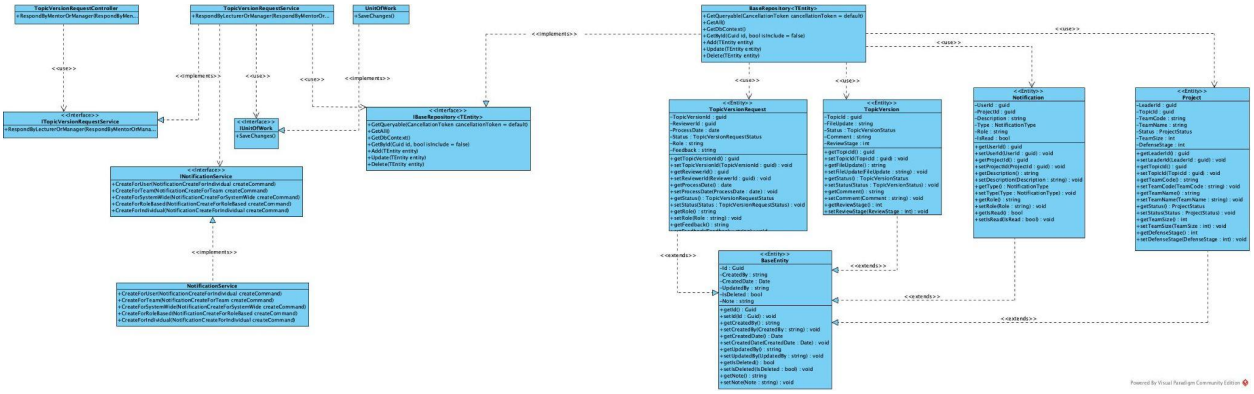


3.3.1.2 Sequence Diagram

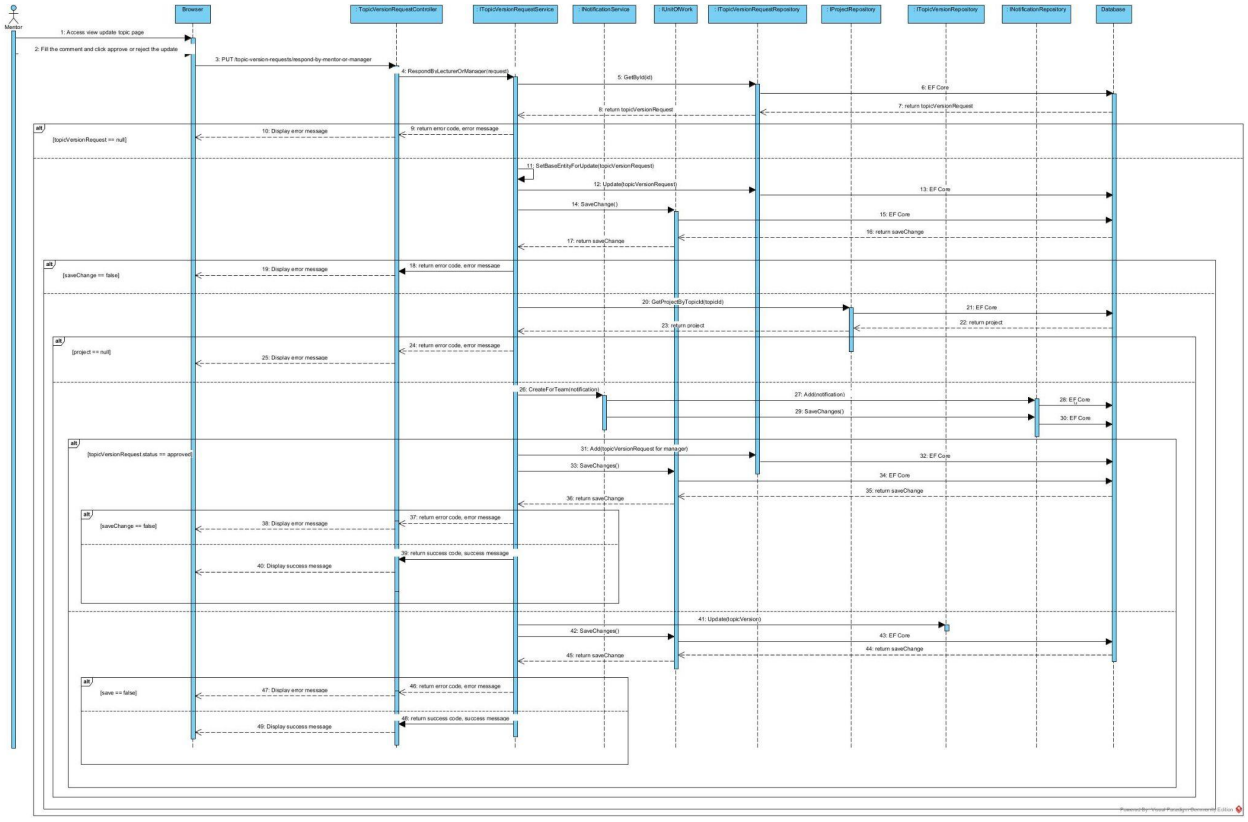


3.3.2 Mentor respond updating topic

3.3.2.1 Class Diagram

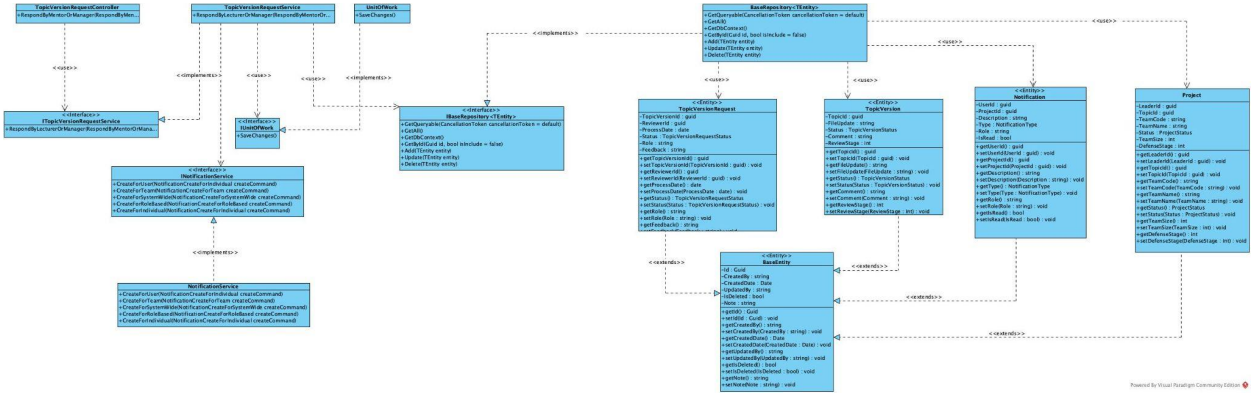


3.3.2.2 Sequence Diagram

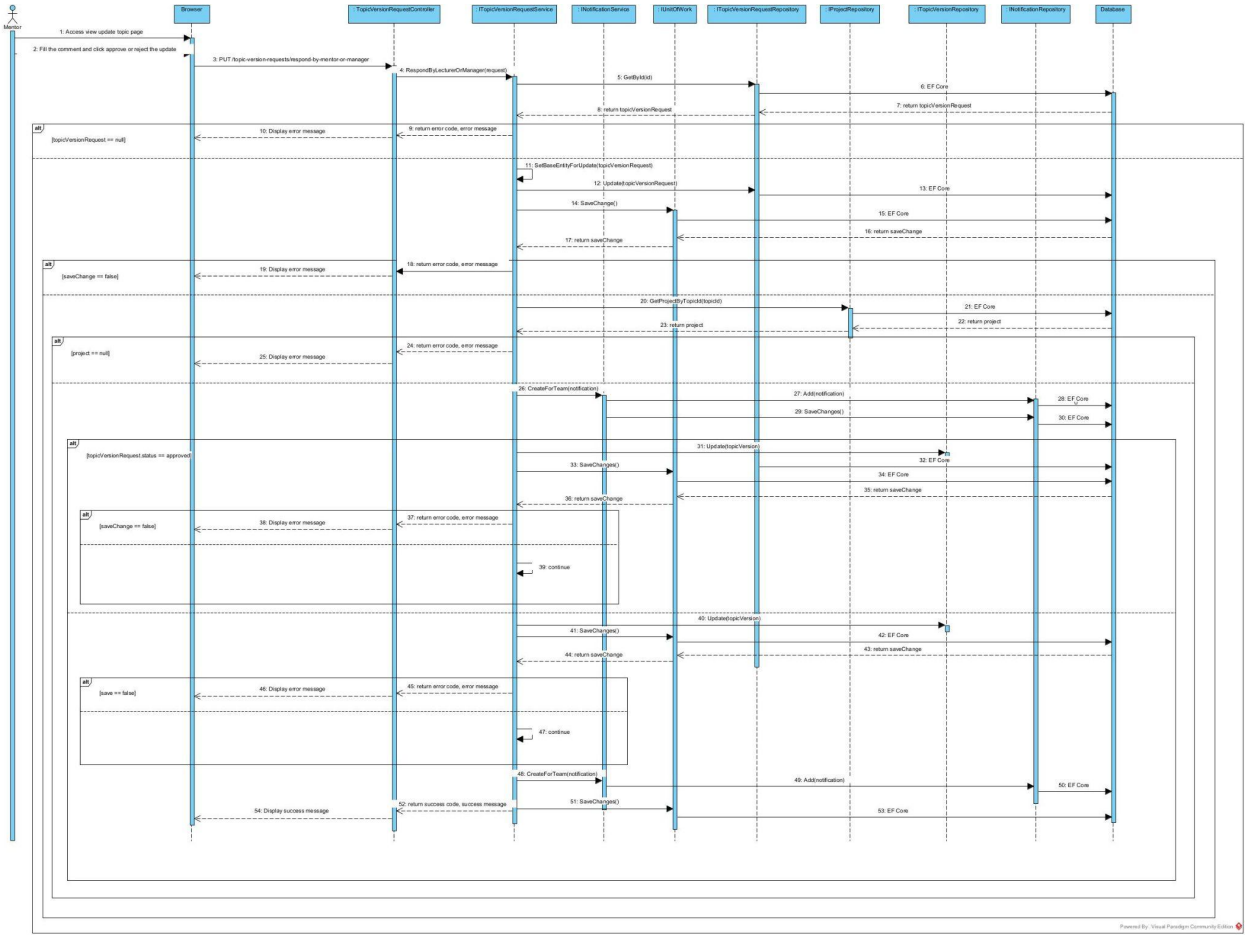


3.3.3 Manager respond updating topic

3.3.3.1 Class Diagram



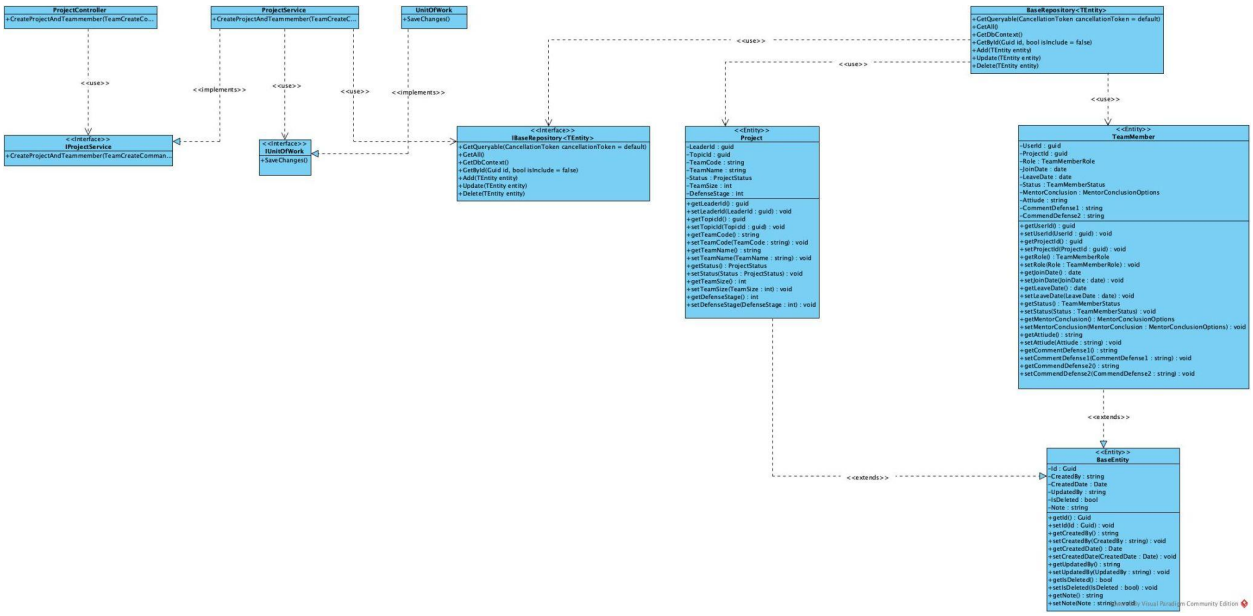
3.3.3.2 Sequence Diagram



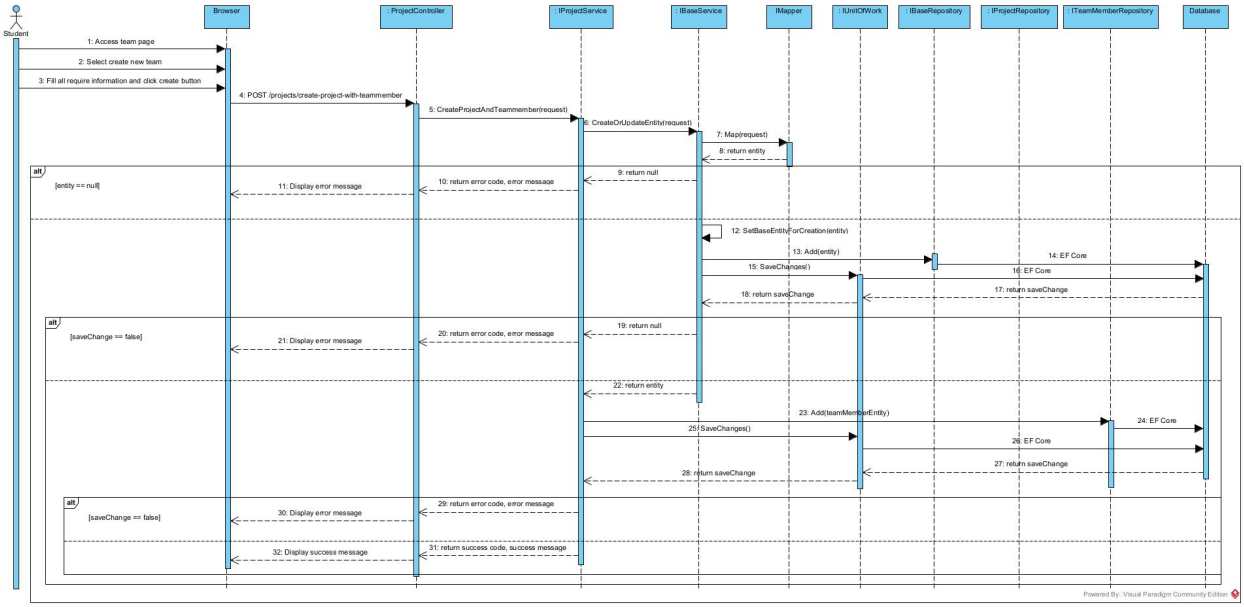
3.4 Team Management

3.4.1 Create a team.

3.4.1.1 Class Diagram

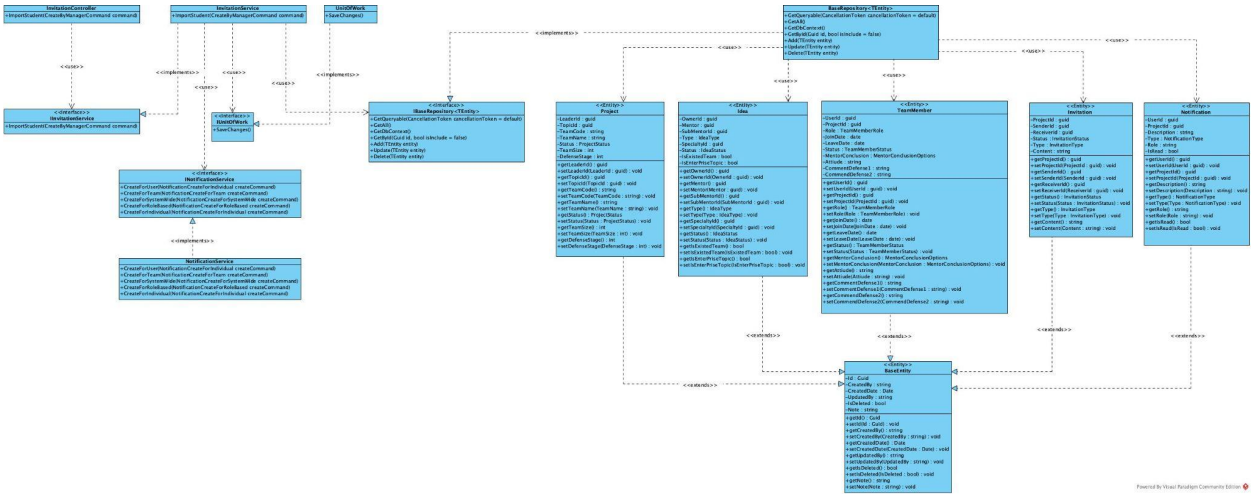


3.4.1.2 Sequence Diagram

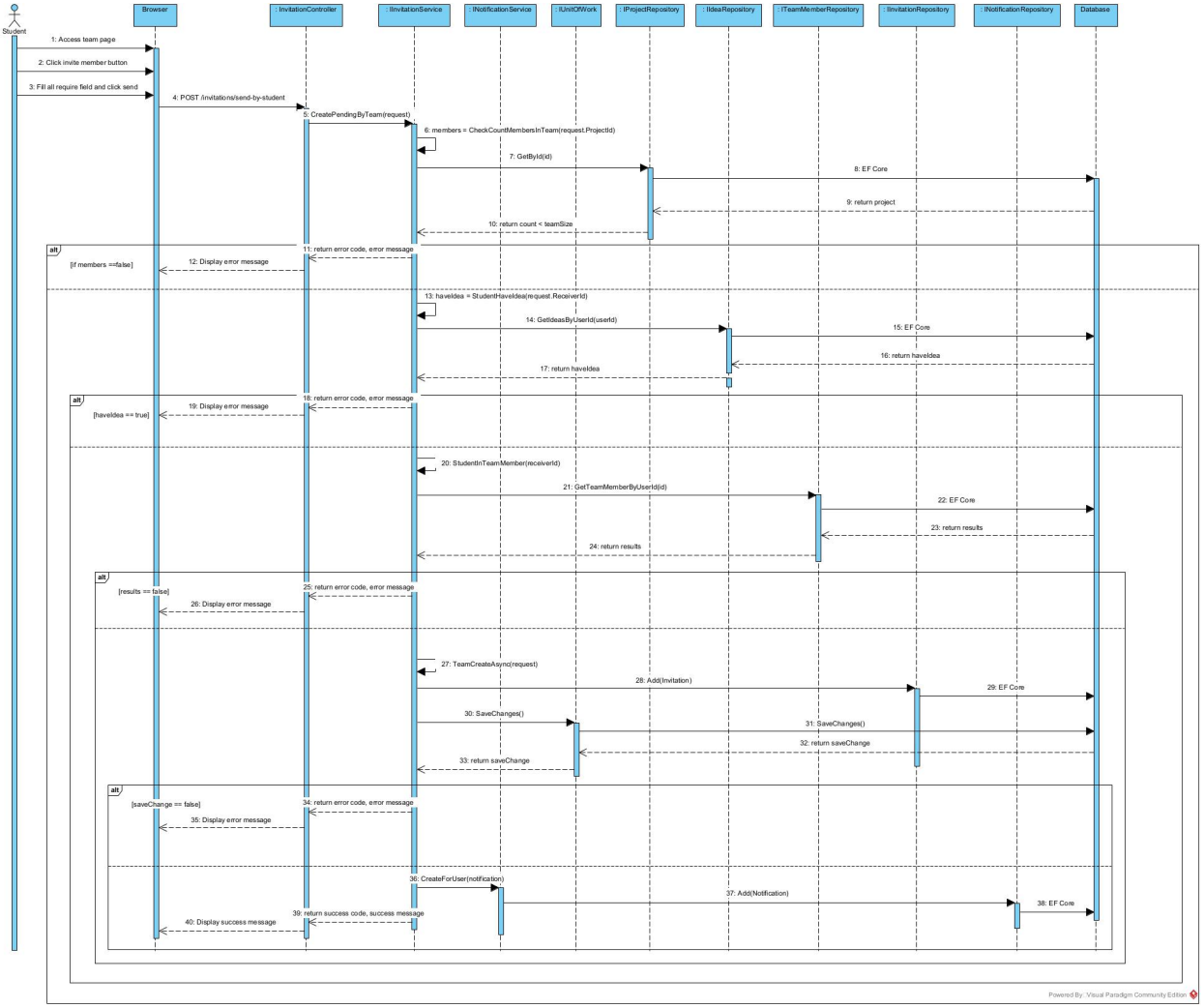


3.4.2 Invite team members.

3.4.2.1 Class Diagram

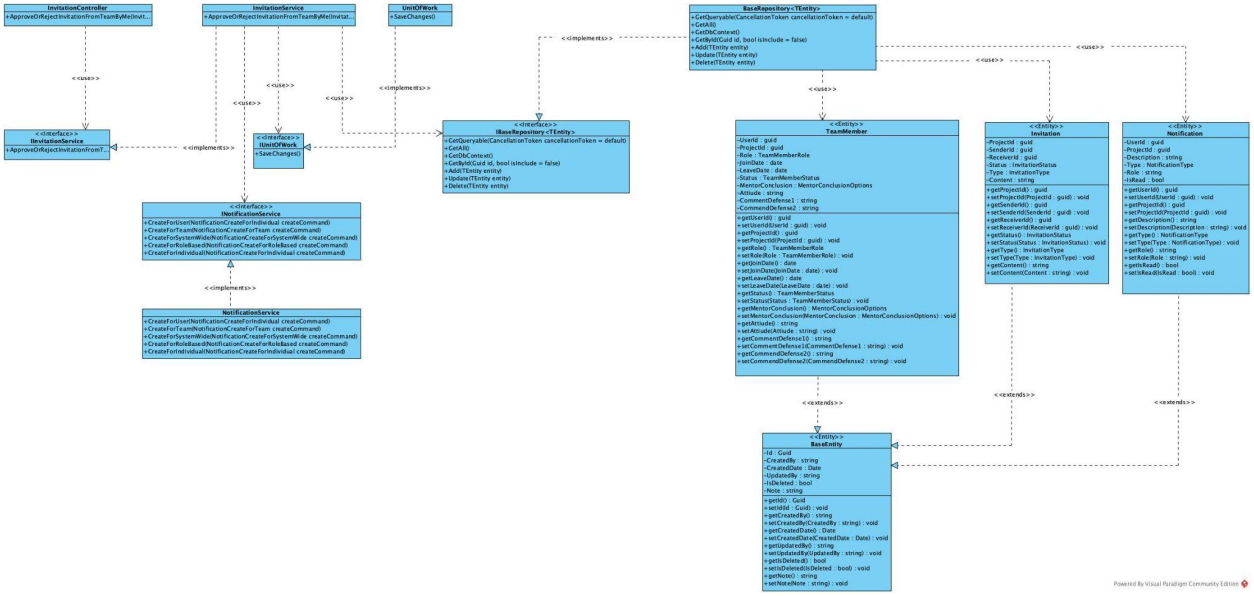


3.4.2.2 Sequence Diagram

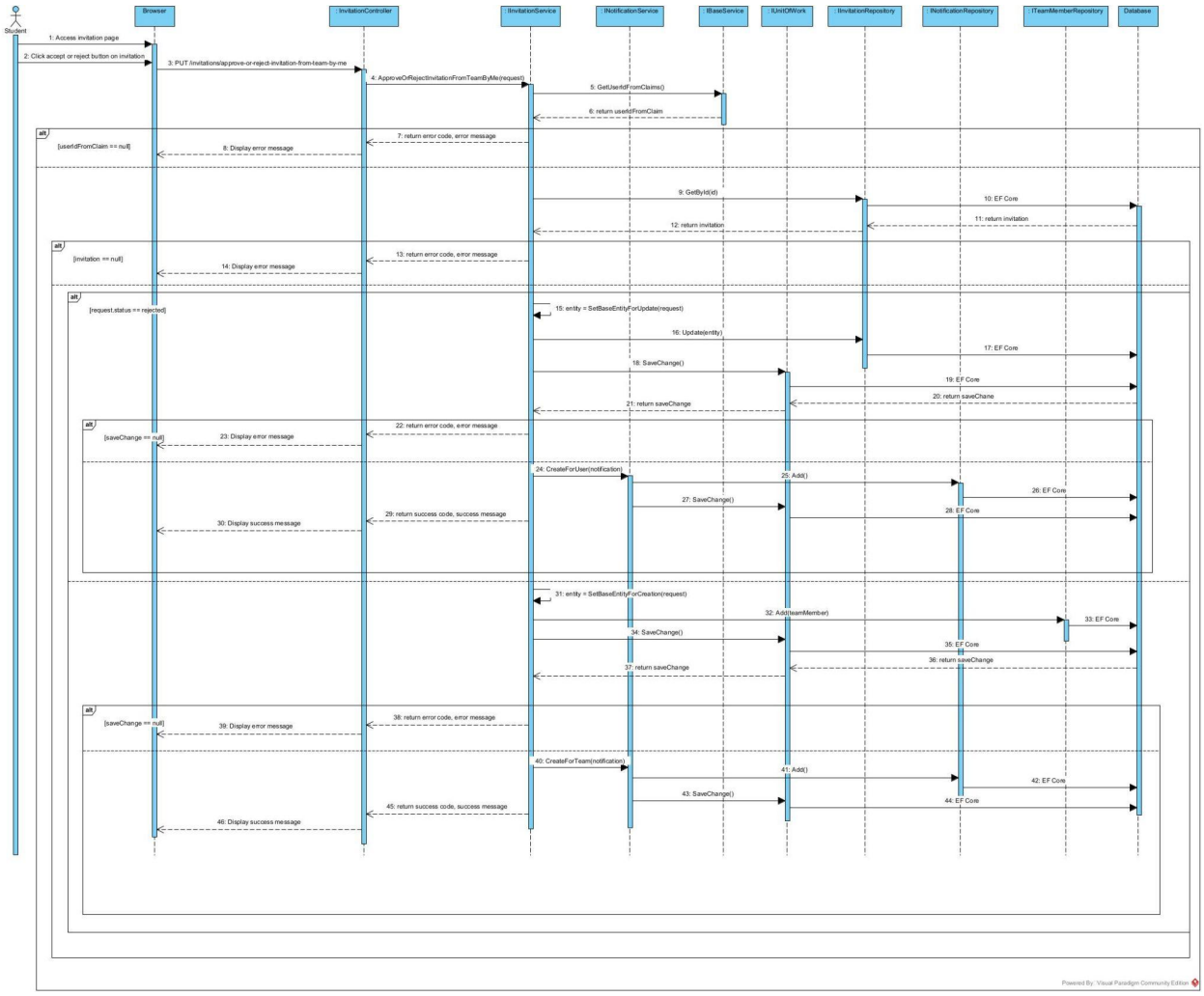


3.4.3 Respond to invitation.

3.4.3.1 Class Diagram

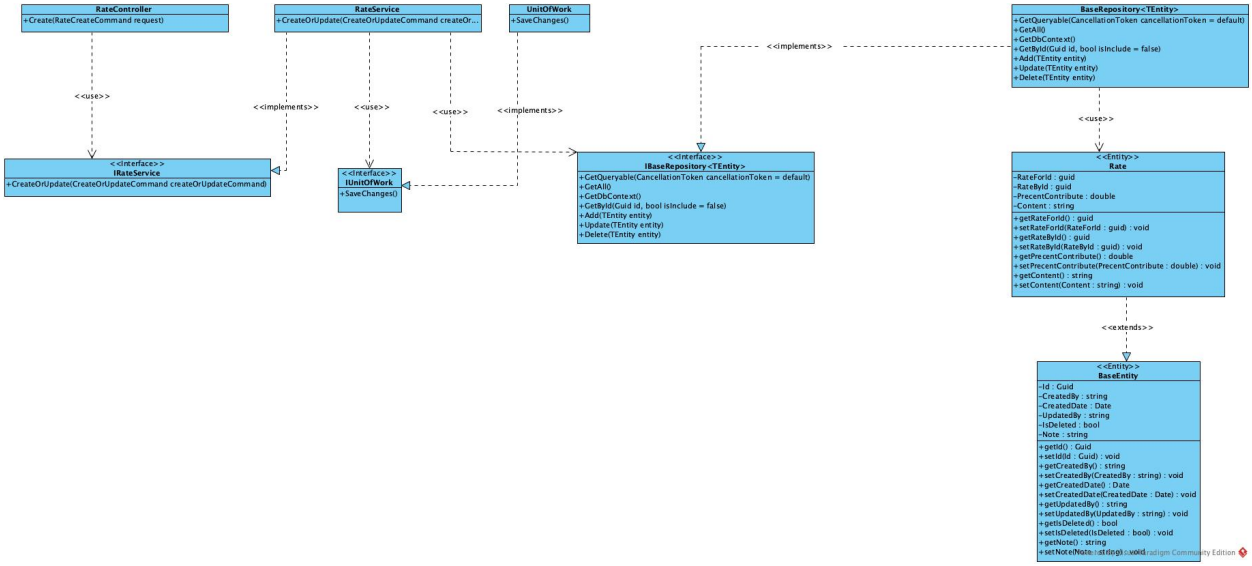


3.4.3.2 Sequence Diagram

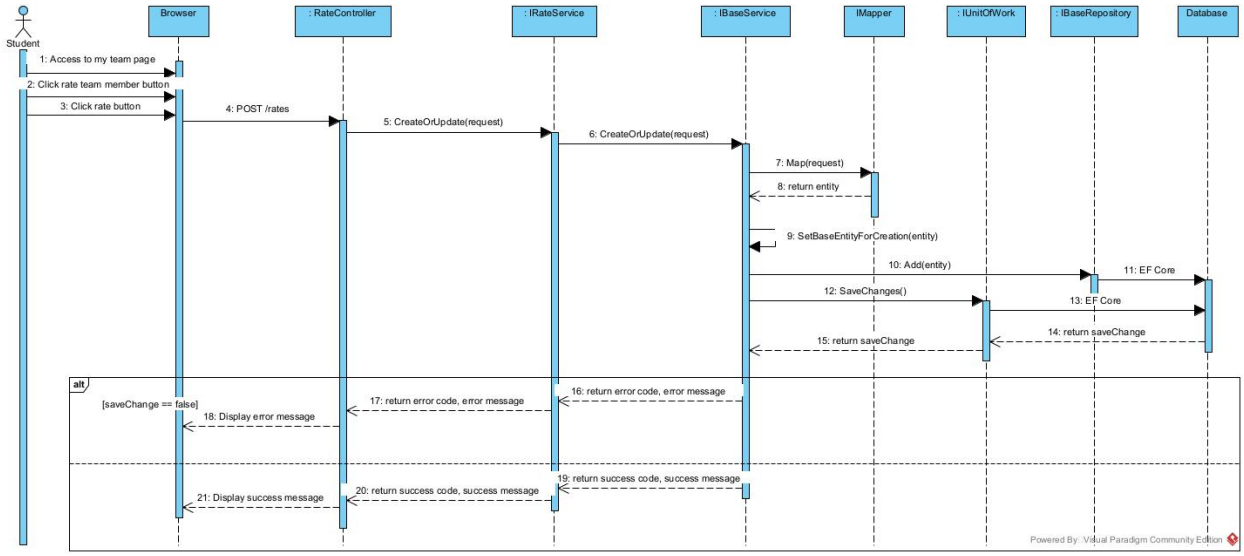


3.4.4 Leader rate member.

3.4.4.1 Class Diagram

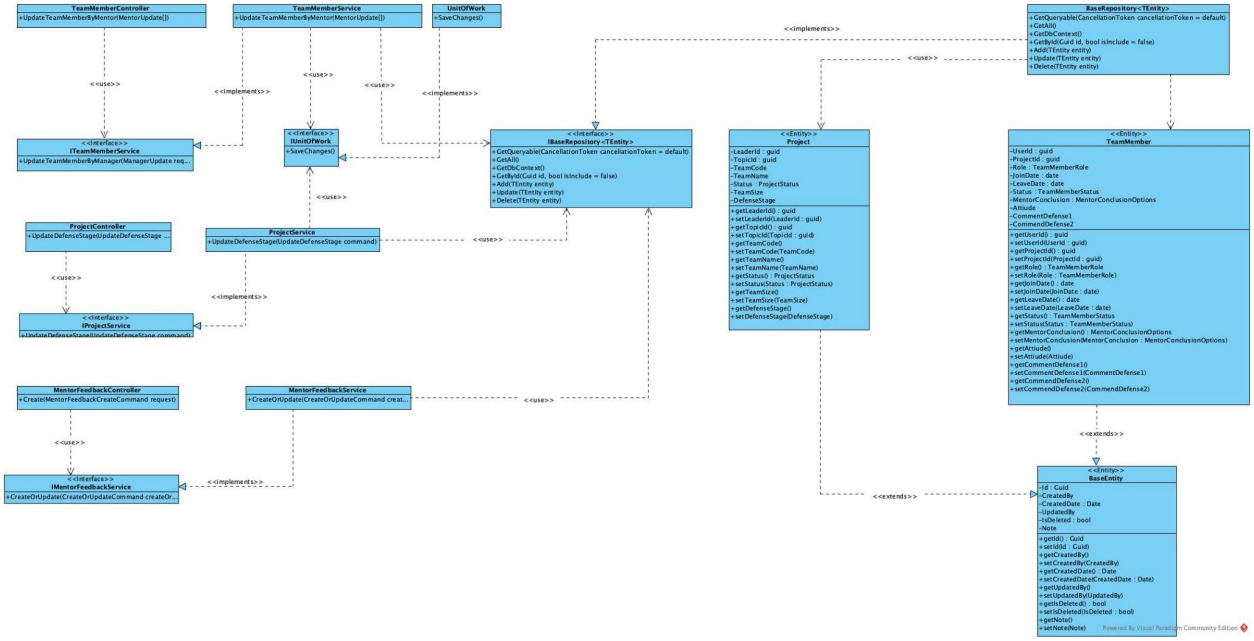


3.4.4.2 Sequence Diagram

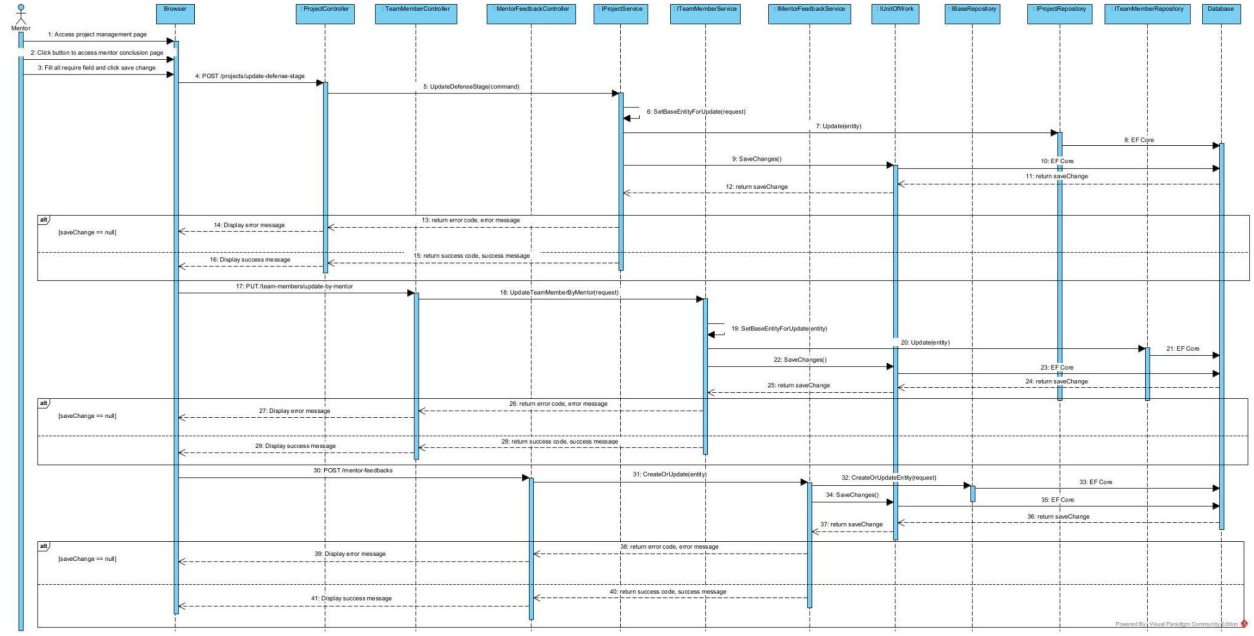


3.4.4 Mentor give conclusion evaluation

3.4.4.1 Class Diagram



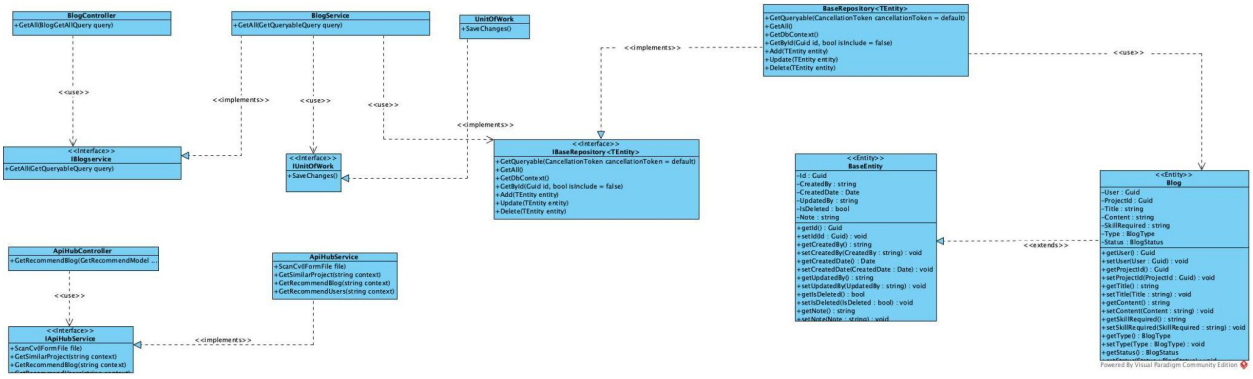
3.4.4.2 Sequence Diagram



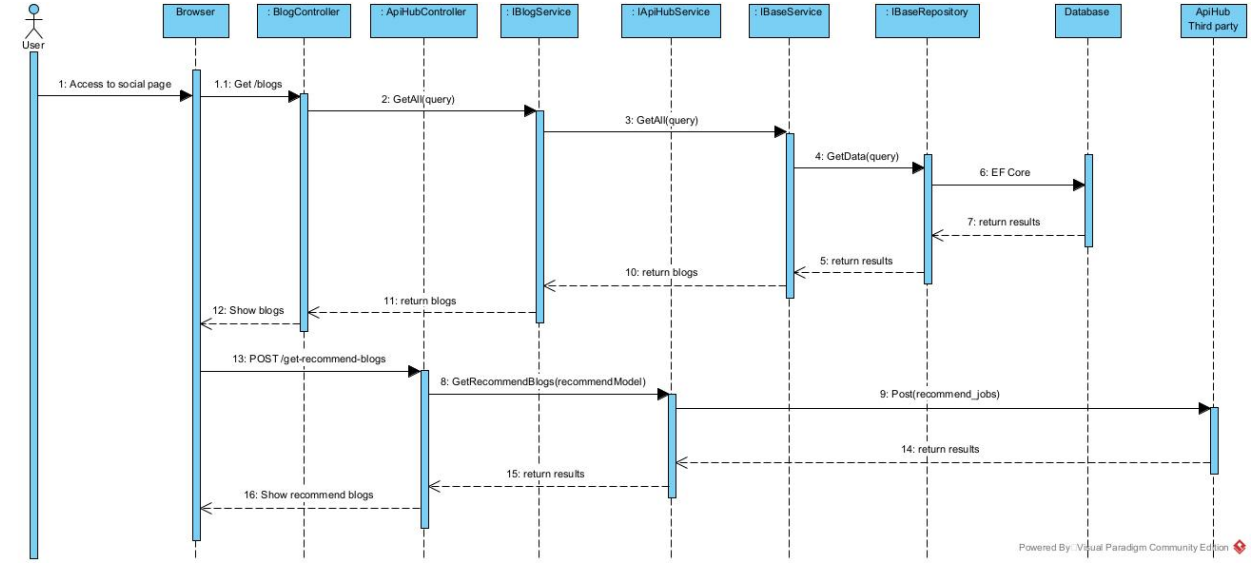
3.5 Blog Management

3.5.1 View blog

3.5.1.1 Class Diagram

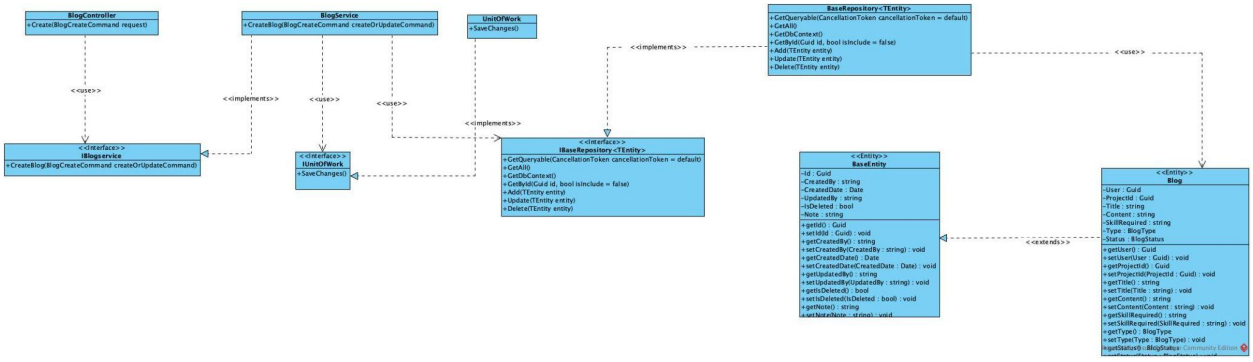


3.5.1.2 Sequence Diagram

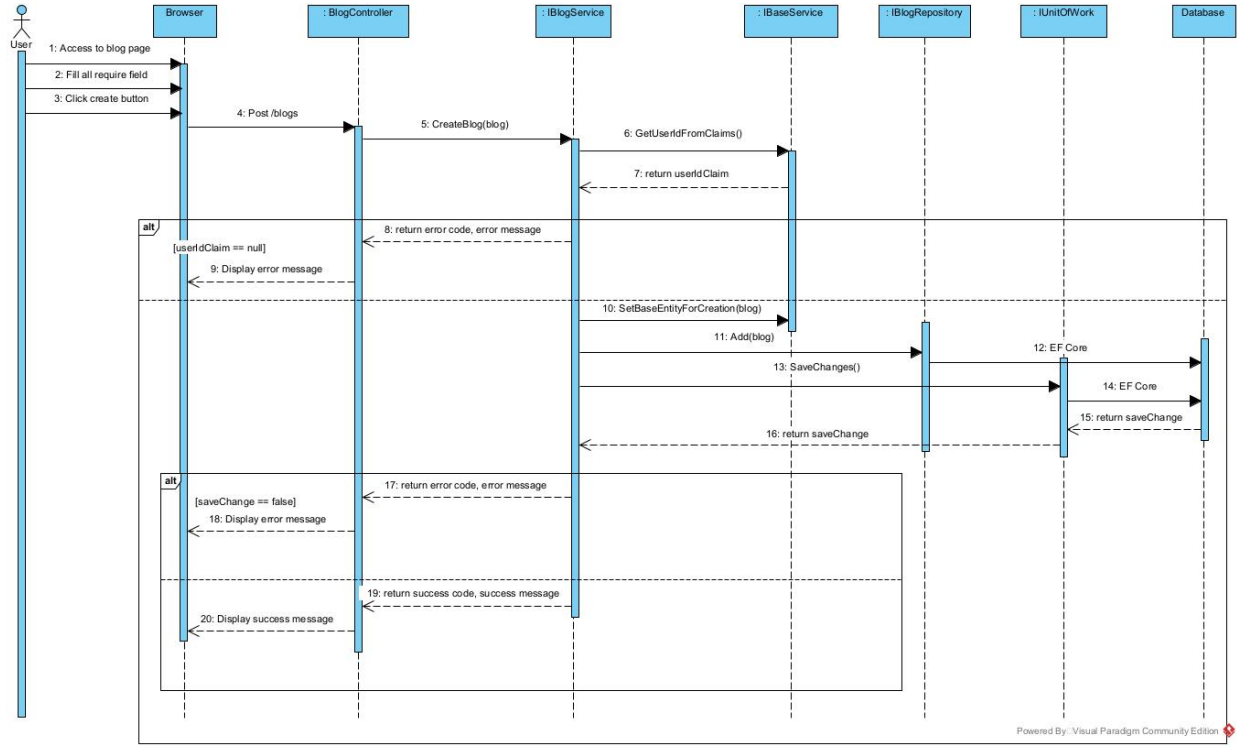


3.5.2 Create blog

3.5.2.1 Class Diagram

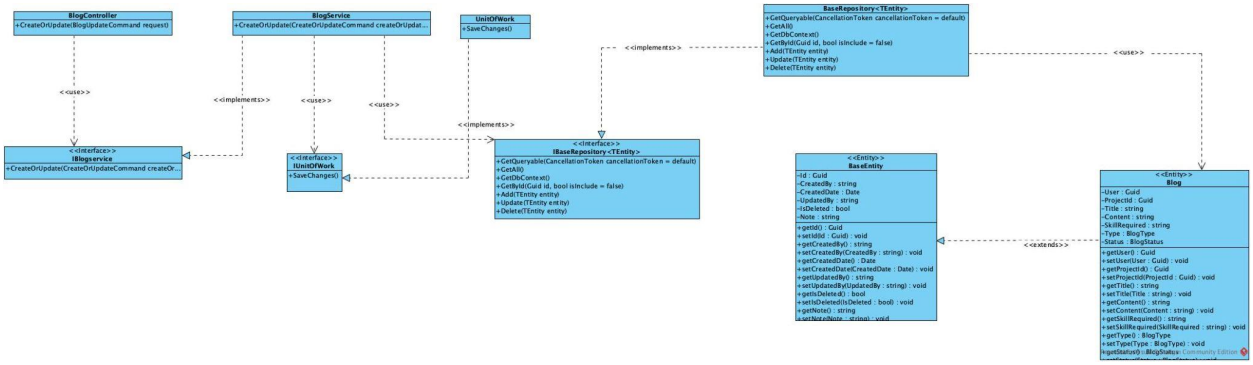


3.5.2.2 Sequence Diagram

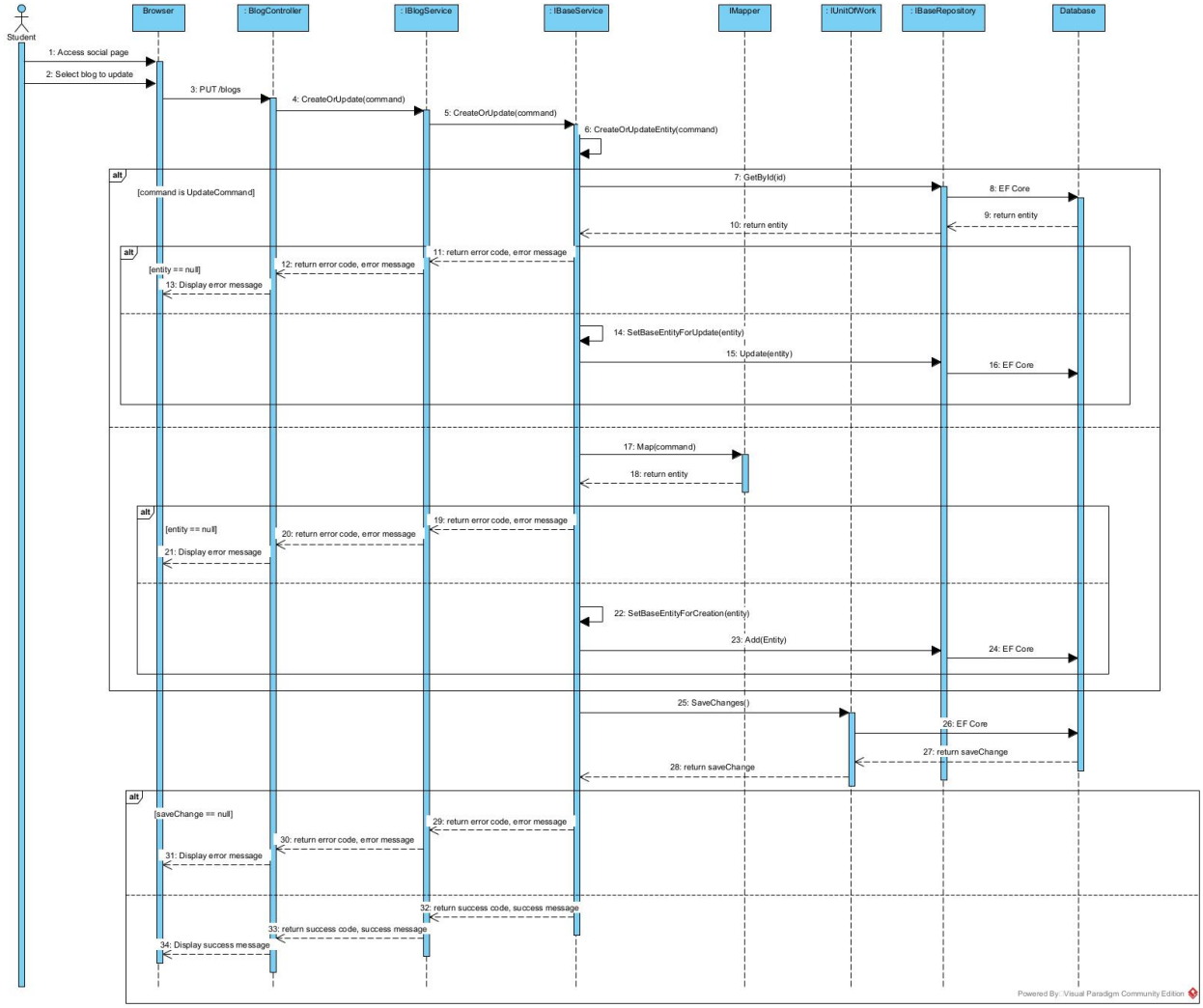


3.5.3 Update blog

3.5.3.1 Class Diagram

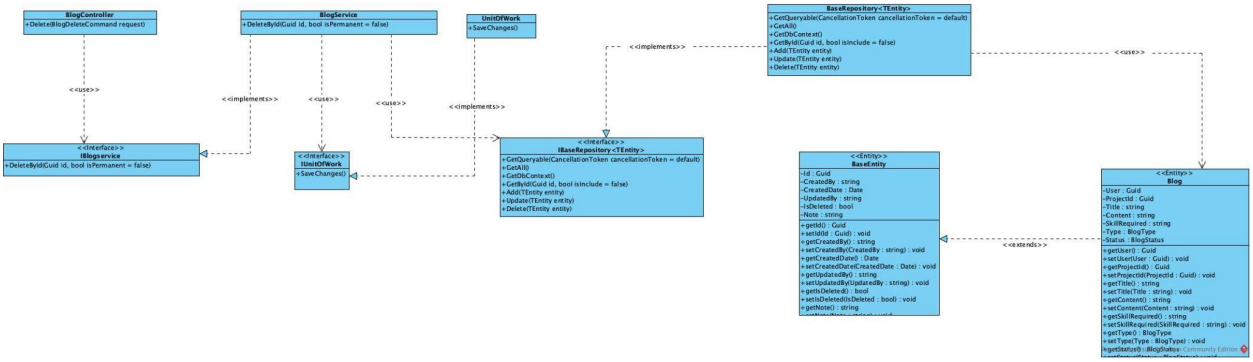


3.5.3.2 Sequence Diagram

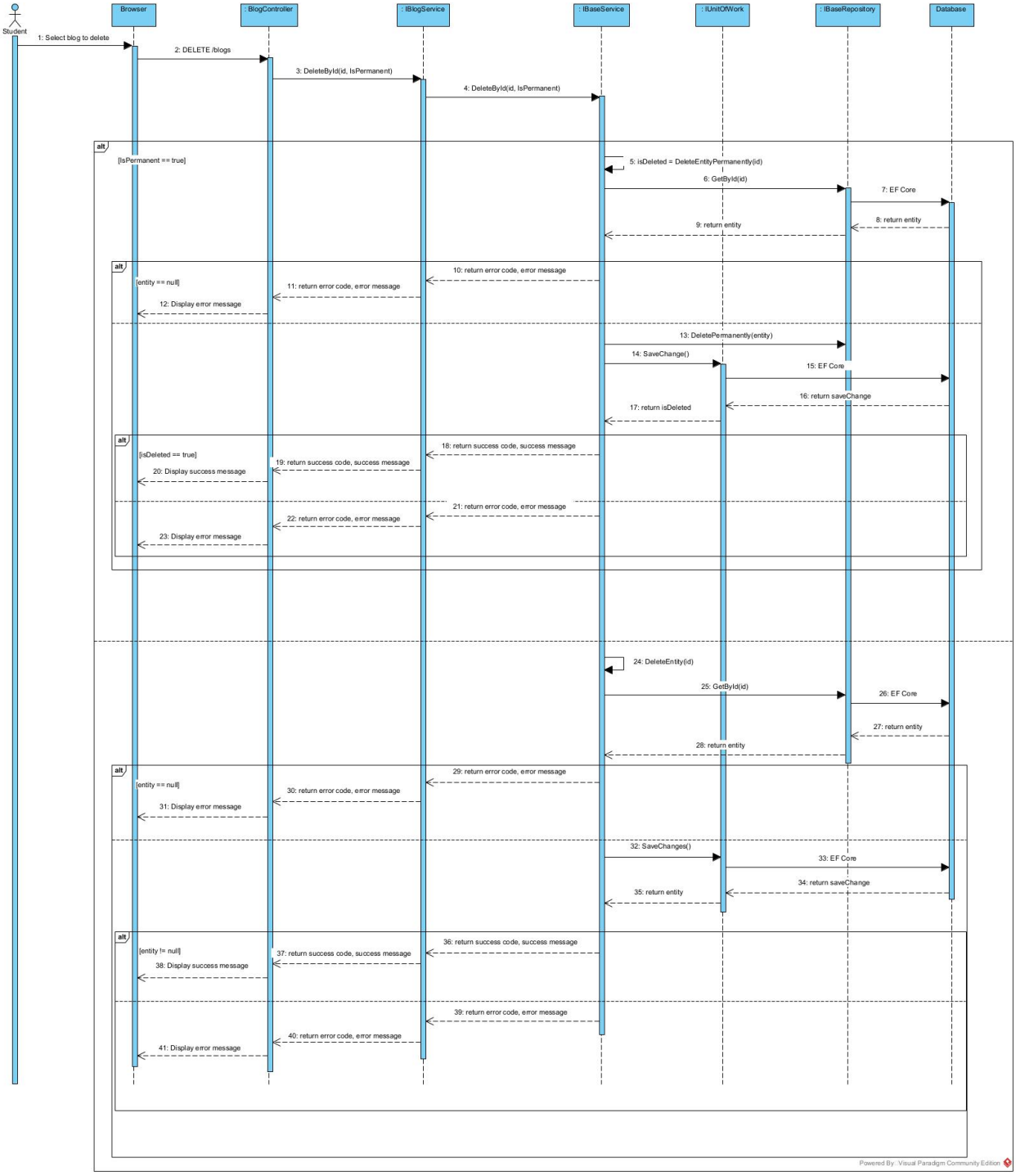


3.5.4 Delete blog

3.5.4.1 Class Diagram

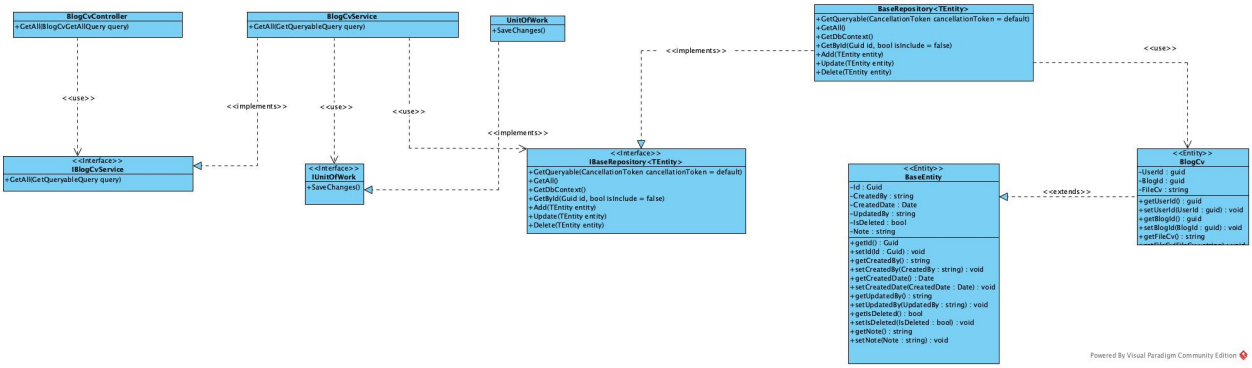


3.5.4.2 Sequence Diagram

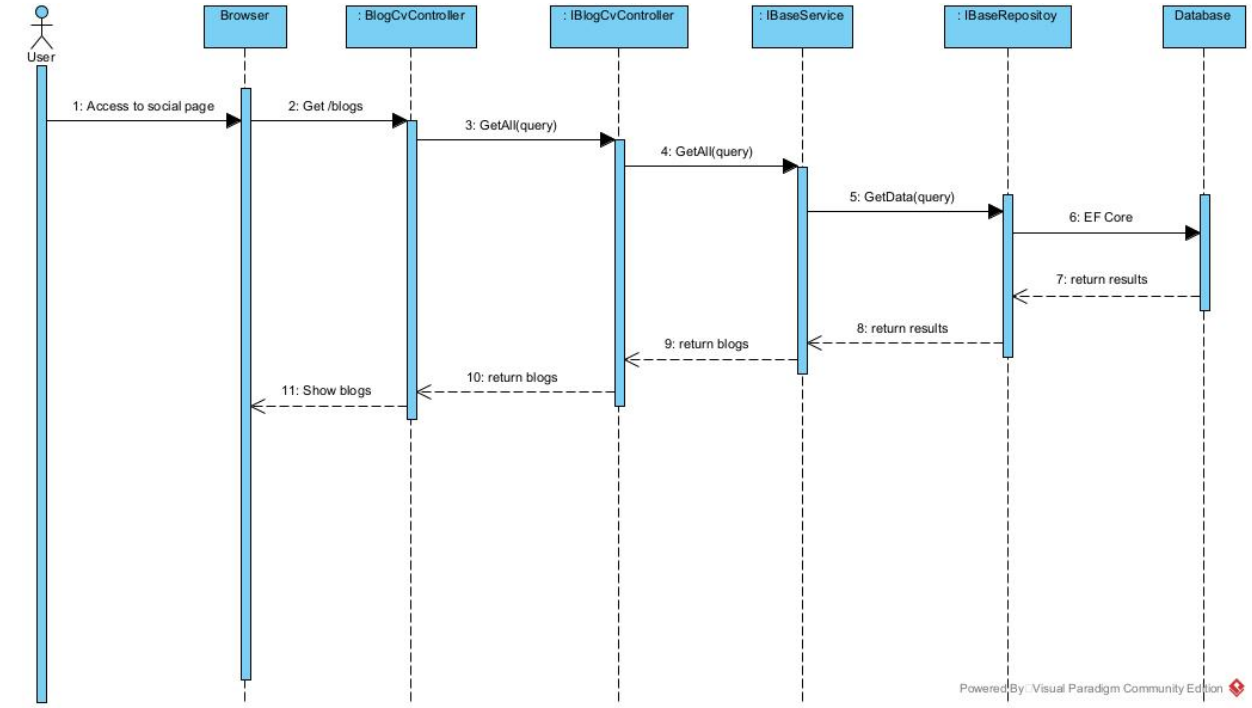


3.5.5 View apply CV

3.5.5.1 Class Diagram

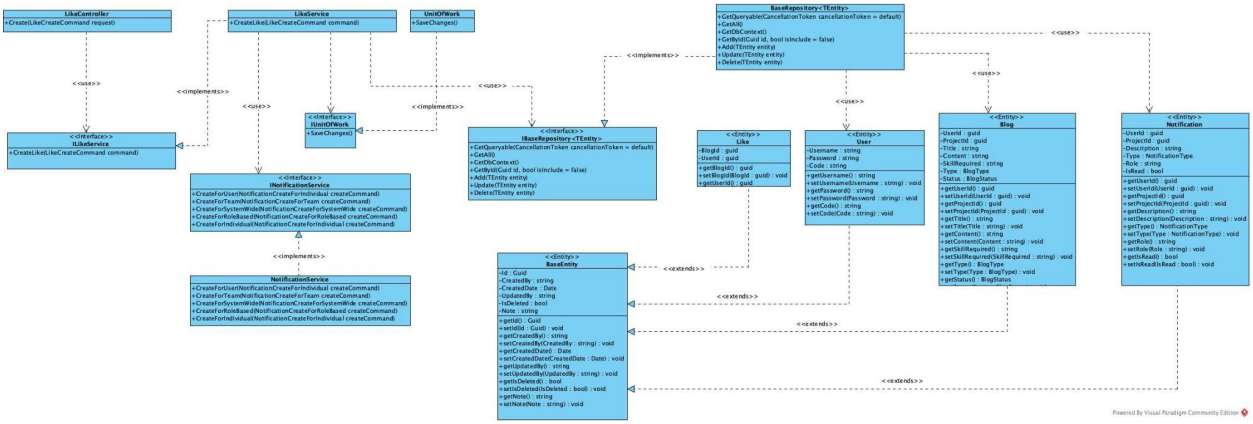


3.5.5.2 Sequence Diagram

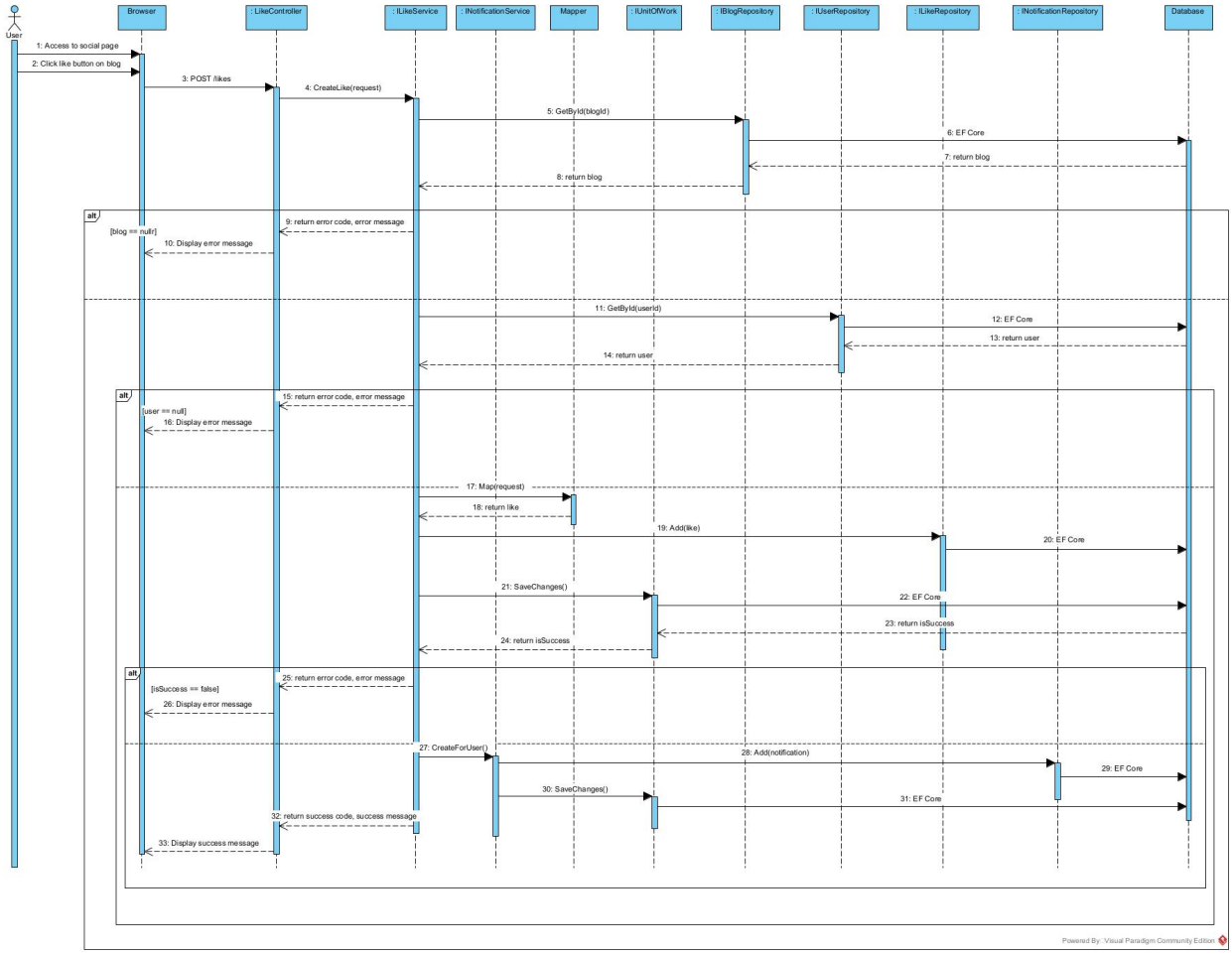


3.5.6 Like blog

3.5.6.1 Class Diagram

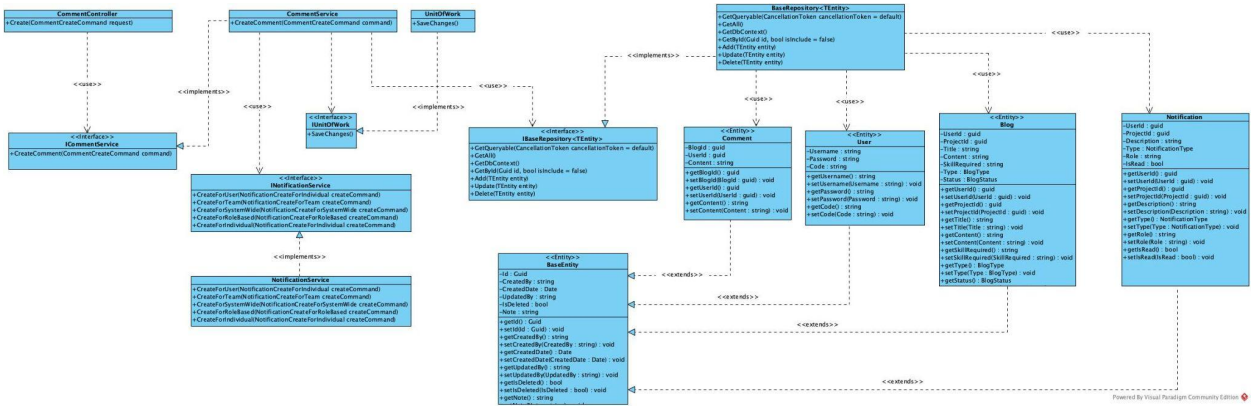


3.5.6.2 Sequence Diagram

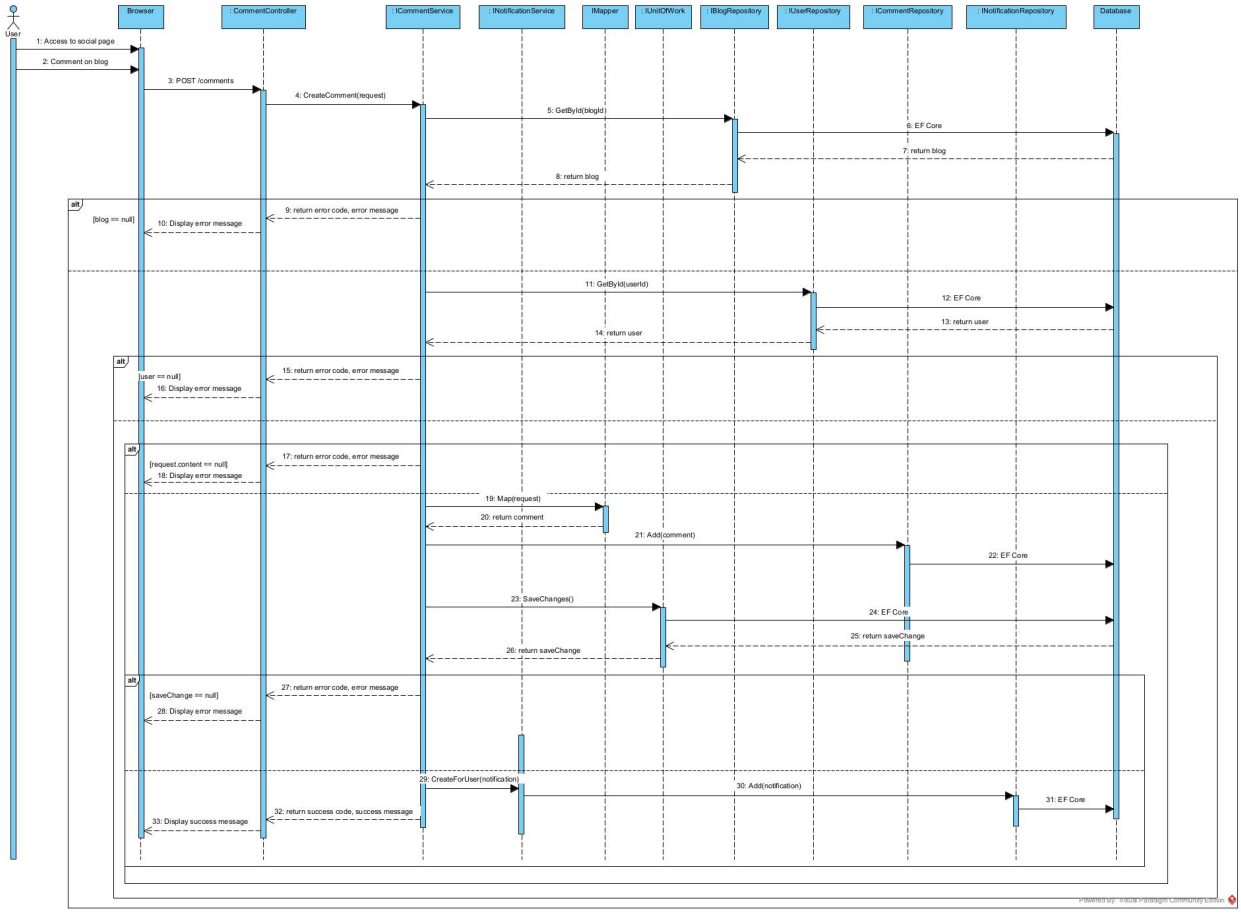


3.5.7 Comment blog

3.5.7.1 Class Diagram

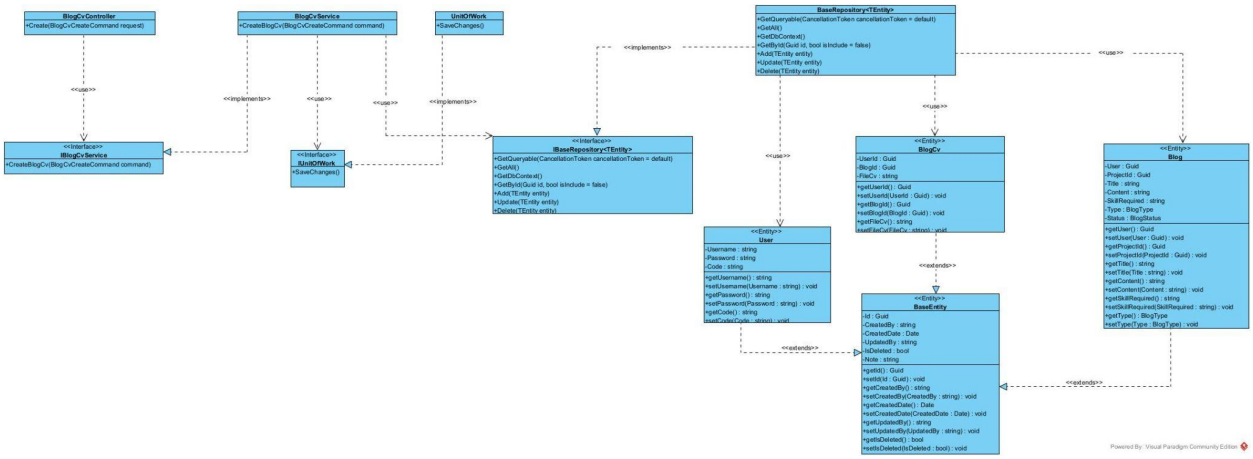


3.5.7.2 Sequence Diagram

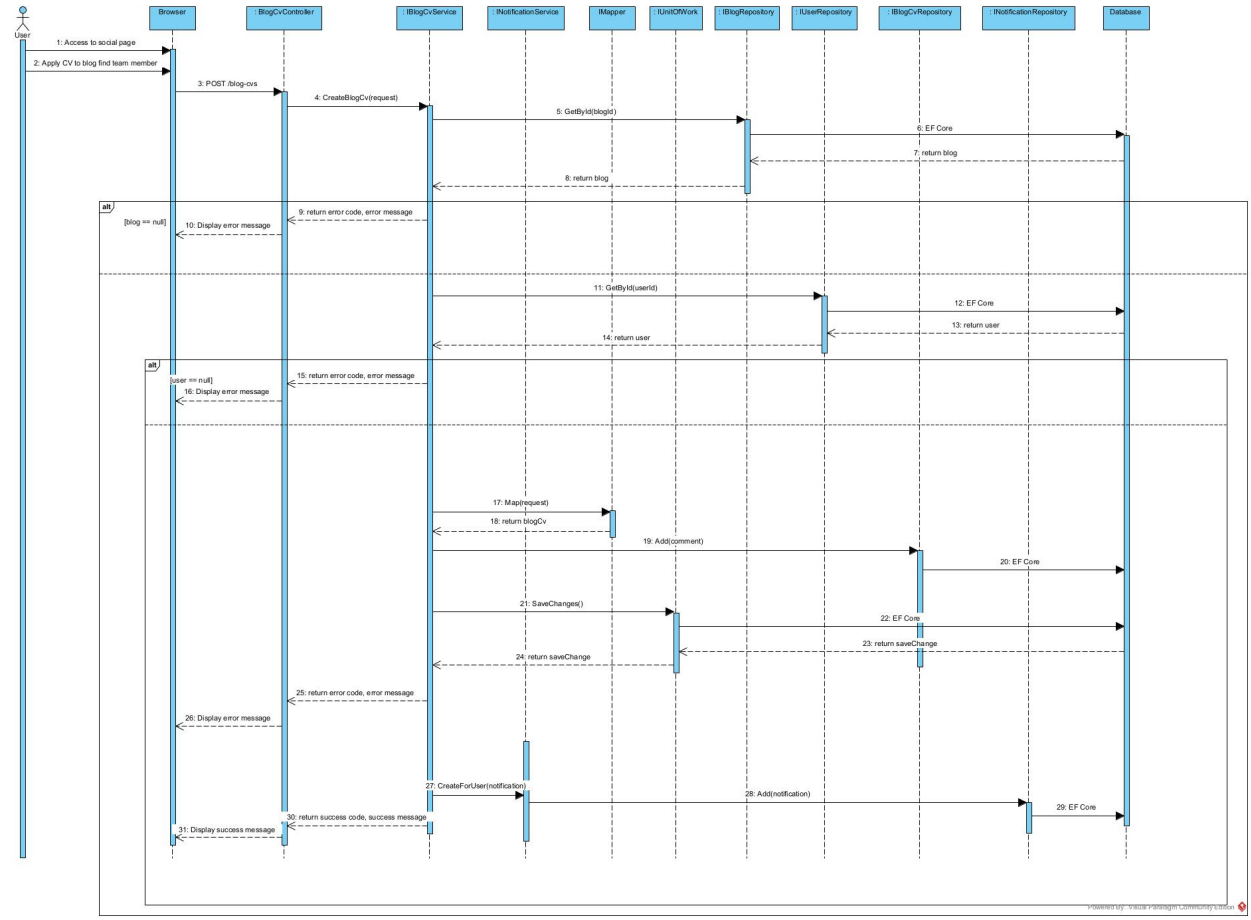


3.5.8 Appy CV

3.5.8.1 Class Diagram



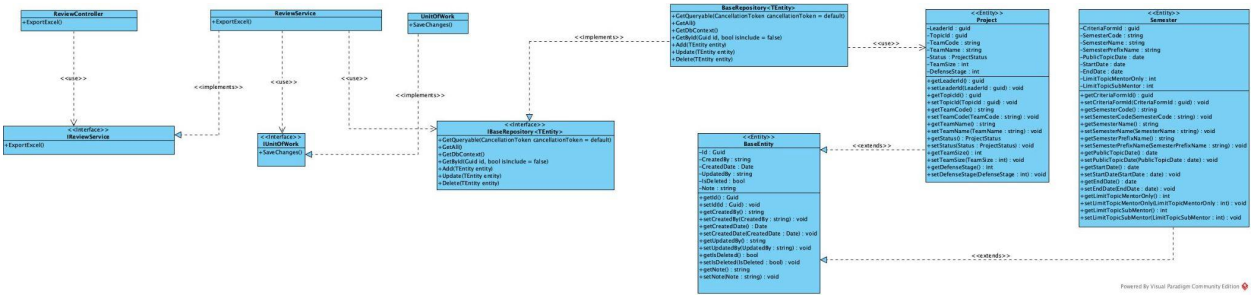
3.5.8.2 Sequence Diagram



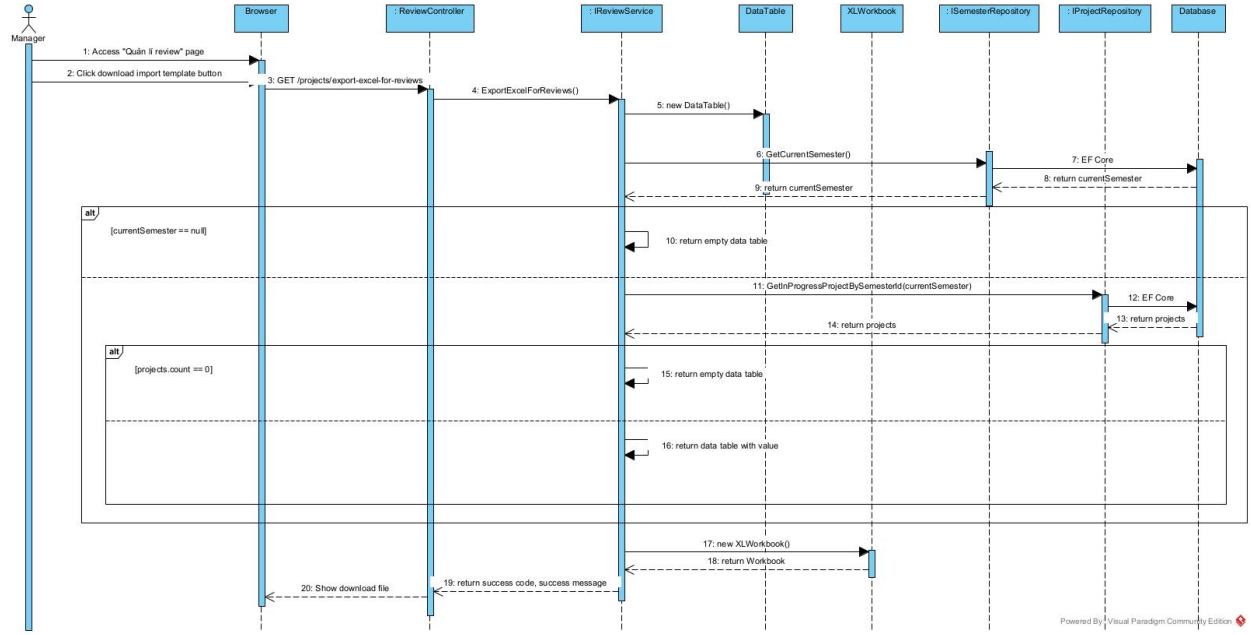
3.6 Review Management

3.6.1 Export review

3.6.1.1 Class Diagram

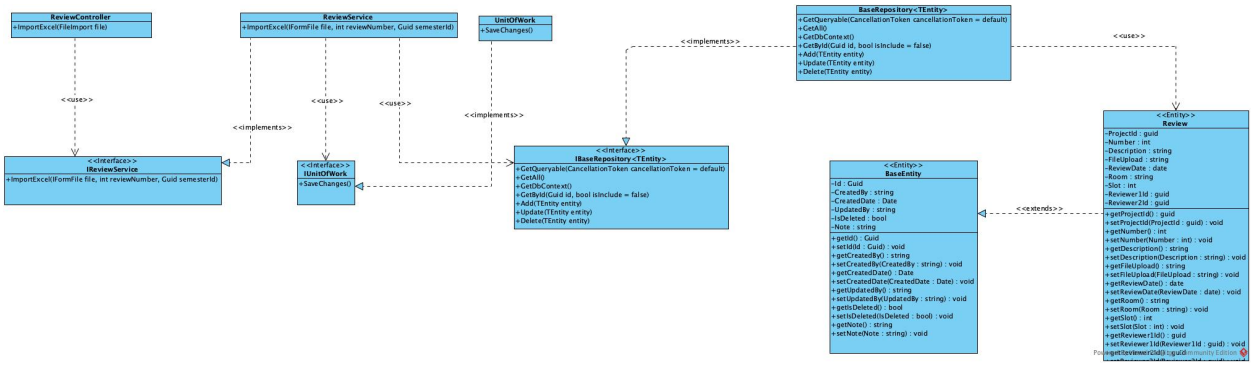


3.6.1.2 Sequence Diagram

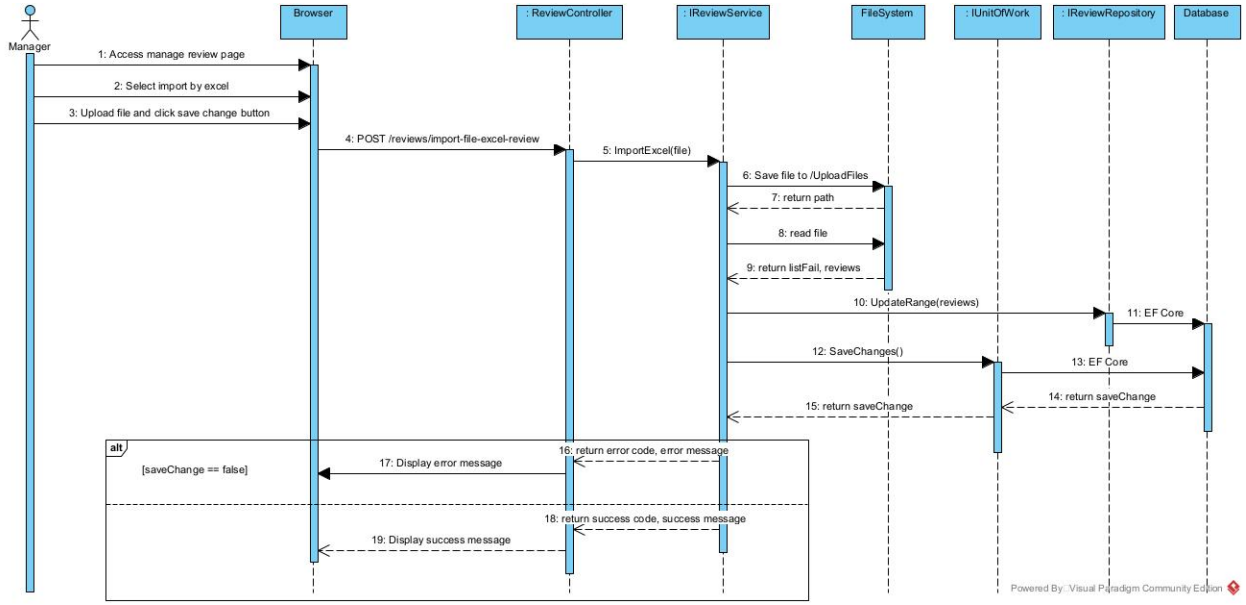


3.6.2 Import review

3.6.2.1 Class Diagram

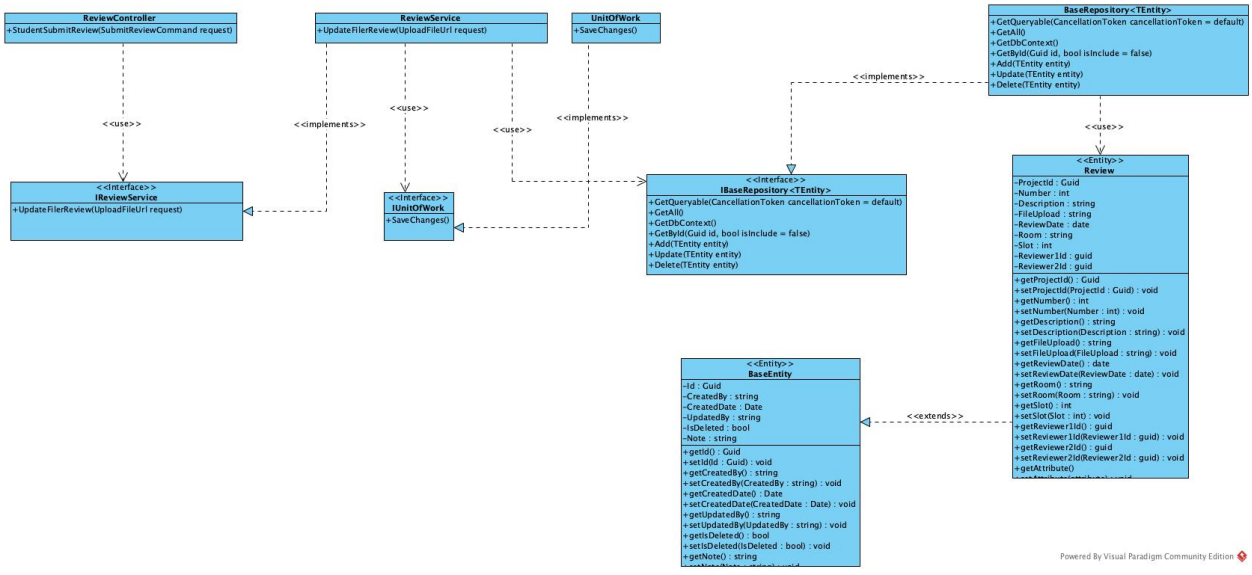


3.6.2.2 Sequence Diagram

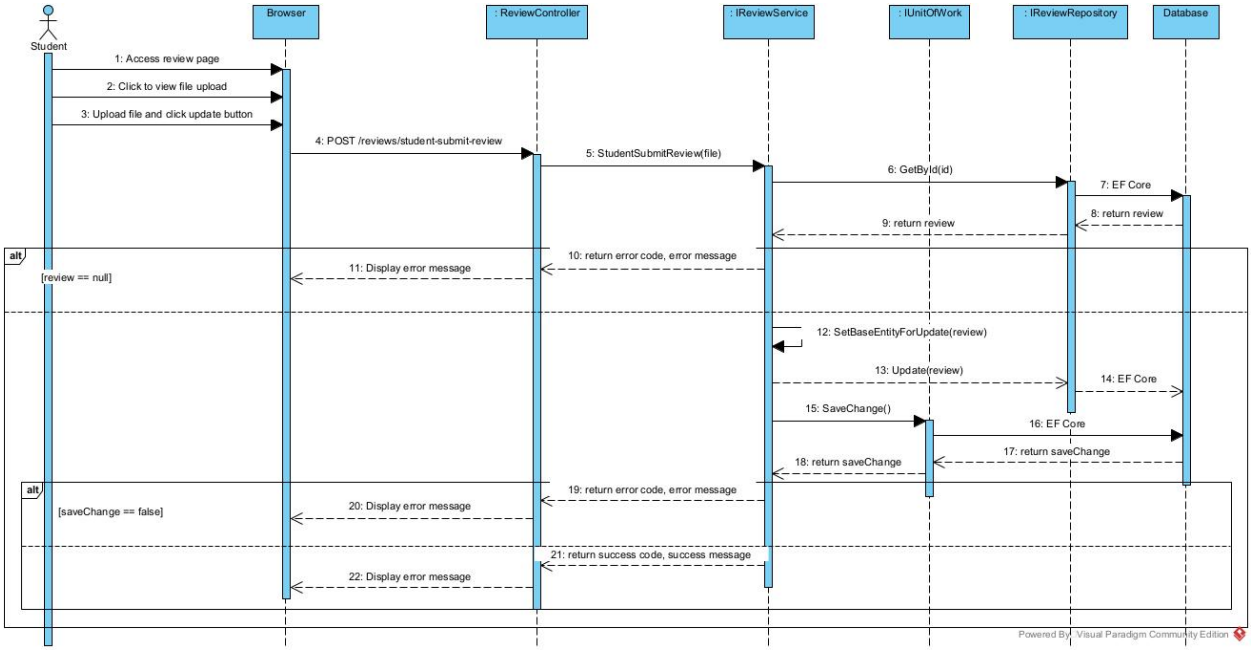


3.6.3 Student submit checklist file

3.6.2.1 Class Diagram



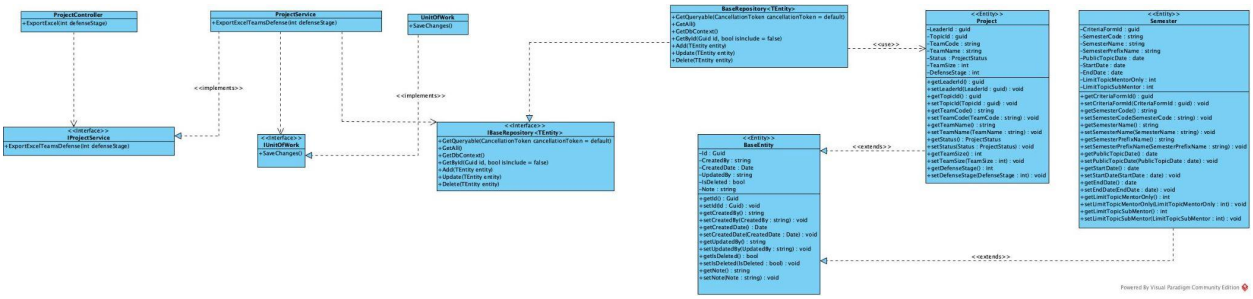
3.6.2.2 Sequence Diagram



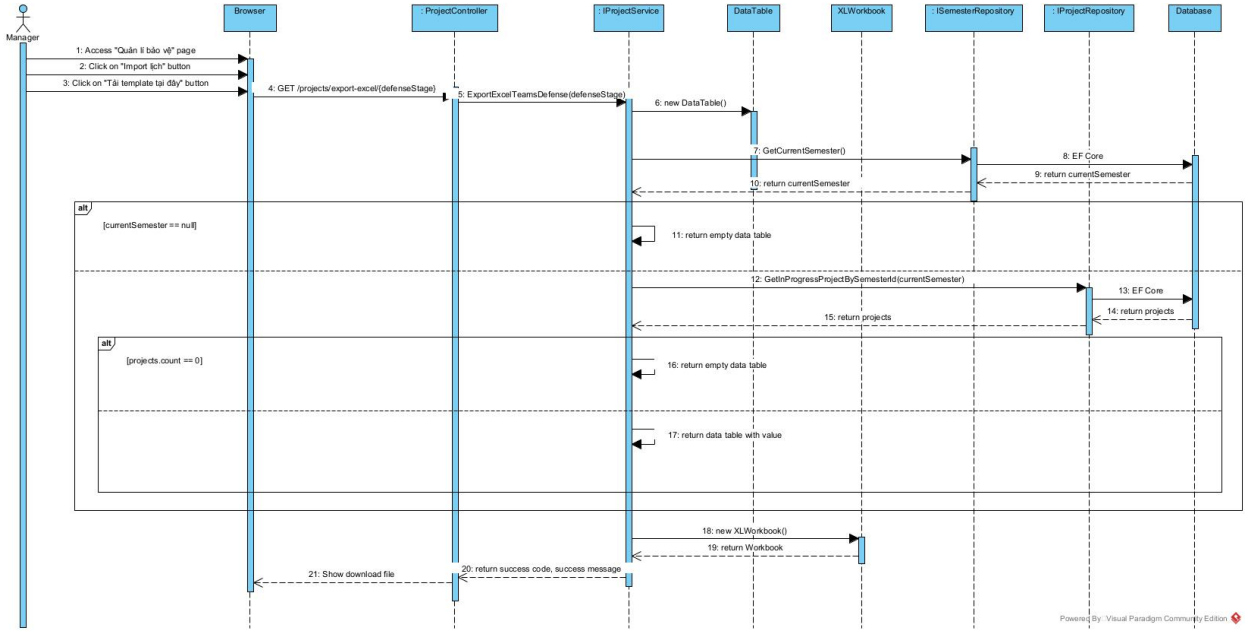
3.7 Defense Management

3.7.1 Export template defense schedule

3.7.1.1 Class Diagram

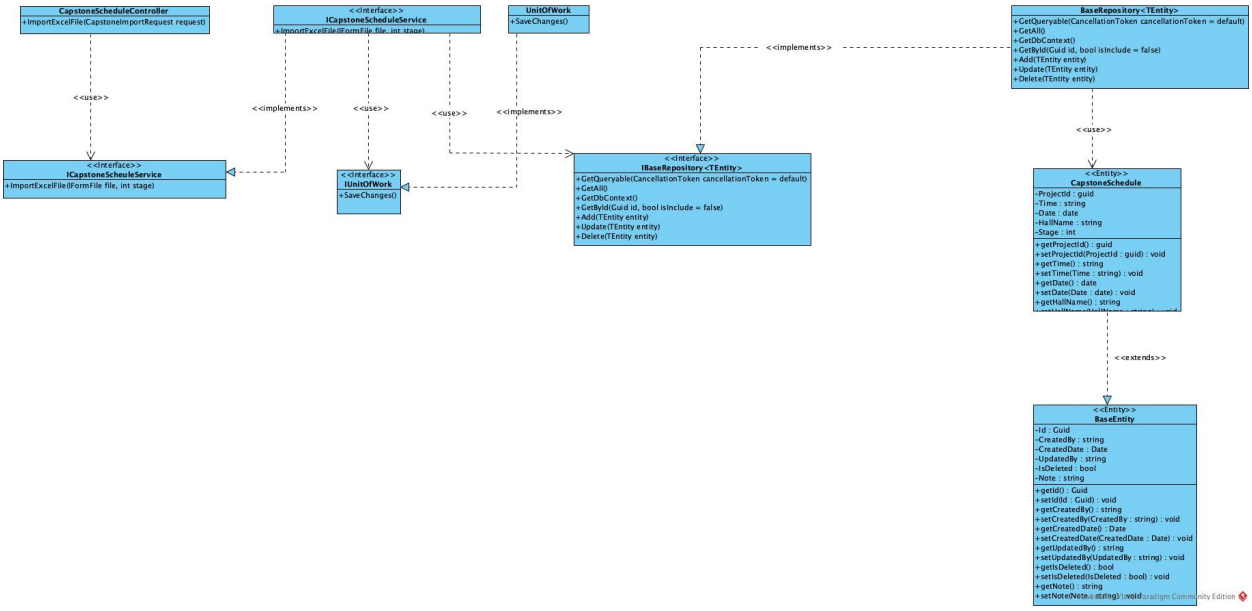


3.7.1.2 Sequence Diagram

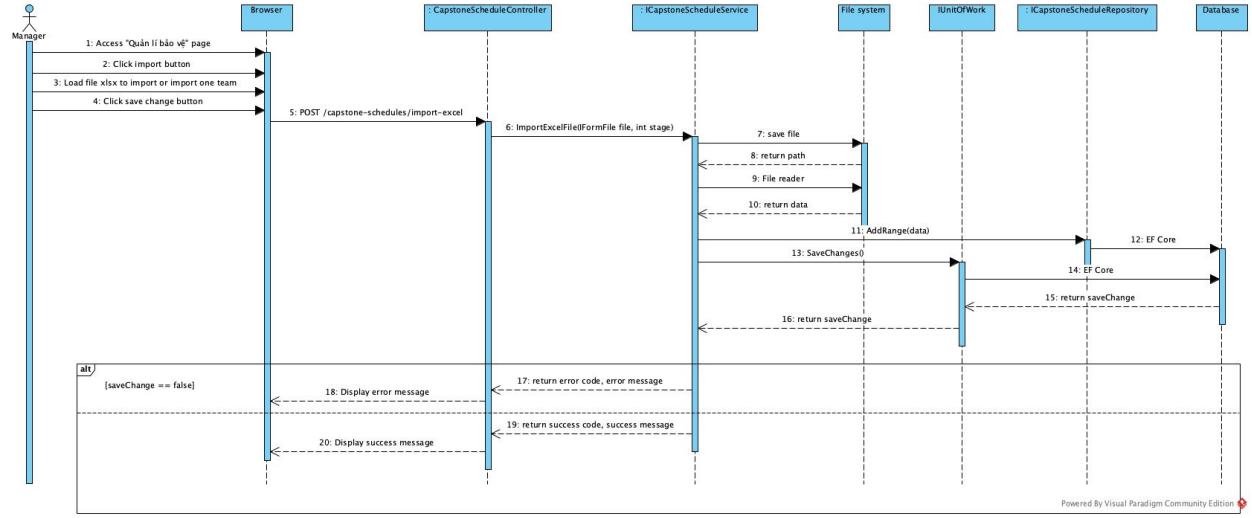


3.7.2 Import defense

3.7.2.1 Class Diagram

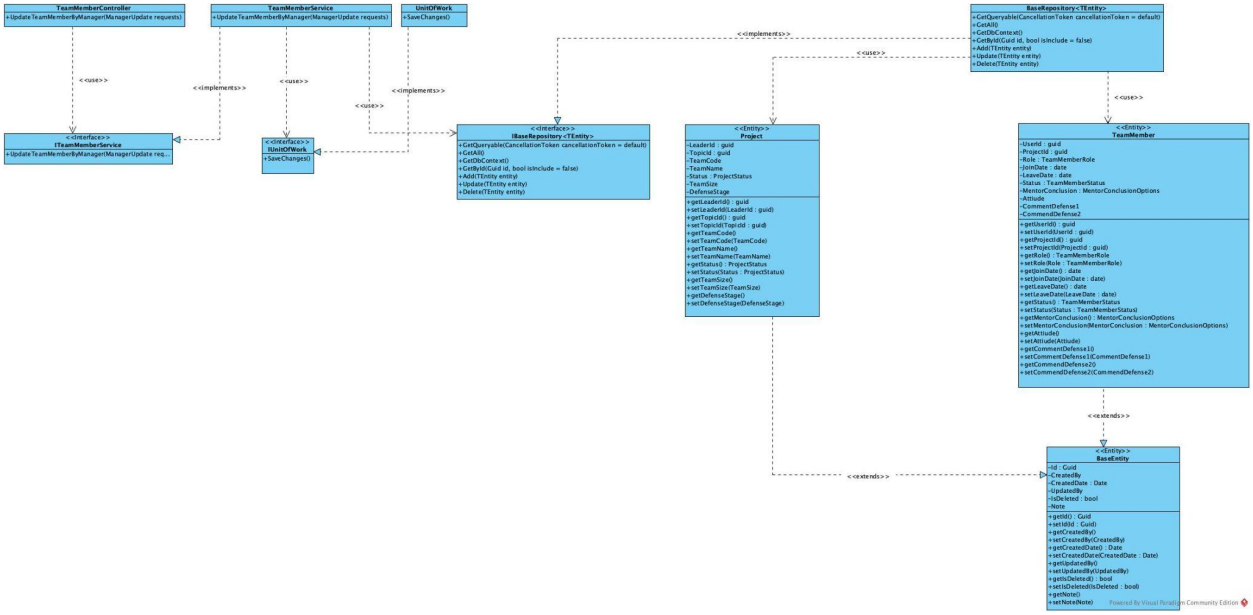


3.7.2.2 Sequence Diagram

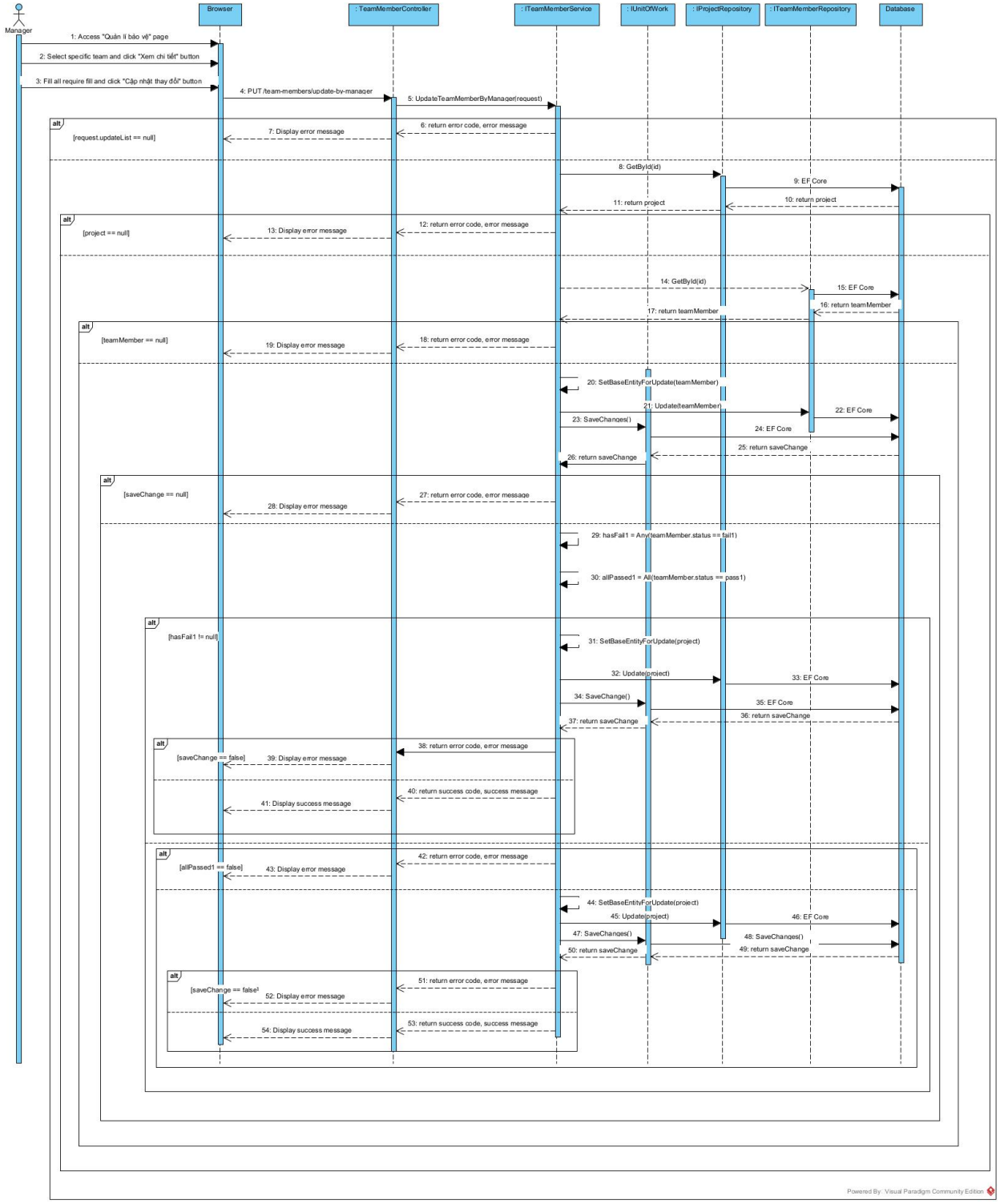


3.7.3 Manager update defense stage 1

3.7.3.1 Class Diagram

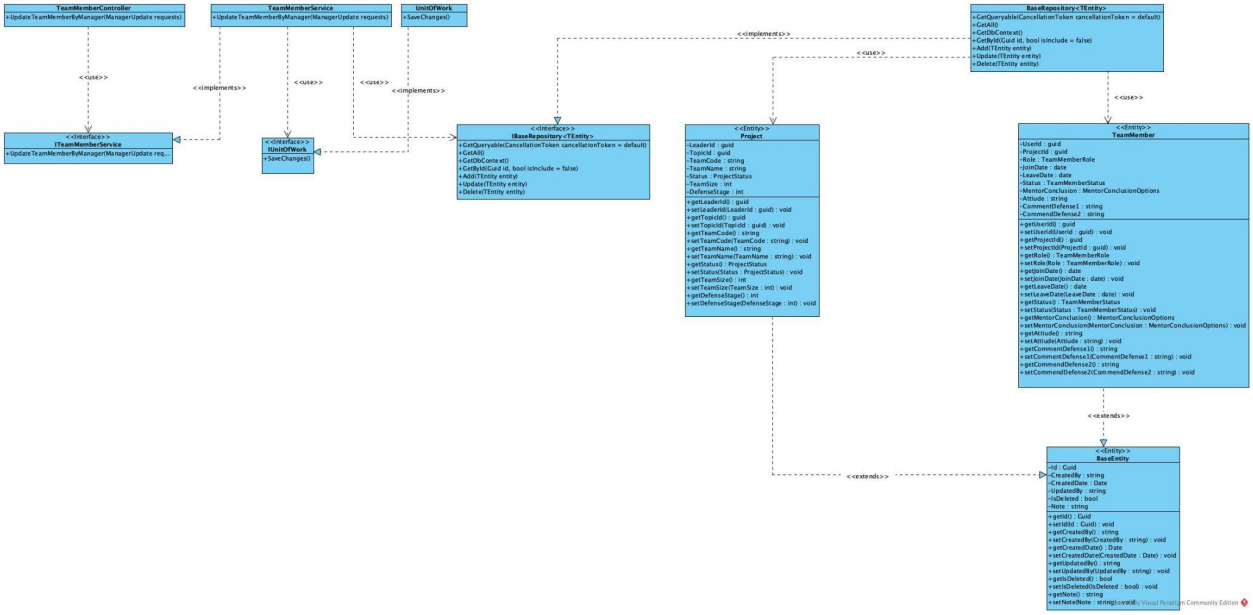


3.7.2.2 Sequence Diagram

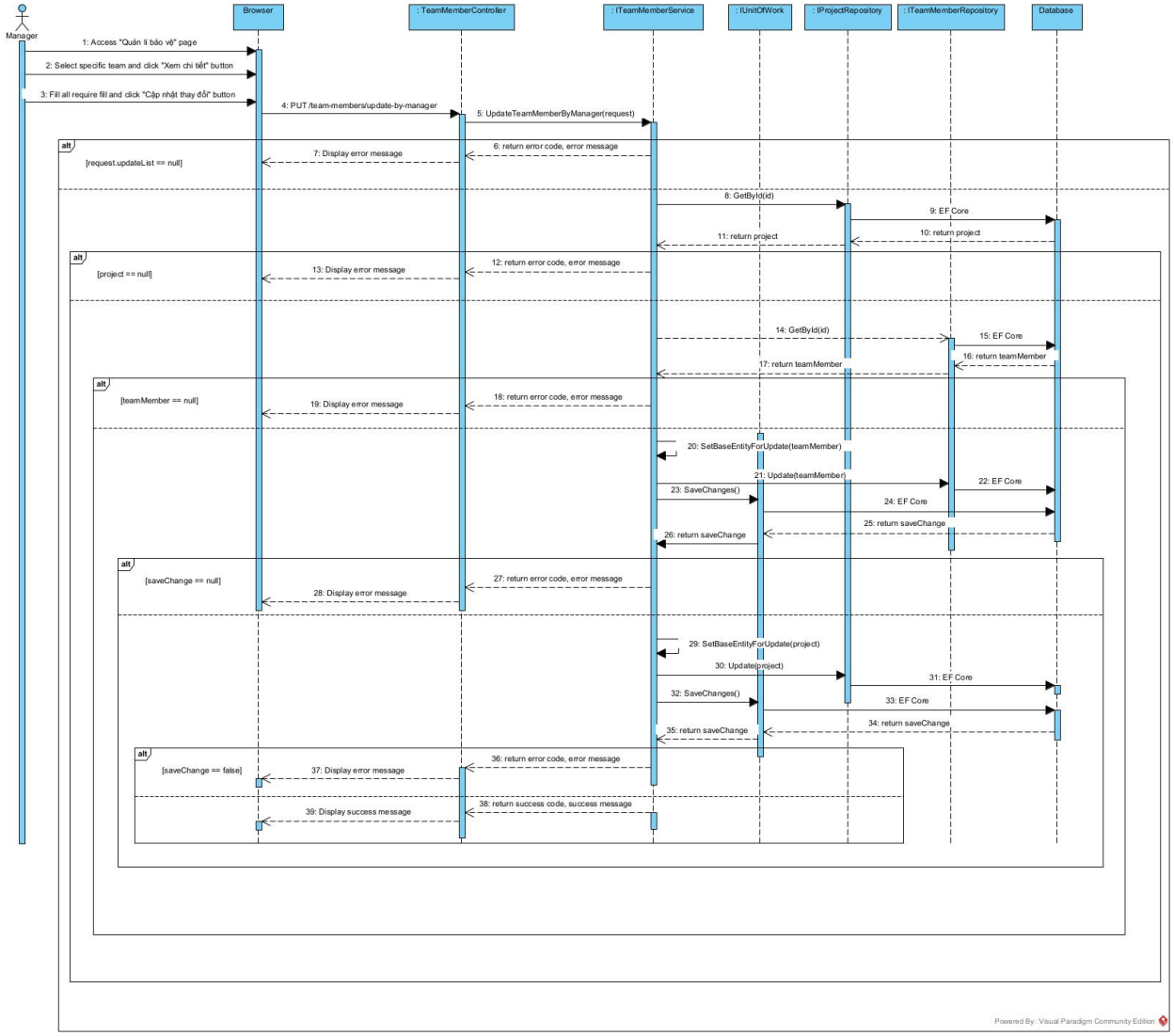


3.7.3 Manager update defense stage 2

3.7.3.1 Class Diagram



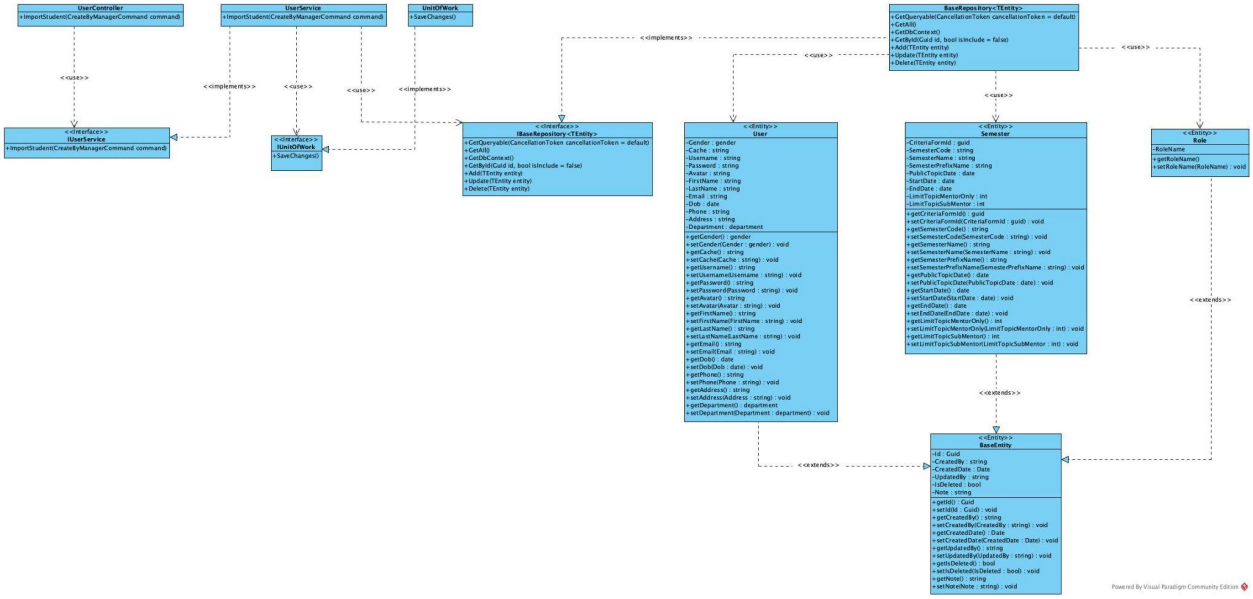
3.7.3.2 Sequence Diagram



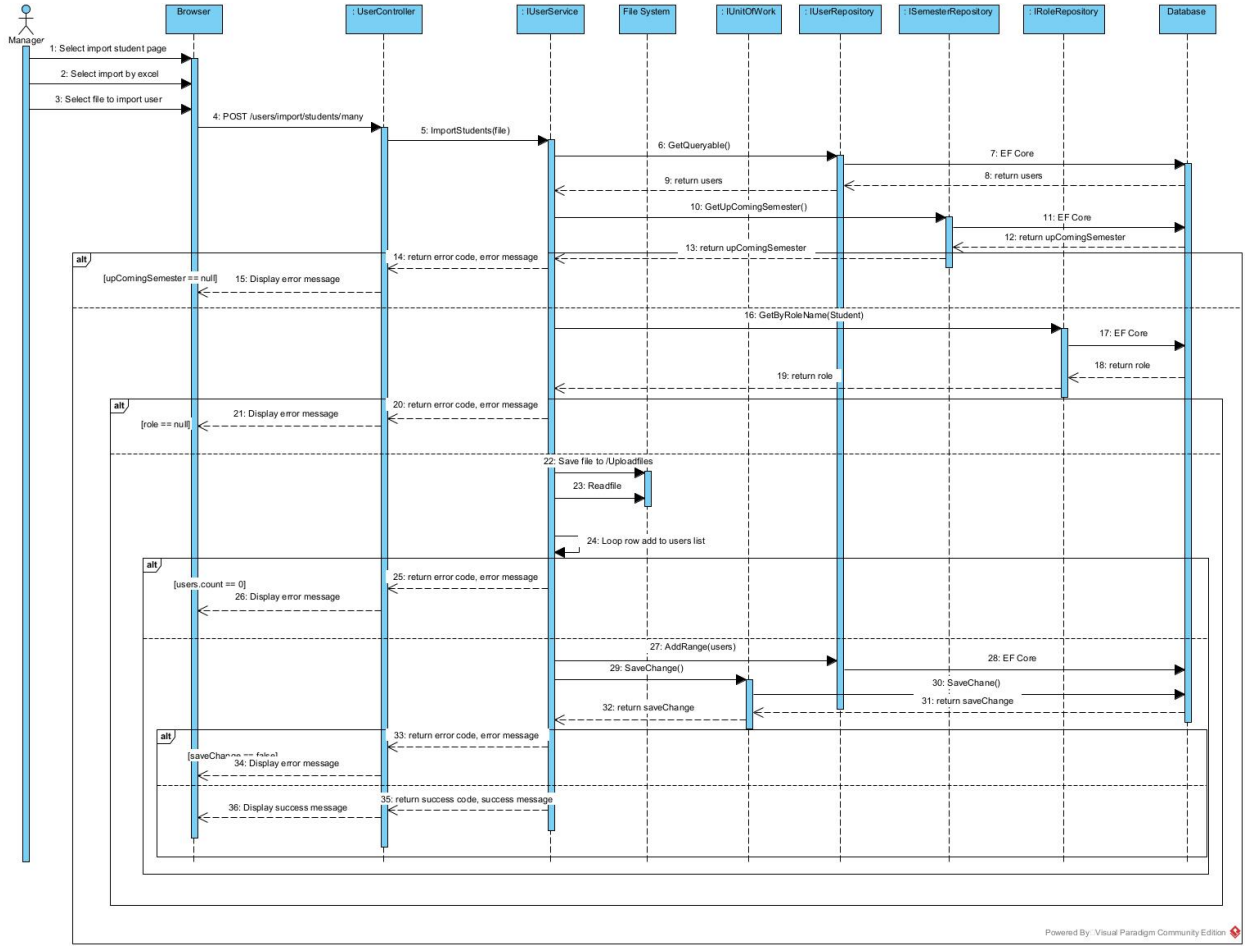
3.8 User management

3.8.1 Import user

3.8.1.1 Class Diagram



3.8.1.2 Sequence Diagram



V. Software Testing Documentation

1. Scope of Testing

- Scope:

- **Features to be tested**:
  - Create exams, exam questions, and import student lists.
  - Automatically grade student submissions based on their source code.
  - Check plagiarism and evaluate source code quality using SonarQube.
  - Deploy exams via Docker and generate test cases using AI.
  - Store data and export grade reports.
- **Features not to be tested**:
  - Integration with external platforms outside the project's scope.
  - Performance testing for extremely large-scale use cases (>10,000 students).

- Testing Levels:

- **Unit Testing**:
    - Responsible: Developers.
    - Inputs: Individual modules like exam creation, grading, plagiarism checking.
    - Focus: Validate module-level functionalities meet design specifications.
    - Acceptance Criteria: Modules function as expected in isolation.
- **Integration Testing**:
    - Responsible: Development and QA teams.
    - Inputs: Interactions between modules (e.g., AI test case generation with grading).
    - Focus: Ensure seamless communication and data flow among system components.
    - Acceptance Criteria: Modules integrate correctly without errors or inconsistencies.
- **System Testing**:
    - Responsible: QA team.
    - Inputs: Entire system with all features integrated.
    - Focus: Validate the complete functionality of the system.
    - Acceptance Criteria: The system meets all functional and non-functional requirements.

- Constraints and Assumptions:

- Testing is limited to a simulated environment with a maximum of 1,000 students.
- Assumes stable internet and infrastructure during testing.

2. Test Strategy

2.1 Testing Types

- Unit Testing:

- **Objective**: Verify individual modules' correctness.
- **Technique**: White-box testing with automated tools (e.g., Jest for TypeScript).
- **Completion Criteria**: All unit tests achieve a pass rate of at least 95%.

- Integration Testing:

- **Objective**: Validate interactions between modules (e.g., AI test cases and Docker deployments).
- **Technique**: API and interface testing.
- **Completion Criteria**: All integration tests pass with no major defects.

- System Testing:

- **Objective**: Confirm the entire system functions as expected.
- **Technique**: Black-box testing.
- **Completion Criteria**: All functional and non-functional requirements are met.

- Performance Testing:

- **Objective**: Assess the system's behavior under various workloads.
- **Technique**: Load testing using JMeter or Locust.
- **Completion Criteria**: System maintains response times <3s under 1,000 users.

2.2 Test Levels

- Unit Testing: Focused on individual modules like grading, plagiarism checking.
- Integration Testing: Covers module interactions, especially between the database and AI test case generator.
- System Testing: Comprehensive testing of the entire system.

2.3 Supporting Tools

- **Unit Testing**: Jest, Mocha (for TypeScript).
- **Integration Testing**: Postman, Newman (for API testing).
- **System Testing**: Selenium (for UI testing).
- **Performance Testing**: JMeter, Locust.
- **Quality Check**: SonarQube.

3. Test Plan

3.1 Human Resources

- **Test Lead**: Oversees all testing activities and ensures test coverage.
- **Test Engineers**: Perform manual and automated testing.
- **Developers**: Handle unit testing and bug fixes.

3.2 Test Environment

- **Software**: Docker, Jest, SonarQube, Selenium, Postman.
- **Hardware**: Servers with minimum 16 GB RAM, multi-core processors.
- **Infrastructure**: Staging environment replicating production setup.

3.3 Test Milestones

- **Unit Testing Completion**: Month 1.
- **Integration Testing Completion**: Month 2.
- **System Testing Completion**: Month 3.

4. Test Cases

- Unit Test Cases: See Report5_Unit Test.xls.
- Other Test Cases (Integration, System, Acceptance): See Report5_Test Report.xls.

5. Test Reports

- Provide detailed test results, defect statistics, and analysis in Report5_Test Report.xls.
- Summarize pass/fail rates and recommendations for improvement.

VI. Release Package & User Guides

1. Deliverable Package

No.	Deliverable Item	Description
1	Schedule/Task Tracking	We use Trello to track our progress, Google Drive to store schedules, notes and documents: ● Trello ● Google Drive
2	Project Backlog	Trello is a management tool that uses boards, lists, and cards to help organize our backlog
3	Source Codes	We host our source code for the Back End, Front End on Github: ● Back End: FPT Team Matching Backend ● Grading Service: FPT Team Matching Grading Service ● Front End: FPT Team Matching Frontend
4	Database Script(s)	N/A
5	Final Report Document	A document file that synthesises the project's reports
6	Test Cases Document	A document that outlines the test cases required for validating each feature of the project
7	Slide	A powerpoint file that shows team members, actor lists, project context, technologies, functional requirements, non functional requirements, and main core flows of the project

2. Installation Guides

2.1 System Requirements

2.1.1 Web Application

Component	Minimum	Recommended
CPU	Intel® Core™ i5-10400F	Intel® Core™ i7-12700F
Memory	At least 16GB RAM	More than 32GB RAM
Storage Space	512GB SSD	1000GB SSD
Network	100 Mbps	300 Mbps
Web Browser	Chromes (v110 or higher)	Chrome latest stable version

2.2 Installation Instruction

2.2.1 Backend

Step 1: Download & Install Visual Studio Code: Visual Studio Code

Step 2: Download & Install git: Git

Step 3: Download & Install Docker: Docker

Step 4: Download & Install Newman: Newman

Step 5: Download & Install Postman: Postman

Step 6: Navigate to the folder that you want to install the project, click on address bar of your file explorer, type **cmd** and enter

Step 7: Type this command:

"**git clone https://github.com/SEP490-FPT Team Matching/FPT Team Matching-backend.git**"

"**git clone https://github.com/SEP490-FPT Team Matching/FPT Team Matching-grading-service.git**"

"**sudo apt-get install p7zip-full**"

2.2.2 Frontend

Step 1: Download & Install Visual Studio Code: Visual Studio Code

Step 2: Download & Install git: Git

Step 3: Download & Install Nodejs 20: Node.js

Step 4: Navigate to the folder that you want to install the project, click on address bar of your file explorer, type **cmd** and enter

Step 5: Type this command:

"**git clone https://github.com/SEP490-FPT Team Matching/FPT Team Matching-frontend**"

Step 6: Install modules:

- Open your **Visual Studio Code**
- Click **File** and choose **Add folder to workspace**
- Choose the project folder and click **Add**
- Click **Terminal**, choose **New Terminal** and select the project
- Type command "**npm i**". if it is not working, type one of these command: "**npm i –force**" or "**npm i --legacy-peer-deps**"

Step 7: After installing modules, type "**npm run dev**" and wait for the terminal to show "**Ready in…**"

Step 8: Ctrl + Click on **Local address** on the **terminal** to open the website on the browser

3. User Manual

3.1 Overview

User:

- **Authentication:**
    - Login
    - Logout

Admin:

- **User Management:**
    - Create new user
    - Delete user
    - Update user

Manager:

- **Student and Lecturer management:**
    - Import student
    - Import lecturer
    - Add student
    - Add lecturer

- **Updating topic management:**
    - Evaluate updating topic

- **Capstone management**
    - Export template
    - Import capstone schedule
    - Update result

- **Review management**
    - Export template
    - Import review schedule

Lecturer(Mentor):

- **Idea and topic management:**
    - Evaluate the student's idea.
    - Create idea
    - Send idea to council
    - Resubmit topic
- **Student management:**
    - Give conclusion evaluation

- **Updating topic management:**
    - Evaluate updating topic

Lecturer(Council):

- **Idea and topic management:**
    - Evaluate mentor’s topic

Lecturer(Reviewer):

- **Review schedule::**
    - View review schedule

Student:

- **Idea proposal:**
    - Create ideas.

    - Cancel idea.
    - Resubmit idea.
- **Project management:**
    - View review schedule.
    - Submit file checklist.
    - Update topic (after review 1 and review 2)
    - View defense schedule
    - View defense result
    - Leader rate members
    - View leader rating

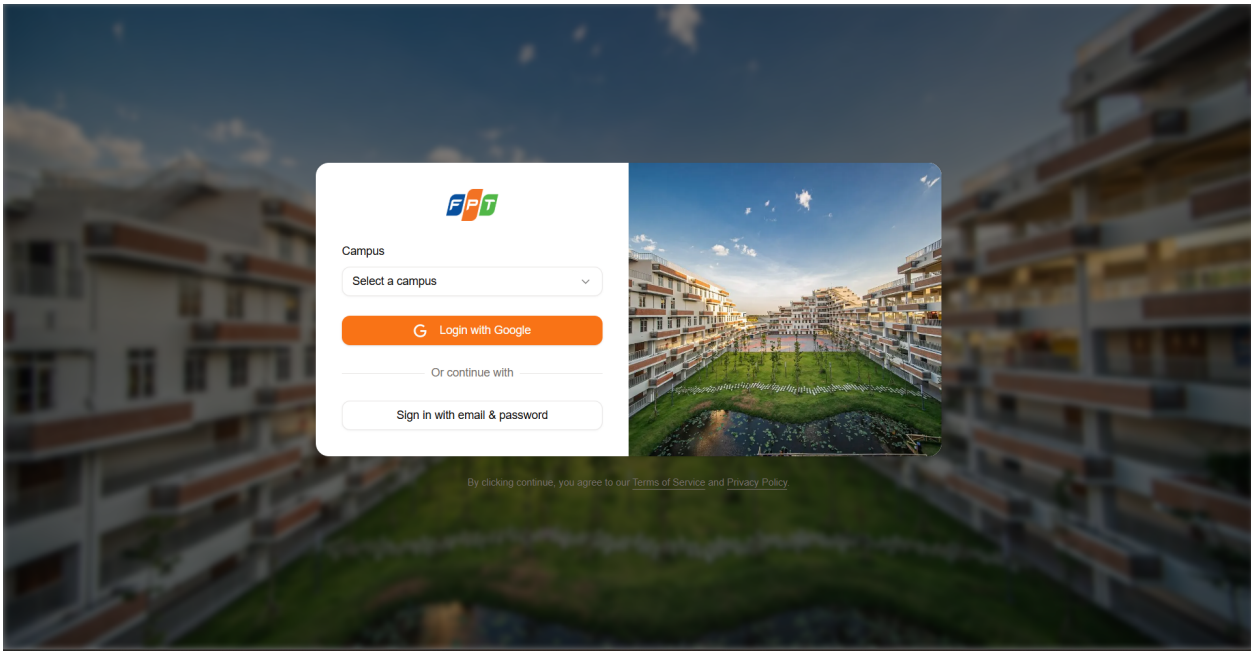
- **Blog management::**
    - Create a blog.
    - Update blog.
    - Delete blog

- **Team management::**
    - Create a team.
    - Invite member
    - Respond invitation
    - Request to join team

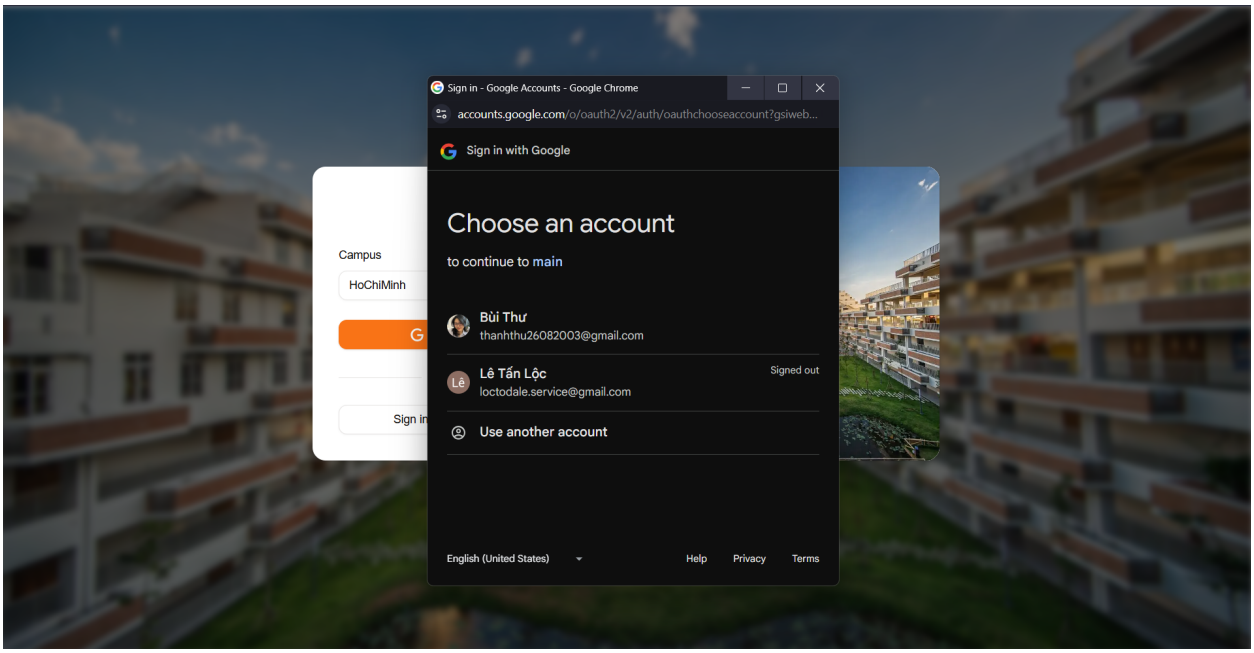
3.2 Guideline

3.2.1 Login

- Step 1: Choose Campus and select **Login with Google** button on Login Page

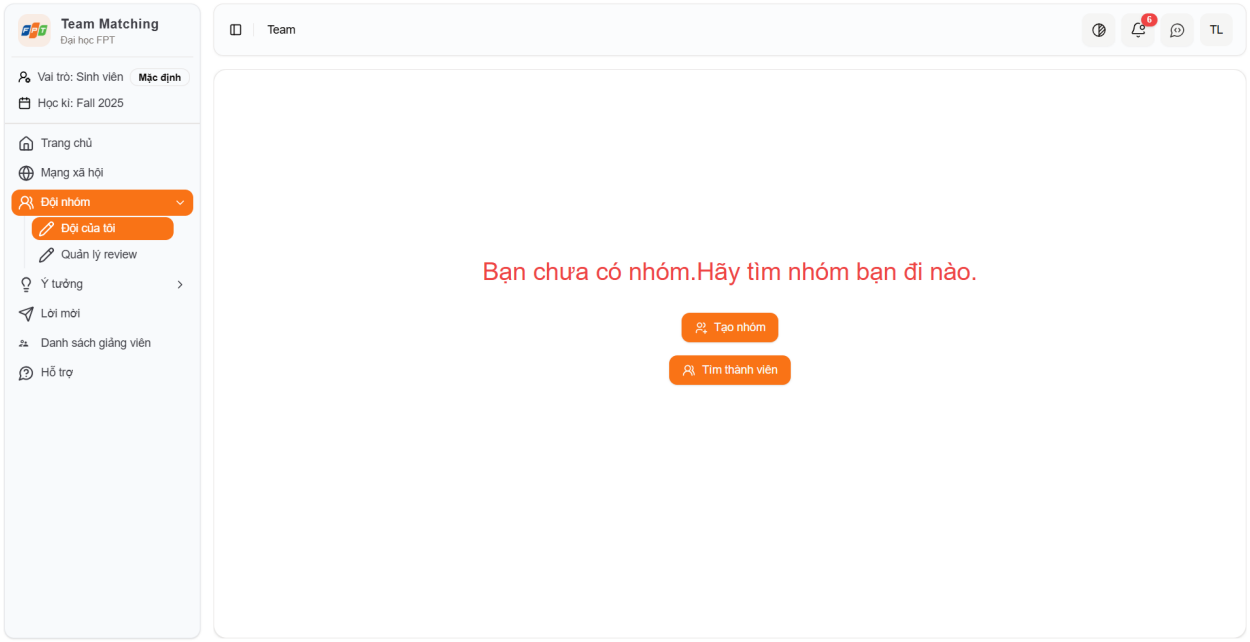


- Step 2: Select Google Account to login

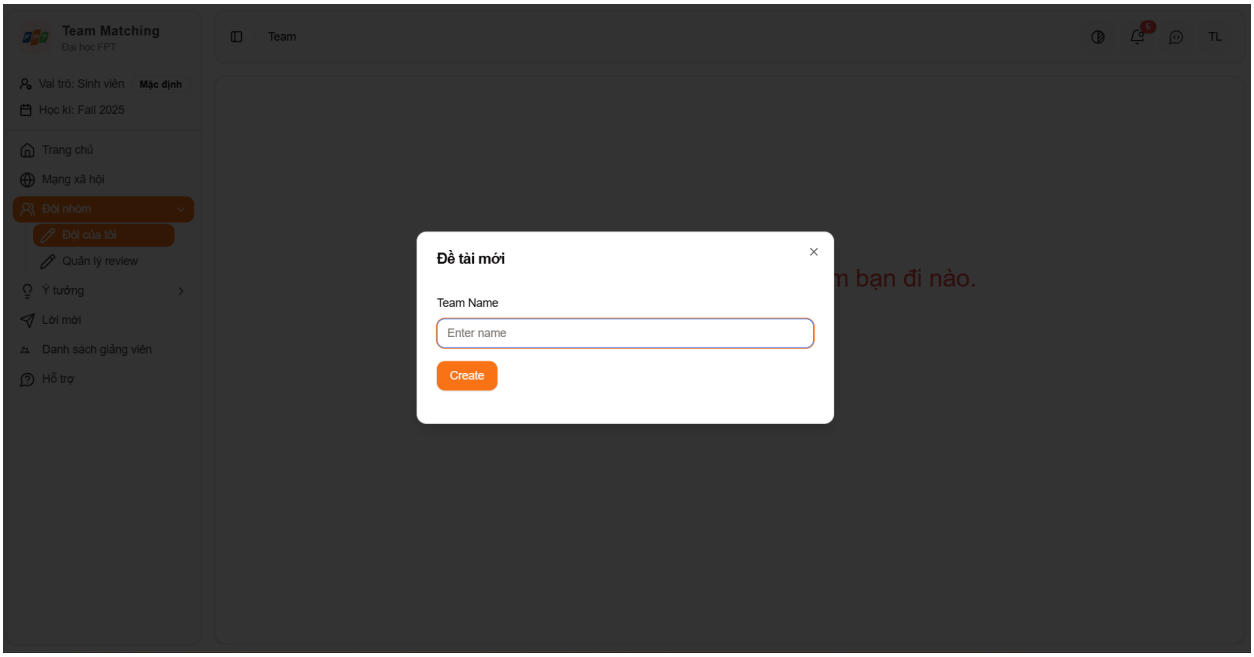


3.2.2 Create team

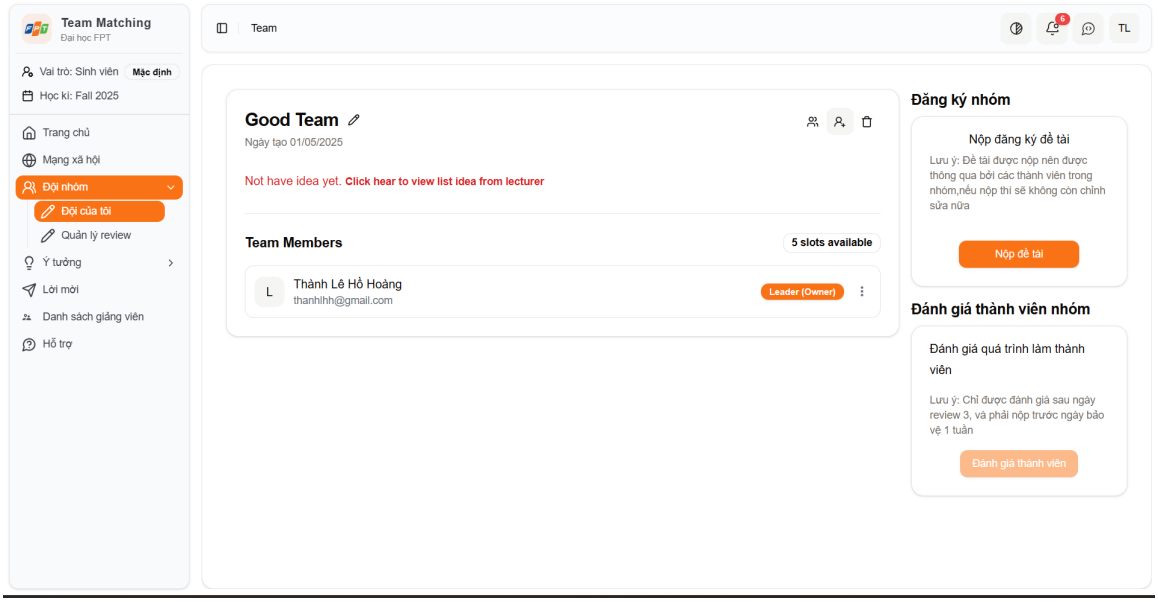
- Step 1: Click on button **Đội nhóm** -> **Đội của tôi**



- Step 2: Click on button **Tạo nhóm**

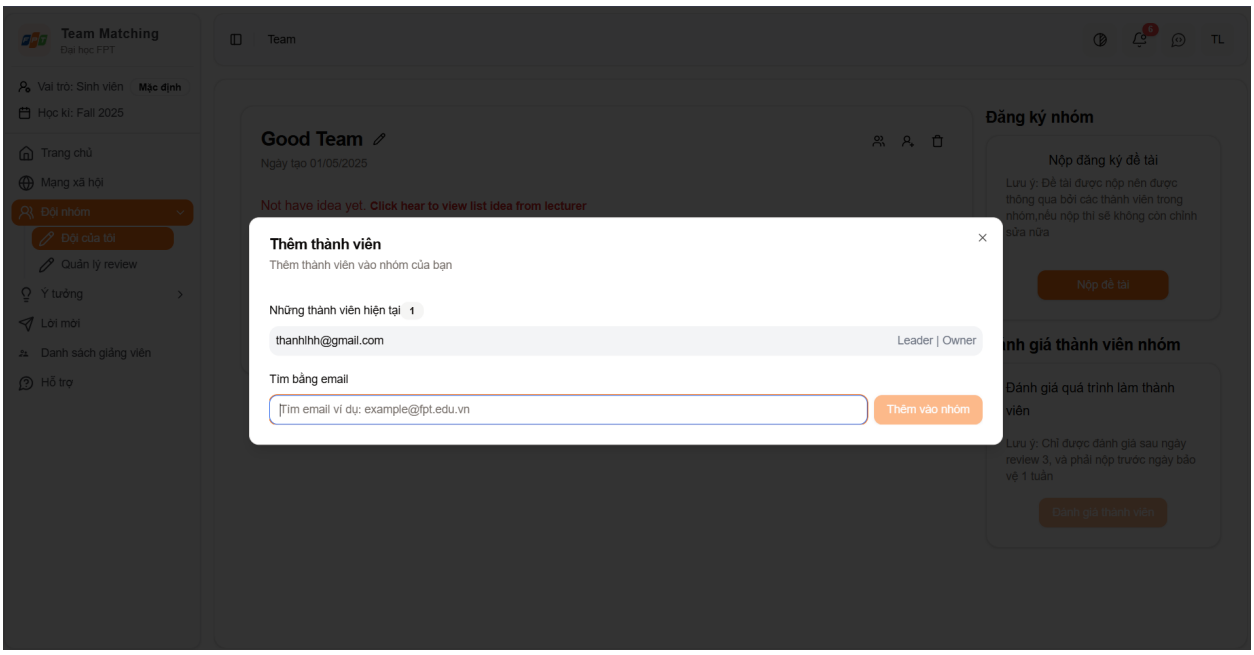


- Step 3: Fill Tên nhóm and click Create

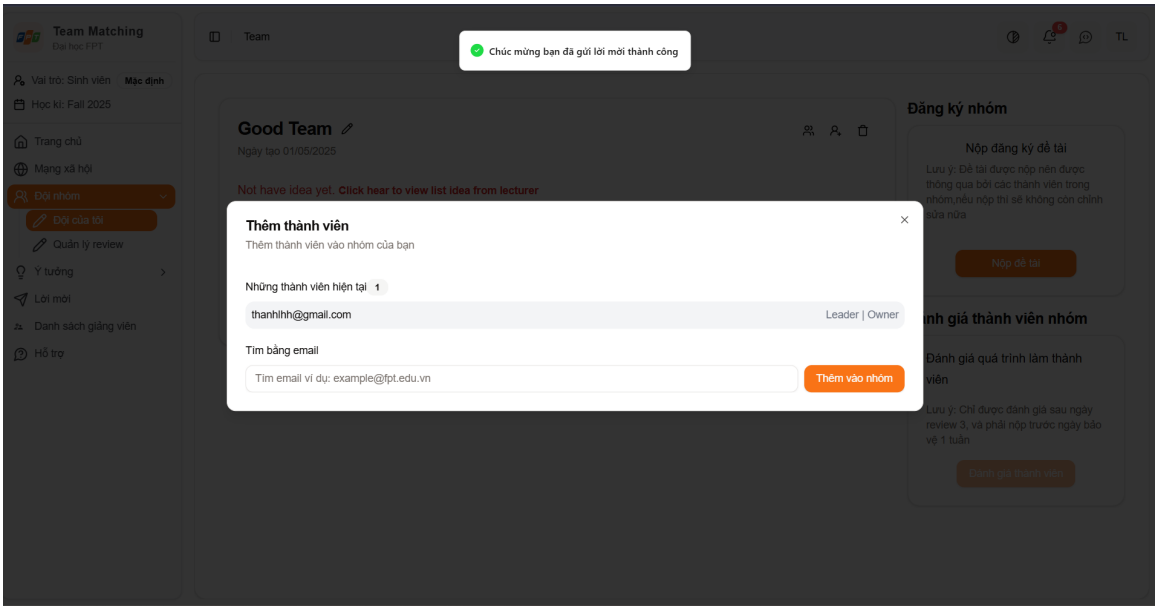


3.2.3 Invite member

- Step 1: Click on button **student icon with plus** on the navigation bar

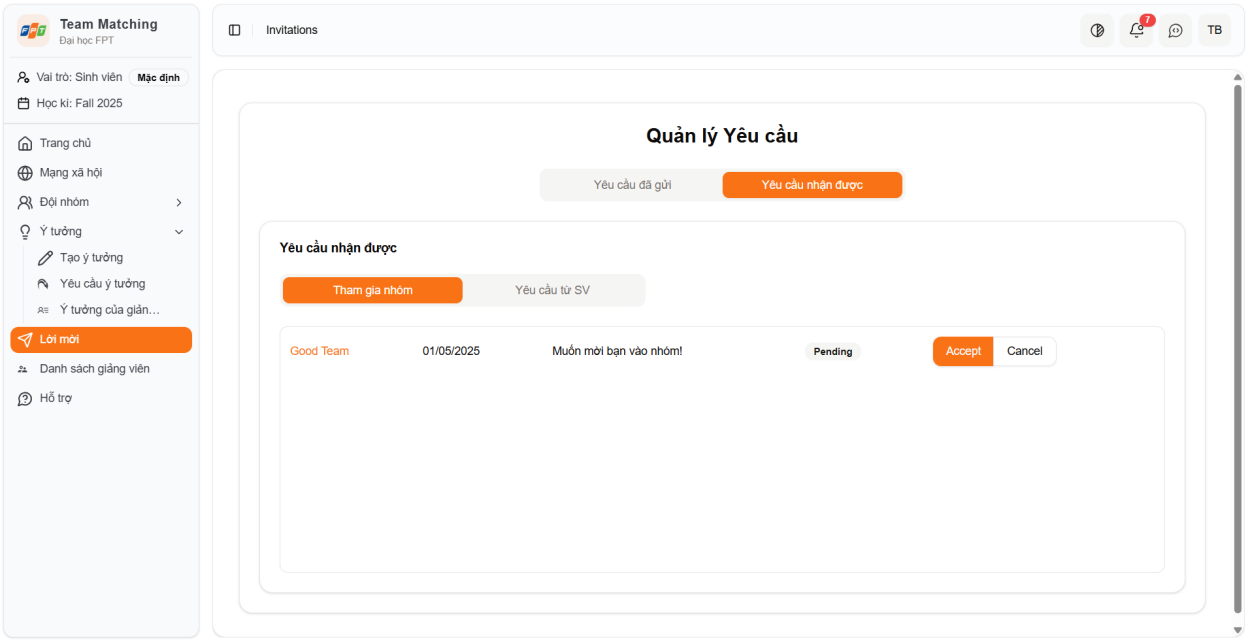


- Step 2: Fill email and click on student **Thêm vào nhóm** on the navigation bar



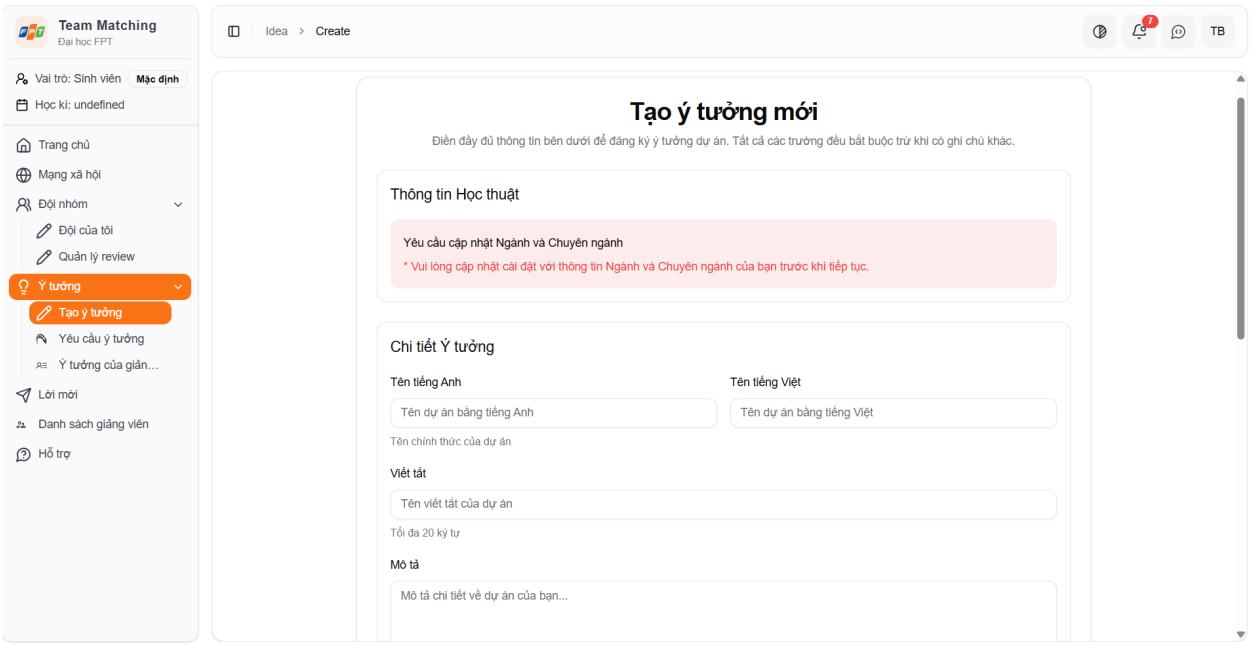
3.2.4 Student respond invitation

- Step 1: Click on button **Lời mời** on the navigation bar **-> Yêu cầu nhận được**

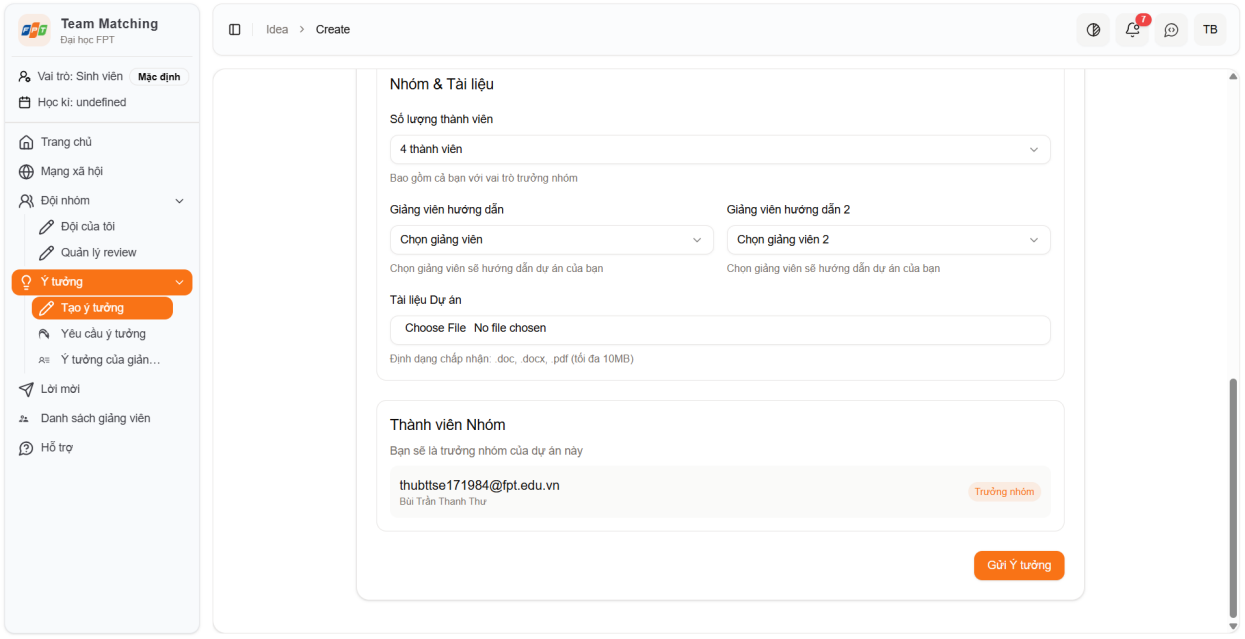


3.2.5 Student create idea

- Step 1: Click on button **Tạo ý tưởng** on the navigation bar

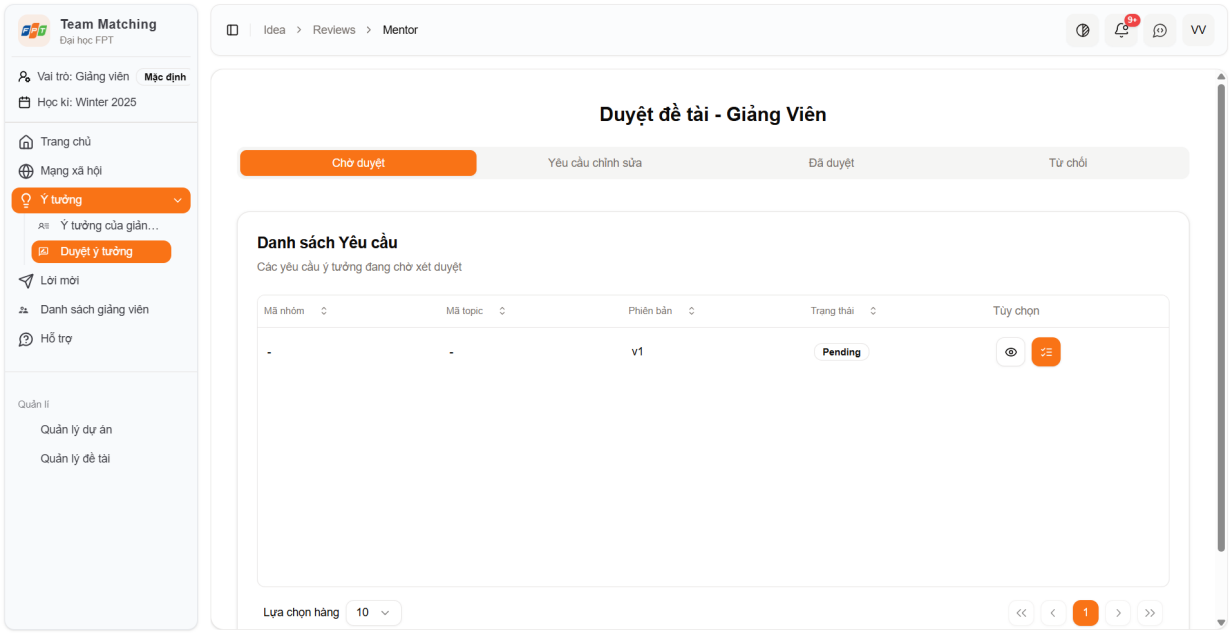


- Step 2: Fill information to registration form modal and click **Gửi ý tưởng** button

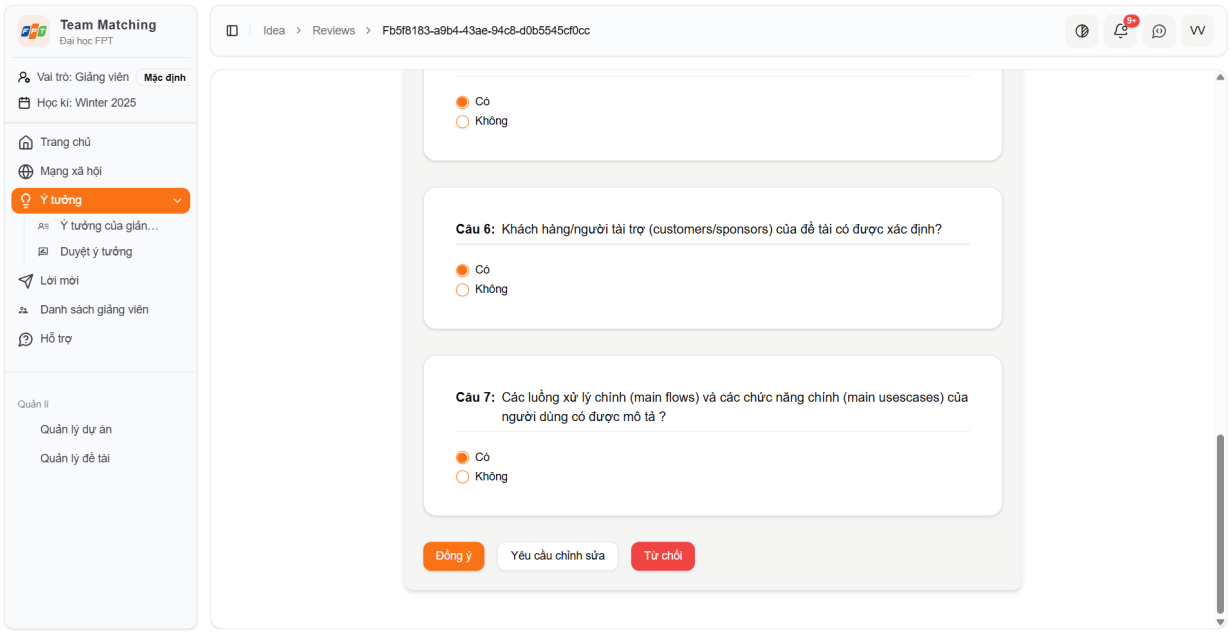


3.2.6 Lecturer evaluate student idea

- Step 1: Click on button **Duyệt ý tưởng**

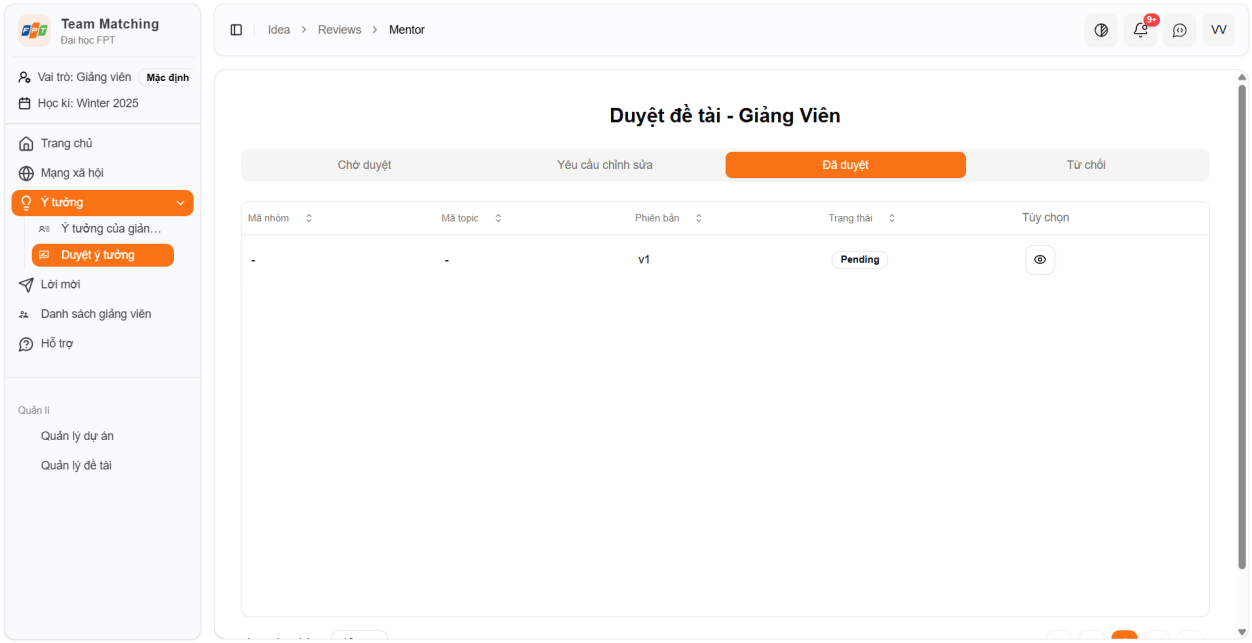


- Step 2: Click on button at the end of record -> fill form and choose final button

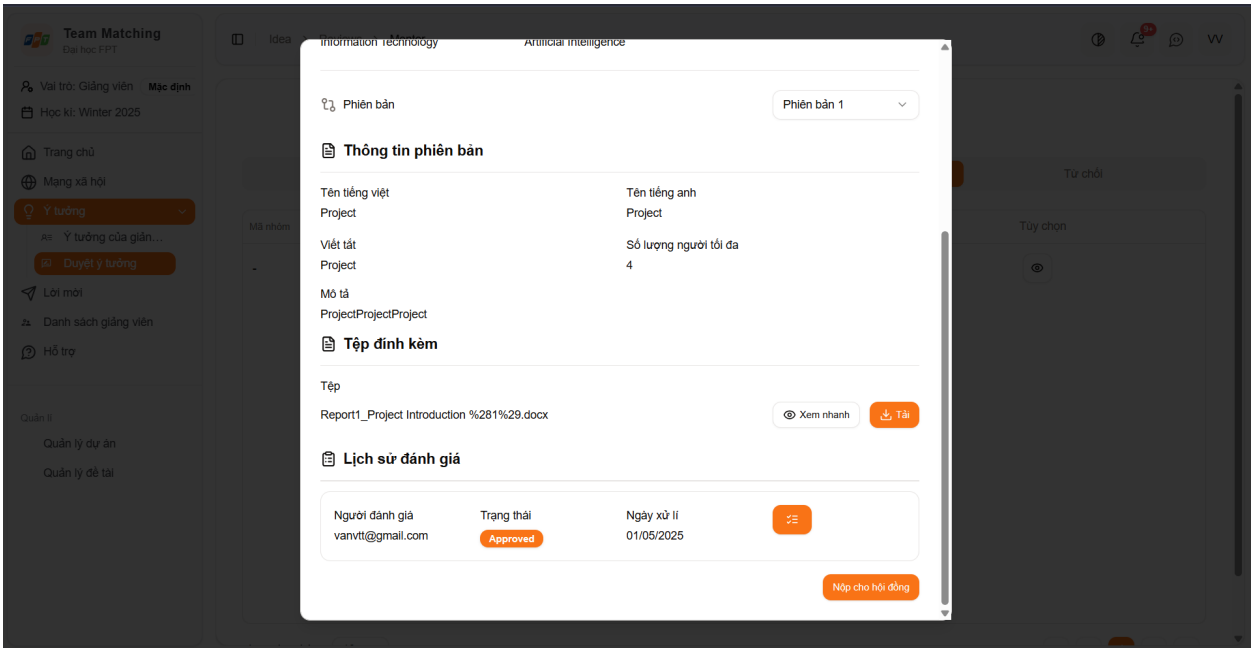


3.2.7 Lecturer submit topic to council

- Step 1: Select **Đã duyệt -> View idea**

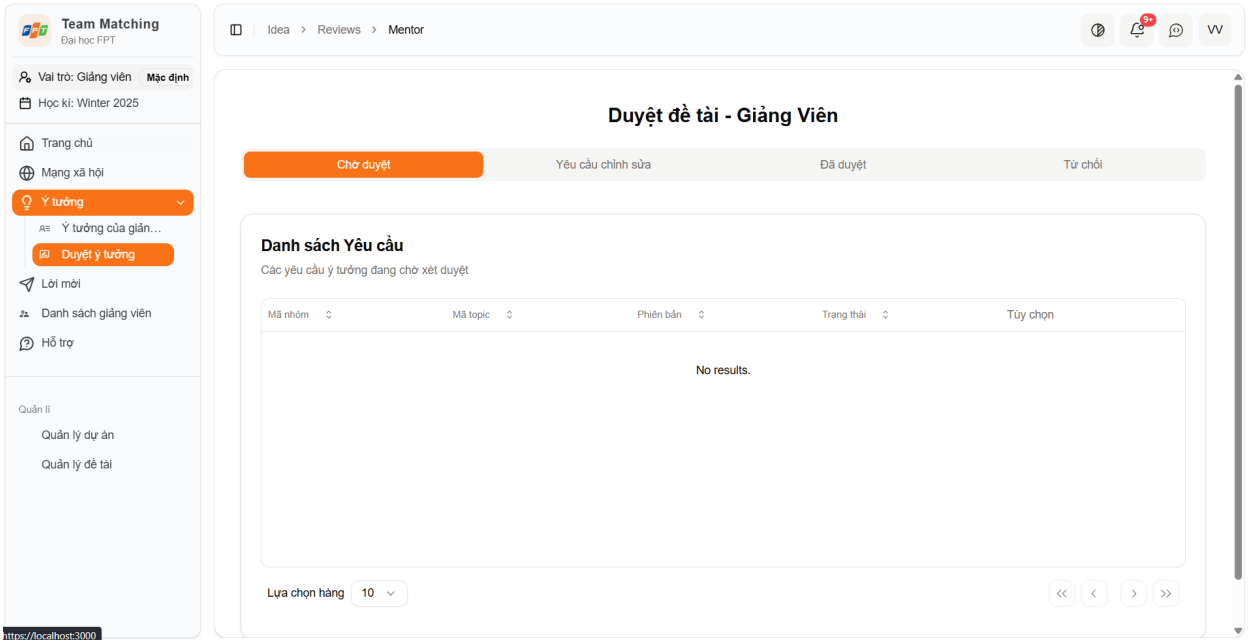


- Step 2: Select **Nộp cho hội đồng**

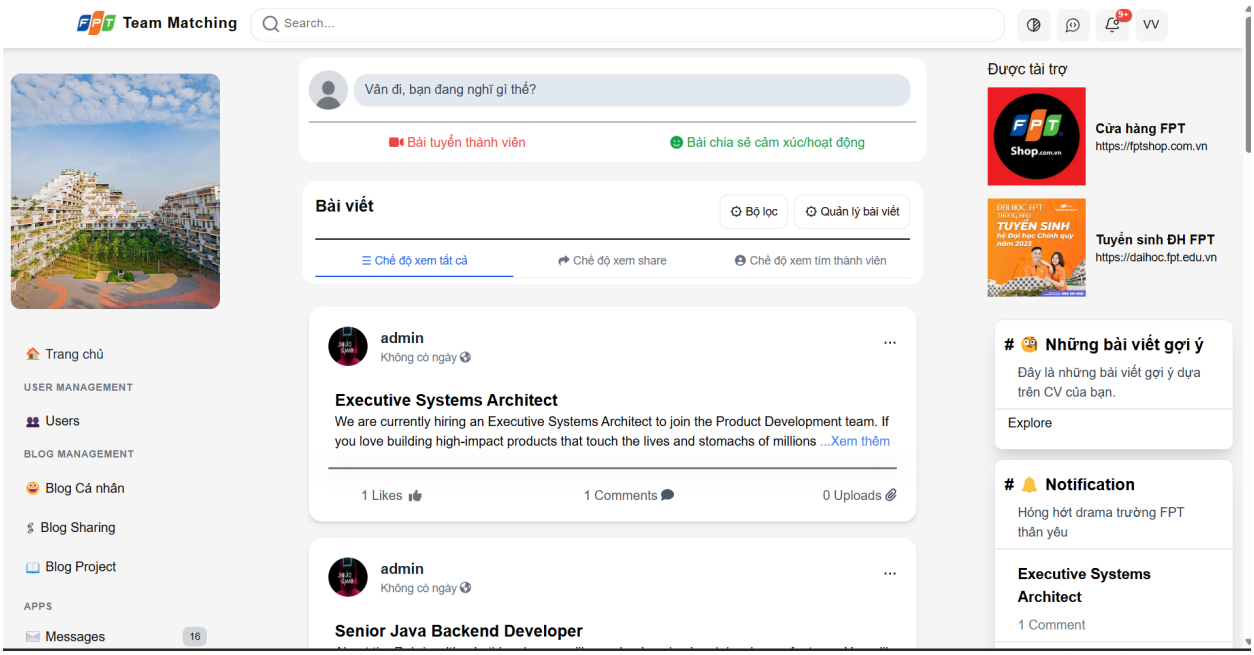


3.2.8 Student create blog

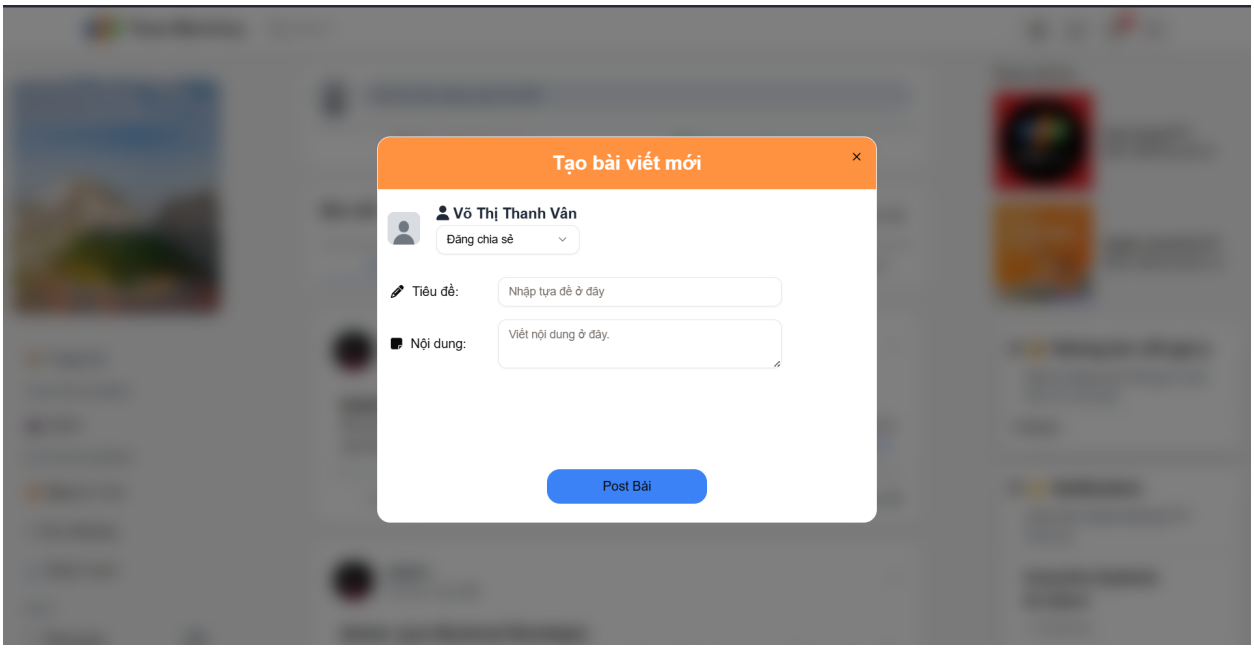
- Step 1: Click **Mạng xã hội** on Navigation bar



- Step 2: Click on **Bạn đang nghĩ gì thế** button on Gherkin-Postman Page

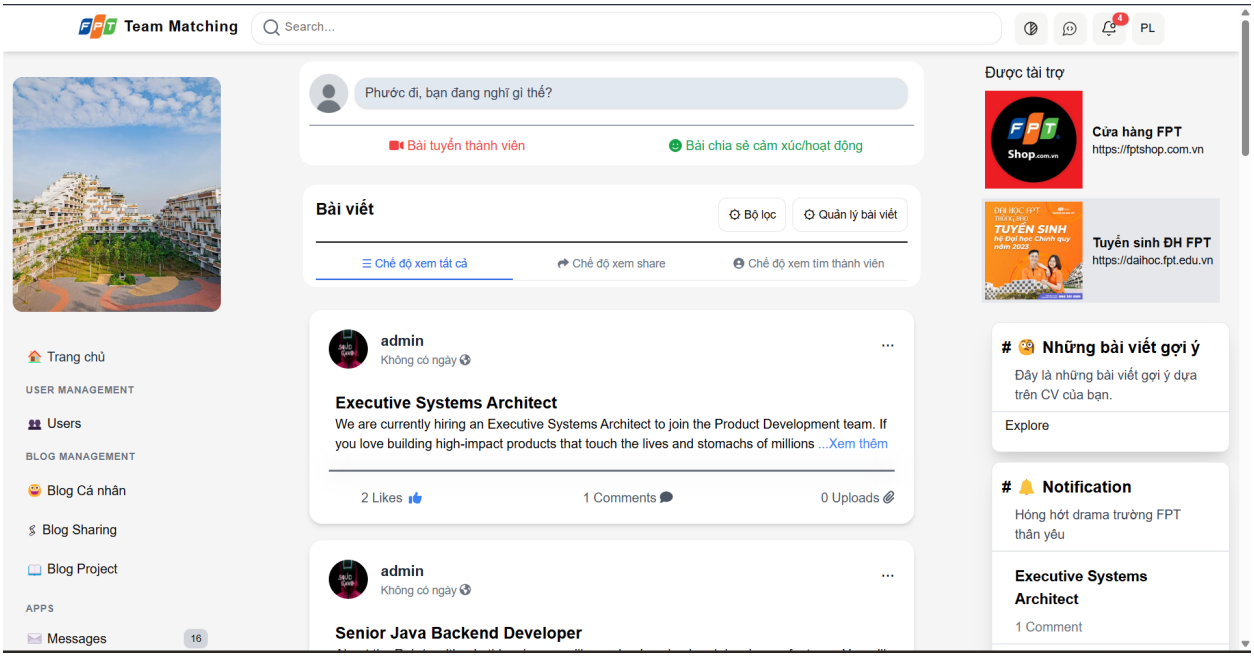


- Step 3: Click on **Bạn đang nghĩ gì thế**



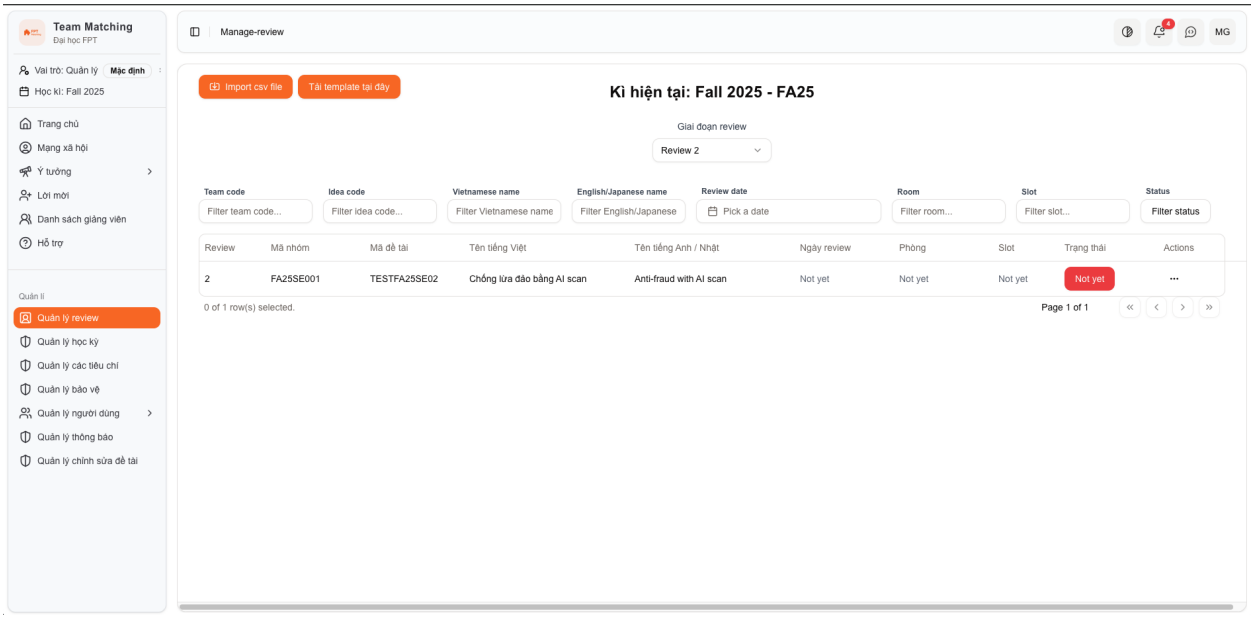
3.2.9 Student apply CV

- Step 1: Select **Uploads** on blog

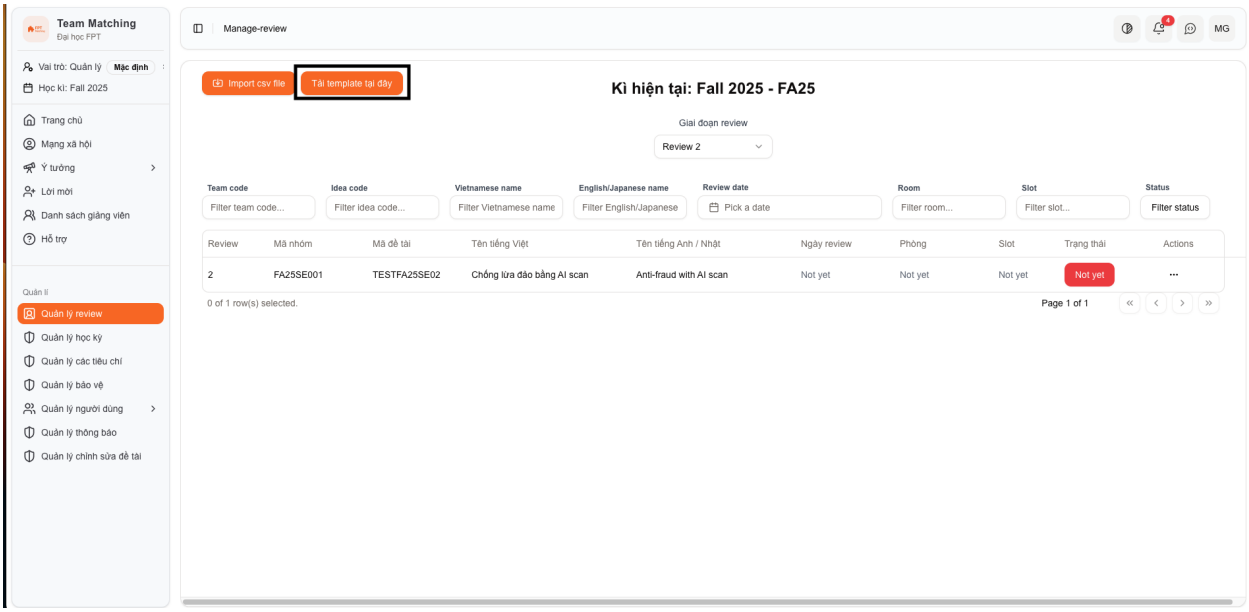


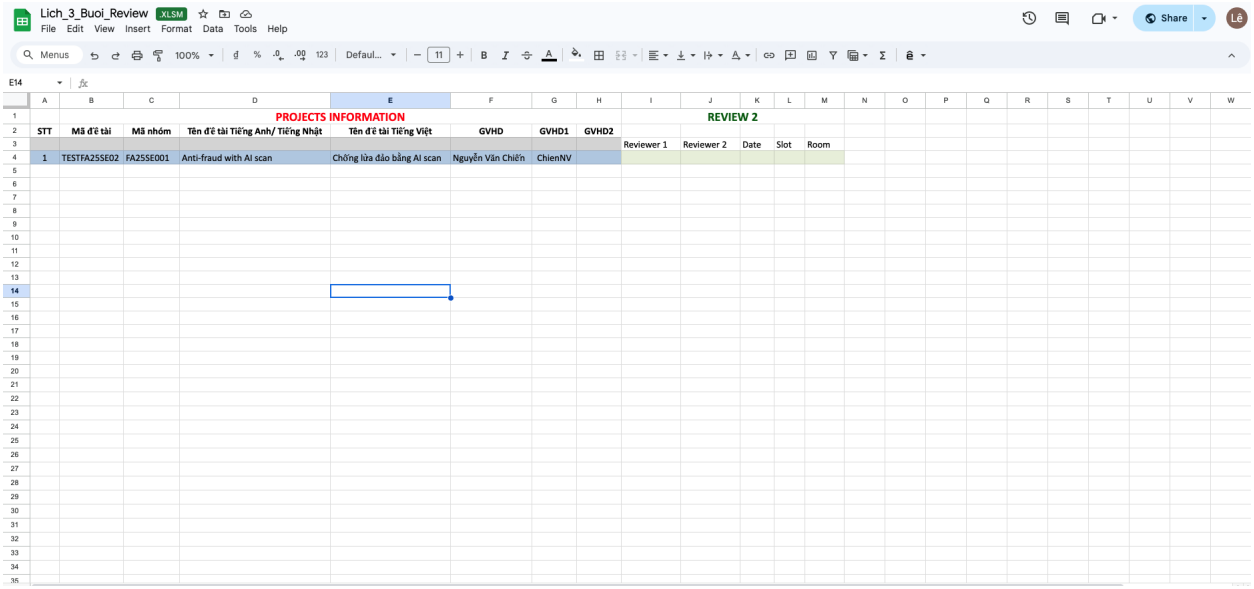
3.2.10 Export template review schedule

- Step 1: Select the “Quản lí review” tab in the sidebar to access the review schedule management page.



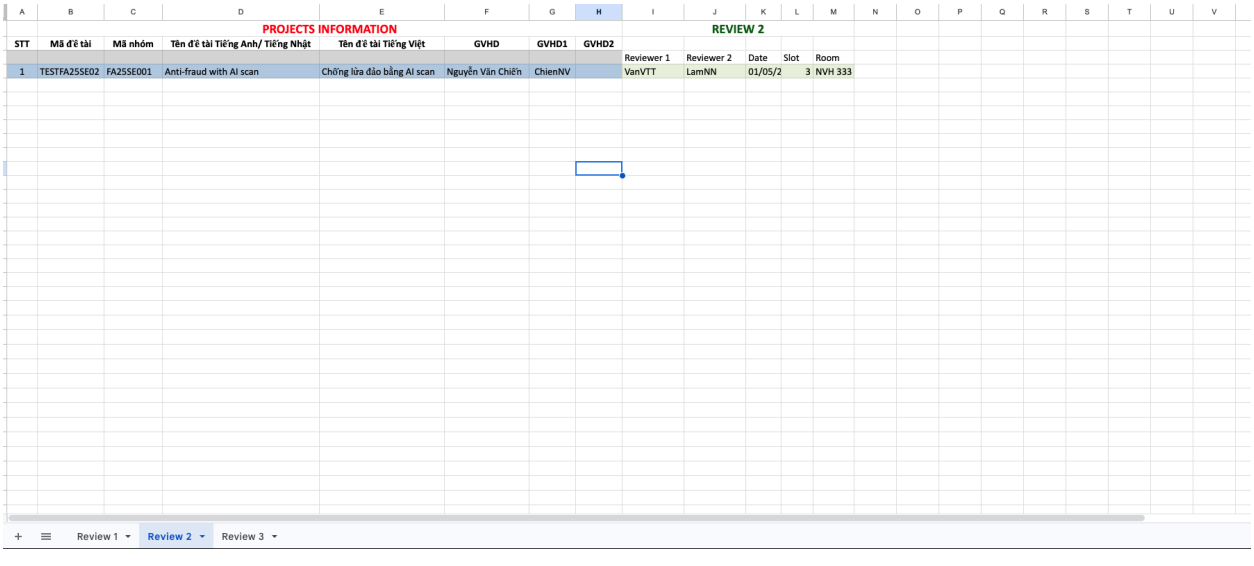
- Step 2: Select “Tải template tại đây” button for download template review schedule.



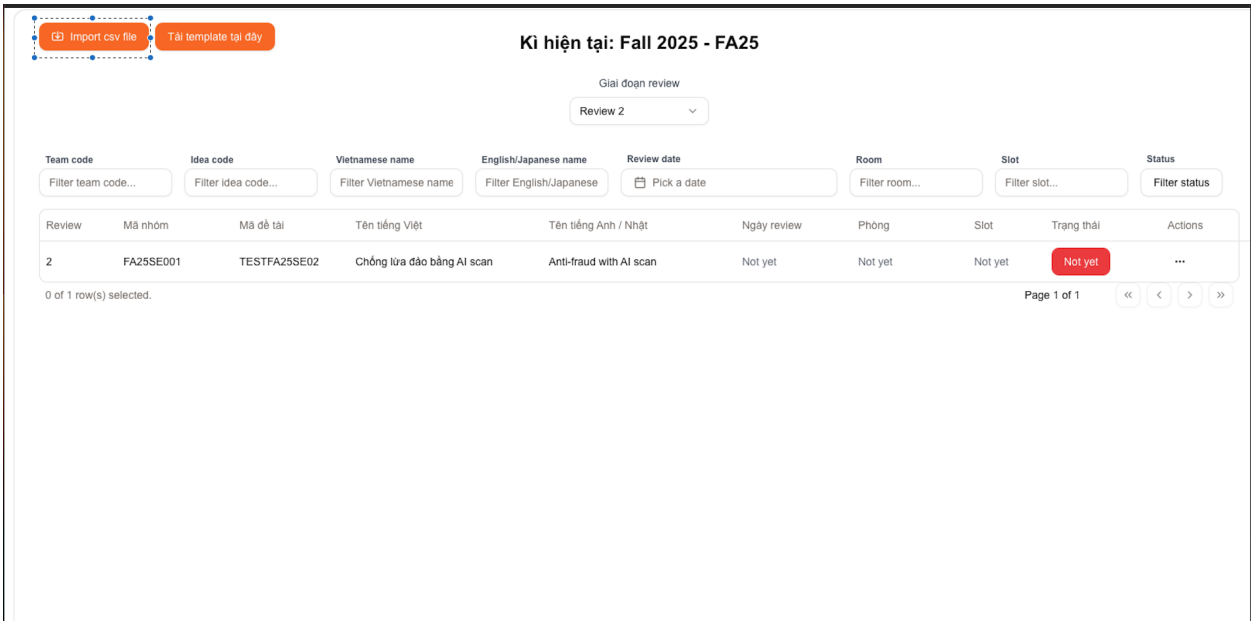


3.2.11 Import review schedule

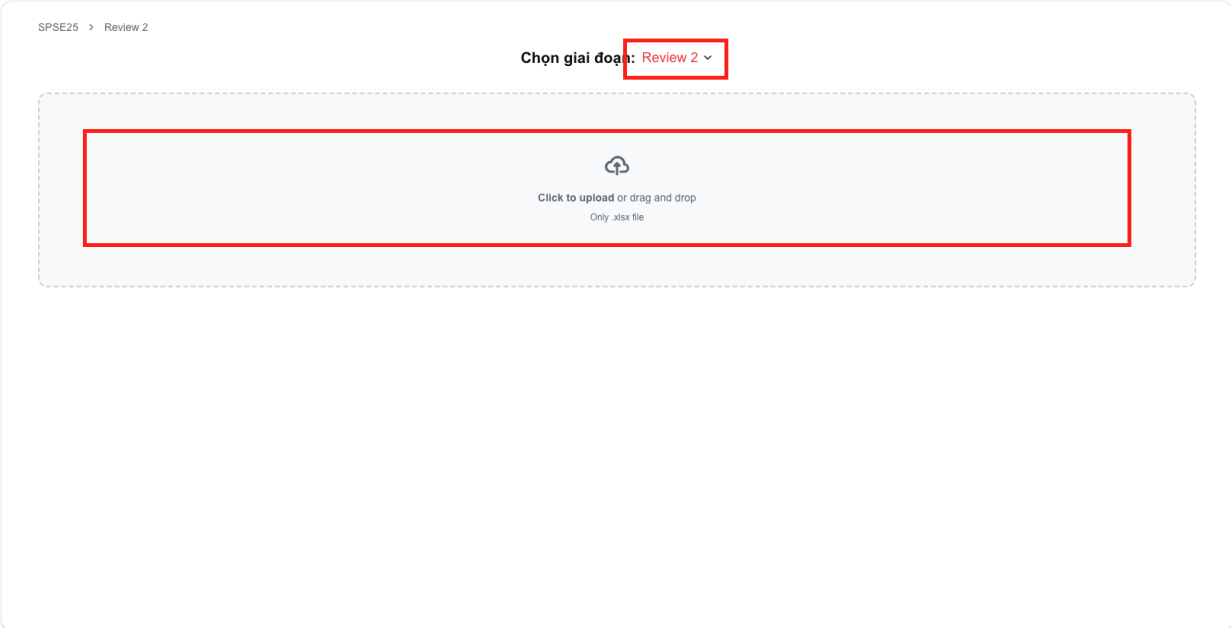
- Step 1: Fill all fields in the file template. Sure that all data is valid.



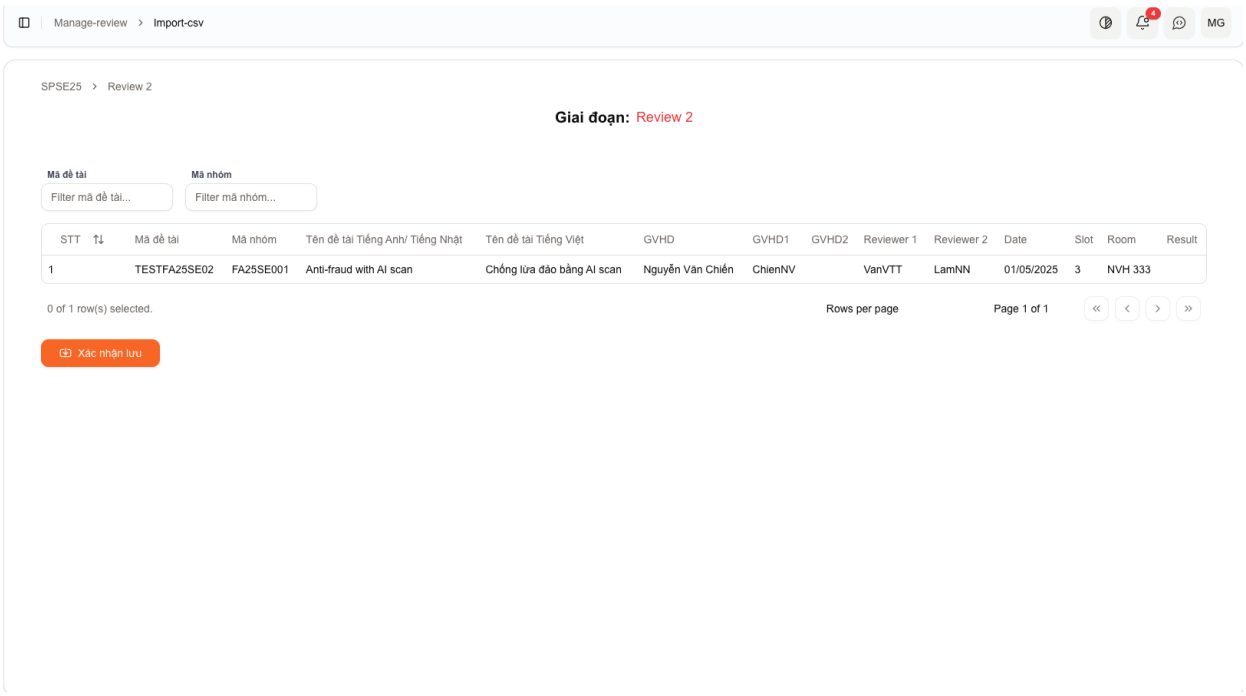
- Step 2: Select “Import csv file” button.



- Step 3: Select review stage and click “Click to upload file button” .

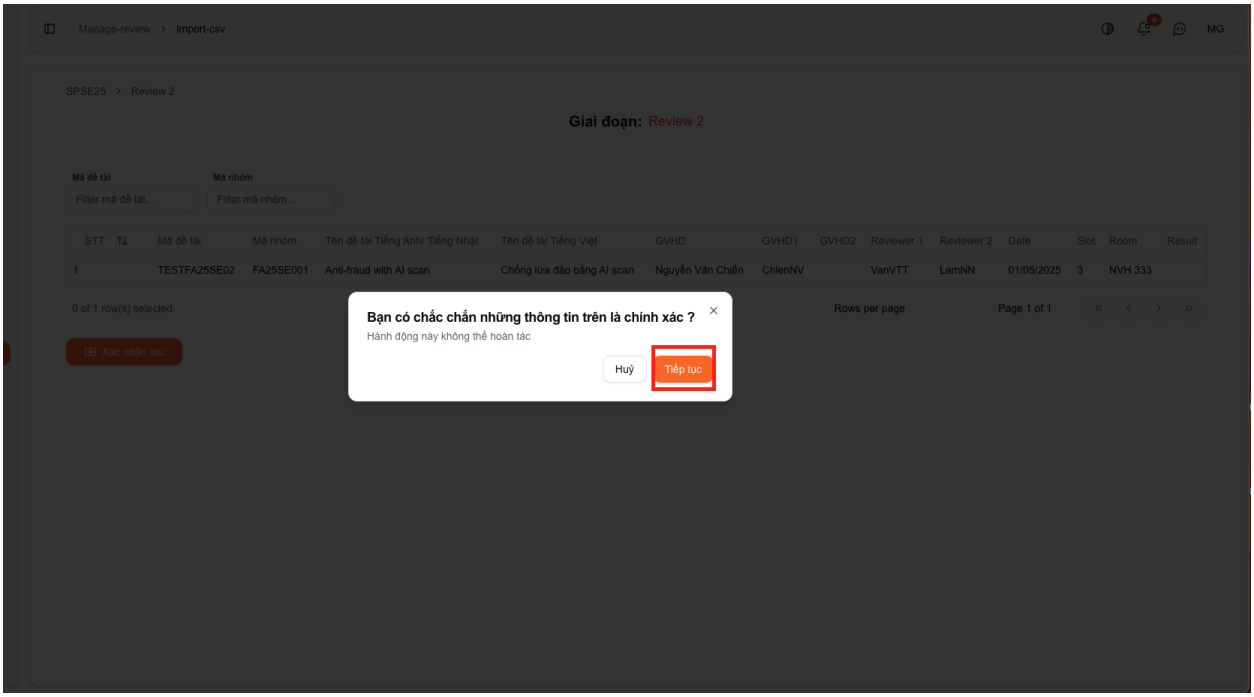


- Select the file that you have information about the review schedule. It will show data inside the file.

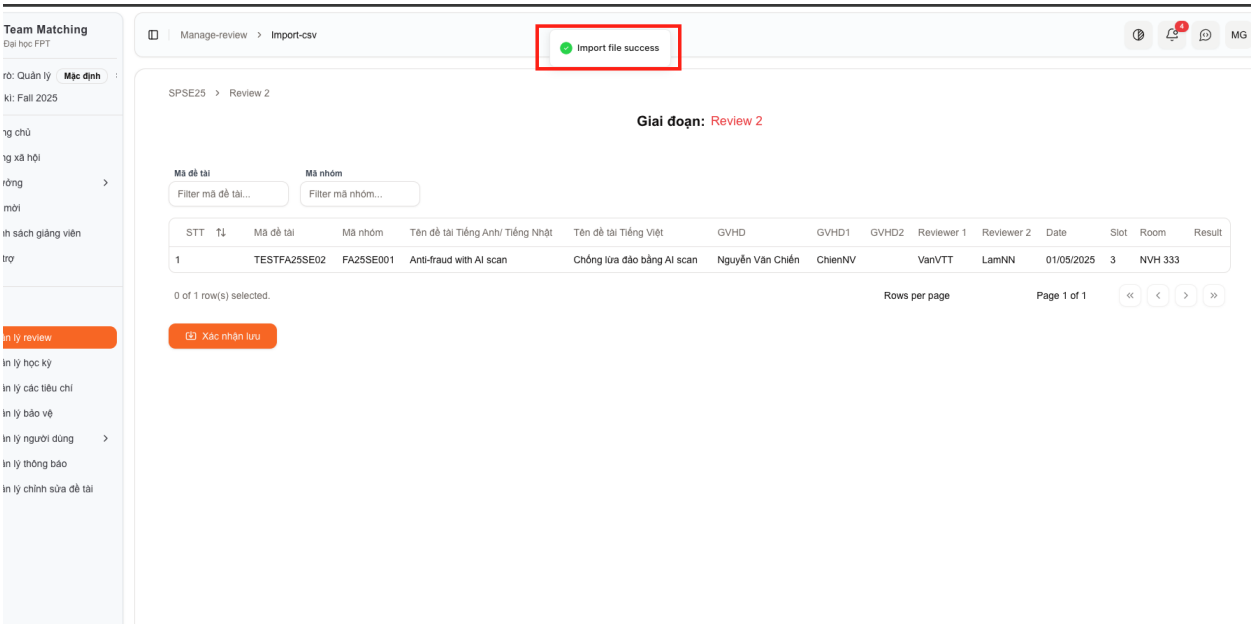


- Select the “Xác nhận lưu” button and confirm from the dialog it will update the review schedule.


SPSE25 › Review 2
Giai đoạn: Review 2
Mã đề tài
Filter mã đề tài...
Mã nhóm
Filter mã nhóm...
STT
Mã đề tài
Mã nhóm
Tên đề tài Tiếng Anh/ Tiếng Nhật
Tên đề tài Tiếng Việt
GVHD
GVHD1
GVHD2
Reviewer 1
Reviewer 2
Date
Slot
Room
Result
1
TESTFA25SE02
FA25SE001
Anti-fraud with AI scan
Chống lừa đảo bằng AI scan
Nguyễn Văn Chiến
ChienNV
VanVTT
LamNN
01/05/2025
3
NVH 333
0 of 1 row(s) selected.
Rows per page
Page 1 of 1
Xác nhận lưu

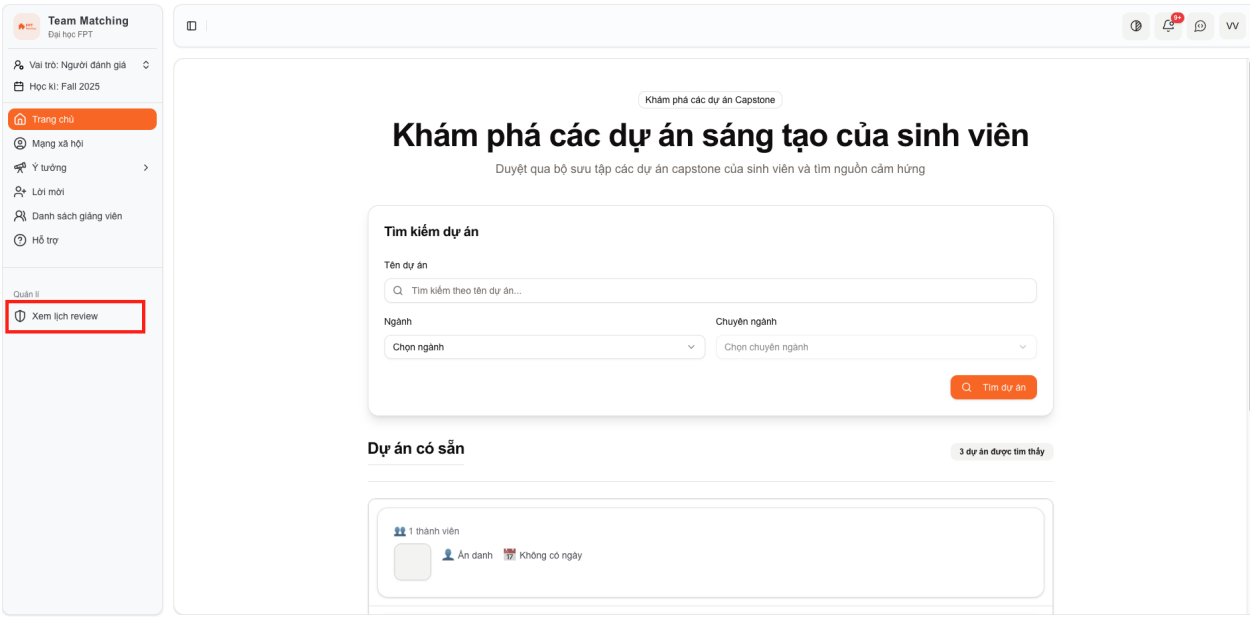

Manage-review › Import-csv
MG
SPSE25 › Review 2
Giai đoạn: Review 2
Mã đề tài
Filter mã đề tài...
Mã nhóm
Filter mã nhóm...
STT
Mã đề tài
Mã nhóm
Tên đề tài Tiếng Anh/ Tiếng Nhật
Tên đề tài Tiếng Việt
GVHD
GVHD1
GVHD2
Reviewer 1
Reviewer 2
Date
Slot
Room
Result
1
TESTFA25SE02
FA25SE001
Anti-fraud with AI scan
Chống lừa đảo bằng AI scan
Nguyễn Văn Chiến
ChienNV
VanVTT
LamNN
01/05/2025
3
NVH 333
0 of 1 row(s) selected.
Rows per page
Page 1 of 1
Xác nhận lưu
Bạn có chắc chắn những thông tin trên là chính xác ?
Hành động này không thể hoàn tác
Huỷ
Tiếp tục

- If it imports success it will show a success toast message.

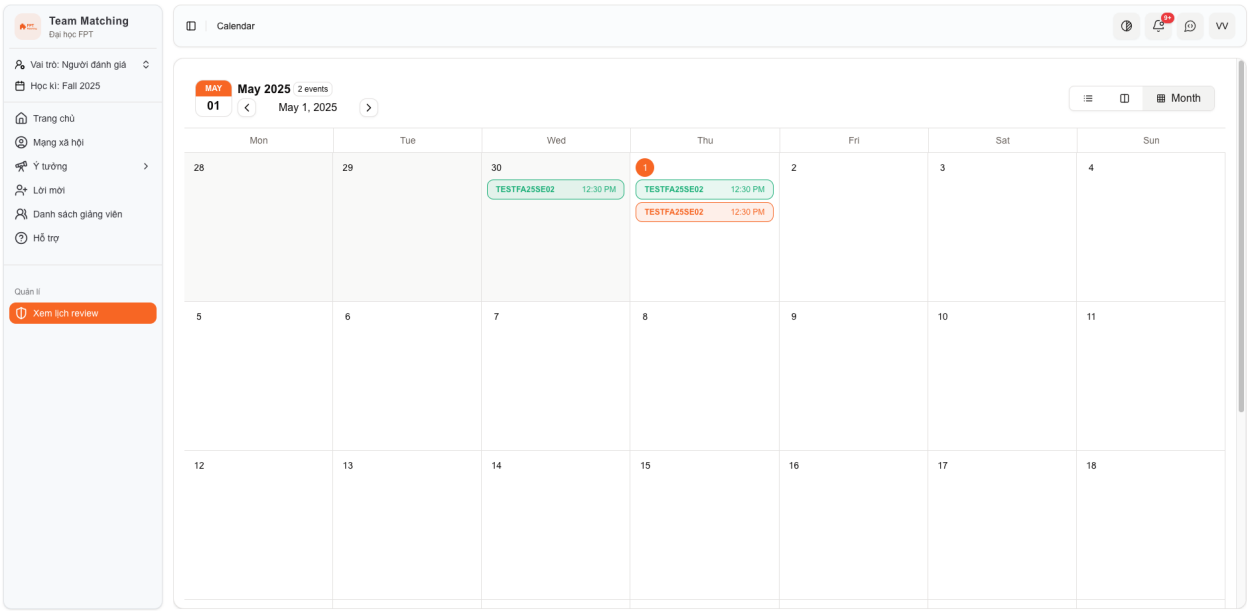


3.2.12 Lecturer (Reviewer) view review schedule

- Step 1: Click on the “Xem lịch review” tab in the sidebar.



- It will show the weekly timetable with event is the review schedule.

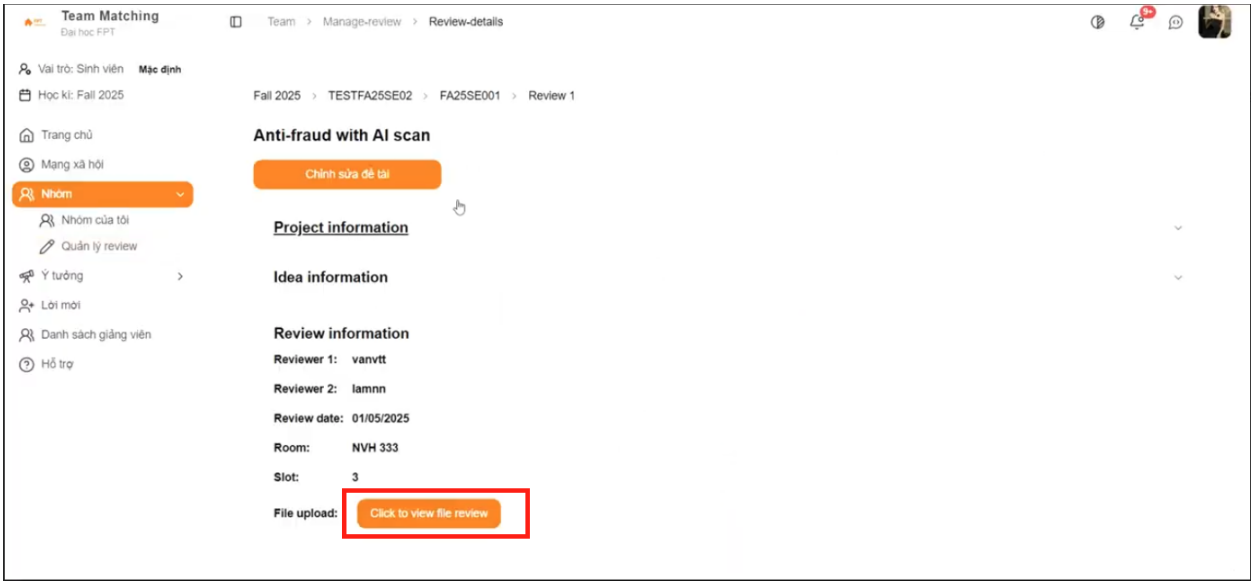


3.2.13 Student submit file checklist

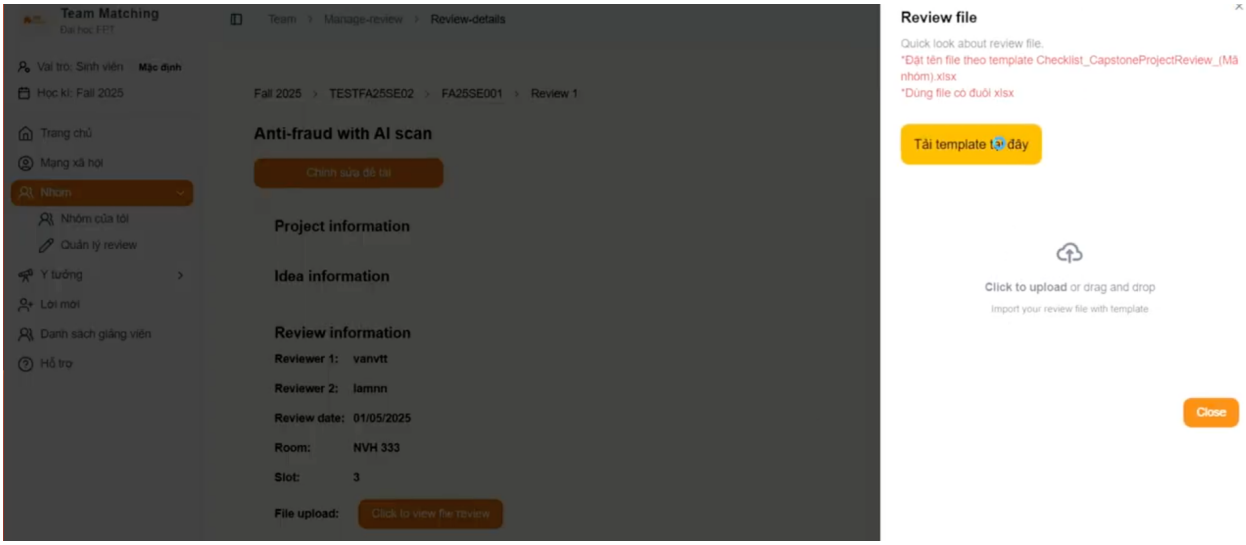
- Step 1: Select the “Quản lí review” button to show all reviews and select the review that you want to add to the checklist file.

- Step 2: Click “View detail” button

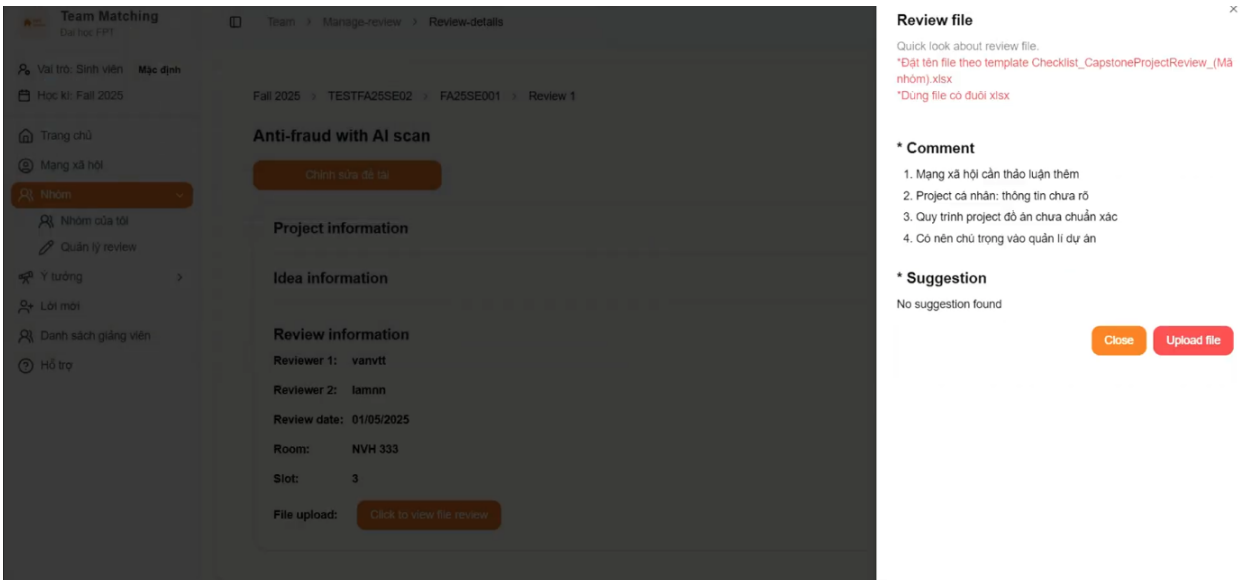
- Step 3: Click button “Click to view detail file” to show ui for upload checklist



- Select “Tải template tại đây” button to download template file.



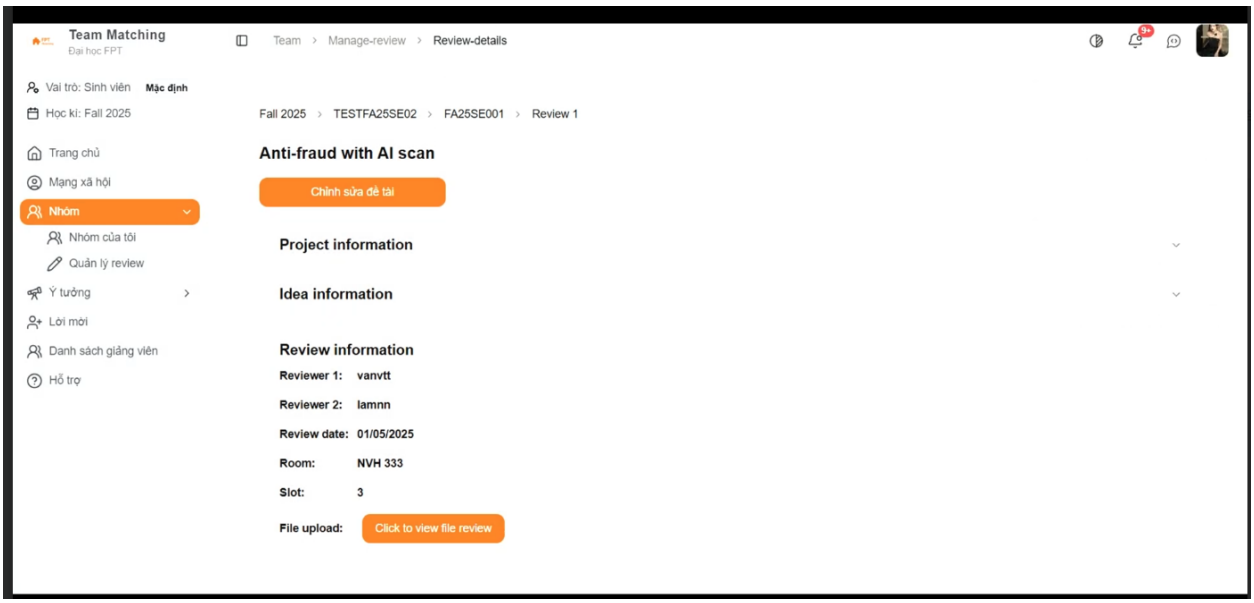
- After uploading the file, it shows content inside the checklist file.



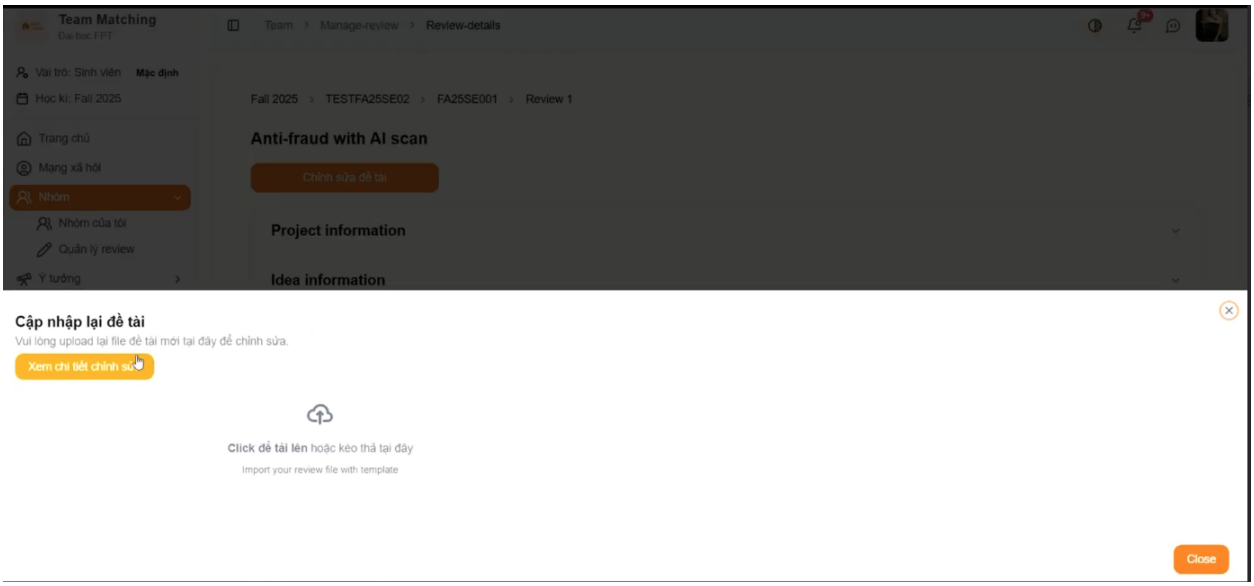
- Click “Upload file” button to save

3.2.14 Student send request update topic

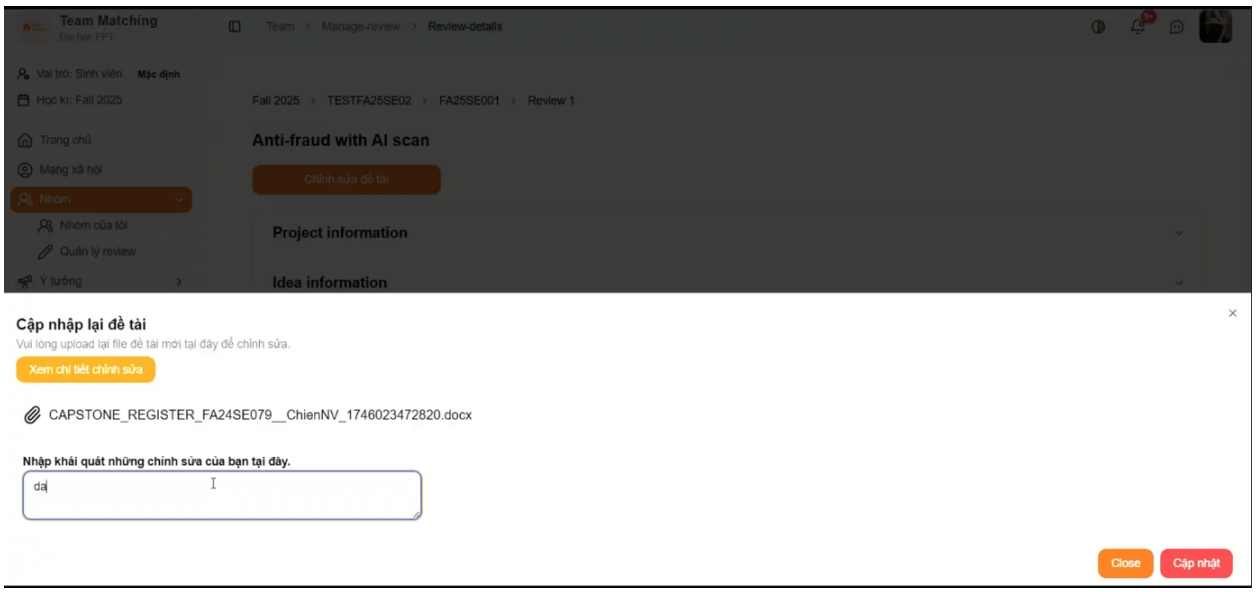
- Step 1: Select “Chỉnh sửa đề tài” button.



- Step 2: Upload file update topic.

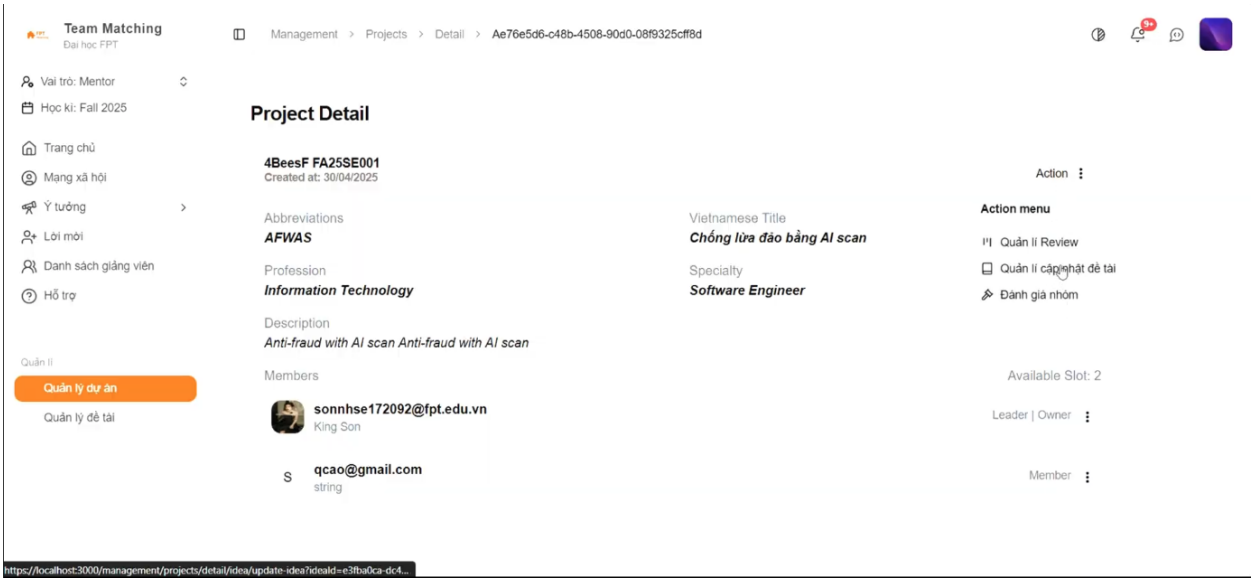


- Step 3: Fill the surmise update information and click “Cập nhật” button to send request update topic.

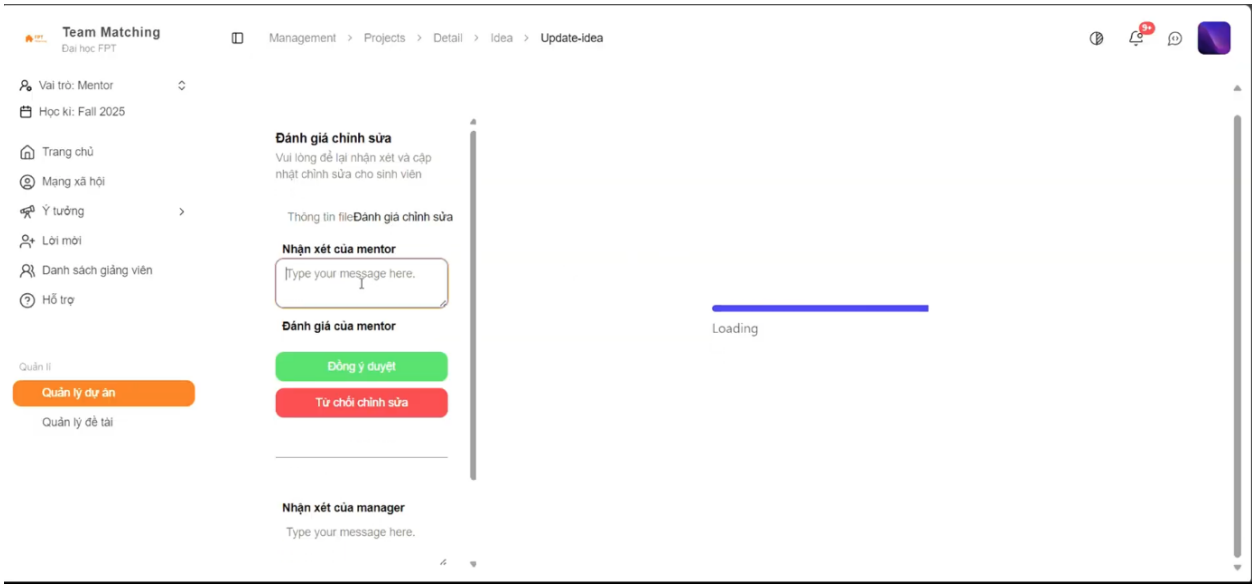


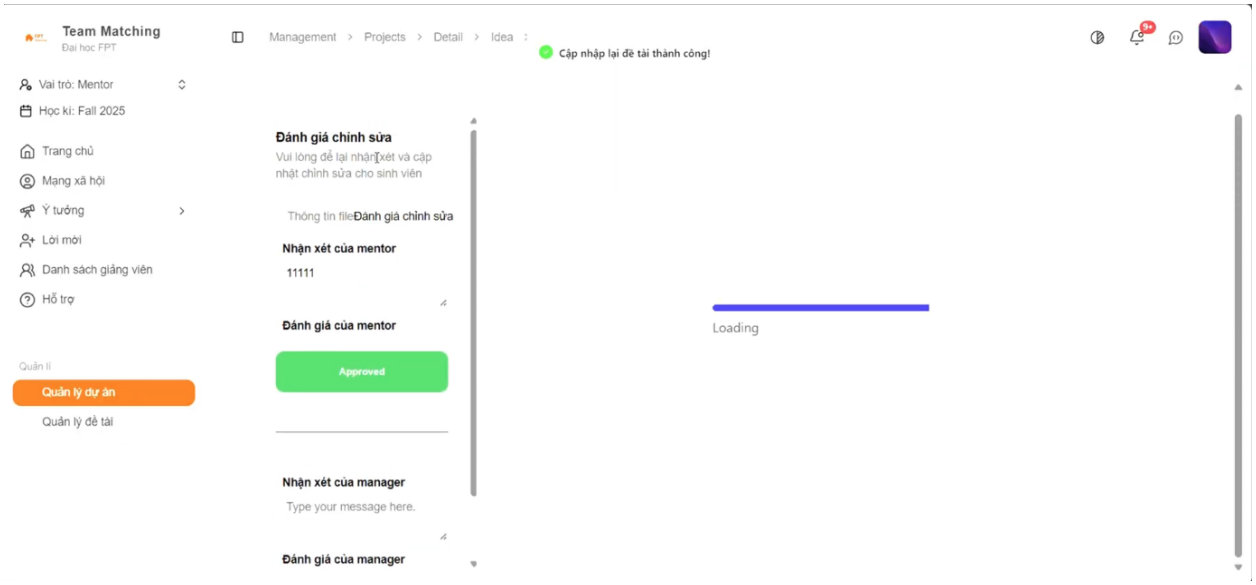
3.2.15 Mentor respond updating topic request

- Step 1: Access to view project detail, select the action button and select option “Quản lí cập nhật đề tài”



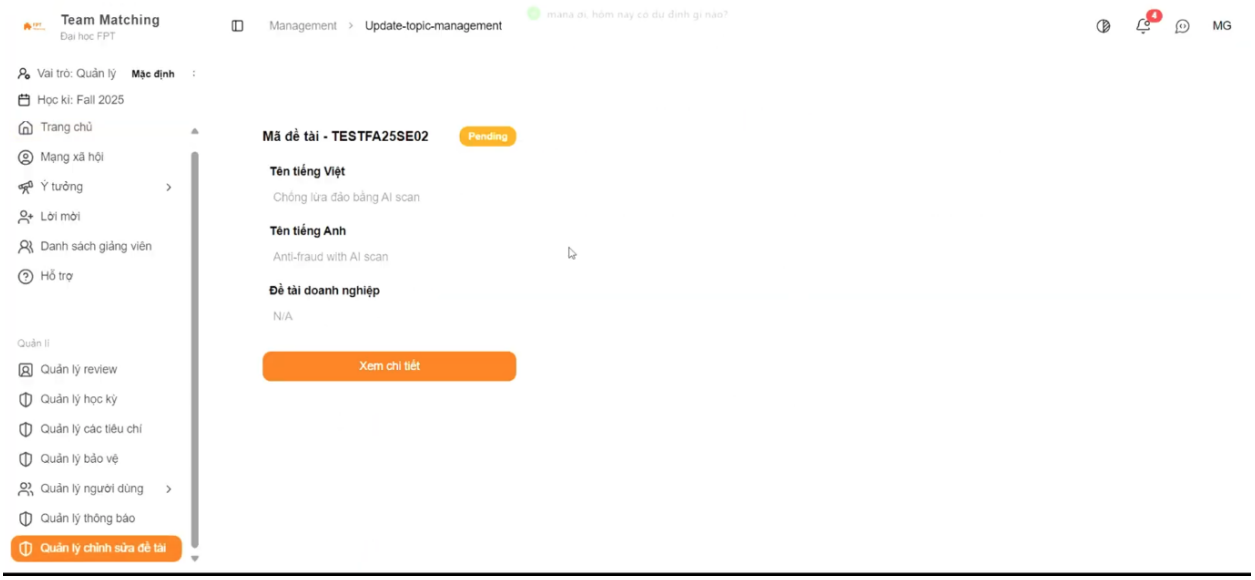
- Step 2: Fill the feedback about the update topic and select give conclusion.



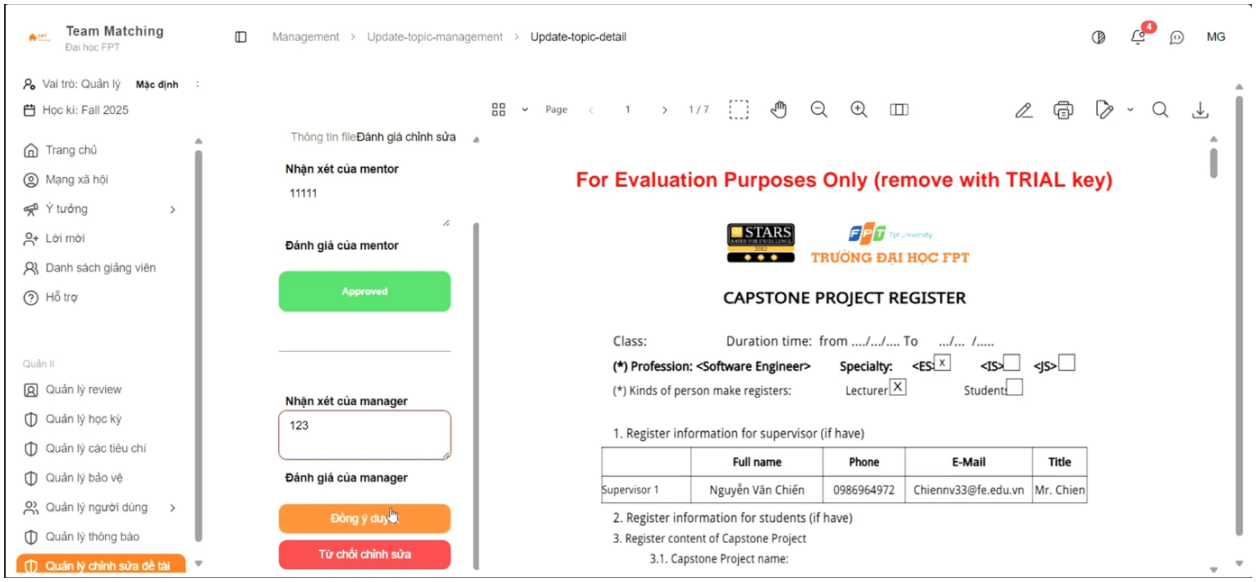


3.2.16 Manager respond updating topic request

- Step 1: Select “Quản lí chỉnh sửa đề tài” tab then click view detail on the topic.

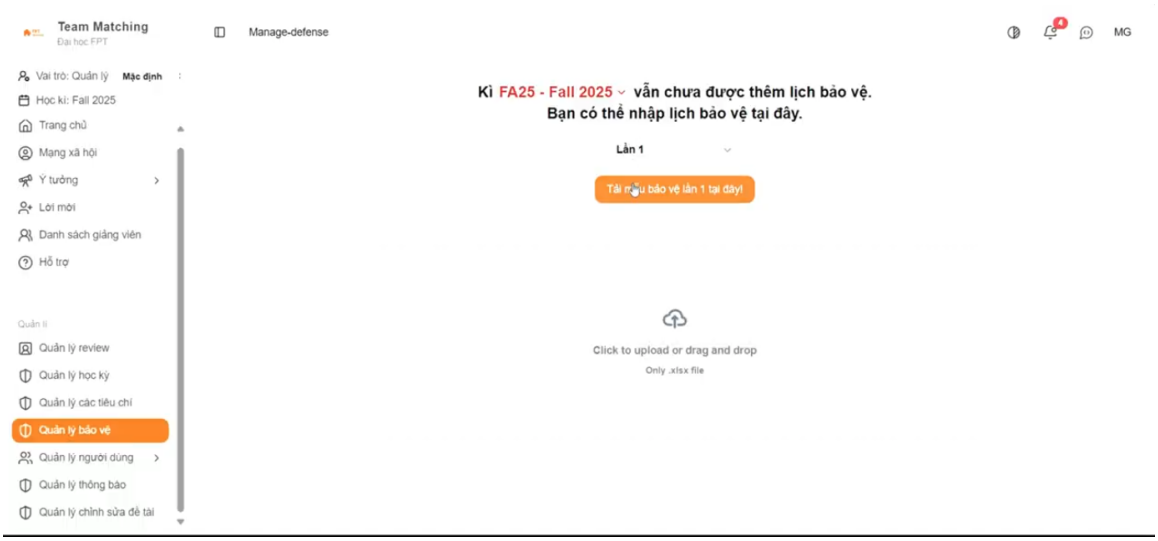


- Step 2: Then manager give feedback and final decision.

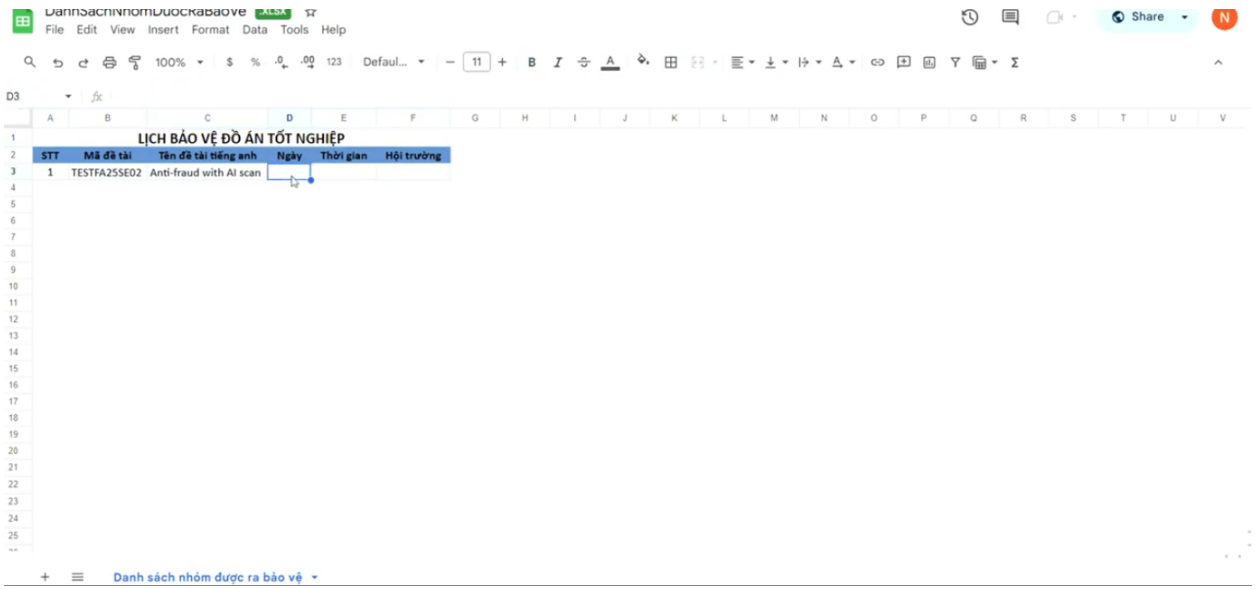


3.2.17 Export template defense schedule

- Step 1: Select “Quản lí bảo vệ” tab and then select the “Tải mẫu bảo về lần … tại đây” button to download template

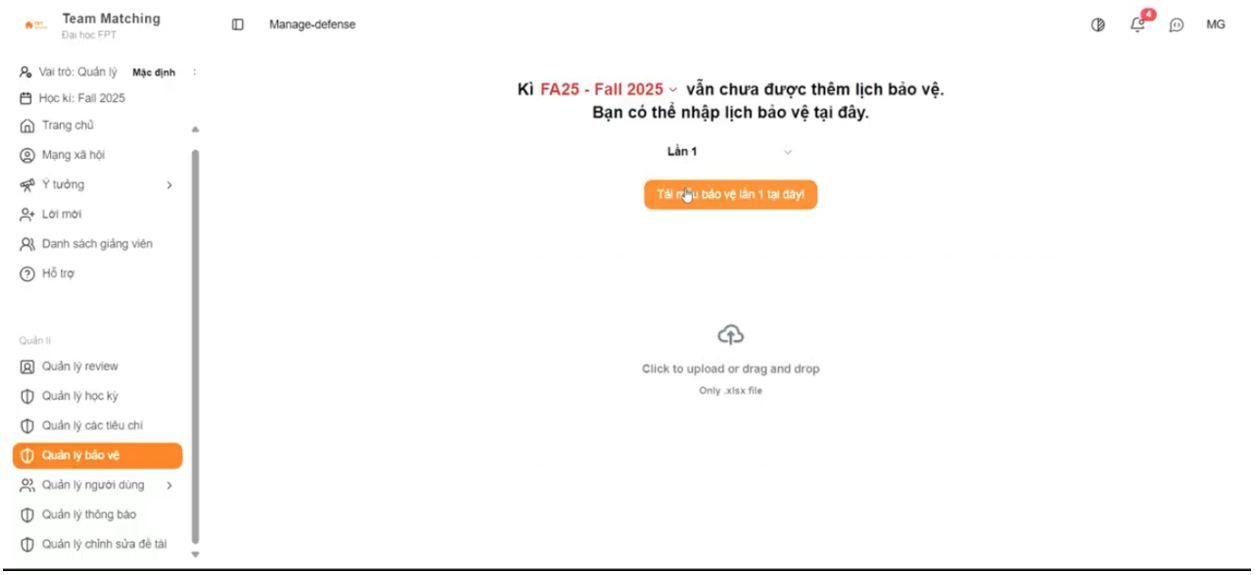


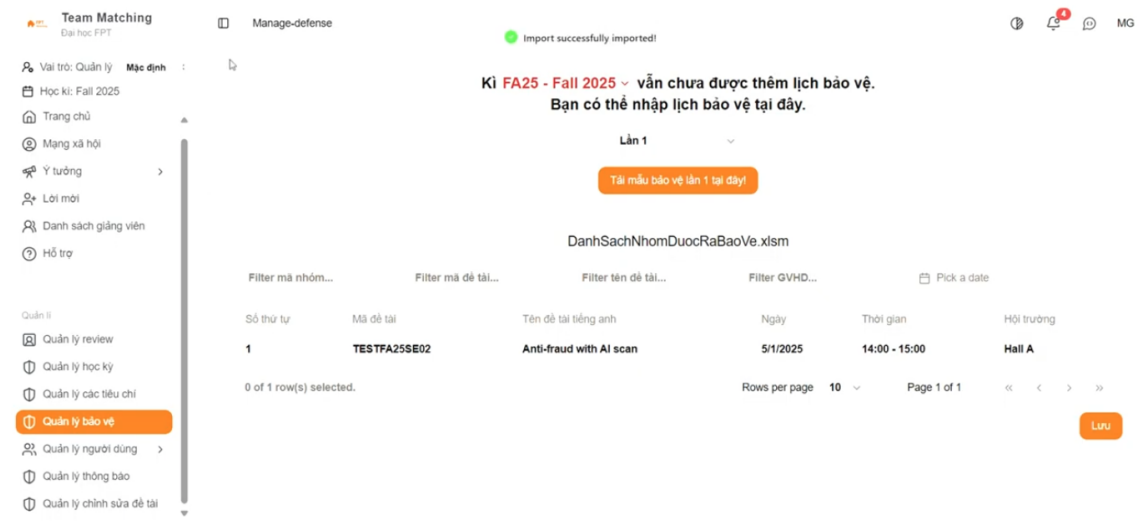
- Template will look like this



3.2.18 Import defense schedule

- Step 1: After having a file template with valid data, then clicking to upload file, it will show content inside the file.





- Step 2: Click “Lưu” button to upload file and save a defense schedule.

3.2.19 Update defense result

- Step 1: Select **Quản lí review** on the navigation bar -> **View Detail**

- Step 2: Fill information and click **Lưu thông tin** button