RadiantIQ

UNIVERSITÀ DI TRENTO
Department of Information Engineering and Computer Science

Project's acronym:	RADIANTIQ
Project's title:	RadiantIQ
Start date:	08/03/2024
Finish date:	DD/MM/YYYY

D1 - RadiantIQ Software Specification

WP1:	Software Specification
Task 1.1:	Define Software Specification
Submission date:	22/04/2024
Responsible:	RadiantIQ
Version:	1.0
Status:	Completed
Author(s):	Lorenzo Cattai, Gabriele Pernici, Anh Tu Duong, Lorenzo Negut
Deliverable type:	DOCUMENT

Version list:

Version	Authors	Date	Description
0.1	Anh Tu Duong	09/04/2024	Define the scheme of the document
1.0	Anh Tu Duong	11/04/2024	Update domain analysis and objectives
1.1	Anh Tu Duong, Gabriele Pernici, Lorenzo Cattai	18/04/2024	Update usecases
1.2	Gabriele Pernici	19/04/2024	Update actors, functional, non-functional requirements
1.3	Gabriele Pernici, Anh Tu Duong, Lorenzo Cattai	19/04/2024	Update context diagram, component diagram
1.4	Gabriele Pernici, Anh Tu Duong, Lorenzo Cattai, Lorenzo Negut	21/04/2024	Update class diagram

Contents

Acronyms

Acronym	Description
FRS	Functional Requirement Specification
FSD	Functional Specification Document
FLOSS	Free/Libre Open Source Software
LLMs	Large Language Models
UML	Unified Modeling Language
UniTn	University of Trento

1 Introduction

The project consists in a platform providing a better learning experience for scientific subjects. The idea is to include standard formal explanations of topics (with associated exercises) accompanied by a small number of interactive minigames. To better involve the students, each exercise will be put in an AI generated context (e.i. a phisics problem related to speeds and distances could be told using the story of Achilles and the turtoise). Moreover, the formal explanations can be genrated by an AI, uploaded by a professor or by a combination of the two. Lastly, AI is used to suggest which topics should be revised for the students using the platform.

2 Domain Analysis

2.1 Domain: Interactive Education

We want to create an application in the education domain and try to make it extremely interactive.

2.2 SWOT analysis (interactive education domain)

- **Strengths:**
  - Use of AI to generate basic information in the articles, with the supervision of admins.
  - Use of AI to generate more advanced information from user's customized necessities and weaknesses
  - Use a more fun and interactive approach in the education process
  - Learn by trying with minigames with immediate feedback or in a more standard way with articles
  - Possibility for experts to contribute with their own articles
  - FLOSS Education Platform
- **Weaknesses:**
  - Developing effective mini-games' experiences might be time consuming from a developer perspective
  - Making fun and yet instructive experiences is hard
  - Careful management of the AI generated information is needed
  - Quality check on article uploaded must be implemented
  - Costs of using LLMs
- **Opportunities:**
  - Using AI in education is an emerging idea that has not yet spread widely and can make this system unique
  - Various studies show the positive effect of interaction and hands-on experiences in the learning process
  - Large demand of easy and complete ways to learn science
  - Lack of effective and proficient communication/cooperation/interaction between students and teachers which creates an ample improvement margin to be enhanced
- **Threats:**
  - Possible limitations of AI technologies from government entities
  - Minigames must be entertaining to be successful
  - Brilliant is a direct competitor (tho it does not currently use AI nor personalize its exercises and lectures)
  - Hard to find funding

3 Project Objectives

3.1 Use of minigames in interactive learning

Create some minigames to make learning more interactive and more intriguing for students.

3.2 Use AI to help develop a more compelling learning experience

Use AI to generate compelling descriptions of the topics, to make learning funnier.

3.3 Make learning science more approachable and enjoyable

Make the platform an accessible starting point in the learning of science, to allow everyone to learn science using intuition and reason.

3.4 Provide single-topic focused content

Providing single topic courses means the possibility to create a learning path specific for the interests of the user.

3.5 Provide private classes as instances of a course

Create classes, from the general courses, to allow teachers/professors to integrate the platform in their standard lectures.

3.6 Allow better student-professor interaction

Using the classes the professors can understand which topics are clearer and then provide feedback. Moreover students can easily determine the level of comprehension of each topic before a test.

3.7 Provide quality guarantees on the material published

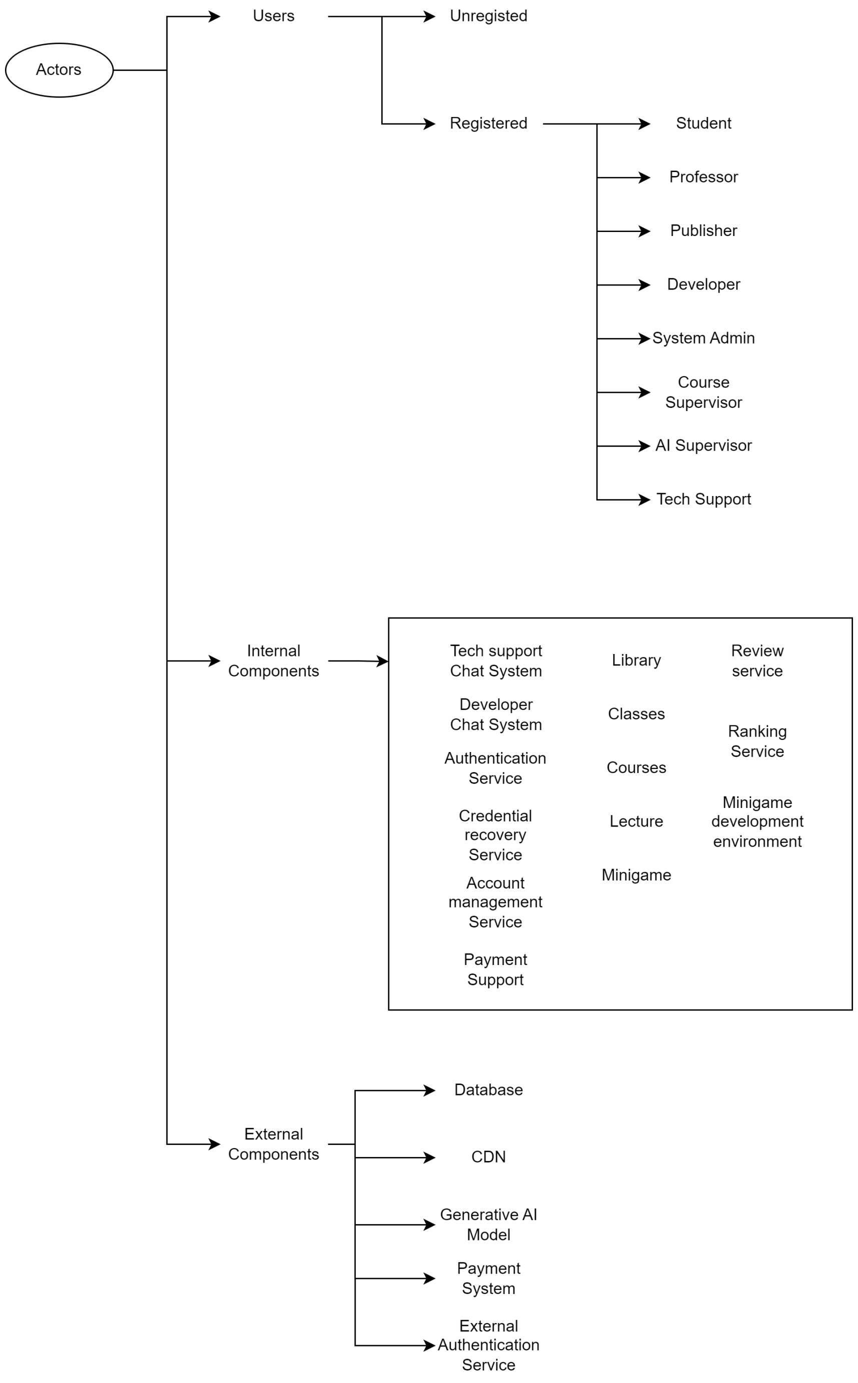
Have personnel checking the validity of the published material, while not interfering with the private resources.

3.8 Collect data and provide a progress history

For the learning user, having feedback on the level of comprehension is fundamental and will help them focus more on the less understood topics.

4 Actors

We created a Mind map summarizing the actors scheme of the system, it is shown below.



4.1 Roles functioning

Each role provides specific and unique functionalities (that are mutually exclusive). Each user can have multiple roles assigned to them simultaneously, in a composite role system similar to the one used in Discord. We chose this model to easily separate functionalities and assign them to users modularly.

5 Functional Requirements

5.1 User stories

1. As a *Registered User*
   I want to *login*
   To *access my personal profile*
2. As a *Registered User*
   I want to *terminate my account*
   To *remove myself from the platform*
3. As a *Registered User*
   I want to *change my username, password and settings*
   To *customize the profile*
4. As a *Registered User*
   I want to *access the support chat*
   To *interact with tech support*
5. As a *Registered User*
   I want to *review the courses*
   To *leave my opinion of the course*
6. As a *Unregistered User*
   I want to *use the platform anonymously*
   To *try out the platform without committing to it*
7. As a *Unregistered User*
   I want to *register*
   To *receive an actual role in the platform and access its benefits*
8. As a *Student*
   I want to *enter a custom class created by a professor*
   To *be able to partake in the lectures of my teacher/professor*
9. As a *Student*
   I want to *look through the library and choose my own preferred subjects*
   To *be able to learn what I care about*
10. As a *Student*
    I want to *customize the AI generated theming*
    To *immerse myself more into the learning process*
11. As a *Student*
    I want to *access my statistics, progresses and ranking*
    To *keep track of how I am doing*
12. As a *Student*
    I want to *access the courses and relative activities*
    To *actually learn*
13. As a *Student*
    I want to *see other member of my custom classes*
    To *make sure I joined the right custom class and to know who my peers are*
14. As a *Professor*
    I want to *create a class from a course*
    To *give my students an interactive course under my supervision*

15. As a *Professor*
    I want to *add students to my classes*
    To *have my students follow the path I prepared*
16. As a *Professor*
    I want to *see the list of students of my custom classes*
    To *make sure my students joined my class*
17. As a *Professor*
    I want to *see the statistics and progresses of students of my custom classes*
    To *see how they are performing*
18. As a *Publisher*
    I want to *publish articles, create and modify courses*
    To *add knowledge to the platform*
19. As a *Publisher*
    I want to *change the visibility of courses I created*
    To *decide to keep the course private or make it available for everyone*
20. As a *Publisher*
    I want to *contact developers to add minigames to my courses*
    To *make the experience funnier for the students*
21. As a *Publisher*
    I want to *use a payment system to compensate the developers for their work*
    To *complete the transaction directly on the platform*
22. As a *Course Supervisor*
    I want to *check and change the courses and their contents*
    To *ensure a high quality level and fix possible errors*
23. As a *Course Supervisor*
    I want to *give proof of my qualification*
    To *obtain the right to be a supervisor*
24. As a *AI Supervisor*
    I want to *be able to change the parameters and supervise the generation of content the AI model*
    To *make the user experience better and the generated content more accurate*
25. As a *Developer*
    I want to *develop minigames on request*
    To *help publishers create better courses*
26. As a *Developer*
    I want to *register on payment systems*
    To *receive the money I earn*
27. As a *System Admin*
    I want to *get debug information about the platform*
    To *make sure everything is running properly*
28. As a *System Admin*
    I want to *interact directly with the backend instance*
    To *fix any kind of problems that may emerge*
29. As a *Tech support*
    I want to *respond to the messages on the support chat*
    To *understand the problems of the users*
30. As a *Tech support*
    I want to *have more access to the system*
    To *fix reported user problems*

5.2 Functional requirements

Functional requirements from the Registered Users point of view

FR 1. Registered Users should be able to login using

FR 2. Registered Users should be able to terminate their account and have it completely removed from existence

FR 3. Registered Users should be able to change their username, password and settings (both core and secondary)

FR 4. Registered Users should be able to open chats with tech support

FR 5. Registered Users should be able to rate the courses according to specific parameters

Functional requirements from the Unregistered Users point of view

FR 6. Unregistered Users should be able to use the platform anonymously

FR 7. Unregistered Users should be able to register

Functional requirements from the Students point of view

FR 8. Students should be able to enter a custom class (created by a professor)

FR 9. Students should be able to look through the library of courses and favorite them at will

FR 10. Students should be able to customize the AI generated theming by setting a basic text prompt that will get passed to the AI model when generating the wrapping for the course

FR 11. Students should be able to access their statistics and progress on every course

FR 12. Students should be able to access the course and relative activities

FR 13. Students should be able to see who is enrolled in a class

Functional requirements from the Professors point of view

FR 14. Professors should be able to create classes where users can join to get monitored by the professor

FR 15. Professors should be able to add students to their classes by inviting them

FR 16. Professors should be able to see the list of students of my custom classes

FR 17. Professors should be able to see the statistics and progresses of students of my custom classes

Functional requirements from the Publishers point of view

FR 18. Publishers should be able to publish articles, create, and modify courses

FR 19. Publishers should have the ability to change the visibility of courses they created, deciding whether to keep them private or make them public

FR 20. Publishers should be able to hire developers to add minigames to their courses through an internal development chat

FR 21. Publishers should be able to utilize a payment system to compensate developers for their work, completing transactions directly on the platform

Functional requirements from the Course Supervisors point of view

FR 22. Course Supervisors should be able to review and modify courses

FR 23. Course Supervisors must provide proof of their qualifications

Functional requirements from the AI Supervisors point of view

FR 24. AI Supervisors should be able to adjust parameters and oversee the generation of content by the AI model, specifically focusing on improving the wrapping of the core lectures and exercises

Functional requirements from the Developers point of view

FR 25. Developers should be able to develop minigames upon request to assist publishers in creating better courses

FR 26. Developers should be able to access development chat to communicate with publishers

Functional requirements from the System Admins point of view

FR 27. System Admins should be able to get debug information about the current state of the platform

FR 28. System Admins should be able to directly interact with the backend instance to debug it

Functional requirements from the Tech Support point of view

FR 29. Tech Support should be able to respond to support chats from users

FR 30. Tech Support should have access to the majority of functionalities of the system

Functional requirements from the System point of view

FR 31. The application must provide a registration and authentication procedure

FR 32. The application must provide an account deletion system

FR 33. The application must allow users to modify their accounts informations and settings

FR 34. The application must provide an internal communication system to provide technical support

FR 35. The application must categorize each registered user with one or more roles

FR 36. The application must allow the creation and modification of courses (both private and public), as a group of articles, minigames and exercises

FR 37. The application must provide an internal communication system, to allow publishers to hire developers to create minigames for their courses

FR 38. The application must provide a payment system to allow the publishers to pay the developers for their work

FR 39. The application must incorporate a generative AI model in order to generate articles and/or context for the exercises

FR 40. The application must provide a support service to check the quality of the material published

FR 41. The application must allow the creation of private instances of courses (called classes) with customizable access limitations

FR 42. The application must allow the dynamic addition or removal of students from classes

FR 43. The application must keep an accessible record of the participants of each class

FR 44. The application must allow students to partake in classes or single courses

FR 45. The application must showcase a list of available classes and courses

FR 46. The application must keep track of registered user's results for each course they take part in and keep a global ranking of them

FR 47. The application must allow the users to locally access part of the content

FR 48. The application must provide a verification system for the Course Supervisor's qualifications

FR 49. The application must provide a grading system for the courses

6 Non-functional Requirements

We analyzed non-functional requirements in eight categories: Performance (how fast does the platform respond), Scalability (how many elements should be supported by the infrastructure), Portability (which systems and devices does the platform run on), Reliability (when could problems arise), Maintainability (how long does it take to fix arisen problems), Availability (when, where and in which circumstances should the platform or its contents be available), Security (*TODO*) and Usability (everything concerning user experience).
On the time considerations we assumed the following: 0.1 seconds is the time limit for seemingly instantaneous response; with, at most, a 1 second response time the user will notice the delay, but the flow of thought won't be interrupted; after 10 seconds the user's attention is completely lost. However, for AI generation more time could be needed, especially for complex prompts.
The maximum concurrent visits estimate (and other numerical data in the Scalability category) was reached after making some estimations/research on UniTn data. So the aim is to possibly allow an entire university's student pool to be active concurrently.

1. **Performance**:
    - The application response time should be always under 5 seconds.
    - Loading time for classes/courses should be under 5 seconds.
    - Loading time for lectures should be under 3-4 seconds.
    - Loading time for mini-games should be at most 5 seconds.
    - Every mini-game should always run at, at least, 30 FPS.
    - Loading time for AI theming generation should be at most 2 minutes.
2. **Scalability**:
    - The platform supports at most 20000 concurrent visits.
    - The platform should support around 1000 courses/classes.
    - The platform should support around 100000 registered users.
    - The platform will use *TODO total material storage* of stored data, with an average of *TODO single course storage* for each course and an average of *TODO single user storage* for each user.
3. **Portability**:
    - The platform should run on every main smoothly browser (Chrome, Safari, Firefox, Edge, Opera).
    - The platform should be available in app form on both android and IOS.
    - Maintain consistent user experience among different screen sizes.
4. **Reliability**:
    - Platform should experience bugs or failures only right after the release of a new unstable version.
    - LLM for AI theming will rely entirely on external implementation so it could lead to fatal errors in that regard (if AI regulations change, if the external API are offline, ...).
5. **Maintainability**:
    - Errors or bugs regarding new unstable versions should be solved within three days after discovery.
    - Unstable versions should be marked as stable within 3 weeks after release.
    - LLM external implementation should be replaced, if possible, within one month of service's absence.

6. **Availability**:
    - The platform should be available in every country with internet access and academic freedom.
    - The platform should be available at any time except for maintenance periods that will be announced at least two weeks in advance and should last no longer than a day.
    - Majority of the material should be available also offline (within the application mode).
7. **Security**:
    - Authentication will be a single factor one with a compulsory strong password selection.
    - Authorization to perform specific actions will be granted only to the required roles.
    - Sensible roles (such as Admin or Course supervisor) will be granted only if specific requirements are met.
    - Data on the DB will be encrypted with a solid encryption algorithm.
    - Payment systems will be verified as reliable and secure.
    - Anonymity of unregistered users will be granted.
8. **Usability**:
    - User interface should be easy to use and appealing to the eye.
    - Mini-games should have intuitive designs and/or detailed instructions.
    - Material research should be easy and intuitive.
    - Class enrollment should be intuitive.
    - Material reviews should be easy to find and post.
    - Support chat should be immediate to reach.
    - Platform should allow for a colorblind mode.
    - Localization in various languages (English, Italian, German, Spanish, Chinese, Portuguese, French, Japanese, Korean).

7 Use Cases

We created both the tables for each use case and the complete diagram. Moreover, we created some partial diagrams containing some use cases and organized logically.

7.1 Tables

The following are the 42 use case tables.

7.1.1 Login

Login	
ID	UC1 - UC_LOGIN
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and has credentials
Sequence	1. The actor selects the login option 2. The actor inserts its credentials 2.1 If the actor uses an external authentication system it's redirected 3. Credentials are verified
Postconditions	The user is authenticated and can access its roles' privileges
Alternative sequence 1	1. The actor inputs the wrong password for the first, second or third time
Postconditions	The user is not authenticated and a notification is sent to the user registered with the inserted username
Alternative sequence 2	1. The actor inputs the wrong password for the fifth time
Postconditions	The user is not allowed to login for a significant time and a notification is sent to the user registered with the inserted username
Alternative sequence 3	1. The actor inputs the wrong username
Postconditions	The user is not authenticated

7.1.2 Logout

Logout	
ID	UC2 - UC_LOGOUT
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the logout option 2. The actor confirms their choice
Postconditions	The user is logged out

7.1.3 Credential recovery

Credential recovery	
ID	UC3 - UC_CREDENTIAL_REC
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and has forgotten credentials
Sequence	1. The actor has forgotten their credentials 2. The actor requires new credentials
Postconditions	The user acquires new credentials on the previously specified recovery channel

7.1.4 Register account

Register account	
ID	UC4 - UC_ACC_REGISTRATION
Actors	Unregistered users
Preconditions	The actor decides to register
Sequence	1. The actor selects the account registration option 2. The actor decides their credentials, core settings and which roles they want to apply for 2.1 The credentials' compliance with security policies is asserted
Postconditions	The user is registered and has now one or more roles assigned
Alternative sequence 1	1. The actor wants to apply for the Course Supervisor role 2. The actor provides the apposite proof of identity
Postconditions	The user is registered as a Course Supervisor
Alternative sequence 2	1. The actor inputs incomplete, incorrect or unacceptable credentials/proof
Postconditions	The user is not registered

7.1.5 Delete Account

Delete Account	
ID	UC5 - UC_ACC_DEL
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered, logged in and decides to delete the account
Sequence	1. The actor selects the unregistration option 2. The actor confirms their choice 3. The actor confirms their identity by inserting the account's password
Postconditions	The user is unregistered

7.1.6 Modify account’s core settings

Modify account’s core settings	
ID	UC6 - UC_MOD_CORE_SETT
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered, logged in and decides to change one or more of the core settings (e.g. password, username, recovery channel, personal information, . . .)
Sequence	1. The actor selects the modify core settings option 2. The actor confirms their identity by inserting the account’s password 3. The actor changes the selected settings 4. The actor confirms the choice
Postconditions	The change in settings is saved
Alternative sequence 1	1. The actor doesn’t confirm their changes or cancels the modification
Postconditions	The settings stay the same

7.1.7 Modify account's secondary settings

Modify account's secondary settings	
ID	UC7 - UC_MOD_SEC_SETT
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered, logged in and decides to change one or more of the secondary settings (e.g. theme, layout, . . .)
Sequence	1. The actor selects the modify secondary settings option 2. The actor changes the selected settings 3. The actor confirms the choice
Postconditions	The change in settings is saved
Alternative sequence 1	1. The actor doesn't confirm their changes or cancels the modification
Postconditions	The settings stay the same

7.1.8 Modify AI theming

Modify AI theming	
ID	UC8 - UC_MOD_AI_THEMING
Actors	Students, AI supervisors, Tech Supports
Preconditions	The user is registered and logged in
Sequence	1. The user selects the AI theming option 2. The user inputs a prompt for the AI theming 3. The user confirms the modification
Postconditions	The theming prompt is modified
Alternative sequence 1	1. The user doesn't confirm the modification
Postconditions	AI theming continues with the previous prompt

7.1.9 Access profile and statistics

Access profile and statistics	
ID	UC9 - UC_ACCESS_PROFILE
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the account display option
Postconditions	The profile with all its statistics is displayed

7.1.10 Change user role

Change user role	
ID	UC10 - UC_USER_ROLE
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and logged in
Sequence	1. The actor selects “User Role” panel 2. The actor selects the one of the user role from the list
Postconditions	The user role changed following by his dashboard role

7.1.11 Create course

Create course	
ID	UC11 - UC_COURSE_CREATE
Actors	Publishers
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the course panel 2. The actor selects the “Create new course” option 3. The actor fills all the mandatory sections for creating a new course 4. The actor confirms to create new course
Postconditions	The course is created successfully
Alternative sequence 1	1. The actor doesn’t fill all the mandatory sections for creating a new course 2. The actor confirms to save the unfinished work
Postconditions	The course is saved as draft

7.1.12 Modify course

Modify course	
ID	UC12 - UC_COURSE_MOD
Actors	Professors, Publishers, AI supervisors, Course Supervisors, System Admins
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the course 2. The actor selects the “Modify course” option 3. The actor modifies the course 4. The actor confirms the modification
Postconditions	The course is updated successfully
Alternative sequence 1	1. The actor doesn’t confirm the modification
Postconditions	The course keeps its previous state

7.1.13 Delete course

Delete course	
ID	UC13 - UC_DEL_COURSE
Actors	Publishers, Course Supervisors, System Admins
Preconditions	The user is registered and logged in
Sequence	1. The actor selects “Delete course” 2. The actor confirms the deletion
Postconditions	The course is deleted

7.1.14 Archive course

Archive course	
ID	UC14 - UC_ARC_COURSE
Actors	Publishers, Course Supervisors, System Admins
Preconditions	The user is registered and logged in
Sequence	1. The actor selects “Archive course” 2. The actor confirms the modification
Postconditions	The course is moved to archive

7.1.15 View course

View course	
ID	UC15 - UC_COURSE_VIEW
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins) and Unregistered users
Preconditions	The user is registered, logged in and has the right to enter the course
Sequence	1. The actor select a course from the dashboard or from a library
Postconditions	The course is displayed, along with the global ranking if any minigame is present

7.1.16 Review course

Review course	
ID	UC16 - UC_COURSE_REV
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered, logged in and has has the right to enter the course. Moreover the course is public and opened
Sequence	1. The actor leave a review (comment) for the course
Postconditions	The review is added to the course

7.1.17 Create class

Create class	
ID	UC17 - UC_CLASS_CREATE
Actors	Professors
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the class panel 2. The actor selects the "Create new class" option 3. The actor fills all the mandatory sections for creating a new class 4. The actor has possibility to invite Student(s) to the class 5. The actor confirms to create new class
Postconditions	The class is created successfully

7.1.18 Modify class

Modify class	
ID	UC18 - UC_CLASS_MOD
Actors	Professors, System Admins
Preconditions	The user is registered, logged in and has sufficient permissions (owns the class or is admin)
Sequence	1. The actor selects the class 2. The actor selects the "Modify class" option 3. The actor modifies the class 4. The actor has possibility to invite Student(s) to the class 5. The actor confirms the modification
Postconditions	The class is updated successfully
Alternative sequence 1	1. The actor doesn't confirm the modification
Postconditions	The class remains its previous state

7.1.19 Terminate class

Terminate class	
ID	UC19 - UC_CLASS_TERM
Actors	Professors, System Admins
Preconditions	The user is registered, logged in and has sufficient permissions (owns the class or is admin)
Sequence	1. The actor selects "Terminate class" 2. The actor confirms the termination
Postconditions	The class is terminated, but all the information about the class remains public

7.1.20 Archive class

Archive class	
ID	UC20 - UC_ARC_CLASS
Actors	Professors, System Admins
Preconditions	The user is registered, logged in and has sufficient permissions (owns the class or is admin)
Sequence	1. The actor selects “Archive class” 2. The actor confirms the modification
Postconditions	The class is moved to archive and all the information about the class is accessible only for Professors and System admins

7.1.21 View class

View class	
ID	UC21 - UC_CLASS_VIEW
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and logged in
Sequence	1. The actor selects a class from the dashboard or from an invitation
Postconditions	The public information of the class is displayed

7.1.22 Display class’ attendees

Display class’ attendees	
ID	UC22 - UC_CLASS_ATTENDEES
Actors	Students, Professors
Preconditions	The user is registered, logged in and enrolled in a class
Sequence	1. The actor selects a class they are enrolled into
Postconditions	The list of people attending the class is displayed

7.1.23 Display class' statistics

Display class' statistics	
ID	UC23 - UC_CLASS_PERFORMANCE
Actors	Professors
Preconditions	The user is registered, logged in and manages a class
Sequence	1. The actor selects a class they manage
Postconditions	The performance and statistics of all the attendees is displayed

7.1.24 Join class

Join class	
ID	UC24 - UC_CLASS_JOIN
Actors	Students
Preconditions	The user is registered, logged in and has the right to join the class. Moreover the class is opened and public
Sequence	1. The actor selects “Join class” option
Postconditions	The actor now joined the class. The actor's information, statistics and progresses for the class is initialized and is public for the class's owner

7.1.25 Leave class

Leave class	
ID	UC25 - UC_CLASS_LEAVE
Actors	Students
Preconditions	The user is registered, logged in and is part of a class
Sequence	1. The actor selects “Leave class” option 2. The actor confirm their choice
Postconditions	The actor leaves the class, but their information, statistics and progresses for the class is saved and is still public for the class's owner

7.1.26 Publish article

Publish article	
ID	UC26 - UC_ART_PUB
Actors	Publisher
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the article panel 2. The actor selects the “Publish new article” option 3. The actor fills all the mandatory sections for publishing a new article 4. The actor confirms to publish new article
Postconditions	The article is published successfully

7.1.27 Modify article

Modify article	
ID	UC27 - UC_ART_MOD
Actors	Publisher, Course Supervisors, System Admins
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the article 2. The actor selects the “Modify article” option 3. The actor modifies the article or its visibility 4. The actor confirms the modification
Postconditions	The article is updated successfully
Alternative sequence 1	1. The actor doesn’t confirm the modification
Postconditions	The article remains in its previous state

7.1.28 Delete article

Delete article	
ID	UC28 - UC_ART_DEL
Actors	Publisher, Course Supervisors, System Admins
Preconditions	The user is registered, logged in and has ownership of the article (or is admin)
Sequence	1. The actor selects “Delete article” 2. The actor confirms the deletion
Postconditions	The article is deleted

7.1.29 Archive article

Archive article	
ID	UC29 - UC_ART_ARC
Actors	Publisher, Course Supervisors, System Admins
Preconditions	The user is registered, logged in and has ownership of the article (or is admin)
Sequence	1. The actor selects “Archive article” 2. The actor confirms the modification
Postconditions	The article is moved to archive

7.1.30 View article

View article	
ID	UC30 - UC_ART_VIEW
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins) and Unregistered users
Preconditions	The user is registered and logged in
Sequence	1. The actor selects an article from the dashboard
Postconditions	The article is opened

7.1.31 Review article

Review article	
ID	UC31 - UC_ART_REV
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered and logged in. The article is opened and public
Sequence	1. The actor leave a review (comment) for the article
Postconditions	The review is added to the article

7.1.32 Create minigame

Create minigame	
ID	UC32 - UC_MINIGAME_CREATE
Actors	Developers
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the developer panel 2. The actor selects the “Create new minigame” option 3. The actor uses the environment to create the minigame 4. The actor confirms to create new minigame
Postconditions	The minigame is created successfully and saved to minigame storage

7.1.33 Modify minigame

Modify minigame	
ID	UC33 - UC_MINIGAME_MOD
Actors	Developers, Course Supervisors, System Admin, Tech Supports
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the minigame 2. The actor selects the “Modify minigame” option 3. The actor modifies the minigame 4. The actor confirms the modification
Postconditions	The minigame is updated successfully
Alternative sequence 1	1. The actor doesn’t confirm the modification
Postconditions	The minigame remains in its previous state

7.1.34 Delete minigame

Delete minigame	
ID	UC34 - UC_MINIGAME_DEL
Actors	Developers, Course Supervisors, System Admin
Preconditions	The user is registered, logged in and owns the minigame (or is admin)
Sequence	1. The actor selects “Delete minigame” 2. The actor confirms the deletion
Postconditions	The minigame is deleted from memory and from all the courses containing it

7.1.35 Archive minigame

Archive minigame	
ID	UC35 - UC_MINIGAME_ARC
Actors	Developers, Course Supervisors, System Admin
Preconditions	The user is registered, logged in and owns the minigame (or is admin)
Sequence	1. The actor selects "Archive minigame" 2. The actor confirms the modification
Postconditions	The minigame is moved to archive and is deleted from all the courses containing it

7.1.36 Register minigame from developer

Register minigame from developer	
ID	UC36 - UC_MINIGAME_REG_DEV
Actors	Developers
Preconditions	The user is registered and logged in
Sequence	1. The actor selects the minigame 2. The actor selects "Register minigame to course" 3. The actor chooses the course(s) to register (add) minigame into 4. The actor fills all the mandatory sections for registering a new minigame to course 5. The actor confirms to register minigame
Postconditions	The minigame is register successfully to the course(s)

7.1.37 Register minigame from observer

Register minigame from observer	
ID	UC37 - UC_MINIGAME_REG
Actors	Publishers, Course Supervisors, System Admins
Preconditions	The user is registered, logged in and has authorization from minigame owner
Sequence	1. The actor selects the course 2. The actor selects the “Modify course” option 3. The actor selects “Register minigame to course” 4. The actor chooses the minigame to add to course 5. The actor fills all the mandatory sections for registering a new minigame to course 6. The actor confirms to register minigame
Postconditions	The minigame is register successfully to the course(s)

7.1.38 Play minigame

Delete course	
ID	UC38 - UC_PLAY
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins) and Unregistered users
Preconditions	The user opens the minigame
Sequence	1. The actor plays the minigame
Postconditions	The actor can play the minigame

7.1.39 Pay developer

Pay developer	
ID	UC39 - UC_PAY
Actors	Publishers
Preconditions	The user is registered, logged in and has tasked a developer with a minigame
Sequence	1. The actor selects the payment option 2. The user selects the external payment method they want to use 3. The actor follows the external payment system iter
Postconditions	The actor has paid the developer

7.1.40 Use tech support chat

Use tech support chat	
ID	UC40 - UC_SUPPORT_CHAT
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins)
Preconditions	The user is registered, logged in and has a technical problem
Sequence	1. The actor opens the tech support chat 2. The user sends/receives a message on the chat
Postconditions	The actor interacts with Tech Supports and starts solving the technical problem

7.1.41 Use development chat

Use development chat	
ID	UC41 - UC_DEV_CHAT
Actors	Publishers, Developers
Preconditions	The user is registered, logged in and has the need to discuss about minigame creation
Sequence	1. The actor opens the development chat 2. The actor communicates with the commissioner/developer
Postconditions	The actor interacts with Developer/commissioner and starts the minigame development process

7.1.42 Search material

Search material	
ID	UC42 - UC_SEARCH
Actors	Registered users (Students, Professors, Publishers, Developers, AI supervisors, Tech Supports, Course Supervisors, System Admins) and Unregistered users
Preconditions	The user wants to find some material on the platform (course, class, mini-game, article, . . .)
Sequence	1. The actor selects the search option 2. The actor inputs the possible search parameters (name, subject, type, . . .)
Postconditions	A list of resources adhering to the parameters is displayed

7.1.43 Remove review

Remove review	
ID	UC43 - UC_DEL_REV
Actors	System Admins
Preconditions	The user is registered and logged in. The chosen review violates some policies of the platform
Sequence	1. The actor removes review from any material
Postconditions	The review is removed entirely

7.2 Diagrams

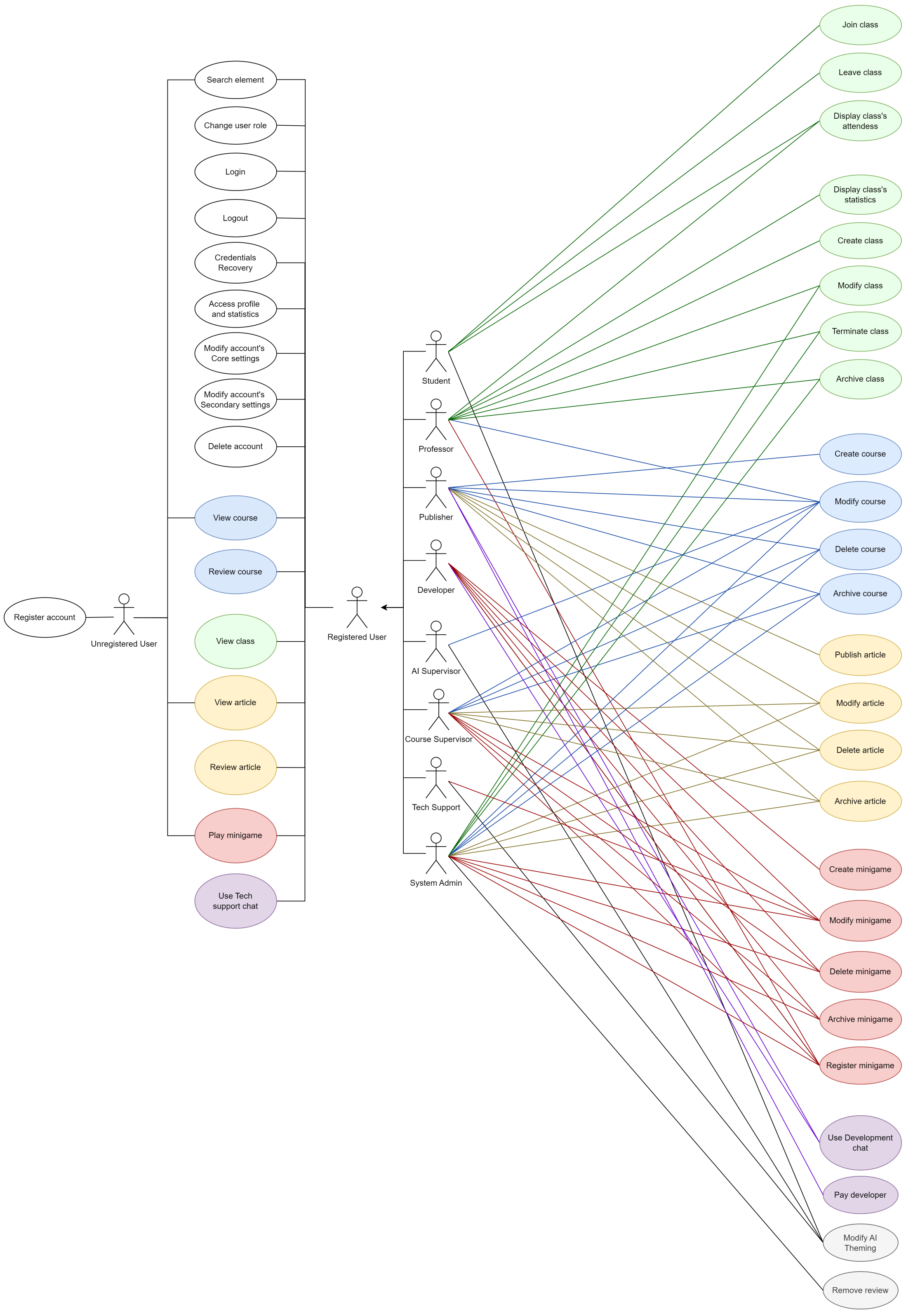


Figure 2: Complete use case diagram

7.2.1 Account and general purpose

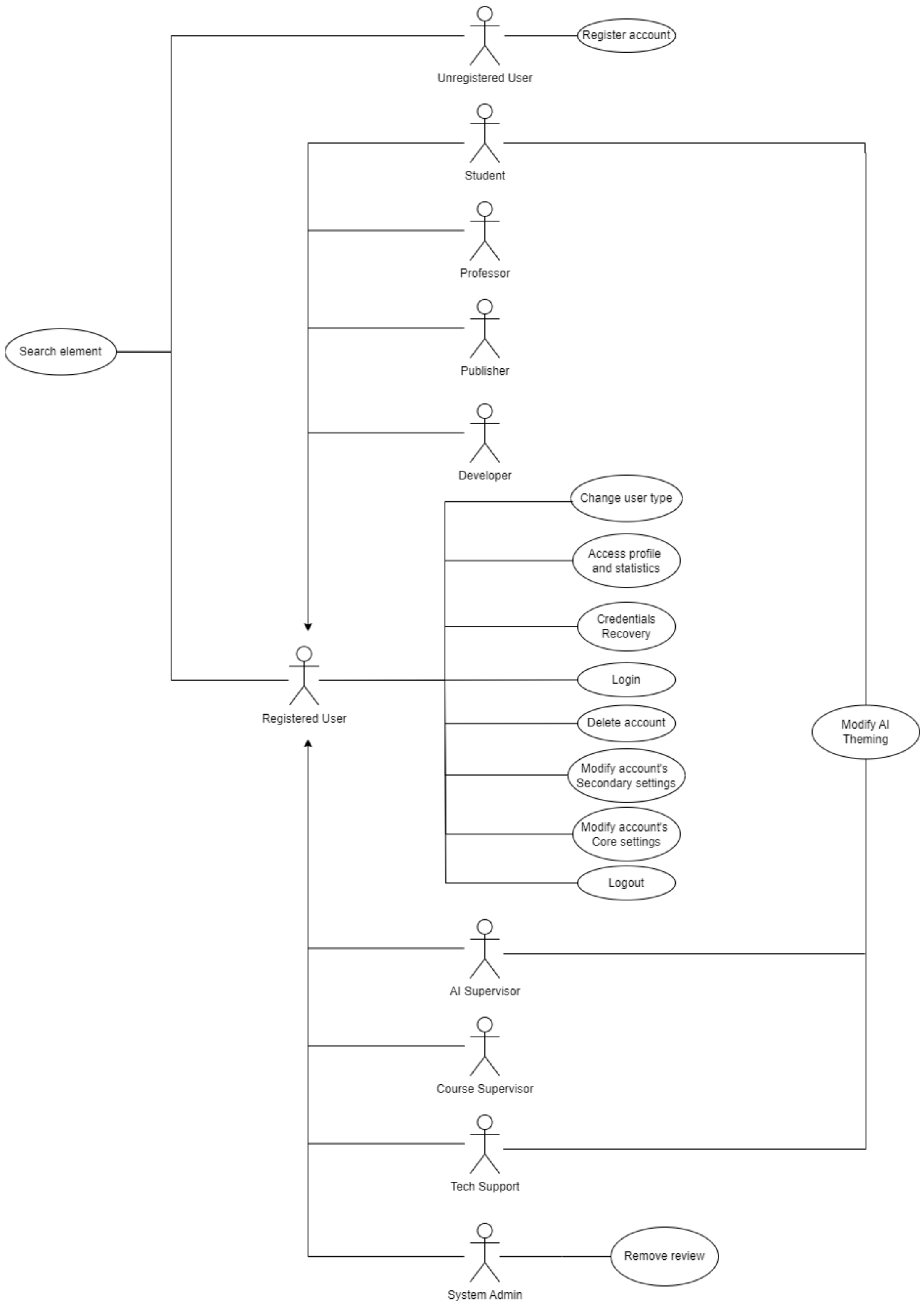


Figure 3: Use case diagram for account system

7.2.2 Article

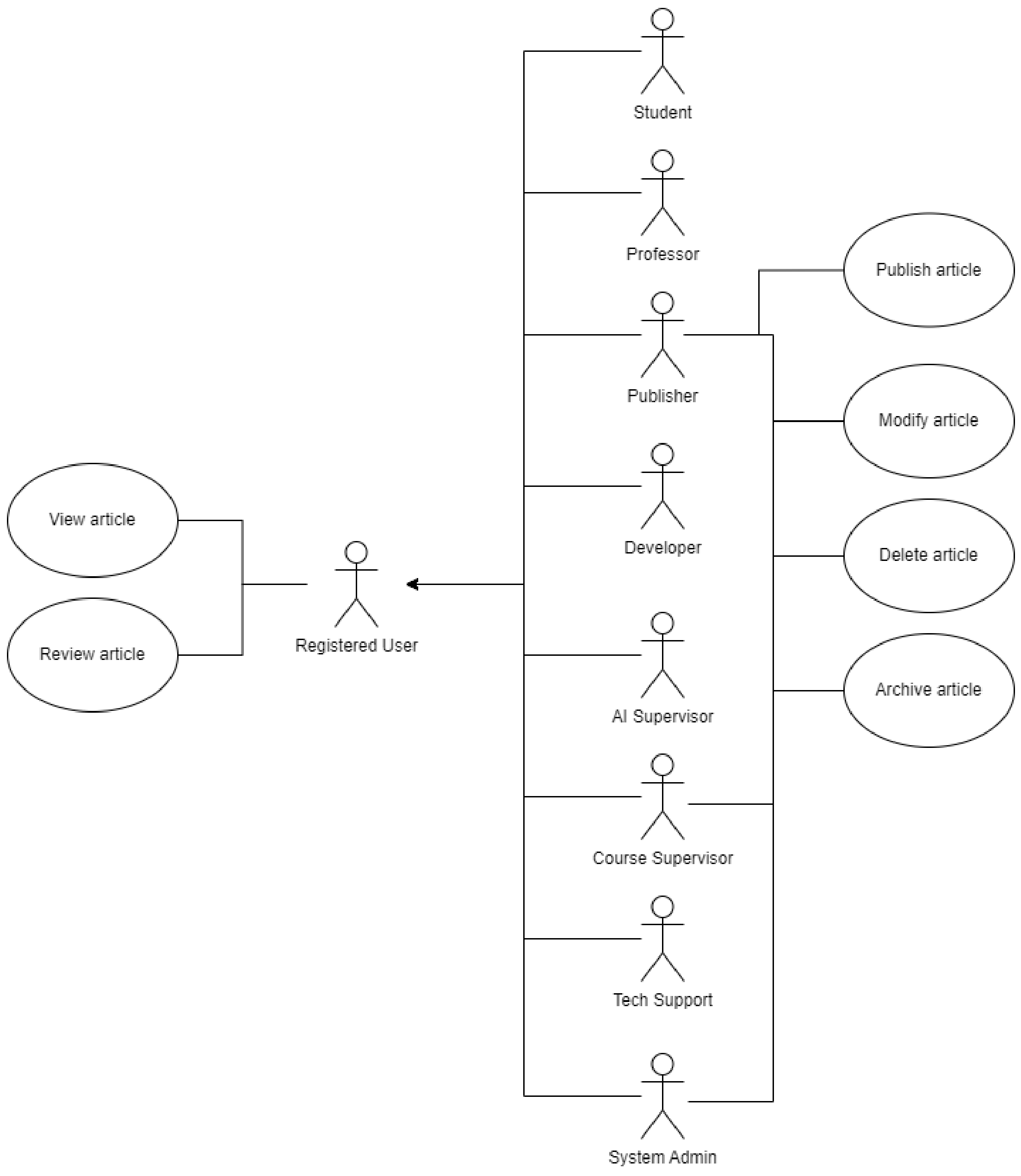


Figure 4: Use case diagram for article system

7.2.3 Chat

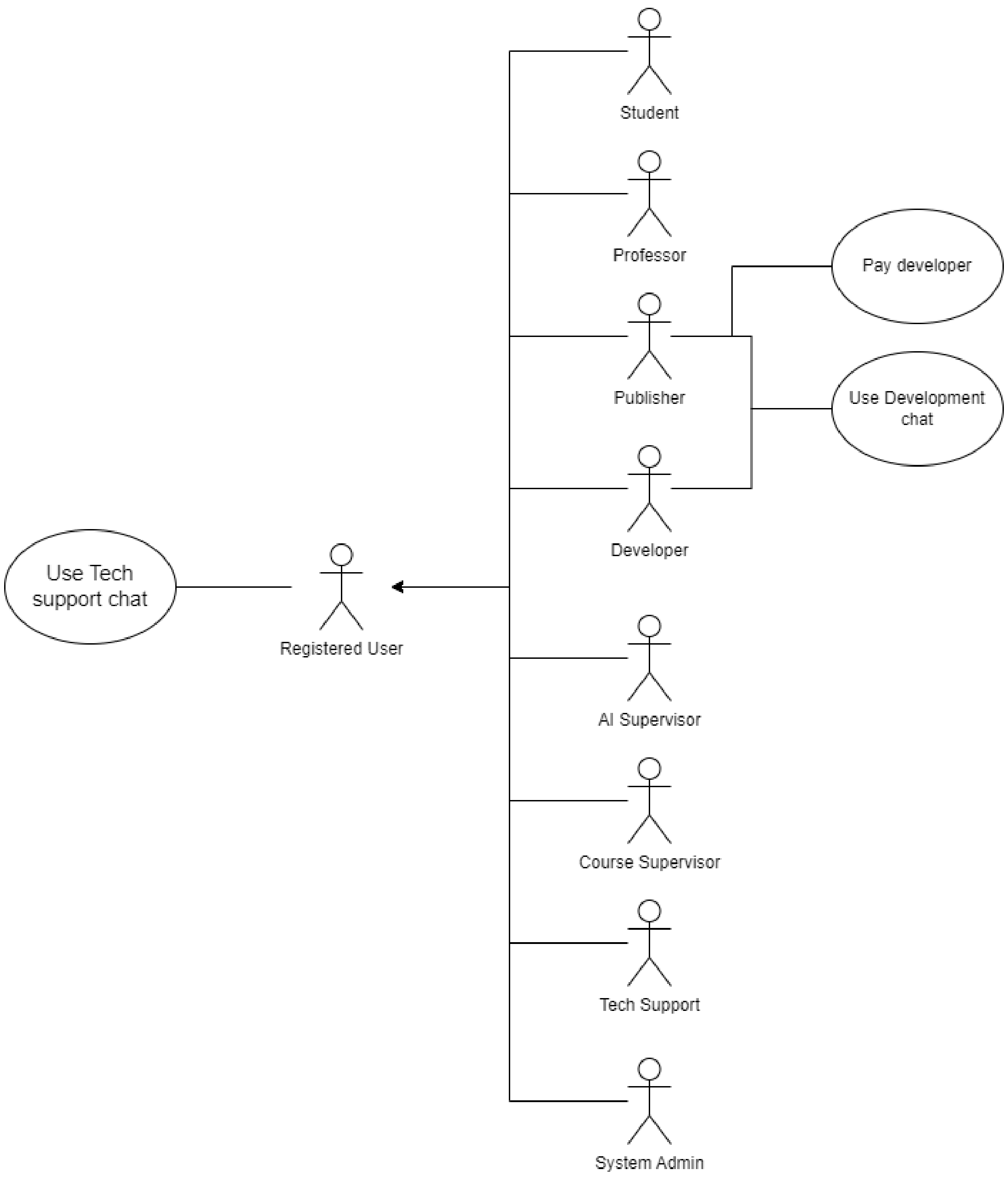


Figure 5: Use case diagram for chat system

7.2.4 Class

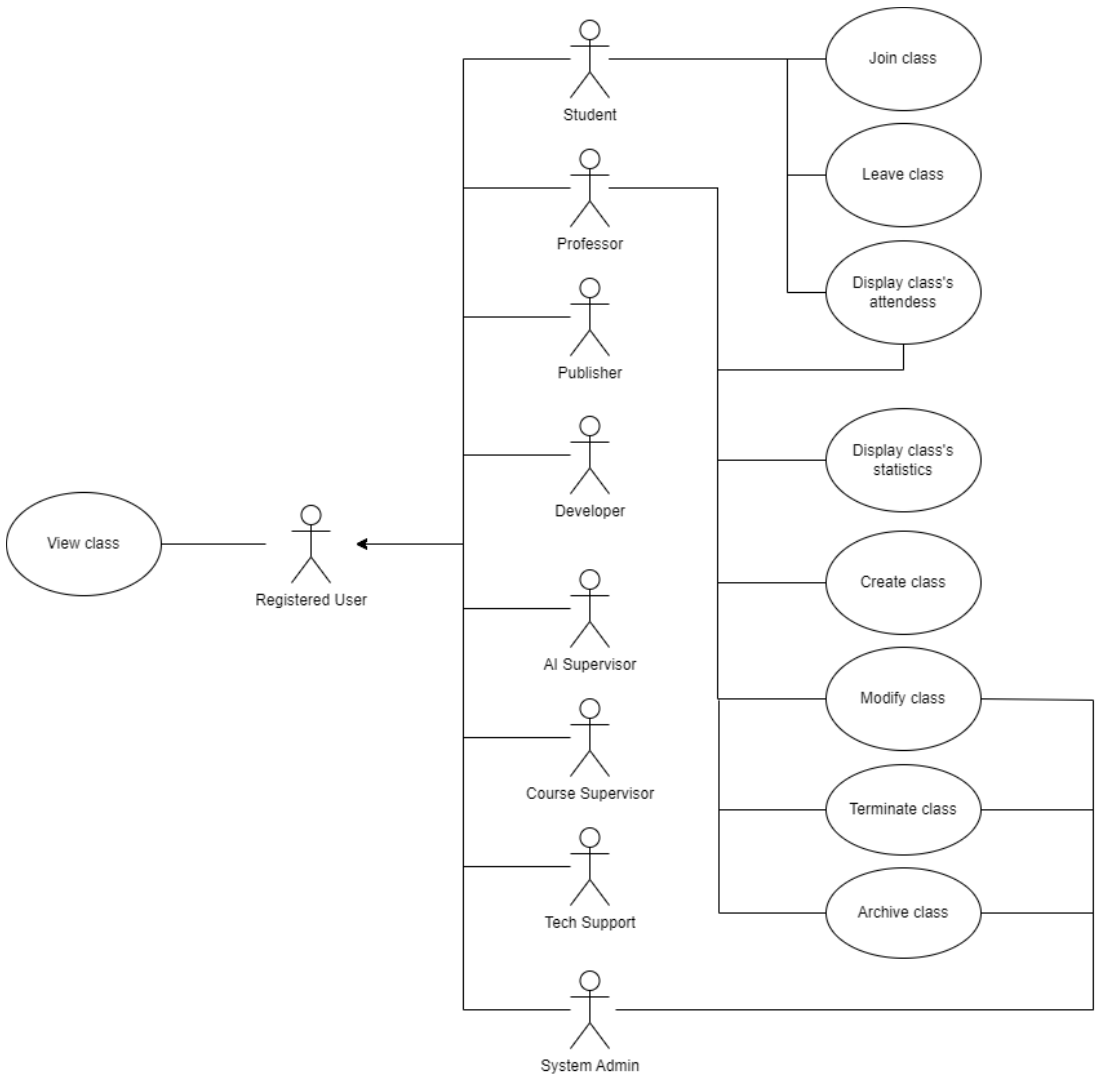


Figure 6: Use case diagram for class system

7.2.5 Course

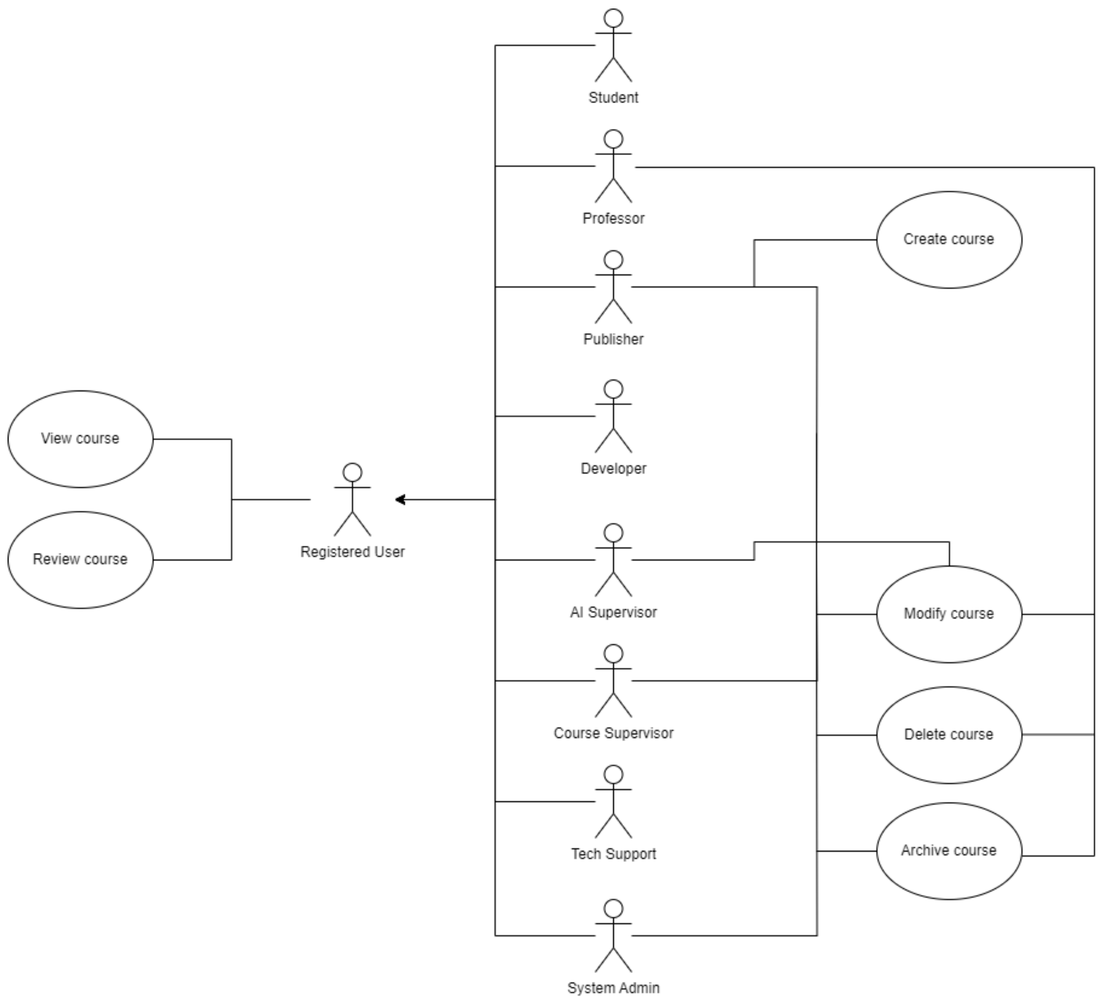


Figure 7: Use case diagram for course system

7.2.6 Minigame

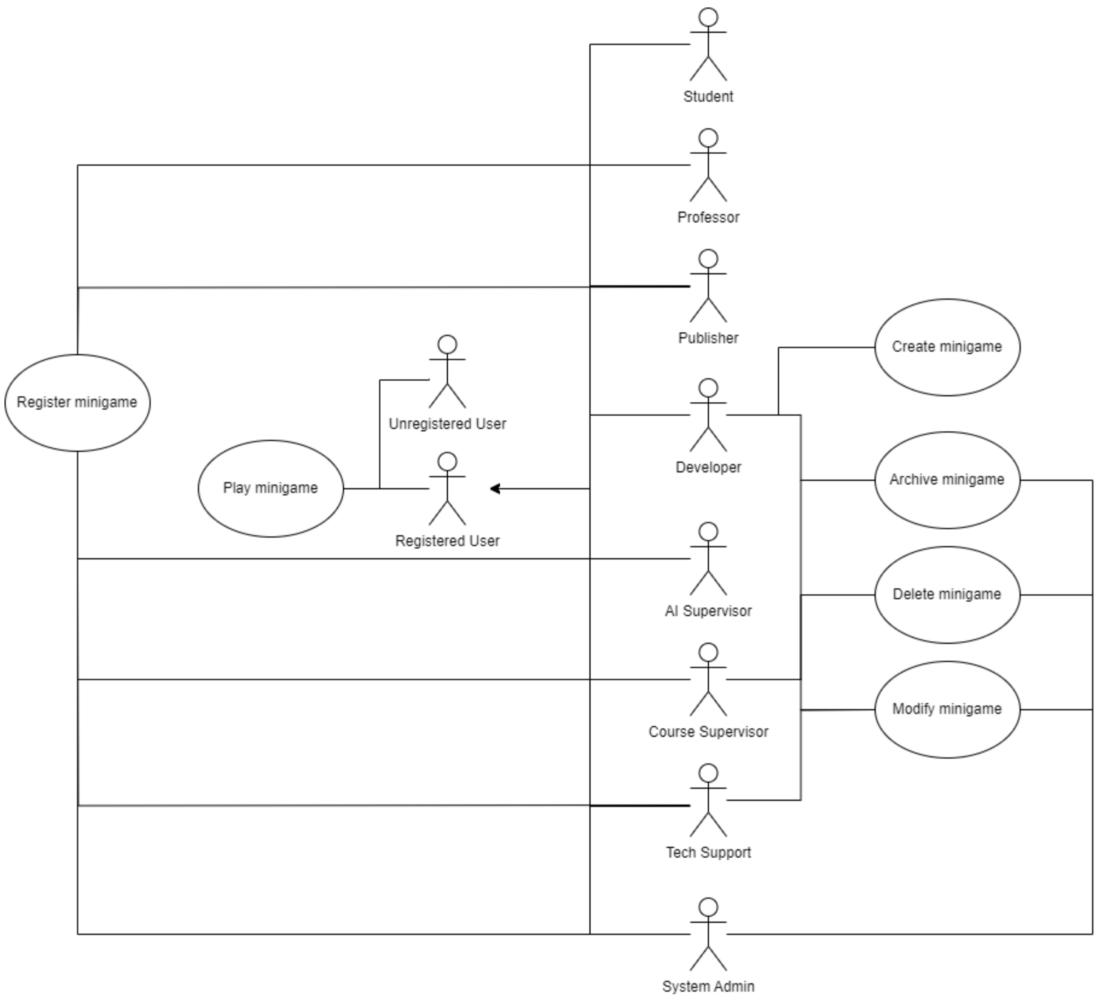


Figure 8: Use case diagram for minigame system

8 Context Diagram

We created a context diagram separating the general roles (registered and unregistered users), the specific roles (all the others), the subsystems (payment, authentication, credential recovery, LLM) and the peer system (database).

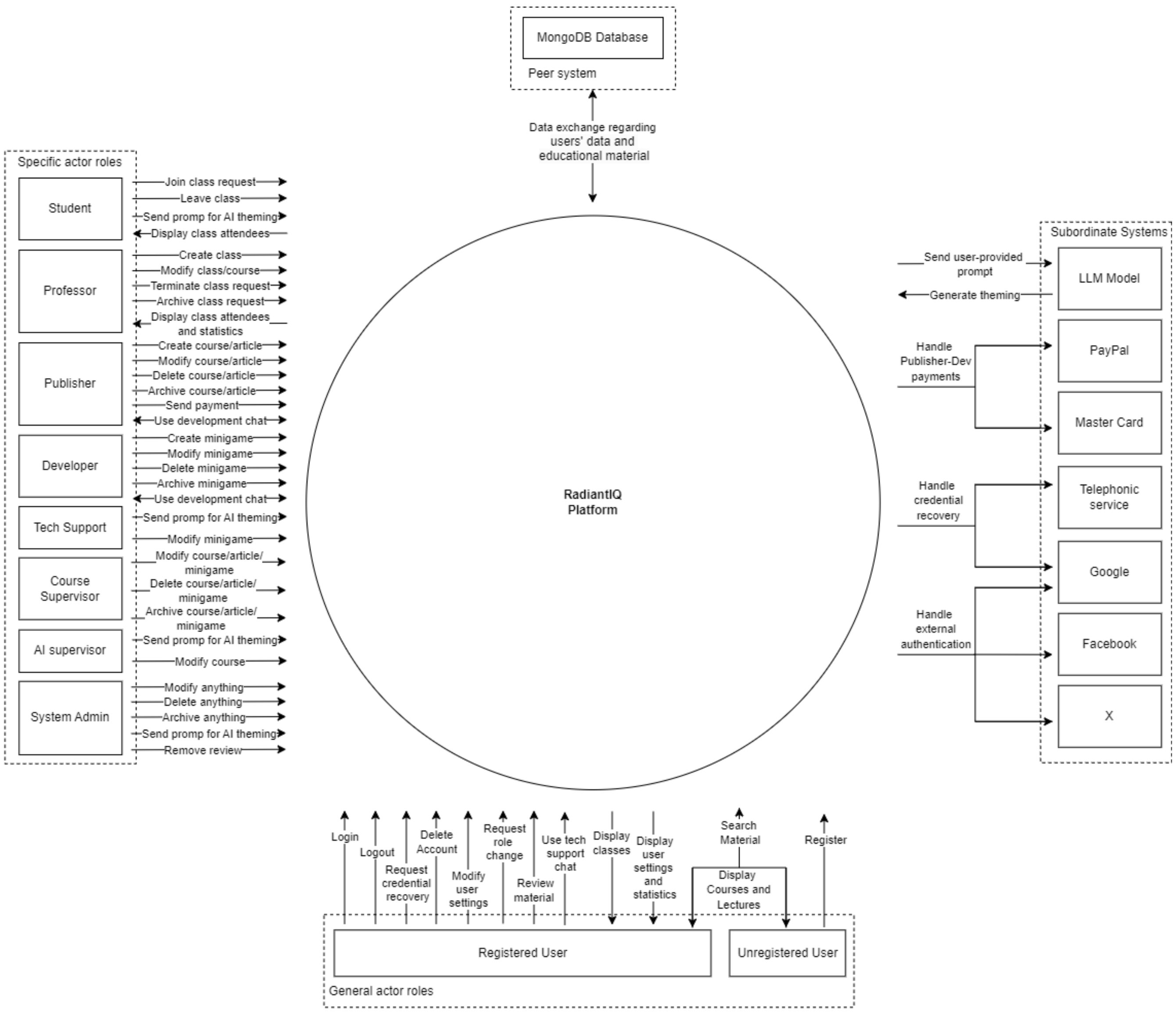


Figure 9: Context diagram

9 Components Diagram

A components diagram was create for better visualization about all the main components of the system.

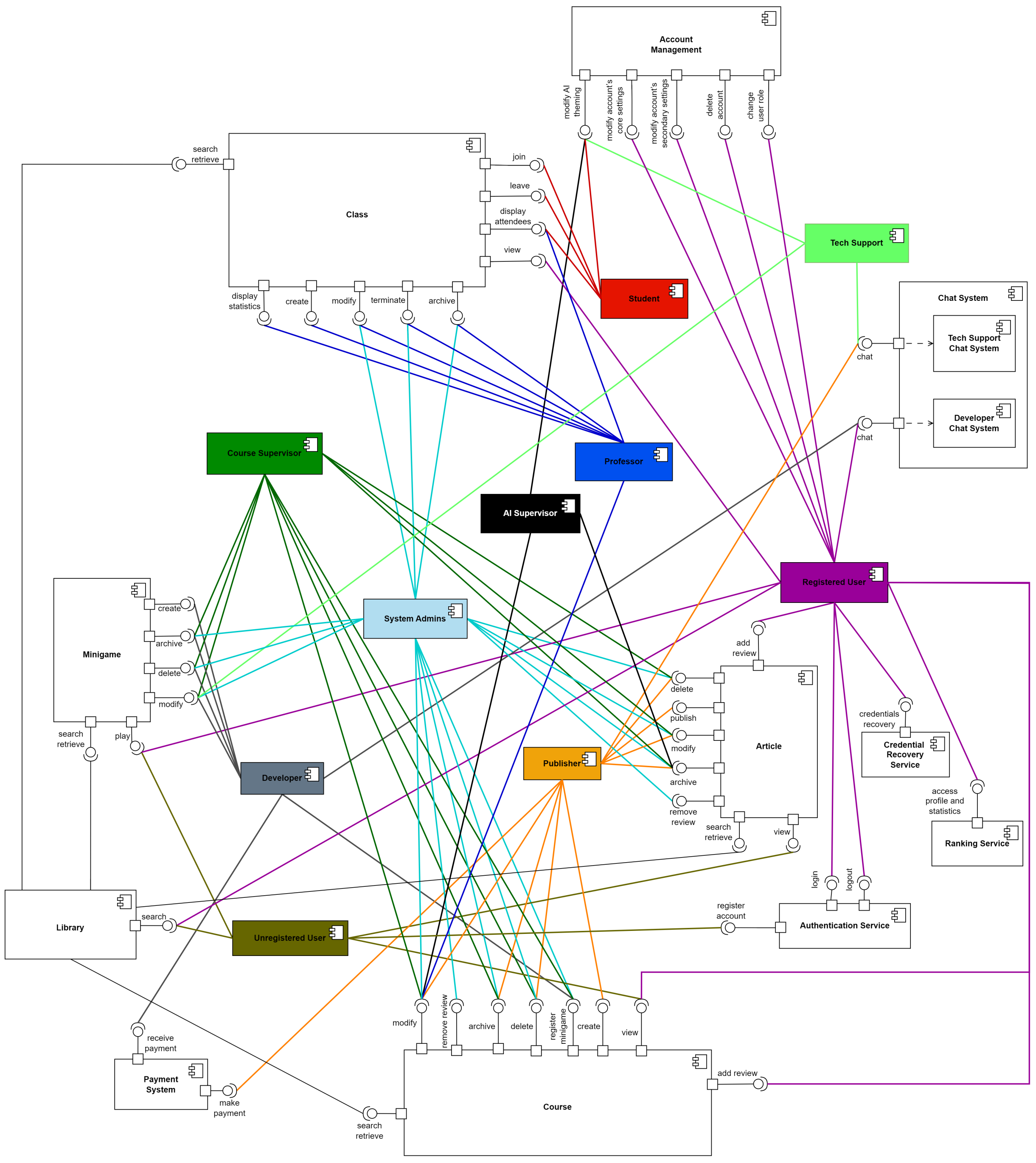


Figure 10: RadiantIQ - Components diagram

10 Class Diagram

The class diagram below in the UML describes the structure of RadianIQ's system by showing the main classes, their attributes, operations and their relationships.

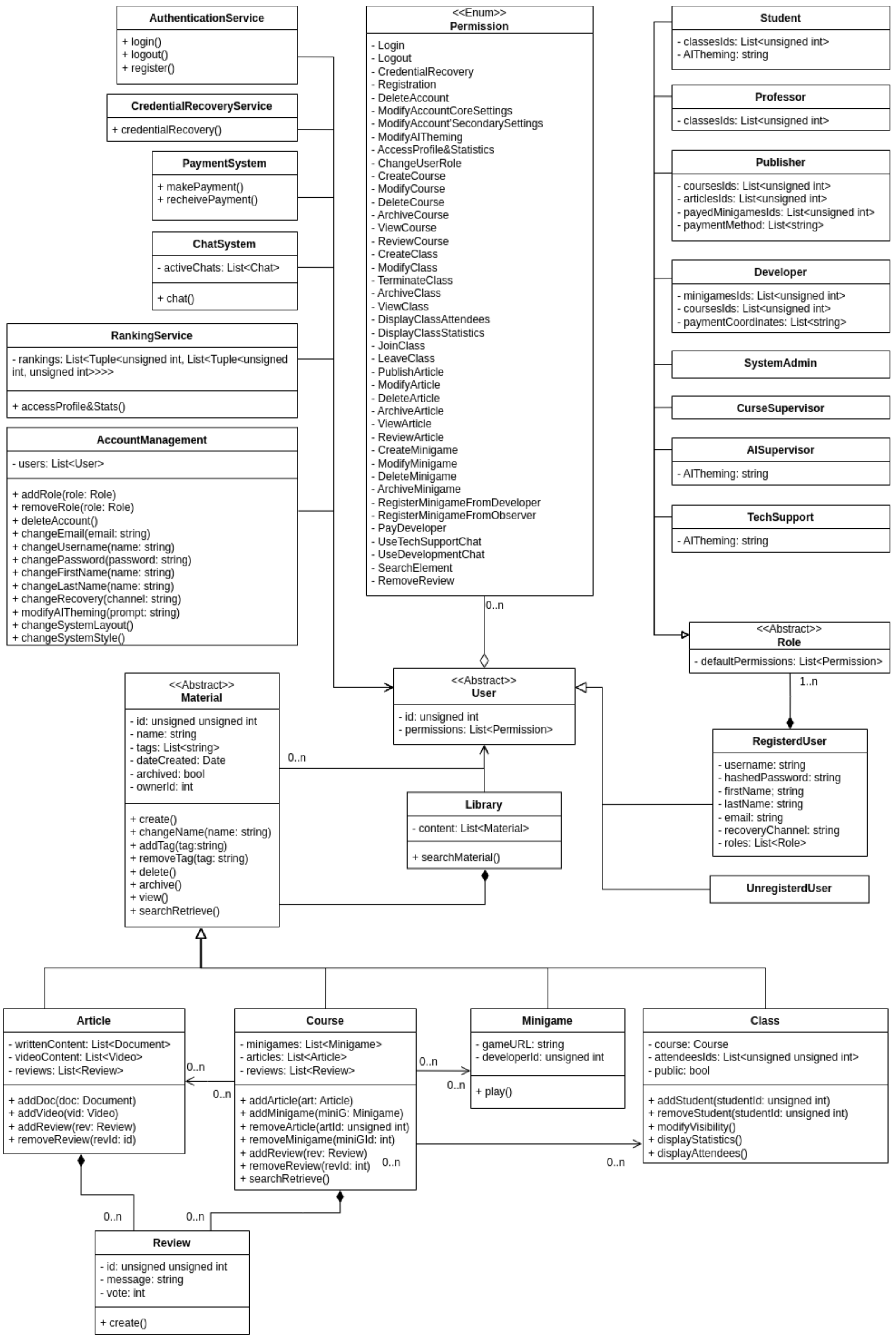


Figure 11: RadiantIQ - Class diagram

11 Conclusion

The project has made significant progress in generating ideas and documenting the Software Specification. This first milestone is meant to finalize Domain Analysis, Objectives, Actors, Function Requirement, Non-functional Requirements, Use cases, Context Diagram, Component Diagram and Class Diagram.

Some difficulties were encountered such as:

- Lack of modeling tools to keep the consistency of the Software Specification across all documents and diagrams, for example while changing things in the Use case table we also want to see the updates to the diagram without redrawing them manually.
- Finding edge cases and all necessary pieces of the system that are required for its creation and proper function.

Moving forward, the focus will be on:

- Reviewing these documents occasionally as we start an implementation to make sure everything stays aligned.
- Finding a modeling tool to keep the UML model up to date. The tool should support both directions of diagram-code transformation.
- Starting to choose and design the core components of the software.