Graduation Project

All In One Academic Hub

Prepared by:

Yassa Alqess Poktor Kamal	CS
Mohamed Ibrahim Elsayed ALI	CS
Abdullah Hakim Abd El-Raouf	CS
Mohamed Salah Abd El-Fatah	CS
Salma Khaled Mohamed Rashed	CS
Ahmed Ashraf Mohamed Mohamed	CS
Mohamed Ali Murad Amen	CS
Abd El-Rahman Ali	IS

Supervised by:

Dr. Ahmed Emad El-din

Co-Supervised by:

Eng. Mohamed Maged

2023-2024

Acknowledgment

Acknowledgment

Both scientifically and practically, we have put a lot of effort into our project. All of this wouldn't have been possible without Allah, Doctors, Assistants, and team members.

We extend sincere appreciation to **Prof. Dr. Nabila Mohamed Hassan**, the dean, for her unwavering support and leadership.

Special recognition goes to **Prof. Dr. Ahmed Emad El-din**, former dean, for the insightful contributions that significantly influenced our project's outcomes.

Gratitude is expressed **to Prof. Dr. Hussam Elbehiery**, our esteemed external supervisor, for dedicated guidance and invaluable insights.

Acknowledgment is conveyed to **Prof. Dr. Hany Girgis**, Nursing Medicine Co-Ordinator, for essential information and consistent support.

Profound thanks **to Eng. Mohamed Maged** for steadfast support and valuable contributions to the project's success.

Table of Contents

Contents

Abstract

Abstract

In response to the challenges faced by educational institutions in managing diverse aspects of the learning process, our application emerges as a strategic solution. Recognizing the need for a streamlined and integrated approach, we propose a comprehensive educational platform.

This platform addresses issues related to attendance tracking, examination management, and classroom collaboration. By providing professors with a versatile tool for generating diverse exam modules & versatile classroom features, we offer a solution that enhances the efficiency and effectiveness of educational delivery.

Additionally, our application fosters a constructive feedback loop, empowering students to share insights about their learning experiences. Moreover, the inclusion of a robust chat service facilitates real-time communication between students and educators, fostering a dynamic and interactive learning environment.

In essence, our solution addresses the complexities of educational administration and interaction, presenting a unified platform that optimizes various facets of the teaching and learning process.

List of Abbreviation

List of Abbreviations

- AIO: All in one
- UI: User Interface
- UX: User Experience
- HTML: Hypertext Markup Language
- CSS: Cascading Style Sheets
- JS: JavaScript
- TS: Typescript
- DFD: Data Flow Diagram
- ERD: Entity Relationship Diagram
- NFR: Non-Functional requirement
- CI/CD: Continuous Integration/Continuous Deployment
- SOW: Scope of work
- SPA: Single Page APPlication

List of Figures

List of Figures

1. ………………………… Introduction

1.1 Background and motivation:

As a student, I have personally experienced the challenges of managing my academic life effectively. Keeping track of student attendance, assignments, exams, and extracurricular activities can be overwhelming, and it can be difficult to stay organized and on top of everything. I have also witnessed the challenges faced by educators in tracking attendance and communicating with students.

Our motivation for developing our platform stems from a desire to address these challenges and provide a comprehensive solution that benefits both students and educators. I believe that technology can play a transformative role in enhancing teaching and learning experience, and I am passionate about creating innovative solutions that empower students and educators to succeed.

1.2 Problem statement:

We address the following key problems:

- Students struggle to manage their assignments and exams effectively, leading to stress and reduced productivity.
- Communication between students and instructors, as well as among peers, is often fragmented and inefficient, hindering collaboration and learning.
- Traditional attendance tracking methods are often manual and error-prone, leading to inaccurate records and administrative inefficiencies.
- Students and educators lack a centralized platform to access and share academic resources, feedback, and other relevant information.

1.3 Project objectives and solutions:

The primary objectives of our platform are to:

- Provide a user-friendly platform for students to manage their assignments, exams, and extracurricular activities.
- Facilitate effective communication between students and instructors, as well as among peers, through in-app messaging, and discussion forums.
- Implement accurate and reliable attendance tracking using advanced technologies such as Geo-Fencing technology.
- Create a central repository for academic resources, feedback, and other relevant information, accessible to both students and educators.

1.3.1 Features for student:

We offer a range of features tailored to the needs of students, including:

- Assignment and Exam Tracking: Students can access both assignments & exams and track their progress on assignments and exams.
- Resources Access: Students can access course materials and other academic resources.
- Communication Tools: Students can communicate with instructors and peers through in-app messaging and participate in discussion forums for each course.
- Attendance Tracking: Students can mark their attendance using Geo-Fencing or manual entry and receive alerts regarding their attendance status.
- Gradebook Integration: Students can view their grades and track their academic progress over time.
- Feedback and Voting: students can share insights about their learning experiences.

1.3.2 Feature for professor:

We also provide a range of features for professors, including:

- Communication Tools: Professors can send automated reminders for classes, exams, and assignments, send push notifications for important announcements, and communicate with students through in-app messaging.
- Attendance Tracking: Professors can monitor and record attendance for each class session, and view attendance reports for individual students or the entire class.
- Gradebook Integration: Professors can integrate their gradebook with the app to provide students with a consolidated view of their grades and academic progress.
- Feedback and Voting: Professors can collect feedback from students to enhance course materials and teaching methods, and conduct polls or surveys within the app.
- Manage Resources: Professor can upload materials, assignments, and other resources.
- Make and manage exams: Professor can both create exam manually or auto based on our algorithm and can assign this exam to specific range of memberships or let the algorithm to auto assign to memberships.

1.4 Stakeholders:

End Users:

- Administrator
- Professor
- Lab instructor
- Student

2.

Literature Review

2.1 Overview

This section delves into an in-depth exploration of the technologies employed in the project, offering a detailed insight into the pivotal components. Additionally, it provides essential information surrounding the project's vision, ensuring a comprehensive understanding of the overarching goals and aspirations.

2.1.1 Tools & Technologies

Mobile App Development

- React Native
- Expo framework

Front-End Development

- HTML
- CSS
- Tailwind
- ReactJS [TypeScript]
- Vite

Back-End Development

- NodeJS [TypeScript]
- .Net core

Caching

- Redis

Communication & messaging system

- Kafka

Database provider

- Postgres

Containerization and Artifact Hosting

- Docker & Docker Hub

Source Code Management

- Git

Source Code Hosting

- GitHub

Continuous Integration/Continuous Deployment (CI/CD)

- GitHub Actions

Cloud provider

- Azure

Message service

- Twilio

3-rd party tools:

- Vs code
- Discord web hooks
- Husky hooks

Testing:

- Postman
- Swagger
- JMeter
- Jest

Logging & monitoring:

- Winston

2.1.2 Front-End development

2.2.1.1 HTML (Hypertext Markup Language):

Purpose: HTML is the standard markup language used to structure content on the web. It defines the basic structure of your web page using elements such as headings, paragraphs, lists, links, and more.

2.2.1.2 CSS (Cascading Style Sheets):

Purpose: CSS is used for styling HTML elements, enhancing the visual presentation of your web pages. It allows you to define the layout, colors, fonts, and other visual aspects, ensuring a cohesive and appealing design.

2.2.1.3 TypeScript:

Purpose: TypeScript is a superset of JavaScript that adds static typing to the language. It enhances code maintainability and catches errors during development. With TypeScript, you can write more robust and scalable code for your front-end application.

2.2.1.4 Tailwind:

Purpose: Tailwind CSS is a utility-first CSS framework that provides a low-level set of reusable utility classes for building custom user interfaces. It promotes a modular and composable approach to styling, enabling rapid UI development and consistent, responsive designs. Tailwind CSS offers highly customizable utilities, responsive utilities out-of-the-box, and a focus on developer productivity.

2.2.1.5 Mobile Application:

React Native:

Purpose: React Native is a framework for building cross-platform mobile applications (SPA) using React and JavaScript. It allows you to use a single codebase to develop apps for both iOS and Android platforms. React Native facilitates faster development and maintains a native-like performance.

2.1.3 Back-End development

Node.js:

Purpose: Node.js is a server-side JavaScript runtime. It enables the execution of JavaScript code on the server, allowing you to build scalable and high-performance web applications. It's known for its event-driven, non-blocking I/O model, making it suitable for handling concurrent requests.

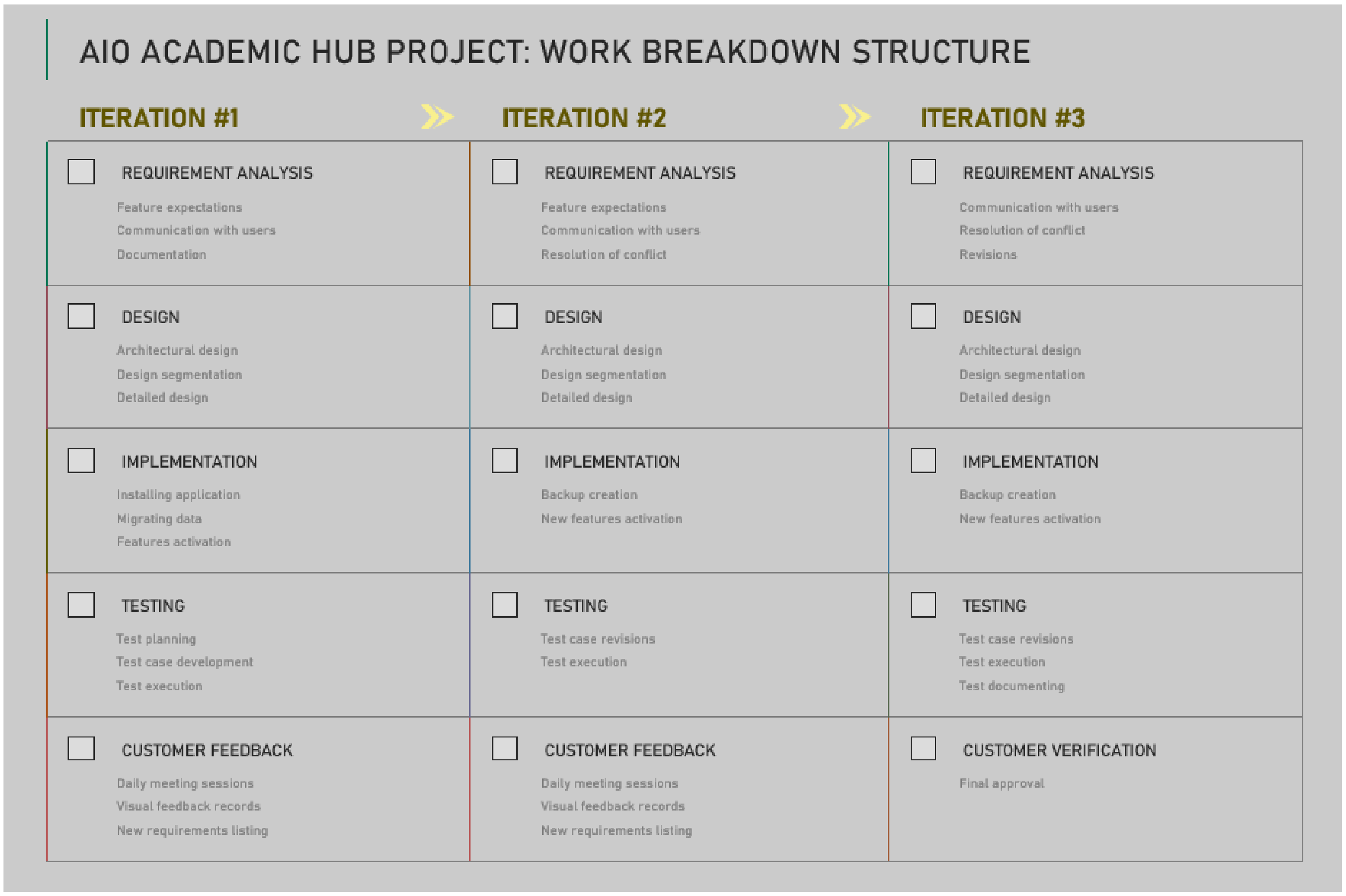
.NET Core:

Purpose: .NET Core is a cross-platform, open-source framework for building modern, cloud-based, and scalable applications. It supports multiple programming languages and is particularly well-suited for building microservices. It provides a robust infrastructure for developing and deploying scalable back-end services.

Microservices Architecture:

Purpose: Microservices architecture involves developing a software system as a collection of small, independent services that communicate with each other through well-defined APIs. This approach enhances scalability, maintainability, and flexibility. Using Node.js and .NET Core as microservices allows you to develop, deploy, and scale individual components independently, promoting a modular and distributed system.

2.1.5 Scope of work (SOW)



AIO ACADEMIC HUB PROJECT: WORK BREAKDOWN STRUCTURE

ITERATION #1	ITERATION #2	ITERATION #3
☐ REQUIREMENT ANALYSIS Feature expectations Communication with users Documentation	☐ REQUIREMENT ANALYSIS Feature expectations Communication with users Resolution of conflict	☐ REQUIREMENT ANALYSIS Communication with users Resolution of conflict Revisions
☐ DESIGN Architectural design Design segmentation Detailed design	☐ DESIGN Architectural design Design segmentation Detailed design	☐ DESIGN Architectural design Design segmentation Detailed design
☐ IMPLEMENTATION Installing application Migrating data Features activation	☐ IMPLEMENTATION Backup creation New features activation	☐ IMPLEMENTATION Backup creation New features activation
☐ TESTING Test planning Test case development Test execution	☐ TESTING Test case revisions Test execution	☐ TESTING Test case revisions Test execution Test documenting
☐ CUSTOMER FEEDBACK Daily meeting sessions Visual feedback records New requirements listing	☐ CUSTOMER FEEDBACK Daily meeting sessions Visual feedback records New requirements listing	☐ CUSTOMER VERIFICATION Final approval

2.2 System Components

- Authentication/Authorization (Identity Module):

Establish a secure identity module with robust authentication and authorization mechanisms to safeguard access, tailored for students and instructors.

- Repository Layer (Database and Caching Module):

Implement a resilient repository layer targeting the database and caching system, ensuring efficient storage and retrieval of student and instructor data while optimizing performance.

- Notification Service:

Develop a notification service that ensures timely communication of important announcements and updates to students and instructors.

- Communication Service:

Create a communication service to integrate with email or messaging tools, fostering effective and seamless communication among students and instructors.

- Dashboard:

Design an intuitive and user-friendly dashboard providing students and instructors with quick access to essential information.

- Attendance Module:

Build a comprehensive attendance module, incorporating features like class attendance tracking for students and instructors to ensure accurate records.

- Class Work Module:

Develop a module to manage and organize class-related activities, tailored for students and instructors, ensuring a streamlined process for recording, and tracking coursework and assignments.

- Exam Module:

Implement an exam module that facilitates the scheduling, recording, and management of student examinations, providing a structured approach to assessment.

2.3 The scalability and performance

Scalability and Performance Optimization:

Implement scalable architecture and performance optimization strategies to accommodate the growth of students and instructors, leveraging database sharding, load balancing, and caching mechanisms to ensure efficient system responsiveness and resource utilization.

2.4 Conclusion

1. The secure identity module ensures a tailored and protected access environment, while the resilient repository layer, incorporating database and caching mechanisms, optimizes data storage and retrieval for improved performance. The notification and communication services foster seamless interaction, keeping students and instructors informed. The intuitive dashboard provides quick access to vital information, and the attendance, class work, and exam modules offer comprehensive tools for efficient record-keeping and assessment. Moreover, the system's scalability and performance optimization, incorporating strategies like database sharding and load balancing, ensure adaptability to the growing needs of the educational community, promising an agile and responsive platform for sustained success.

3. Analysis and design

3.1 Overview

This chapter encompasses a thorough examination of the system, focusing on both analysis and design aspects. It elucidates the system's functionality and non-functional requirements, presenting a comprehensive overview through the exploration of Use Case Diagrams, Sequence Diagrams, Data Flow Diagrams (DFD), Entity Relationship Diagrams (ERD), and other pertinent components.

3.1.1 Requirements and analysis

Agile methodology is a project management and software development approach that prioritizes flexibility, collaboration, and customer satisfaction.
In Agile sprints, the development process is divided into time-boxed iterations (sprints) with specific objectives:

Planning:

- Define and prioritize tasks for the sprint.
- Establish a sprint goal and expected deliverables.

Design:

- Make design decisions, including architecture and user interface.
- Create a blueprint for development.

Development:

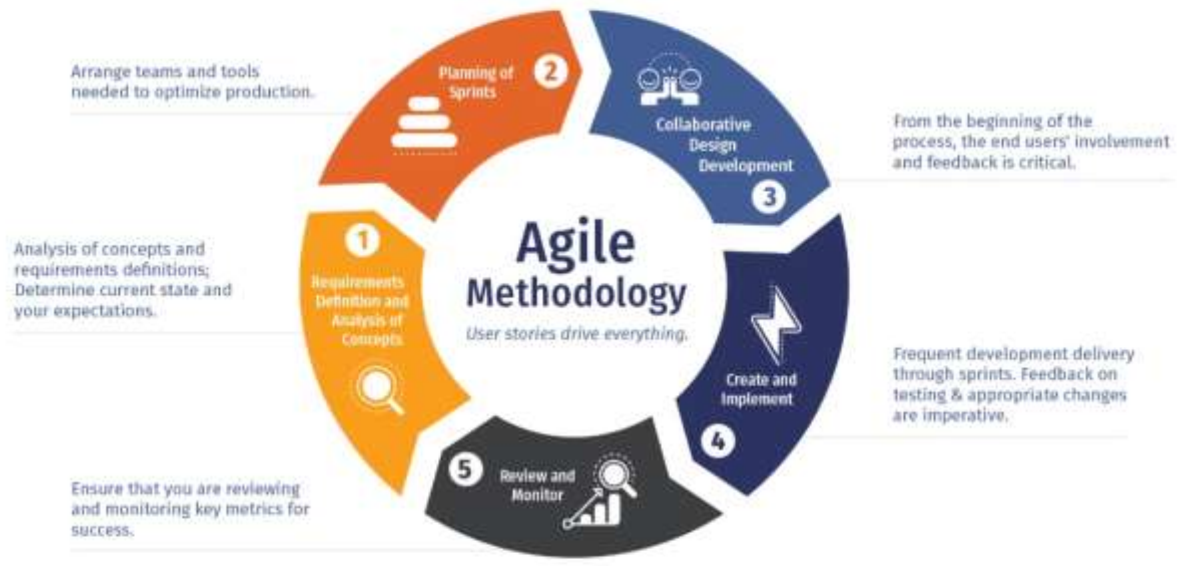
- Implement features incrementally.
- Ensure continuous integration for cohesive coding.

Testing:

- Conduct unit testing by developers.
- Perform integration testing to ensure module compatibility.
- Utilize automated testing for efficiency.

Evaluation & integration:

- Conduct a sprint review to showcase completed work.
- Gather feedback for adjustments.
- Reflect on the sprint in a retrospective meeting for improvements.
- This iterative cycle allows for adaptability, transparency, and the continuous delivery of value to stakeholders.


Agile Methodology
User stories drive everything.
1 Requirements Definition and Analysis of Concepts
Analysis of concepts and requirements definitions; Determine current state and your expectations.
2 Planning of Sprints
Arrange teams and tools needed to optimize production.
3 Collaborative Design Development
From the beginning of the process, the end users' involvement and feedback is critical.
4 Create and Implement
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.
5 Review and Monitor
Ensure that you are reviewing and monitoring key metrics for success.

3.1.2 Functional requirements

A functional requirement (FR) document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users, and the typeof system where the software is used.
Functional requirements as we mentioned above in chapter 1.

3.1.3 Non-functional requirements

A Non-Functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of the system, rather than specific behaviors. Non-functional requirements are often. Called “quality attributes” of a system.
It’s a statement of what a product is or how it will be constructed, or a constraint on how the product will be designed or will behave.

❖ Nonfunctional requirements such as:

- **Performance**:

  Requirement: The system must handle a concurrent user load of [specify the expected number] without significant performance degradation.

  Requirement: All critical transactions, such as authentication and data retrieval, must have a response time of less than [specify the acceptable time].

- **Scalability**:

  Requirement: The system should be easily scalable to accommodate an increase in the number of students, instructors, and concurrent users.

  Requirement: Scalability should be achieved through strategies like database sharding and load balancing.

- **Reliability**:

  Requirement: The system should have a high level of reliability, with an uptime of at least 99%.

  Requirement: Implement automatic backup and recovery mechanisms to ensure data integrity and availability.

- **Security**:

  Requirement: All data transmission should be encrypted using secure protocols (e.g., HTTPS) to ensure the confidentiality and integrity of information.

  Requirement: The system must enforce strong password policies and implement secure authentication mechanisms.

- **Usability**:

  Requirement: The user interface must be intuitive and user-friendly, requiring minimal training for students, instructors, and administrators.

  Requirement: The system should support accessibility standards to ensure usability for users with disabilities.

- **Availability**:

Requirement: The system should be available 24/7, with scheduled maintenance windows communicated in advance.

Requirement: Implement a failover mechanism to ensure continuous service in the event of server or component failures.

- **Scalability and Performance Optimization**:

Requirement: Regularly monitor system performance and optimize database queries, ensuring efficient resource utilization.

Requirement: Periodic scalability tests should be conducted to assess the system's ability to handle increased loads.

- **Interoperability**:

Requirement: The system should support integration with external systems and services, such as external databases or third-party tools.

Requirement: APIs should be well-documented and conform to industry standards for interoperability.

3.2 System Models

3.2.1 The architecture design

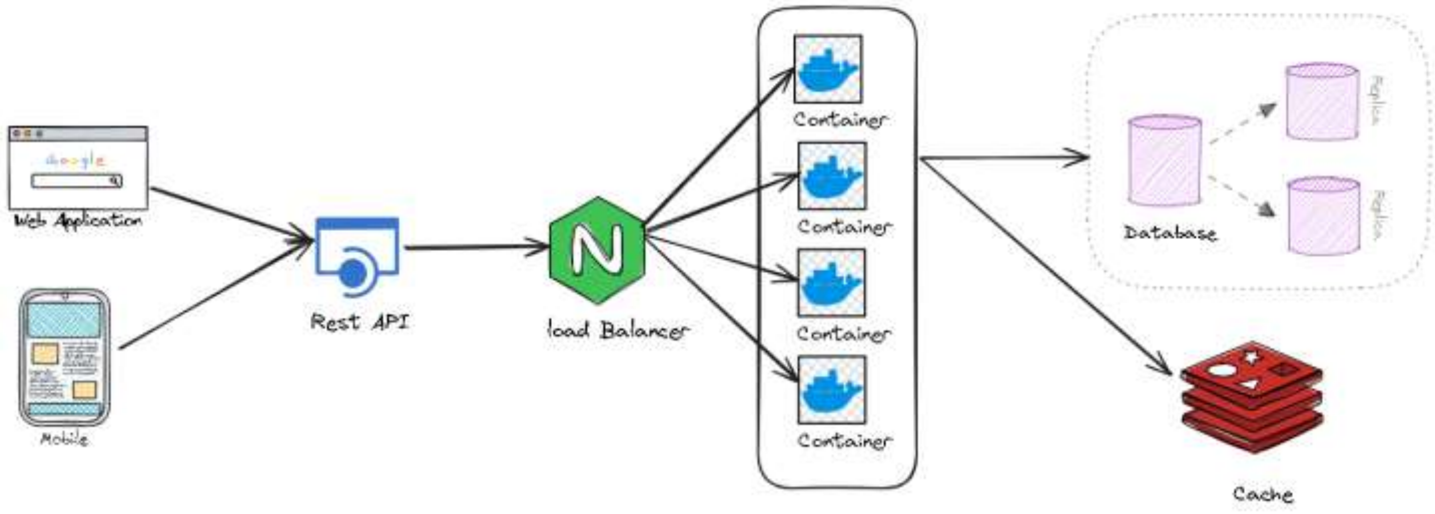


Figure 1 The architecture design

3.2.2 The levels of DF Diagram

The meaning of Data Flow Diagram:

- a graphical representation that illustrates how data flows within a system. It is a visual tool used in the analysis and design phase of system development to depict the movement of data between processes, data stores, and external entities.

3.2.2.1 The Level Zero(0) or in another word Context

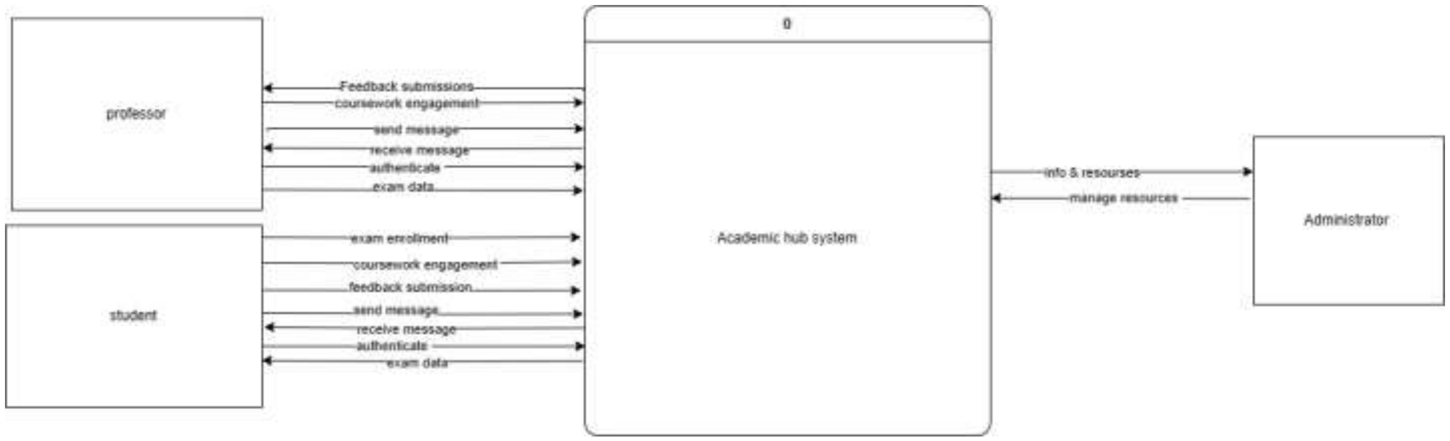


Figure 2 The Level Zero(0) or in another word Context

3.2.2.2 The level one(1)

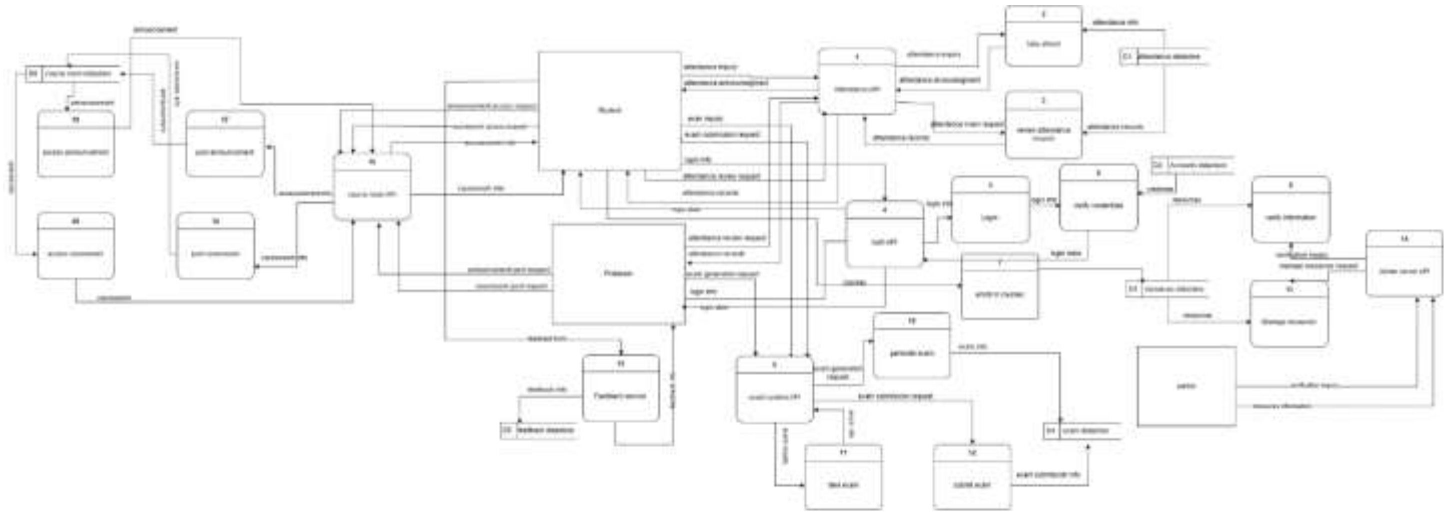


Figure 3 The level one(1)

3.2.2.3 The level two(2)

3.2.2.3.1 Attendance Model

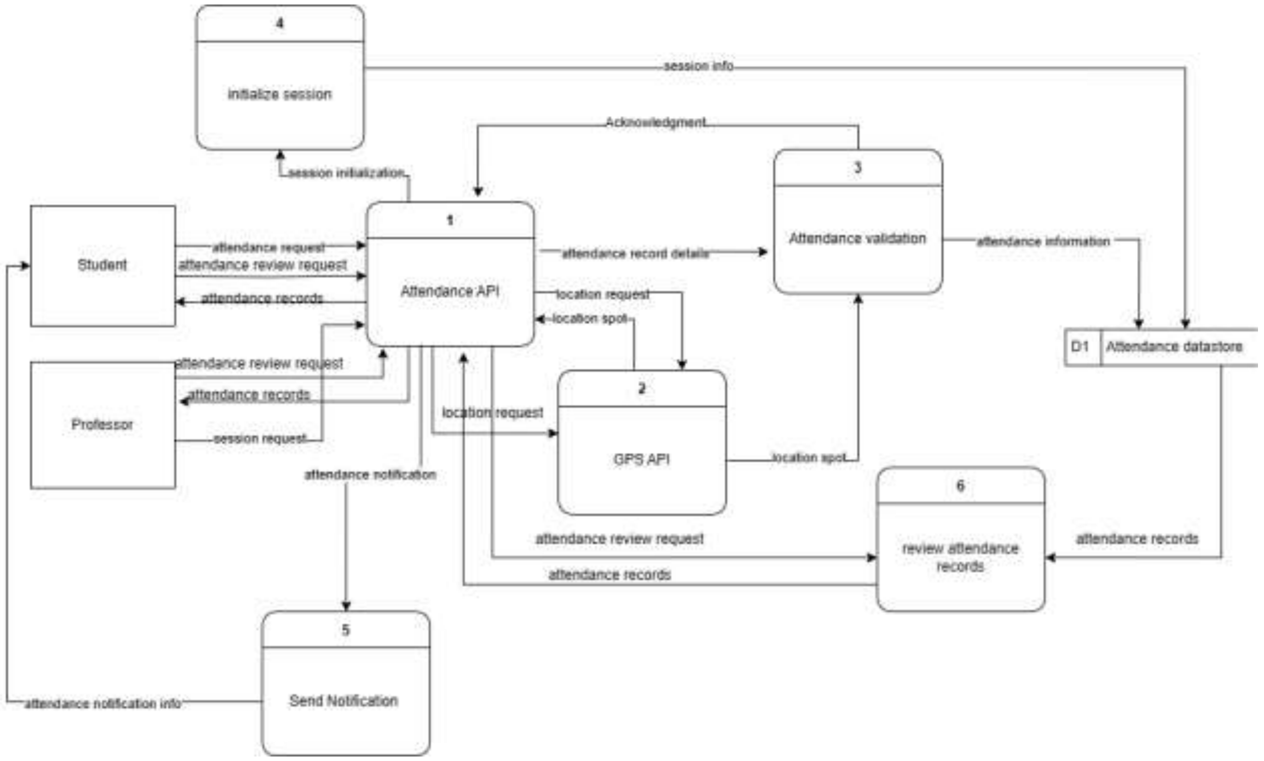


Figure 4 The level two(2)Attendance Model

3.2.2.3.2 Exam module

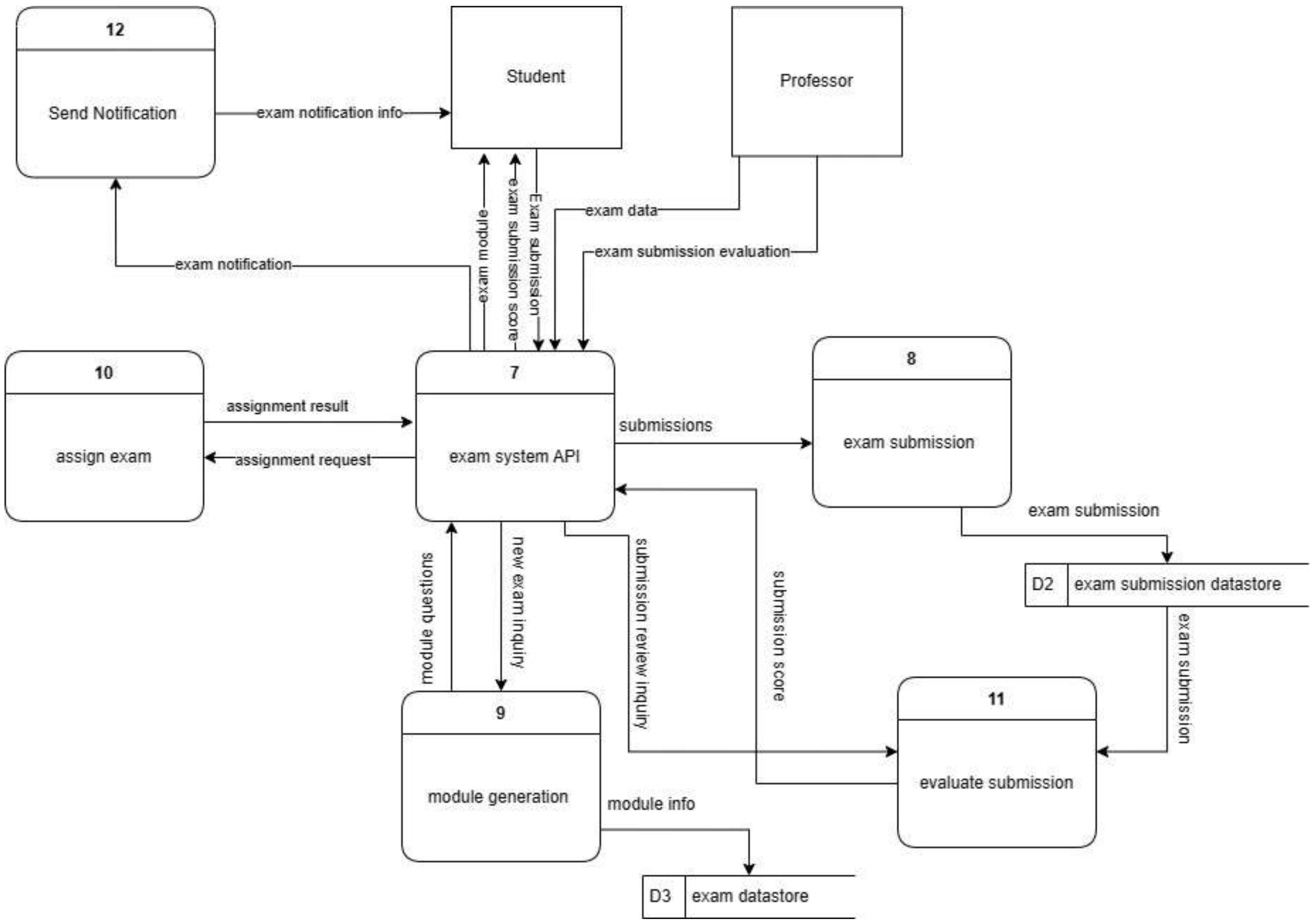


Figure 5 The level two(2) Exam module

3.2.2.3.3 Communication Module

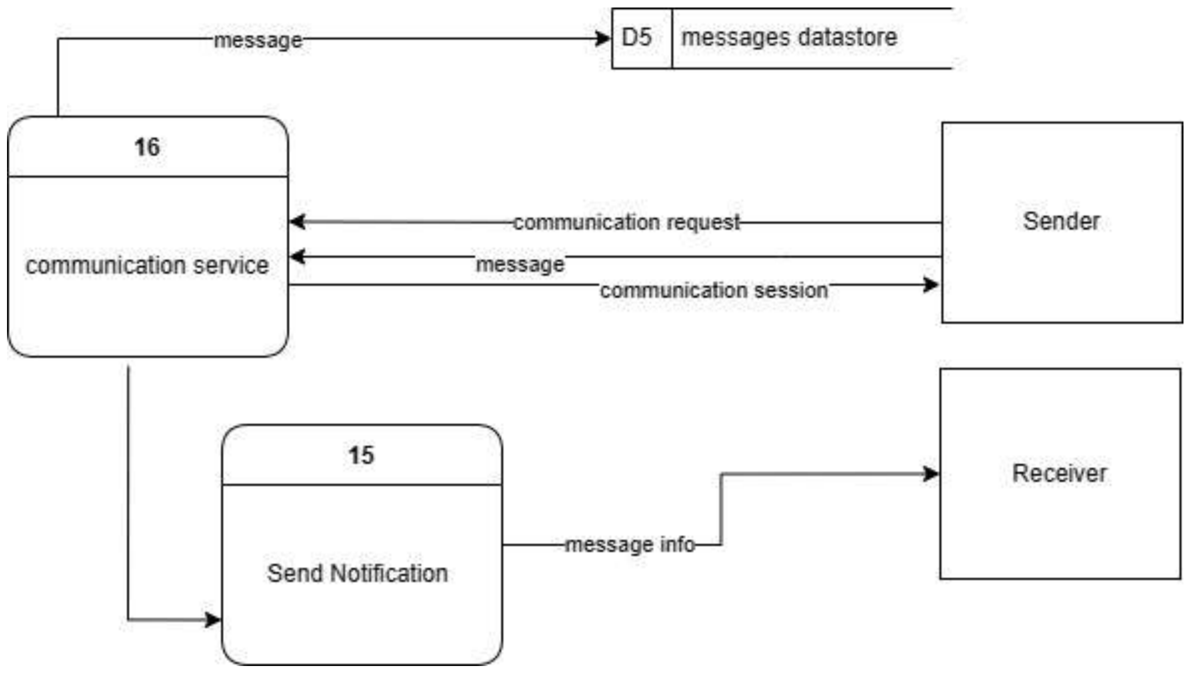


Figure 6 The level two(2) Communication Module

3.2.2.3.4 Feedback Module

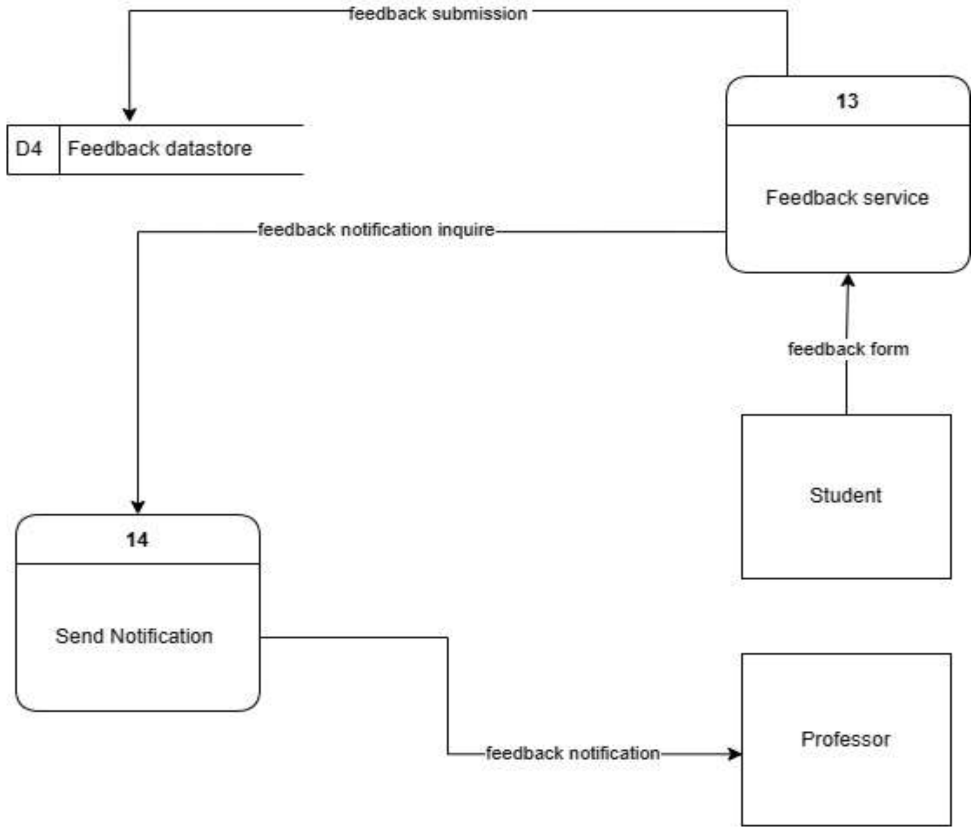


Figure 7 The level two(2) Feedback Module

3.2.2.3.5 CourseRoom Module

For student

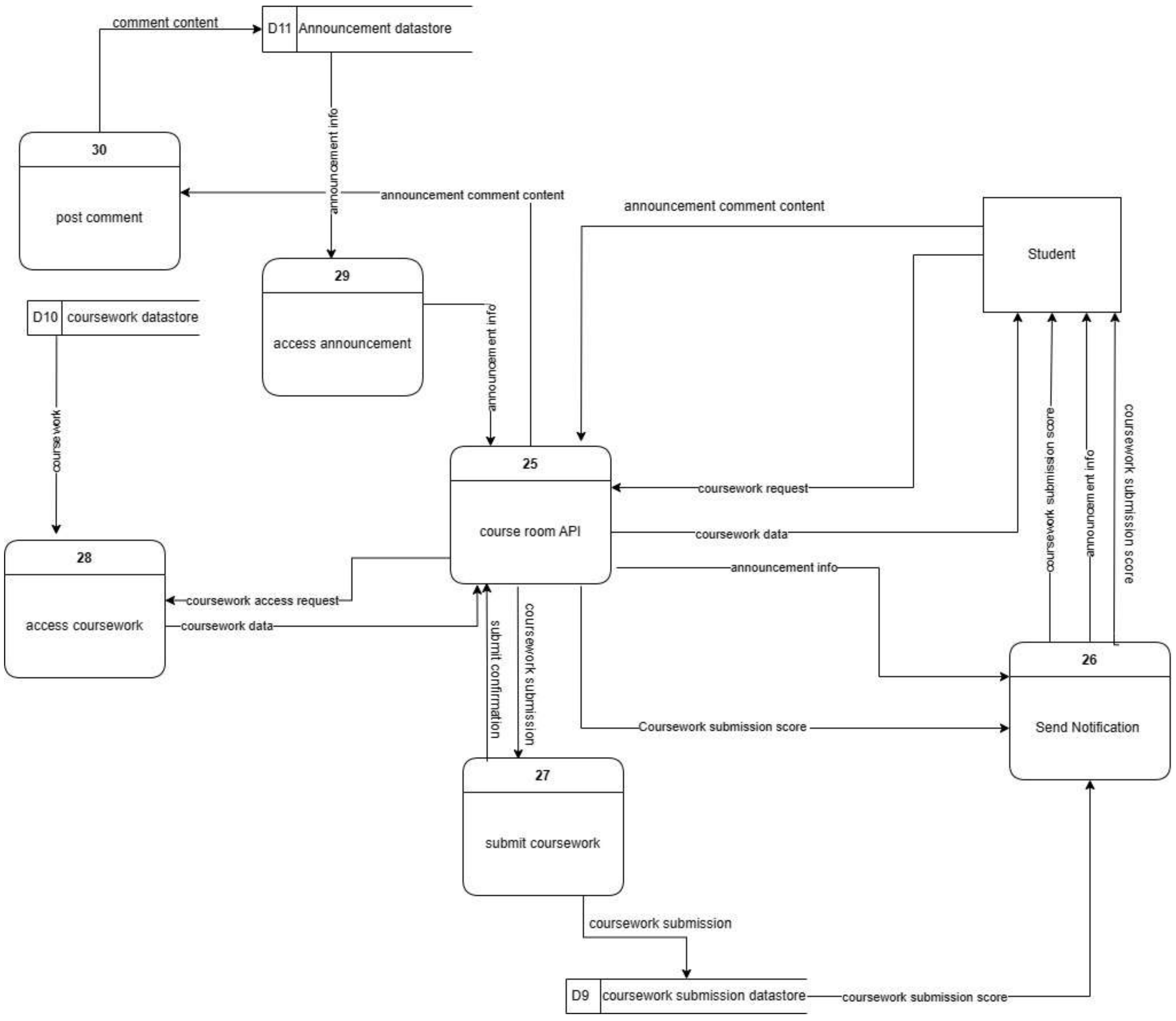


Figure 8 The level two(2) CourseRoom Module For student

For professors

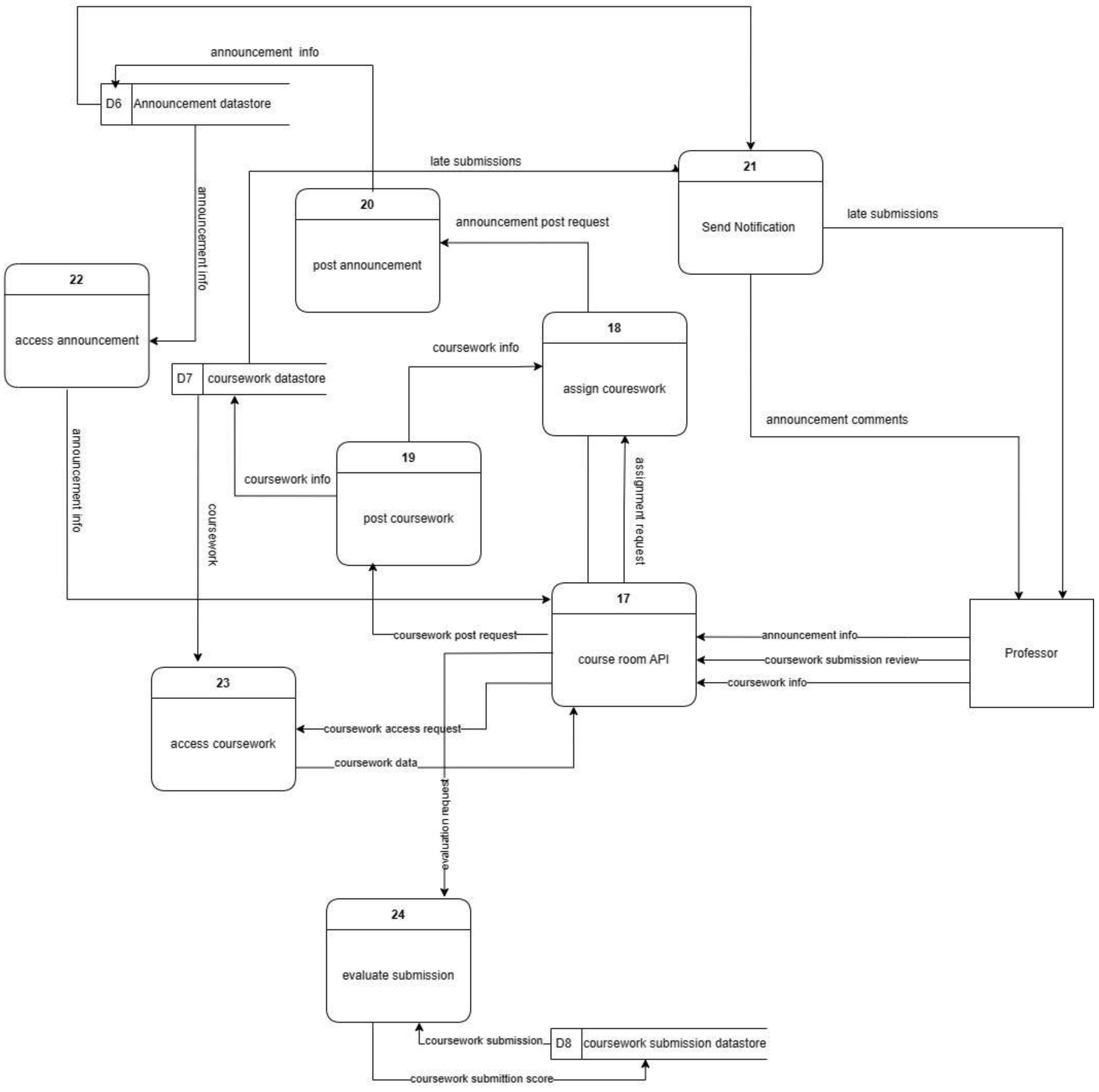


Figure 9 The level two(2) CourseRoom Module For professors

3.2.2.3.6 Identity & administrator Module

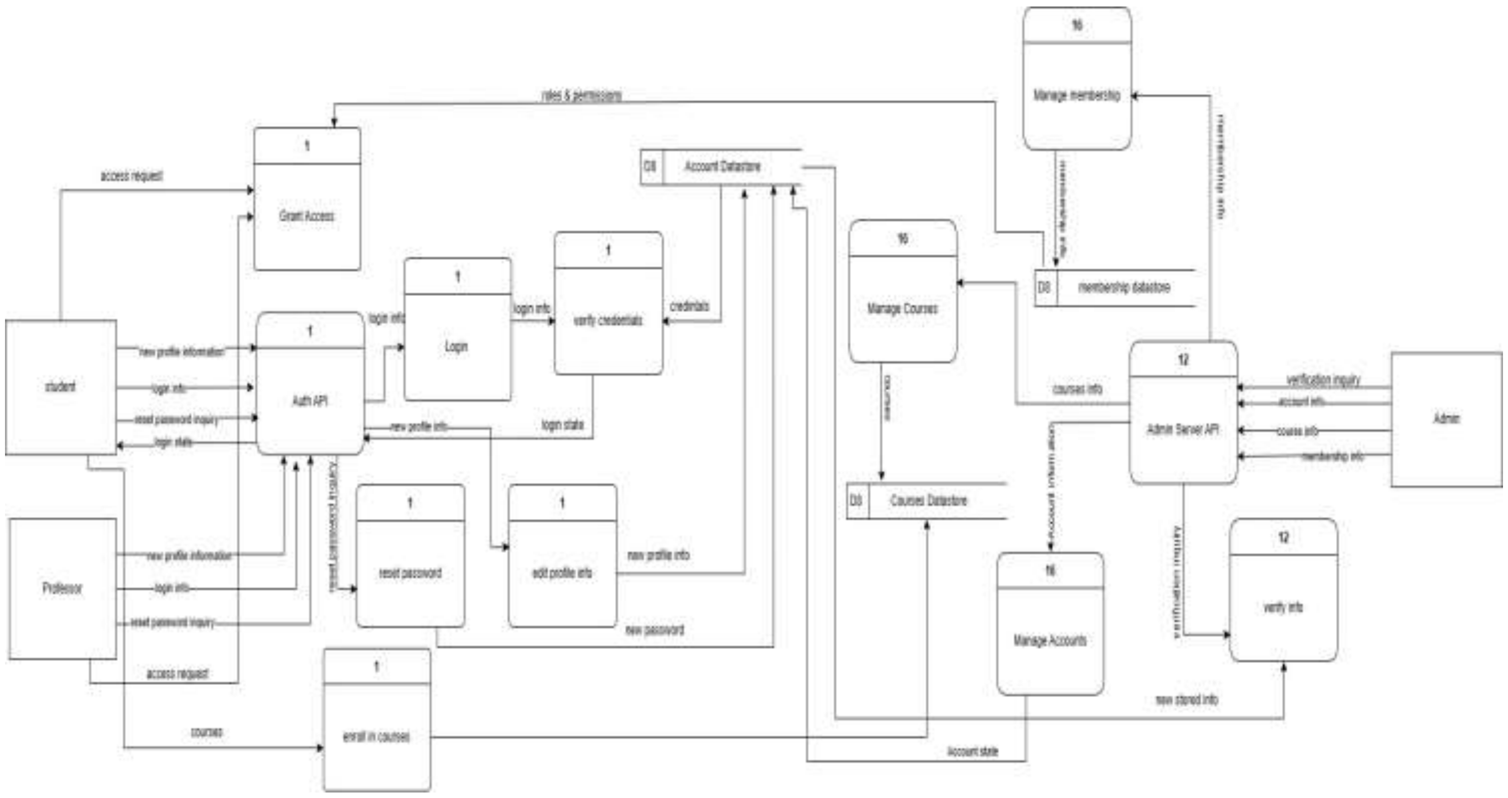


Figure 10 The level two(2) Identity & administrator Module

3.2.3 The use case Diagram

The meaning of UseCase Diagram:

-is type of diagram that illustrates the interactions between users (actors) and a system, focusing on the ways in which the system responds to external stimuli. Use Case Diagrams provide a high-level view of a system's functionality by capturing the various ways users might interact with it and the specific functionalities the system will provide in response.

3.2.3.1 use case in review

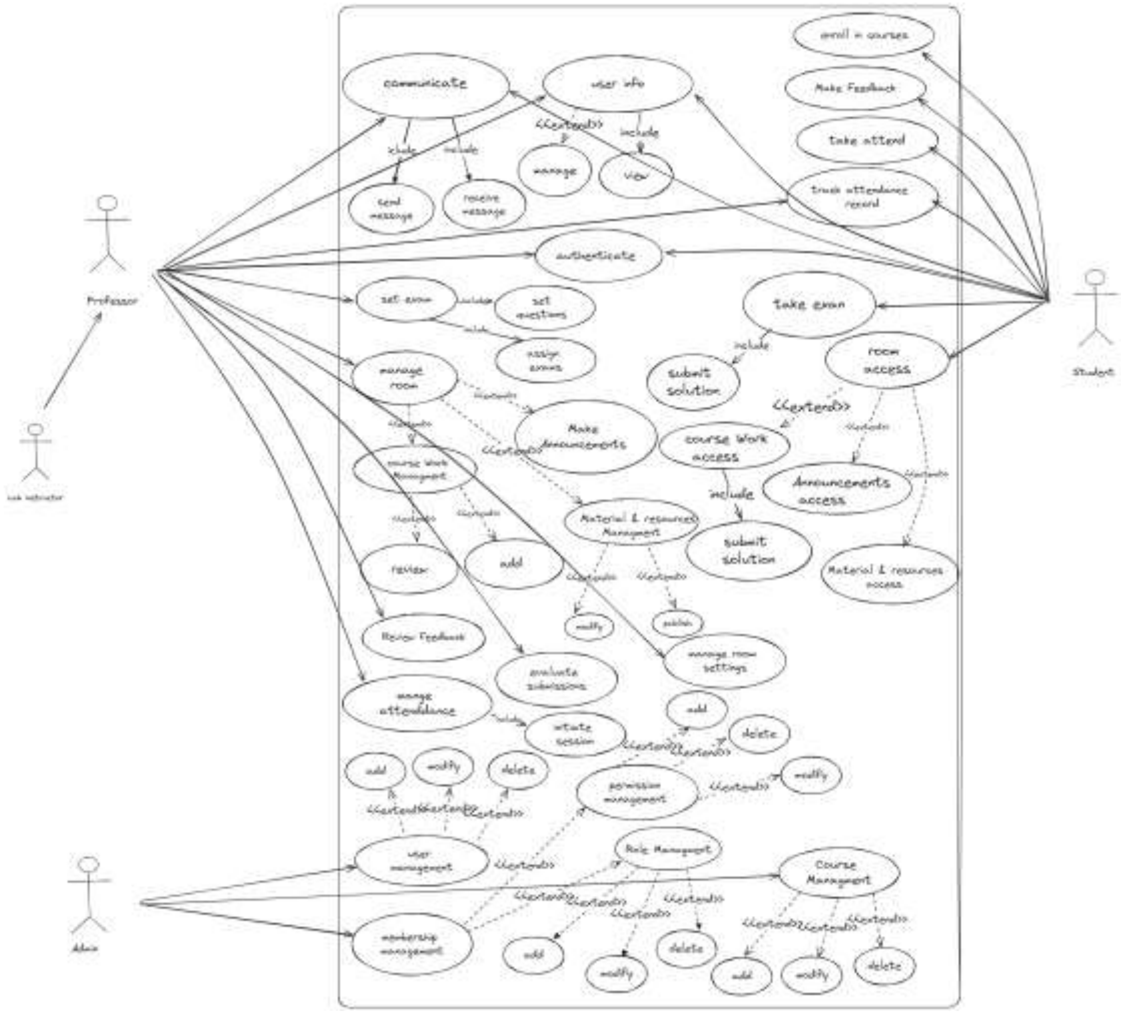


Figure 11 Use Case in Review

3.2.3.2 classroom use case

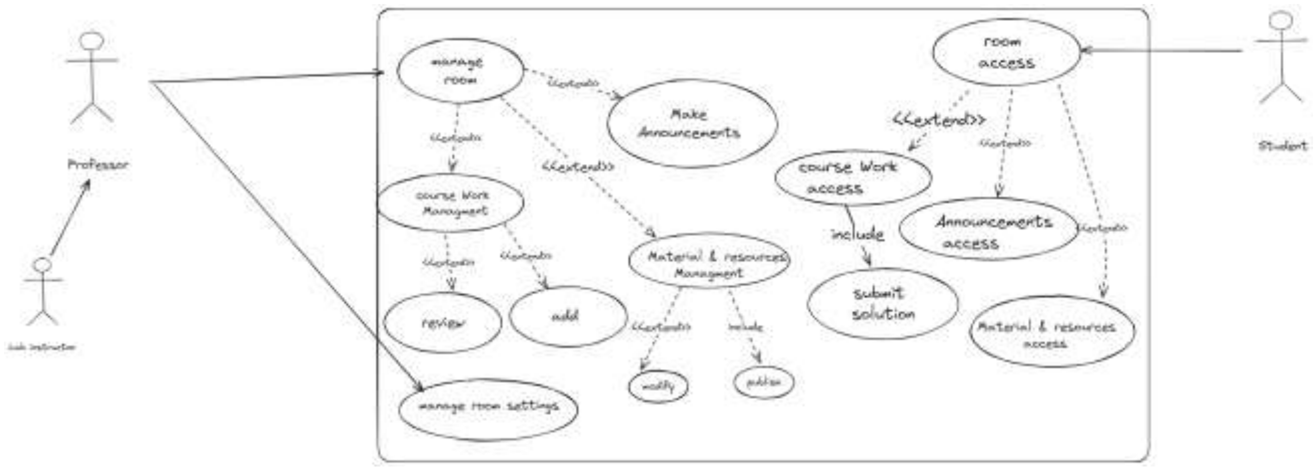


Figure 12 Use Case Classroom Module

3.2.3.3 exam use case

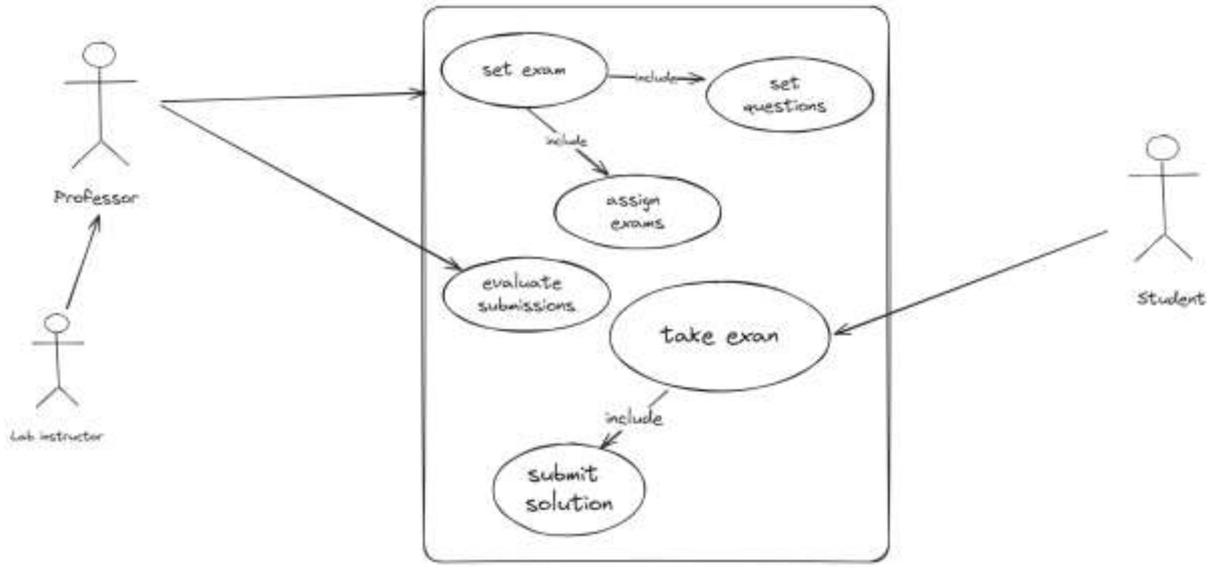


Figure 13 Use Case exam Module

3.2.3.4 attendance use case

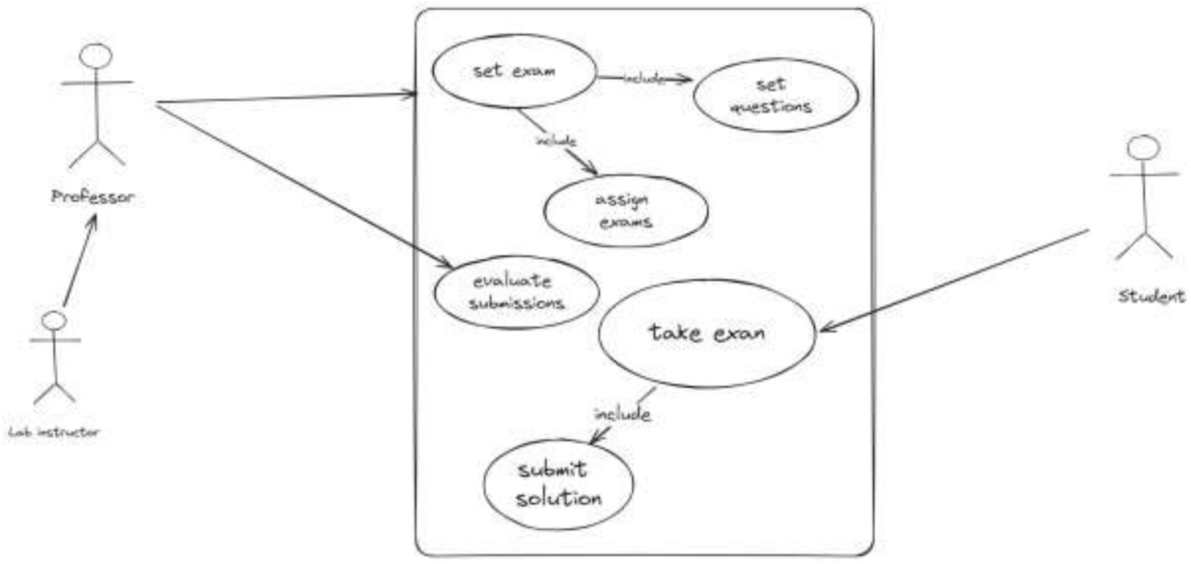


Figure 14 Use Case Attendance Module

3.2.3.5 identity & communication & feedback use case

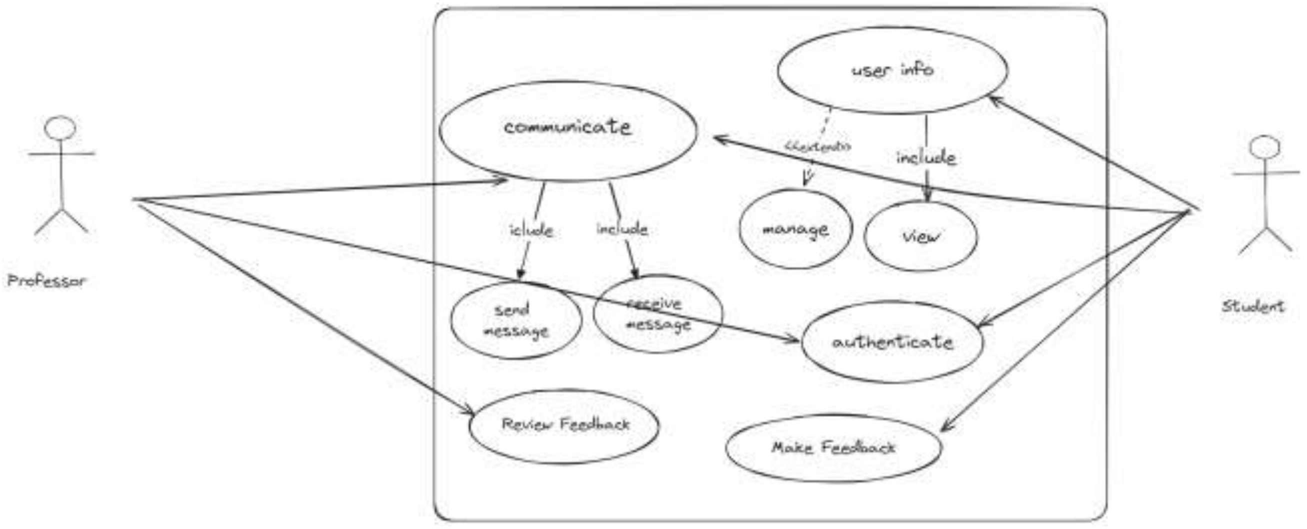


Figure 15 Use Case Identity & Communication & Feedback Module

3.2.3.6 Admin use case

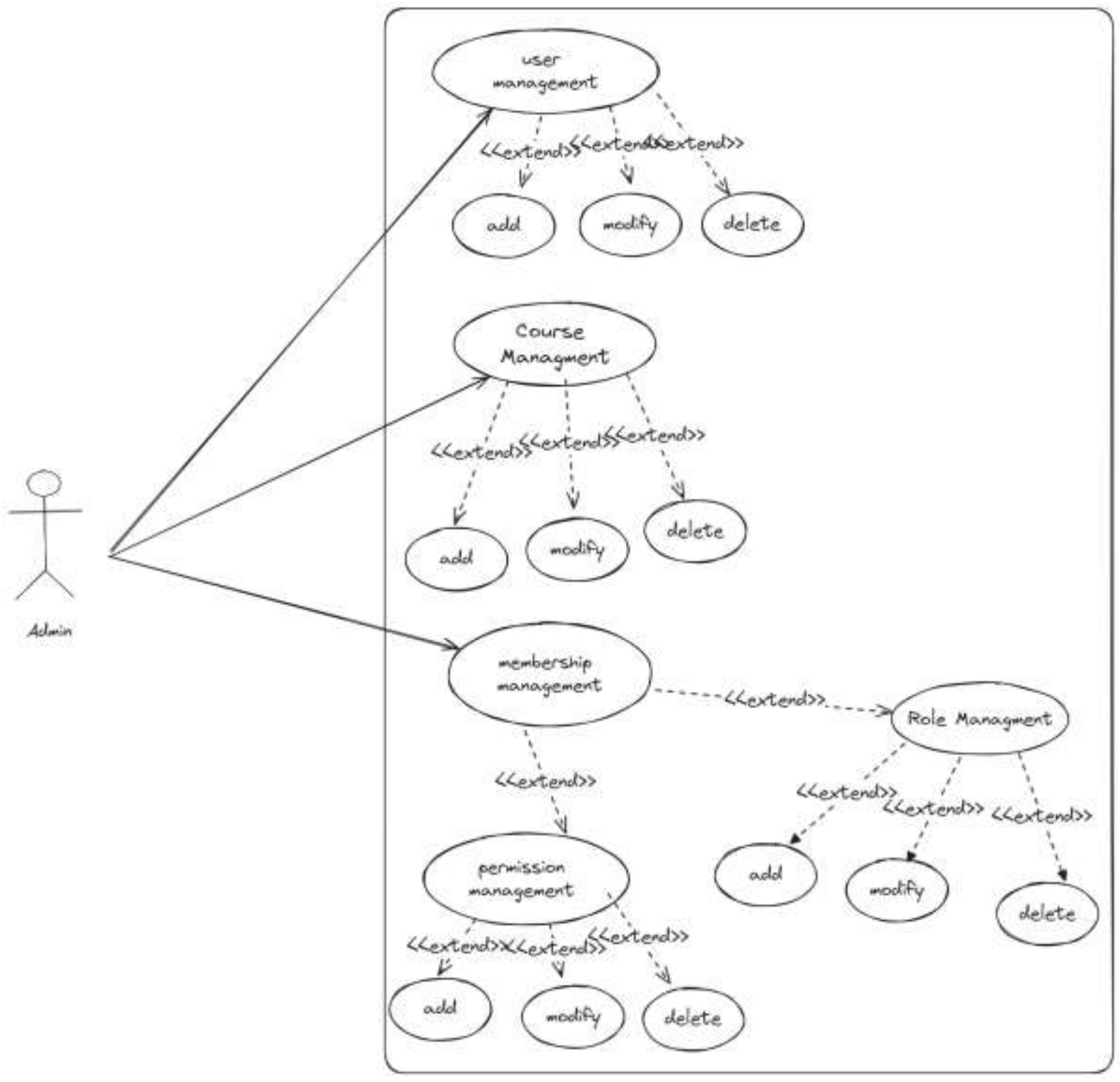


Figure 16 Use Case Admin Module

3.2.4 The ER Diagram

The meaning of ER Diagram:

- is a visual representation of the data model that illustrates the relationships among entities in a database. It is a widely used tool in database design and serves to depict the logical structure of a database in a clear and concise manner.

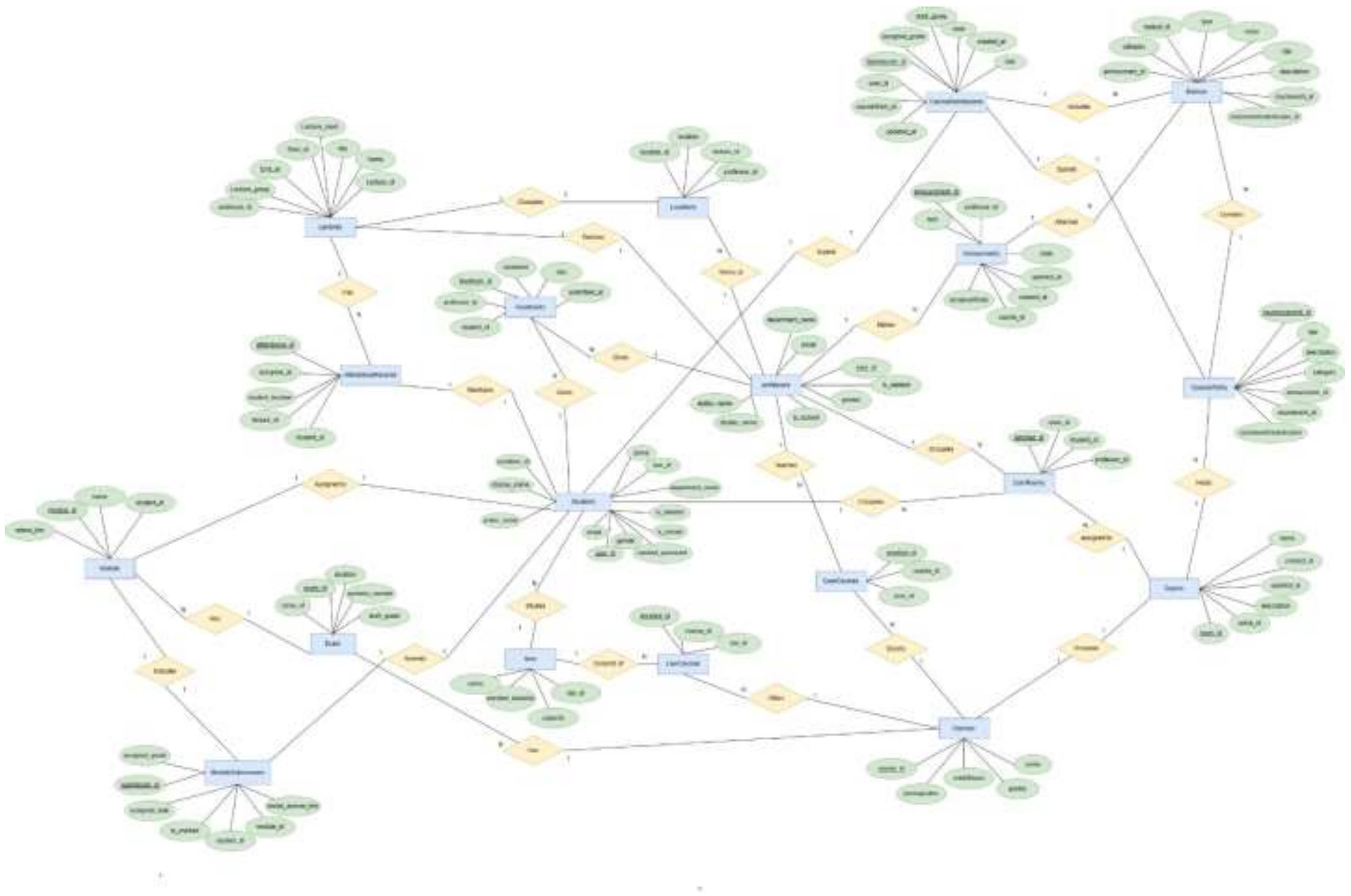


Figure 17 ERD Diagram in Review

3.2.5 Schema Diagram

The meaning of Schema Diagram:

-is a visual representation of a database schema. In the context of databases, a schema defines the structure of the database, including tables, fields, relationships, and constraints. A schema diagram provides an overview of how these components are organized and connected within the database.

3.2.5.1 Business Schema

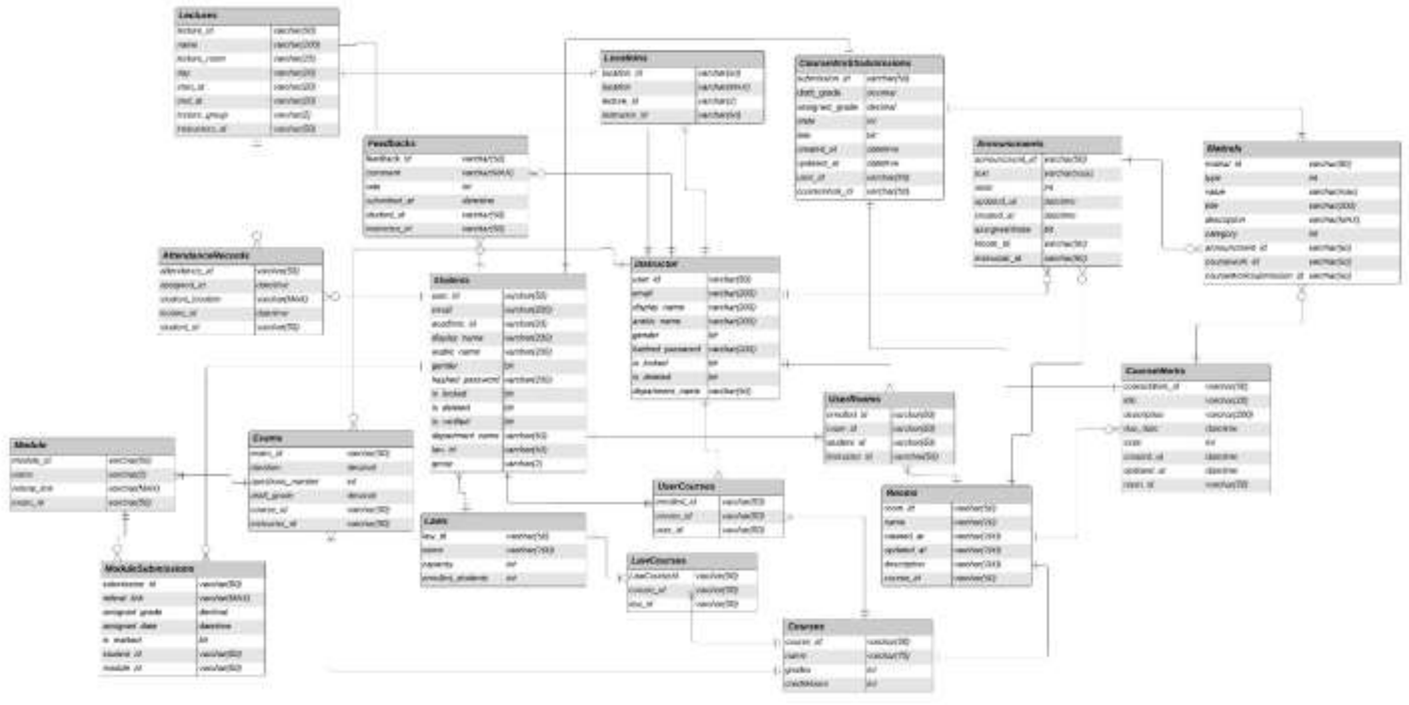


Figure 18 Business Schema in Review

3.2.5.2 Attendance Schema

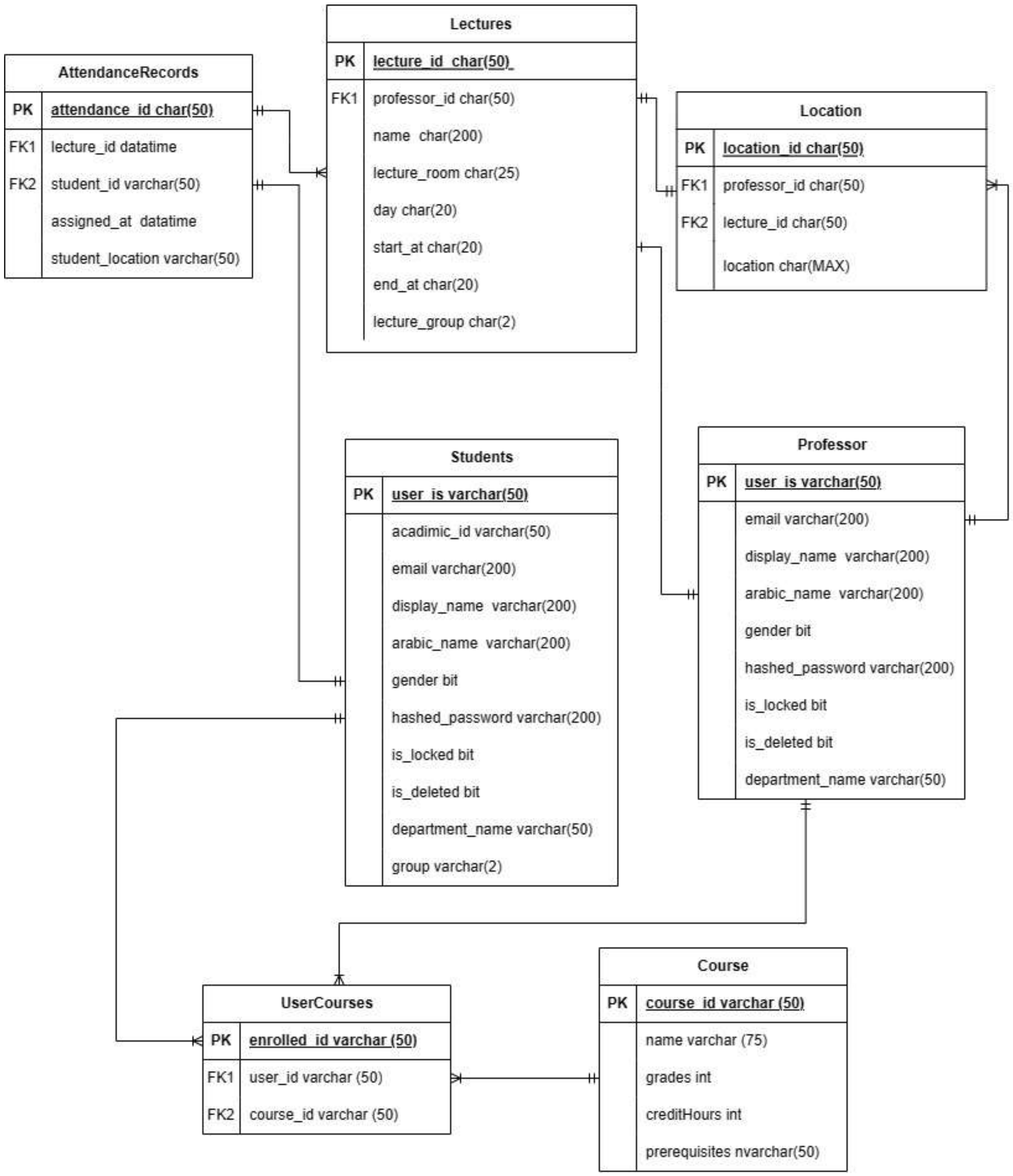


Figure 19 Attendance Schema Module

3.2.5.3 ClassWork schema

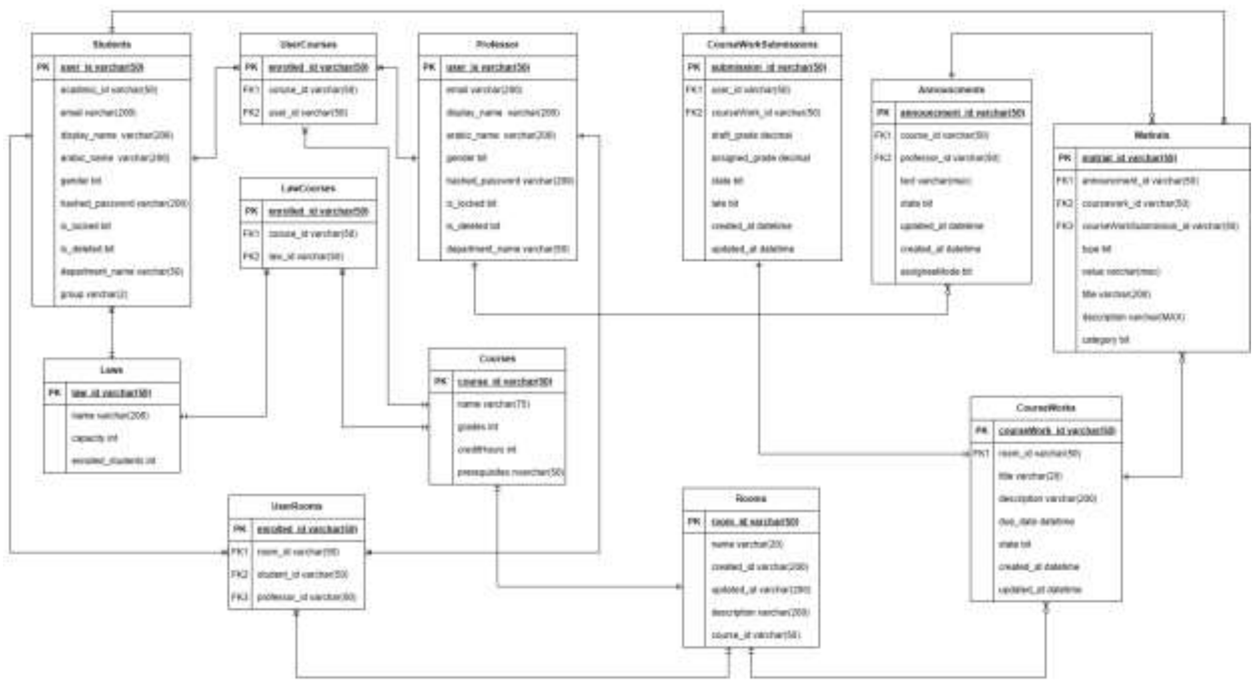


Figure 20 ClassWork Schema Module

3.2.5.4 Exam schema

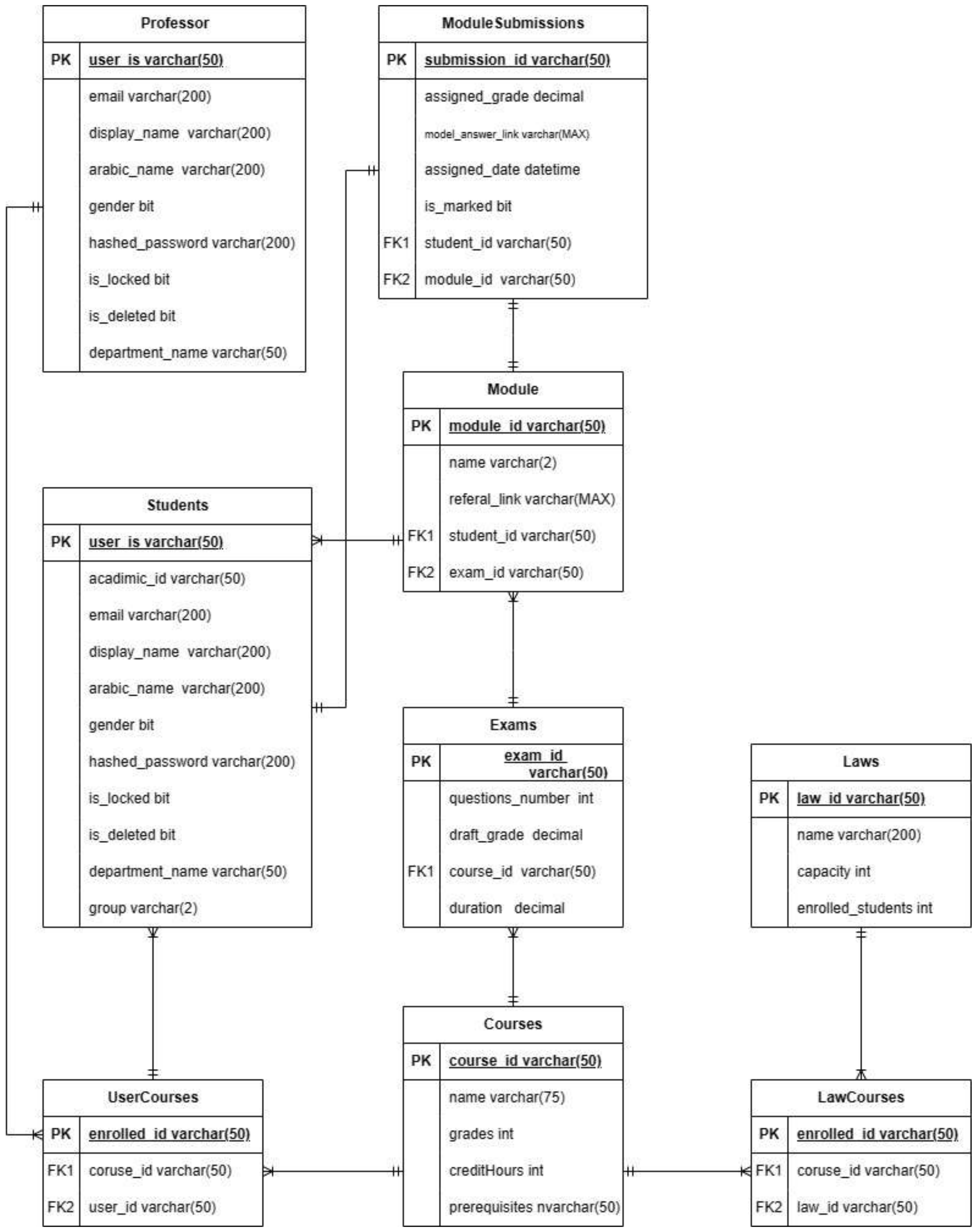


Figure 21 Exam Schema Module

3.2.5.5 Feedback schema

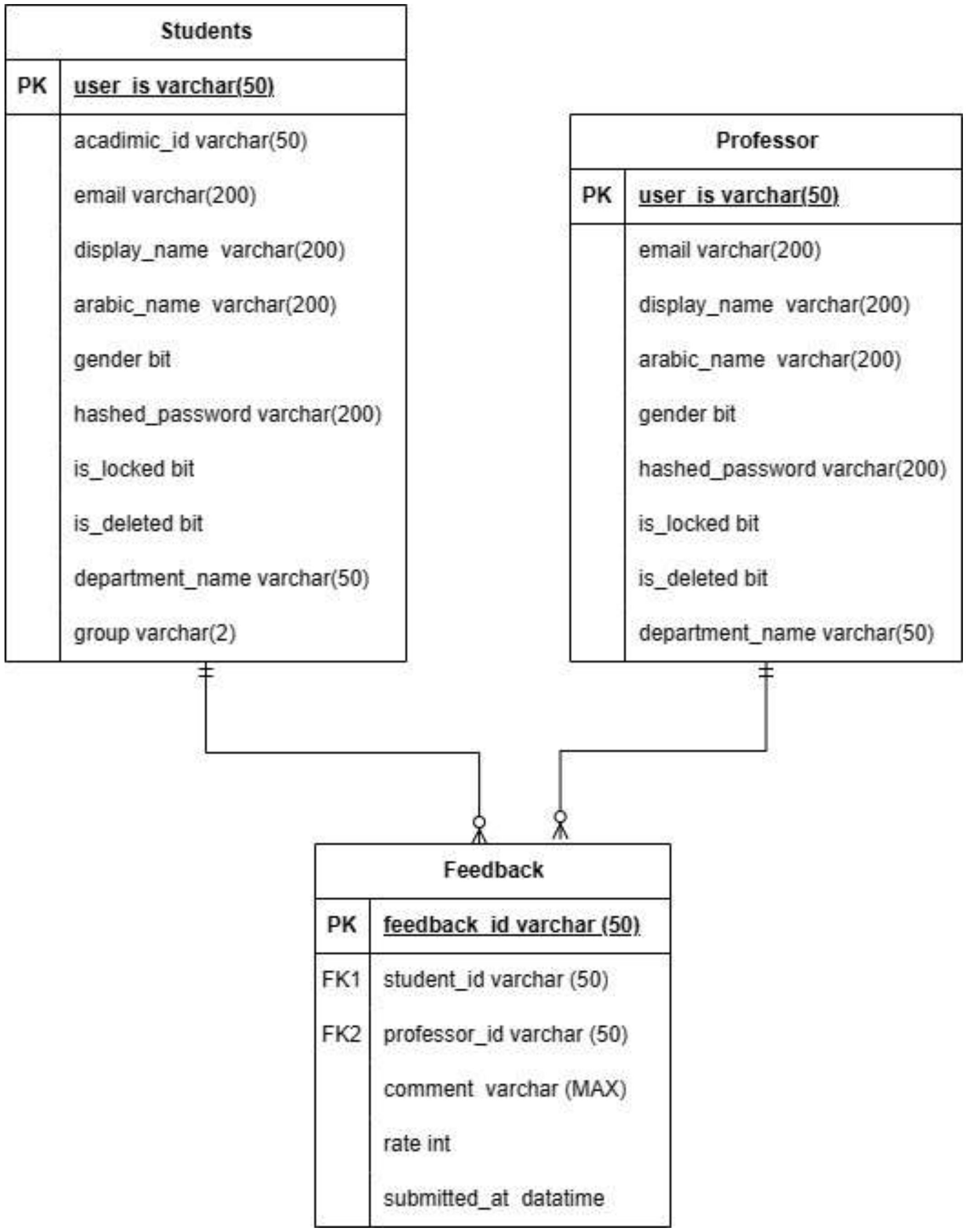


Figure 22 Feedback Schema Module

3.2.5.6 User Management

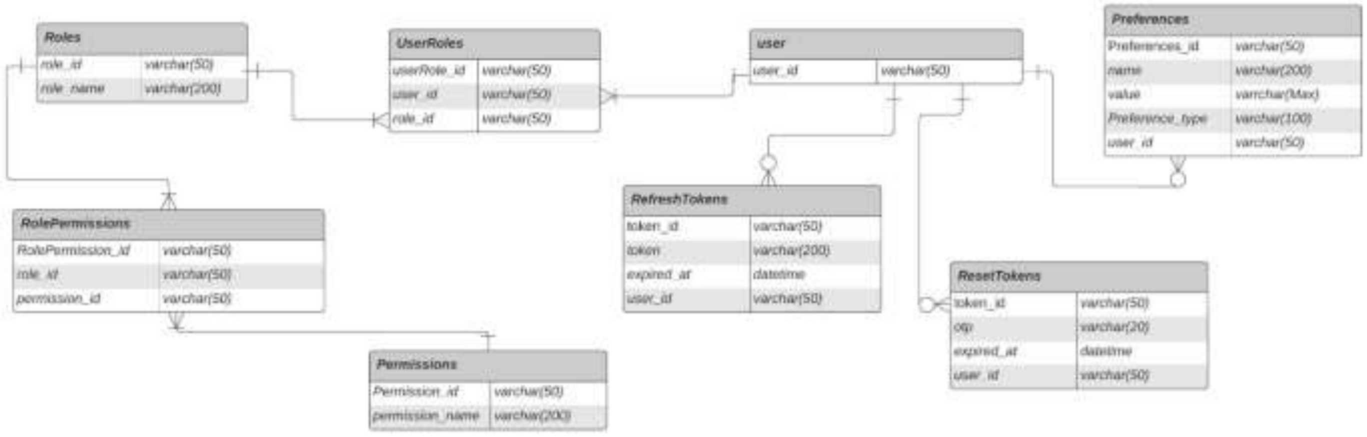


Figure 23 User Management Schema Module

3.2.6 Class Diagram

The meaning of class Diagram:

-is a type of diagram that represents the structure and relationships of classes within a system or application.

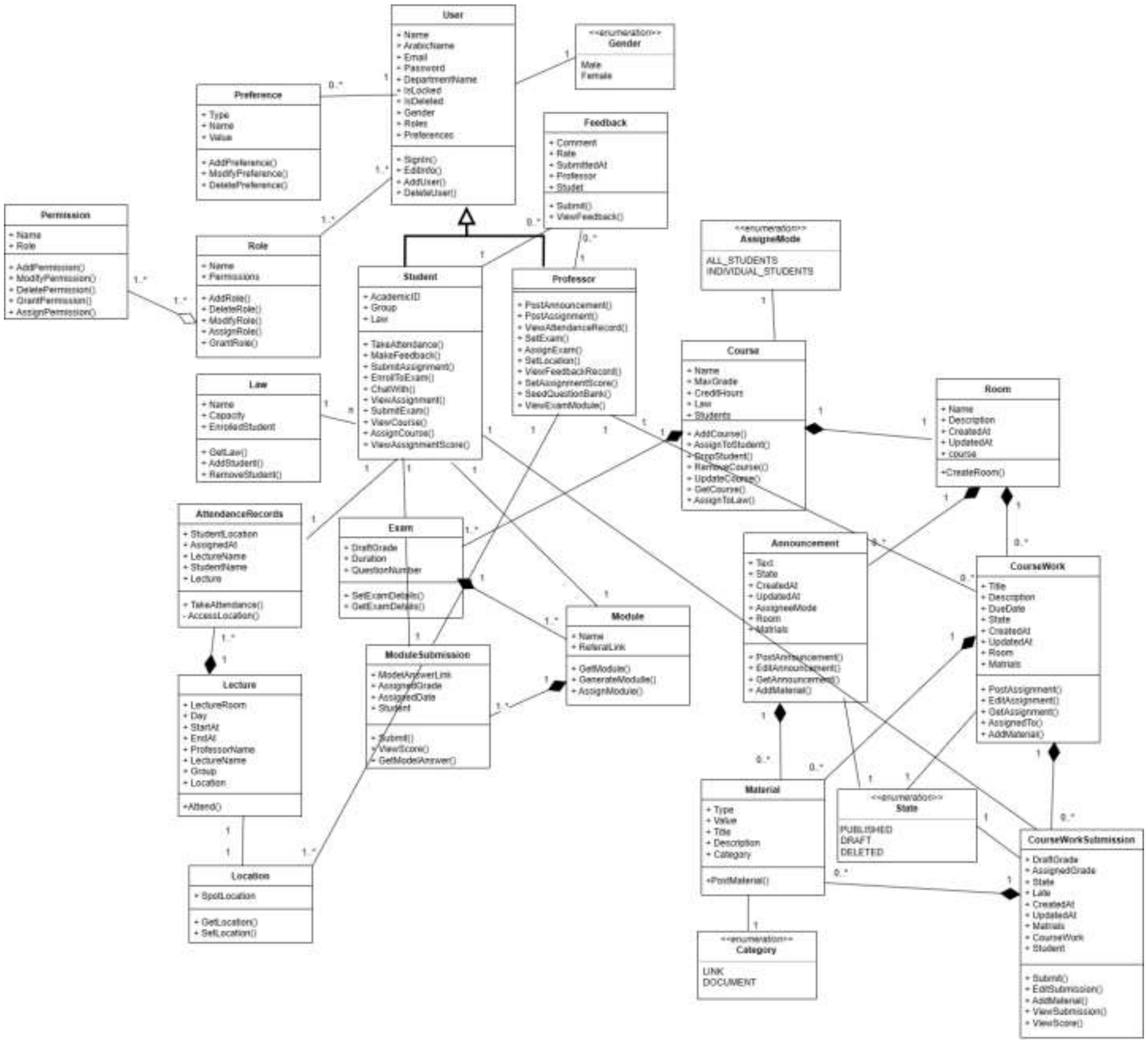


Figure 24 Class Diagram in Review

3.2.7 Class explained

3.2.7.1 Classroom

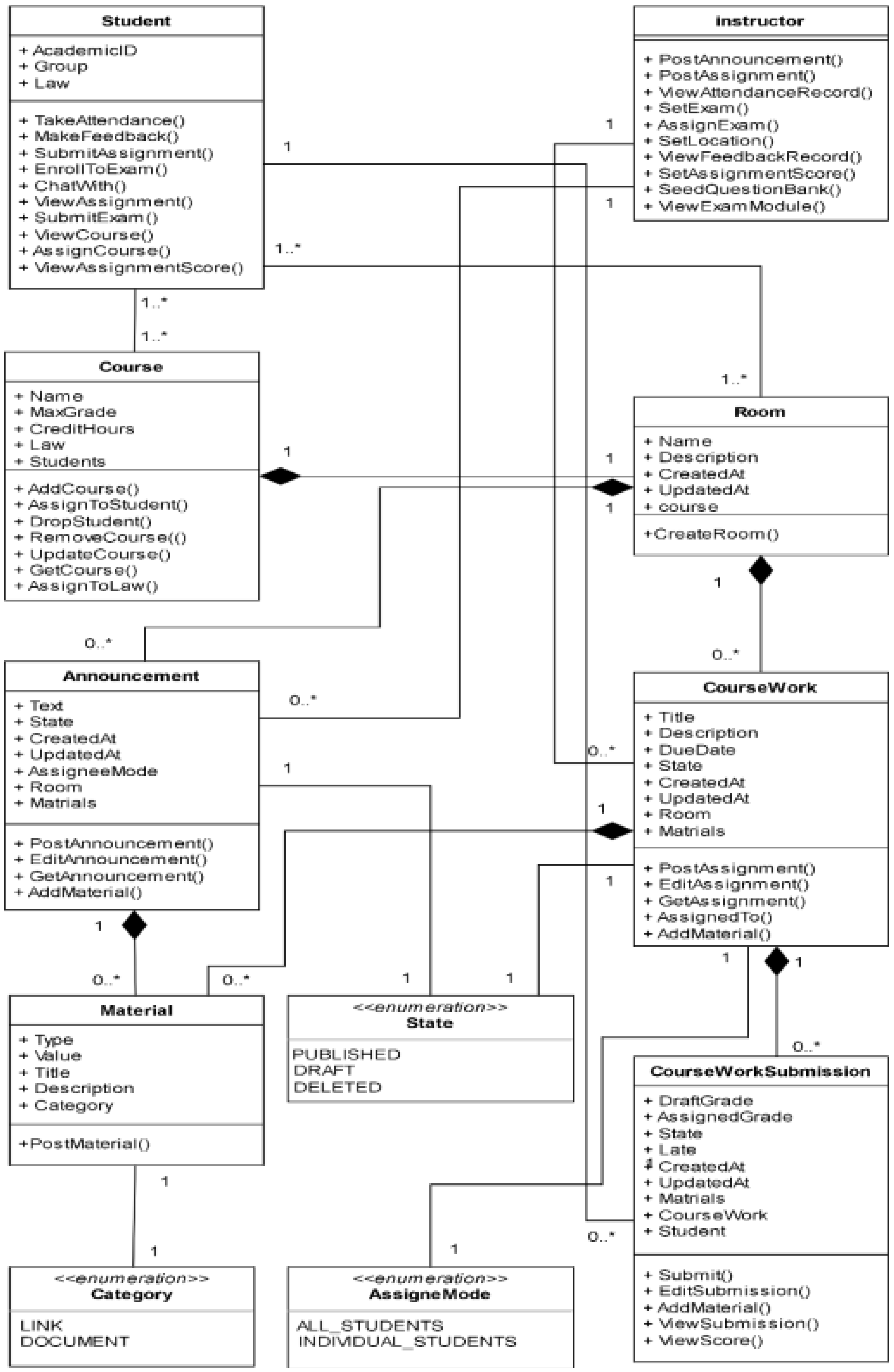


Figure 25 Class Diagram Classroom Module

3.2.7.2 Attendance

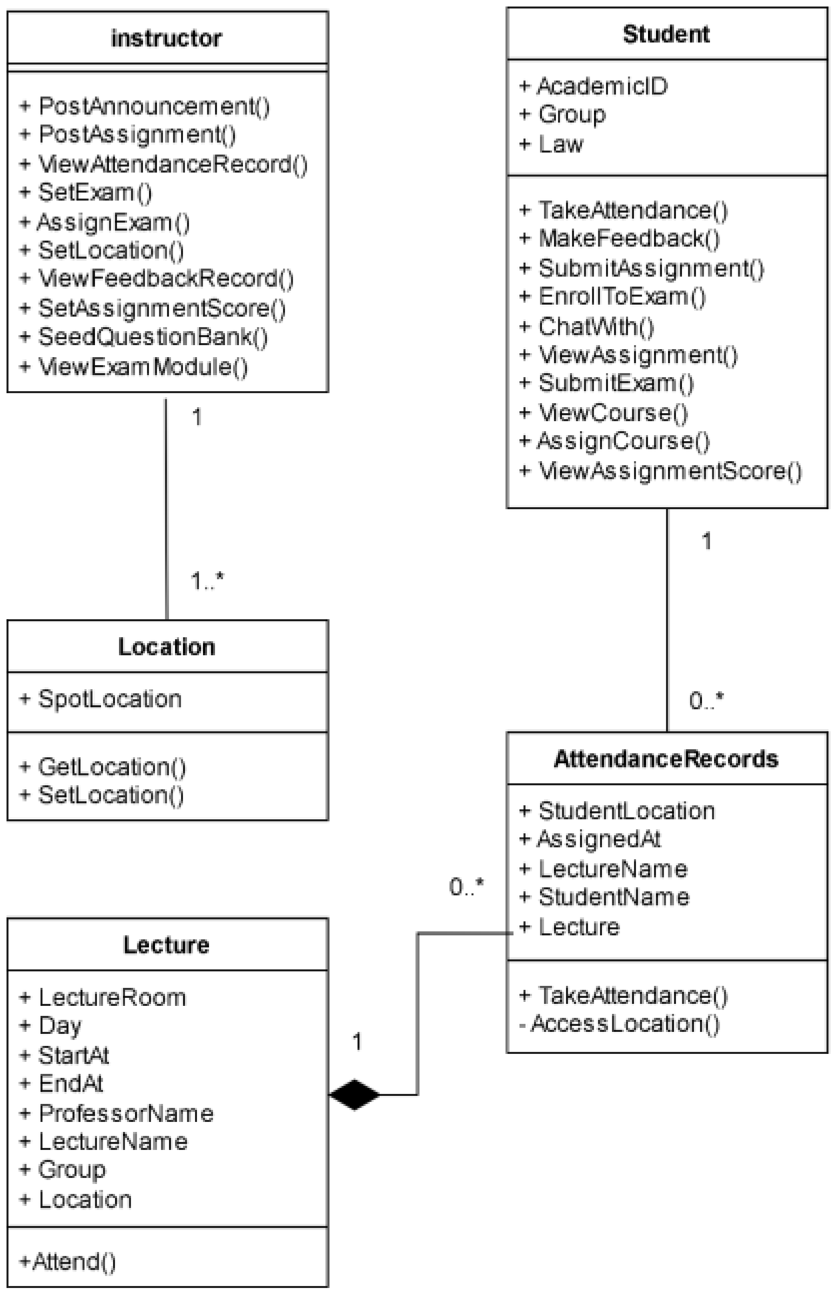


Figure 26 Class Diagram Attendance Module

3.2.7.3 Exam

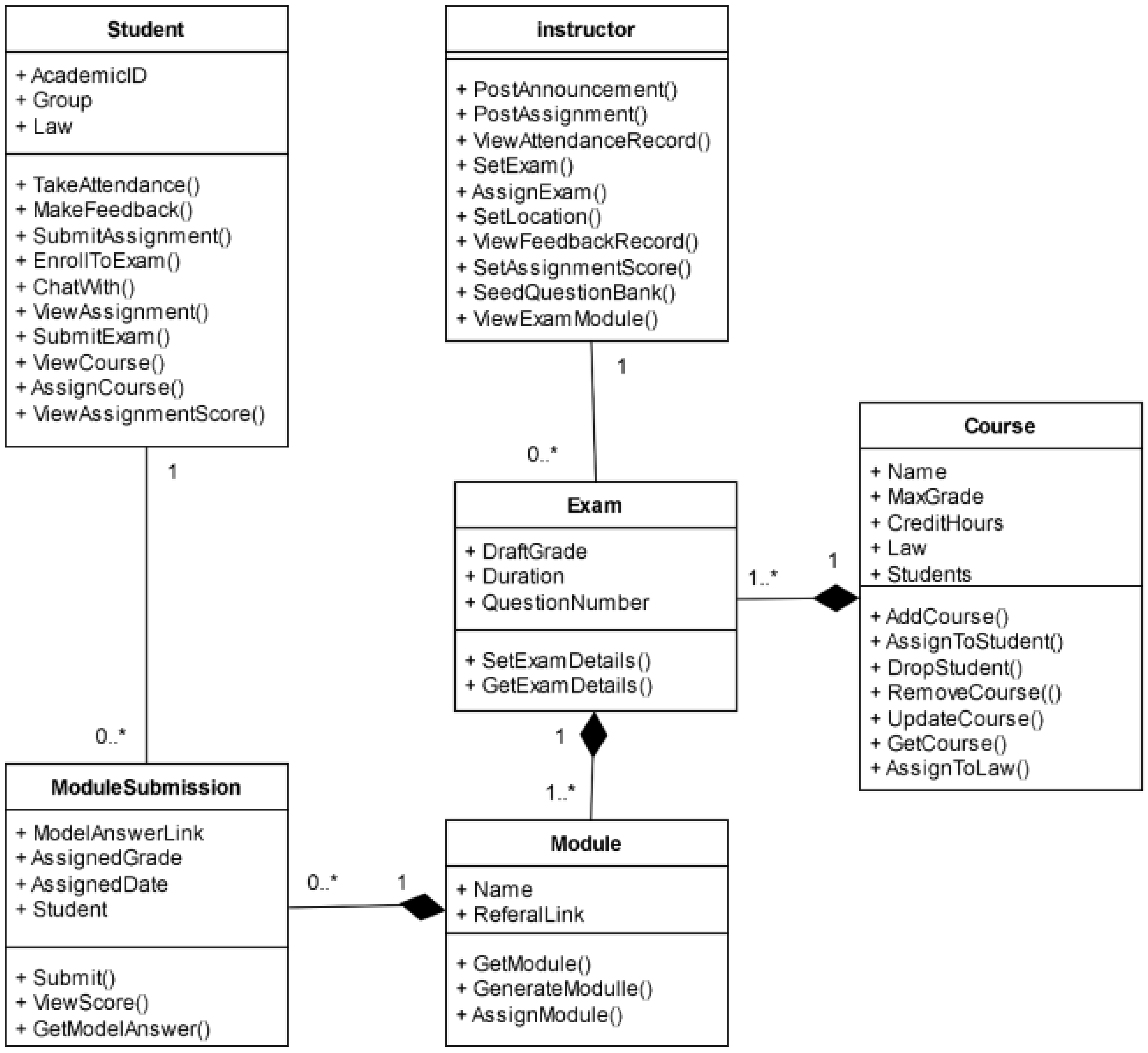


Figure 27 Class Diagram Exam Module

3.2.7.4 Feedback

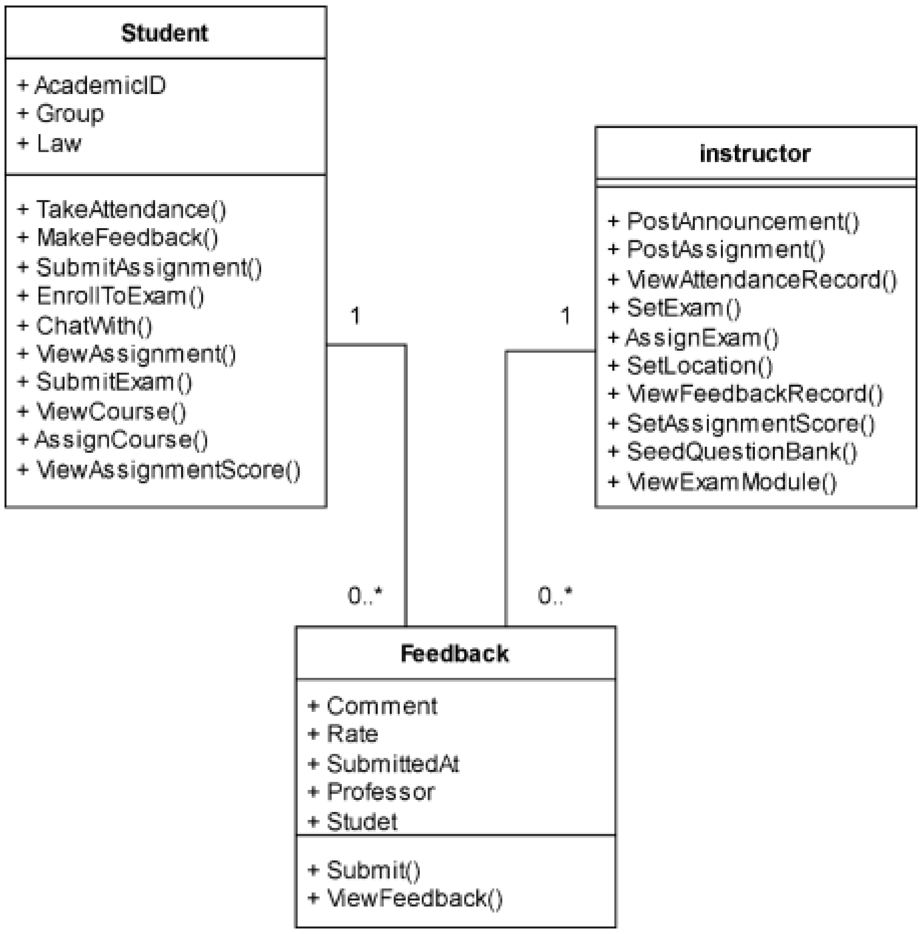


Figure 28 Class Diagram Feedback Module

3.2.7.5 General

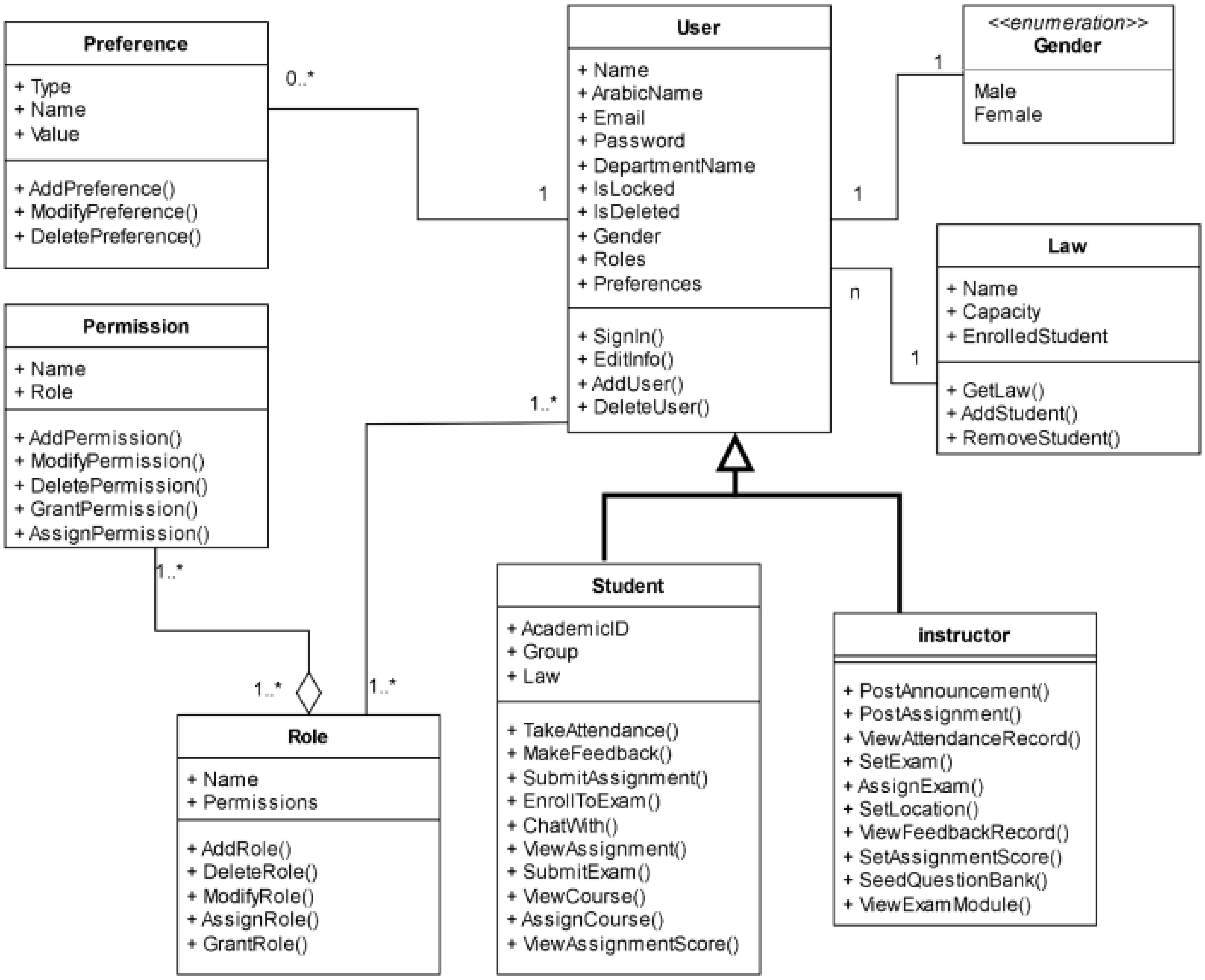


Figure 29 Class Diagram General

3.2.8 Activity Diagram

The meaning of activity diagram:

-is a type of diagram that illustrates the workflow or procedural flow of activities within a system or a business process. It provides a visual representation of the dynamic aspects of a system, showing the sequence of activities, actions, decisions, and transitions.

3.2.8.1 Login and profile complete

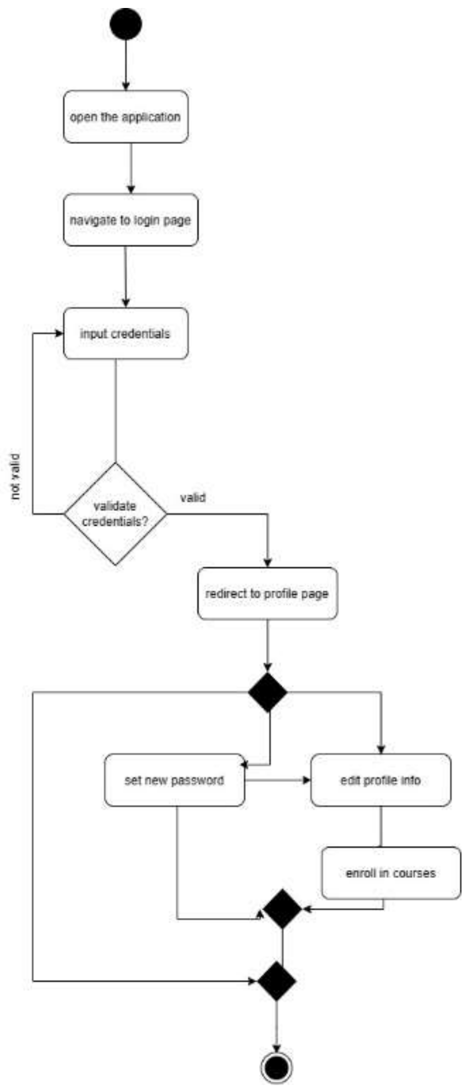


Figure 30 Activity Diagram Login and profile complete Module

3.2.8.2 Attendance

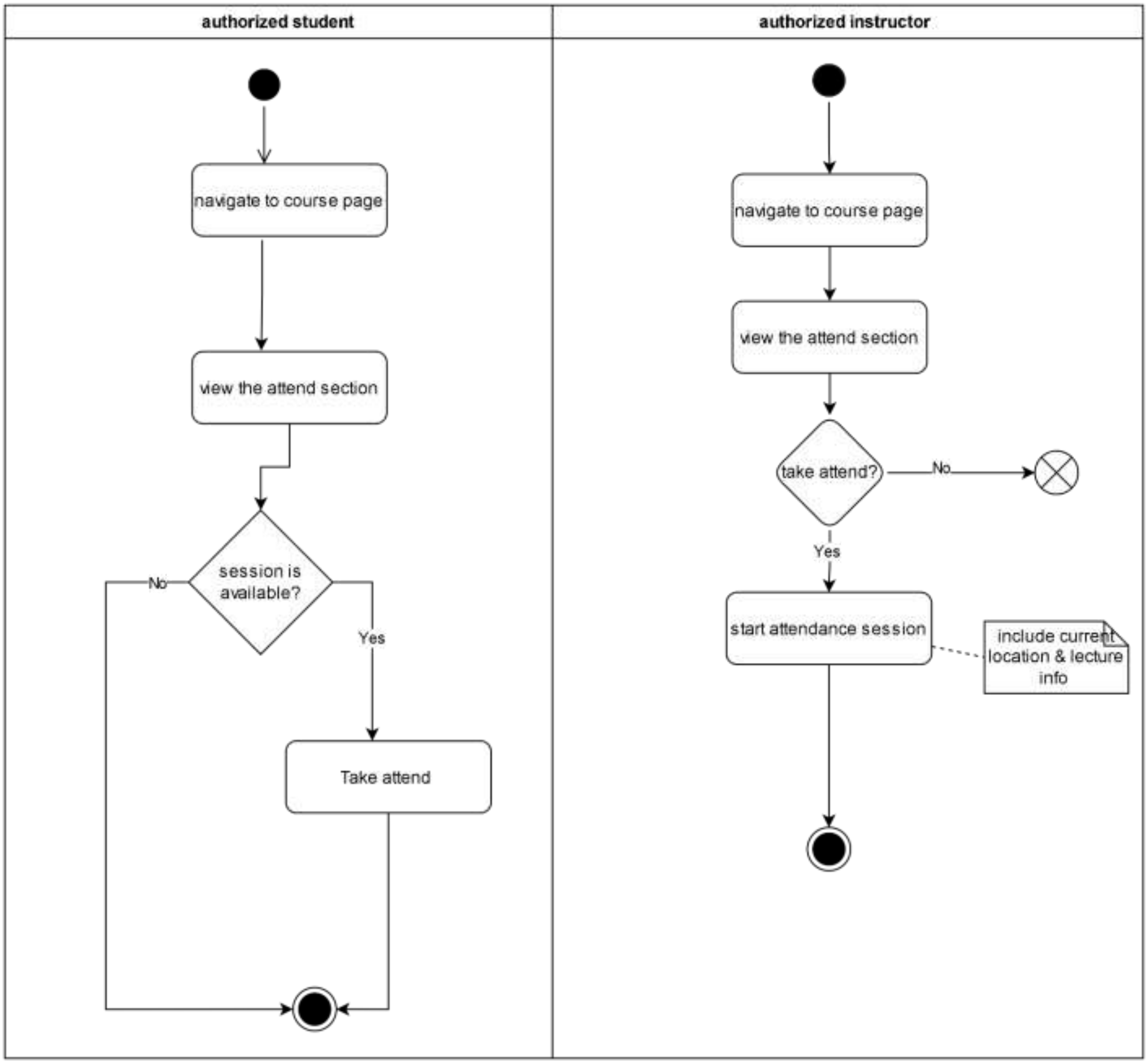


Figure 25 Activity Diagram Attendance Module

3.2.8.3 Exam

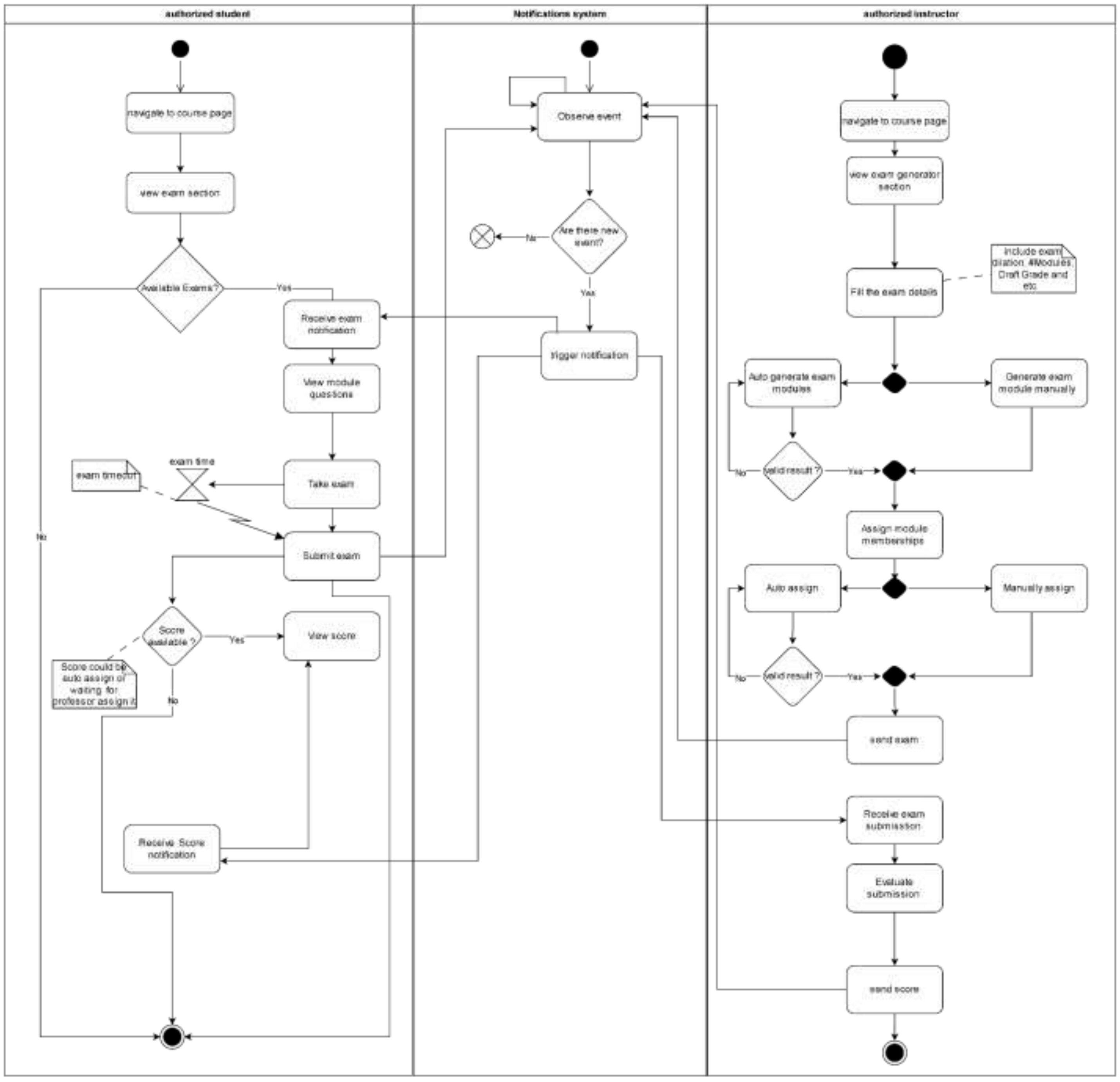


Figure 31 Activity Diagram Exam Module

3.2.8.4 Feedback

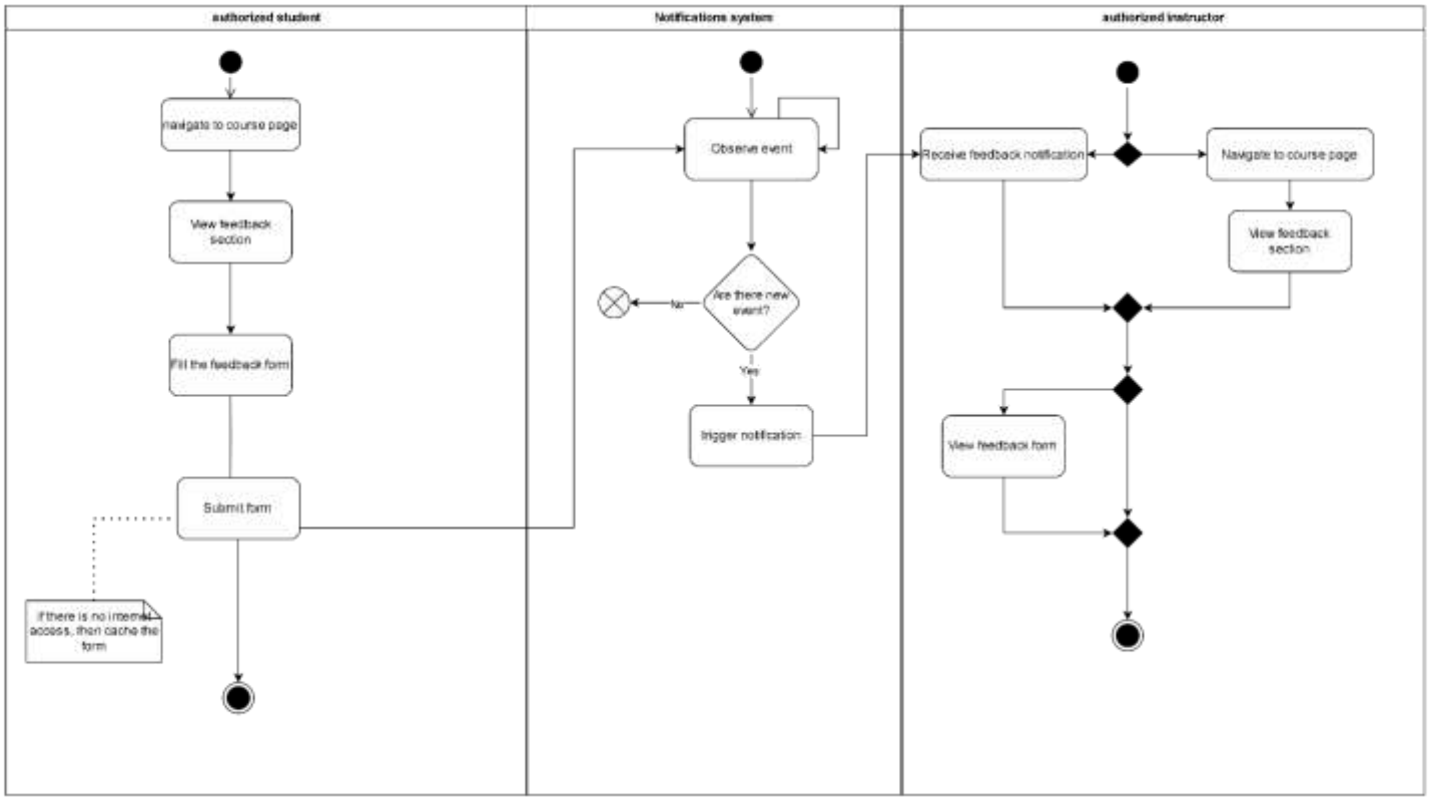


Figure 32 Activity Diagram Feedback Module

3.2.8.5 Communication

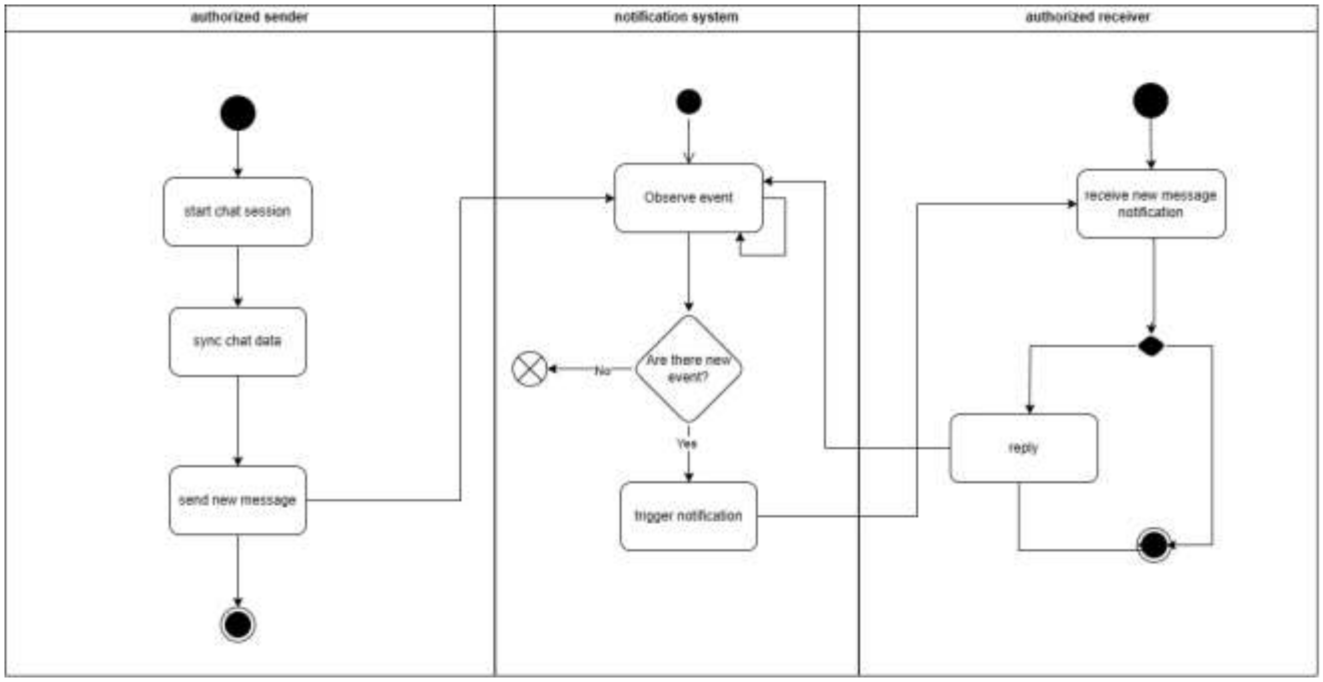
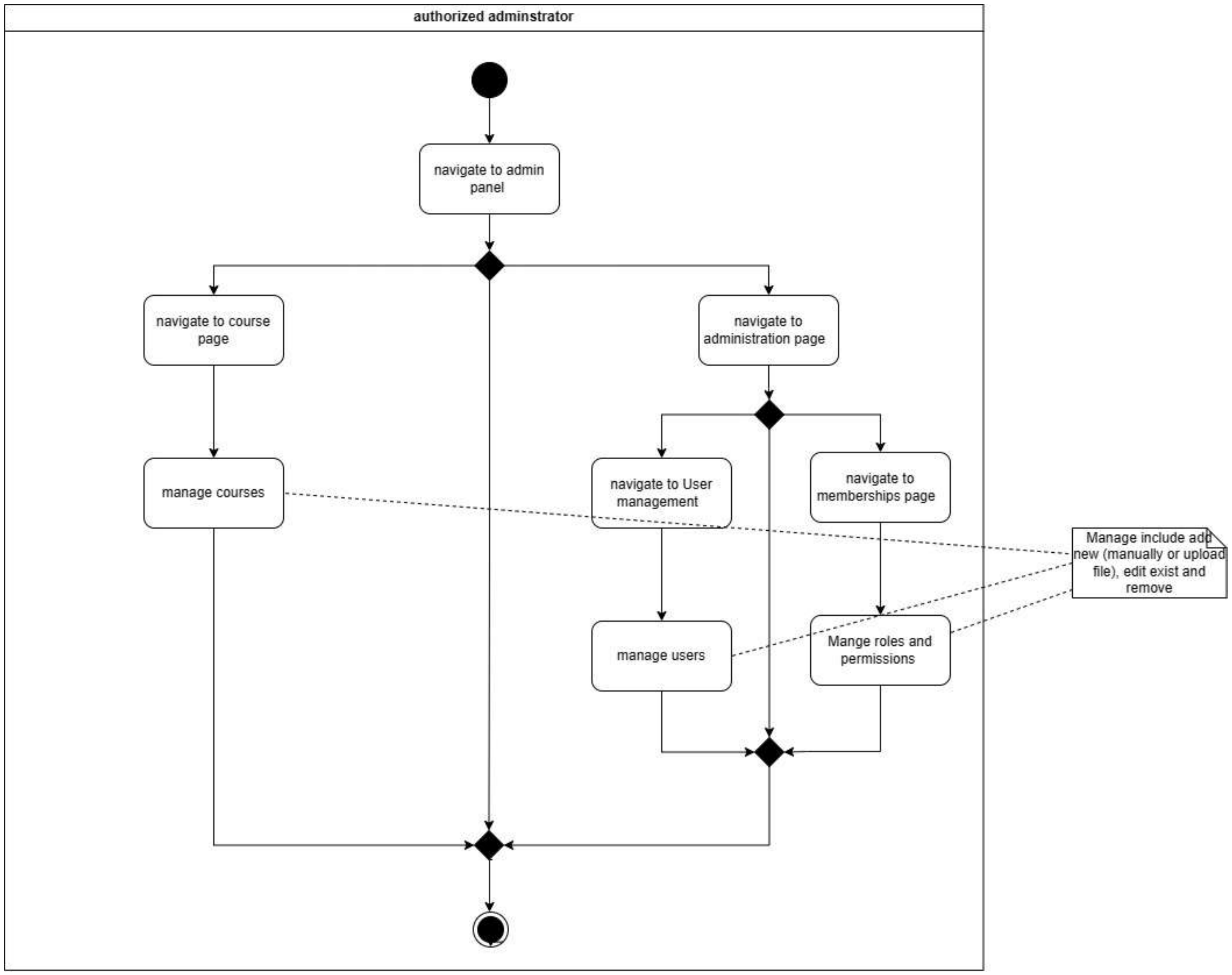


Figure 33 Activity Diagram Communication Module

3.2.8.6 Administrator



authorized adminstrator
navigate to admin panel
navigate to course page
navigate to administration page
manage courses
navigate to User management
navigate to memberships page
Manage include add new (manually or upload file), edit exist and remove
manage users
Mange roles and permissions

Figure 34 Activity Diagram Administrator Module

3.2.8.7 ClassWork

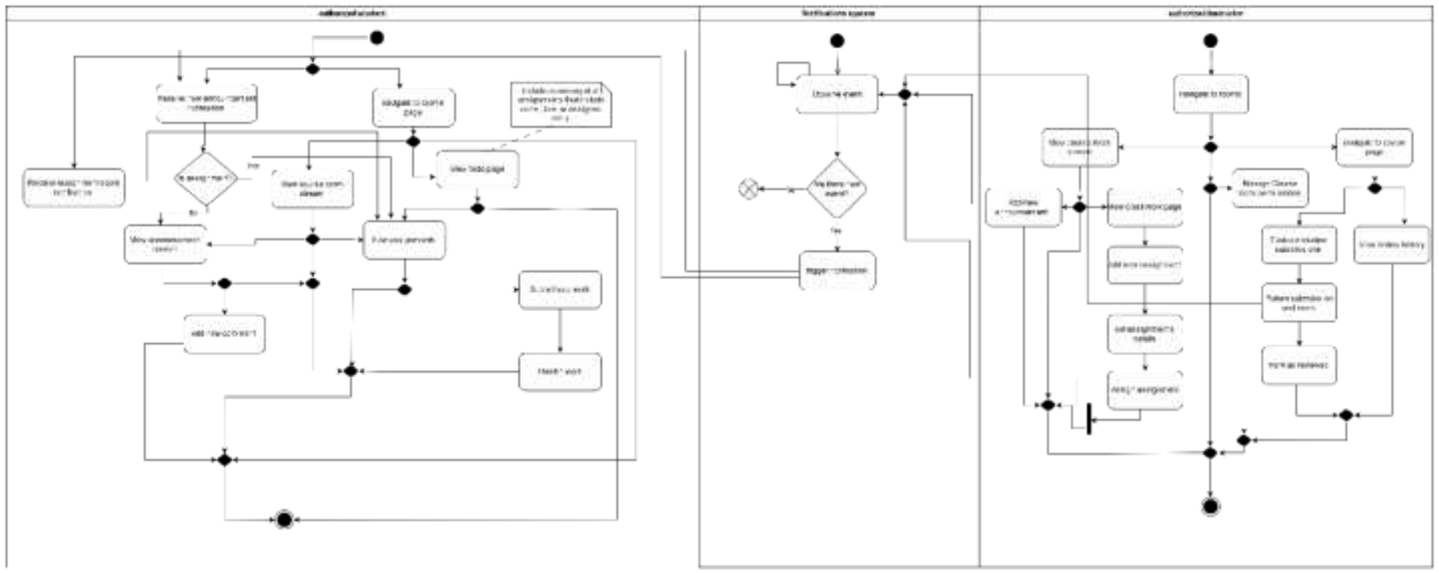


Figure 35 Activity Diagram ClassWork Module

3.2.9 Sequence Diagram

The meaning of sequence diagram:

-is a type of diagram that illustrates the interactions and messages exchanged between different objects or components in a system over a specific period. Sequence diagrams provide a visual representation of the chronological order of interactions among objects, showing how they collaborate to achieve a particular functionality or use case.

3.2.9.1 Authentication

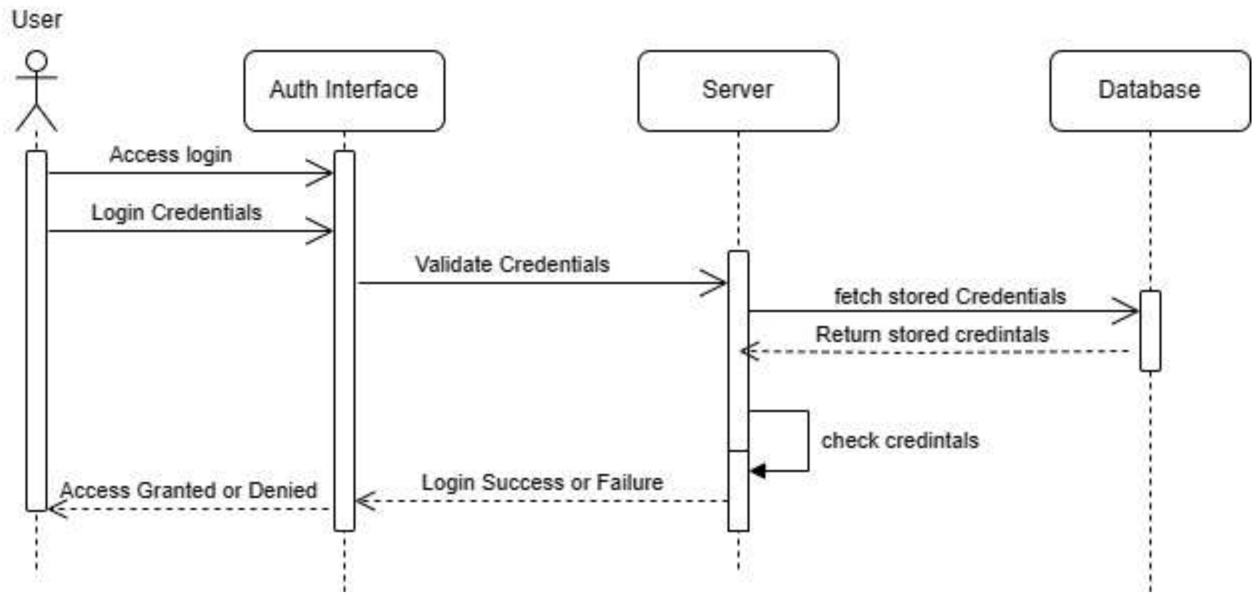


Figure 36 Sequence Diagram Authentication Module

3.2.9.2 Attendance

3.2.9.2.1 Attendance for instructor

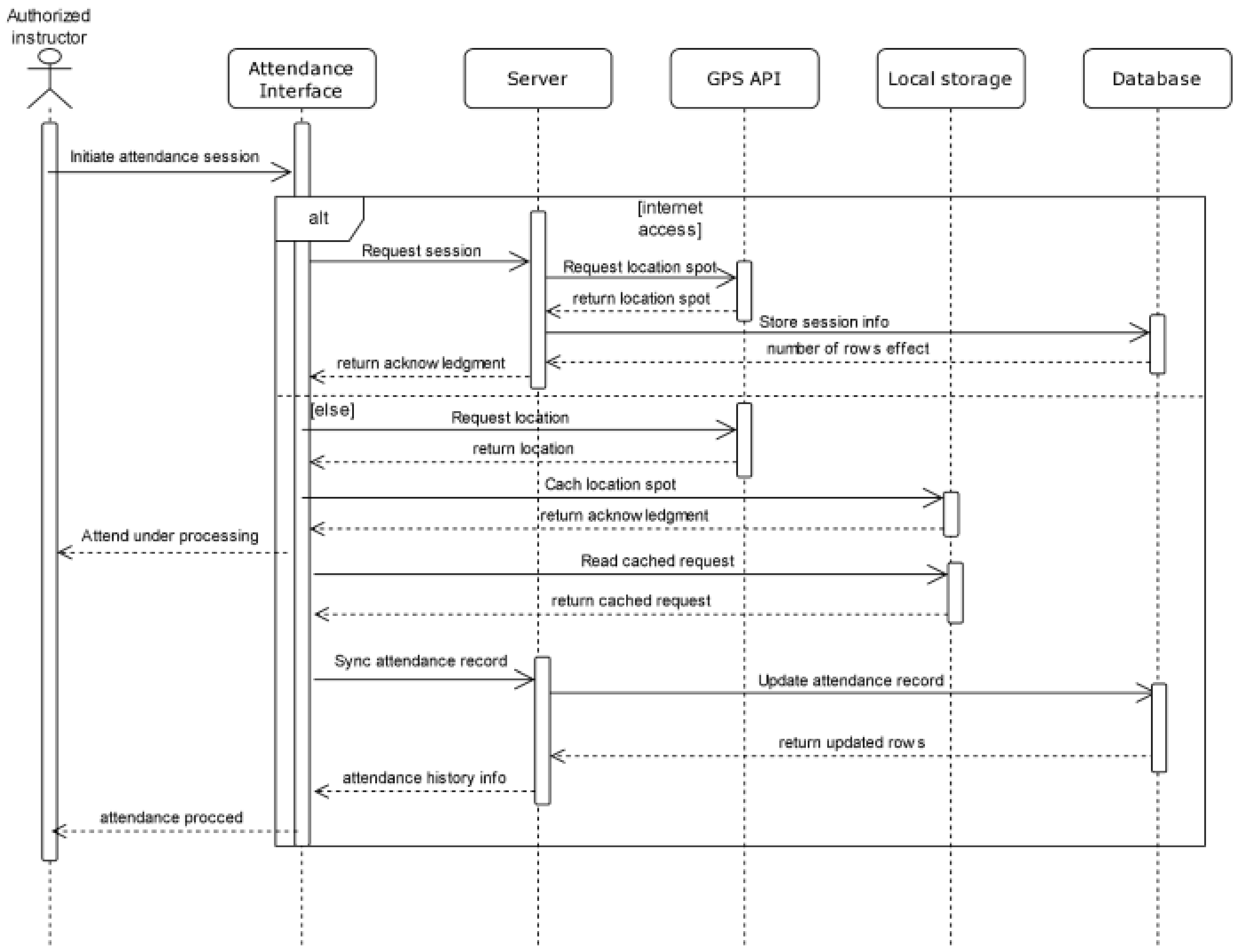


Figure 37 Sequence Diagram Attendance Module For Professor

3.2.9.2.2 Attendance for student

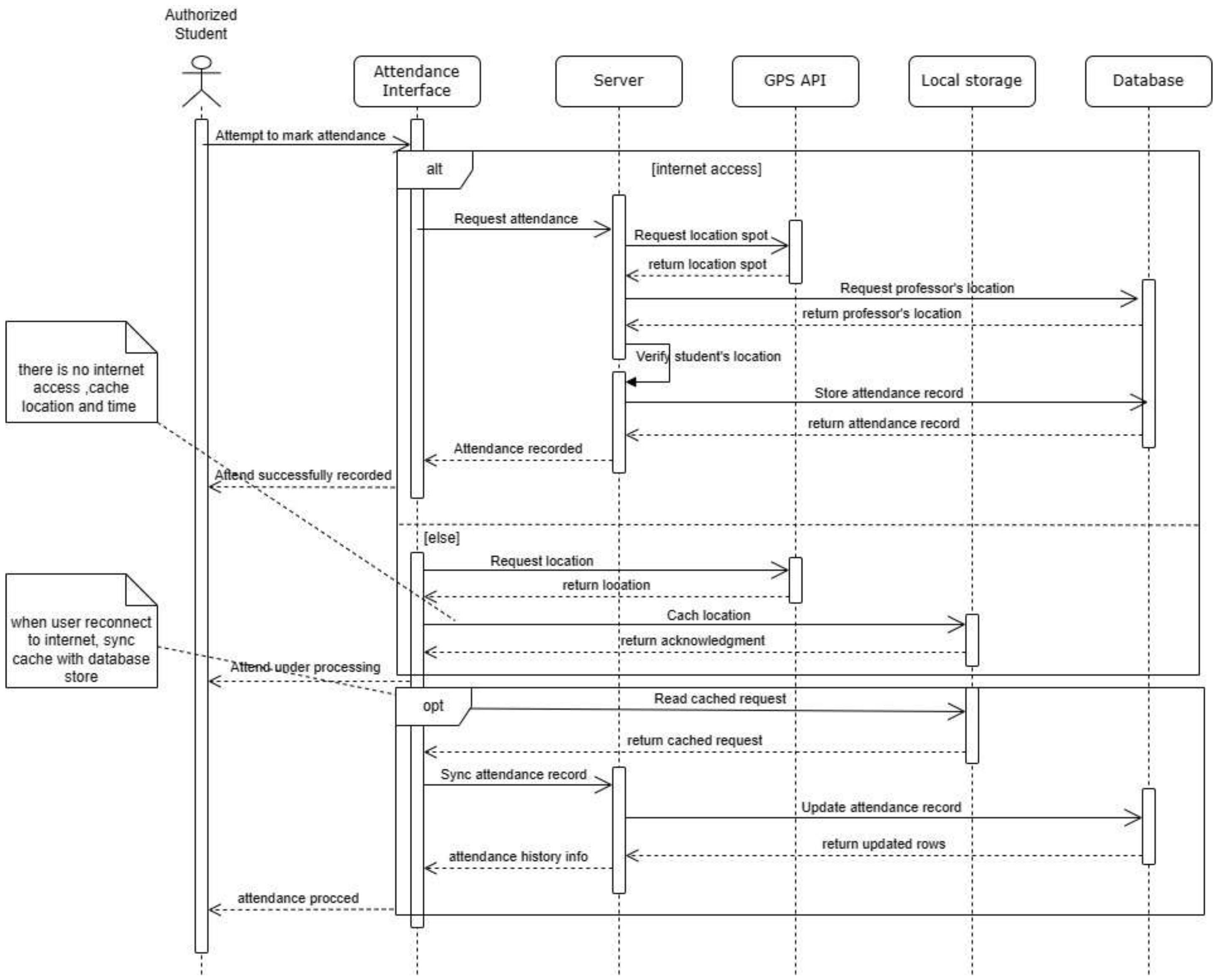


Figure 38 Sequence Diagram Attendance Module For Student

3.2.9.3 Exam module

3.2.9.3.1 Exam generating for instructor

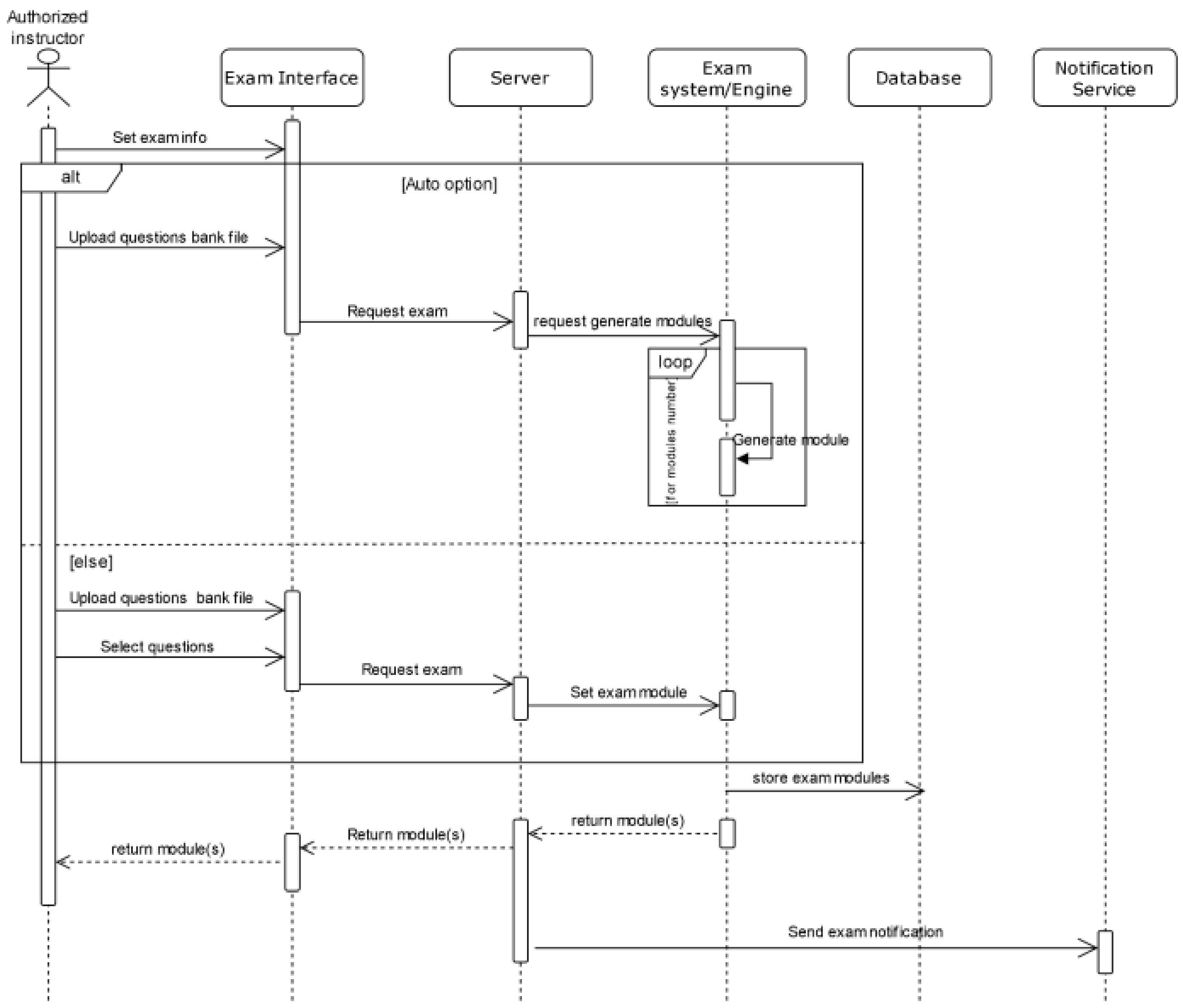


Figure 39 Sequence Diagram Exam Module For Professor

3.2.9.3.2 Exam taking for student

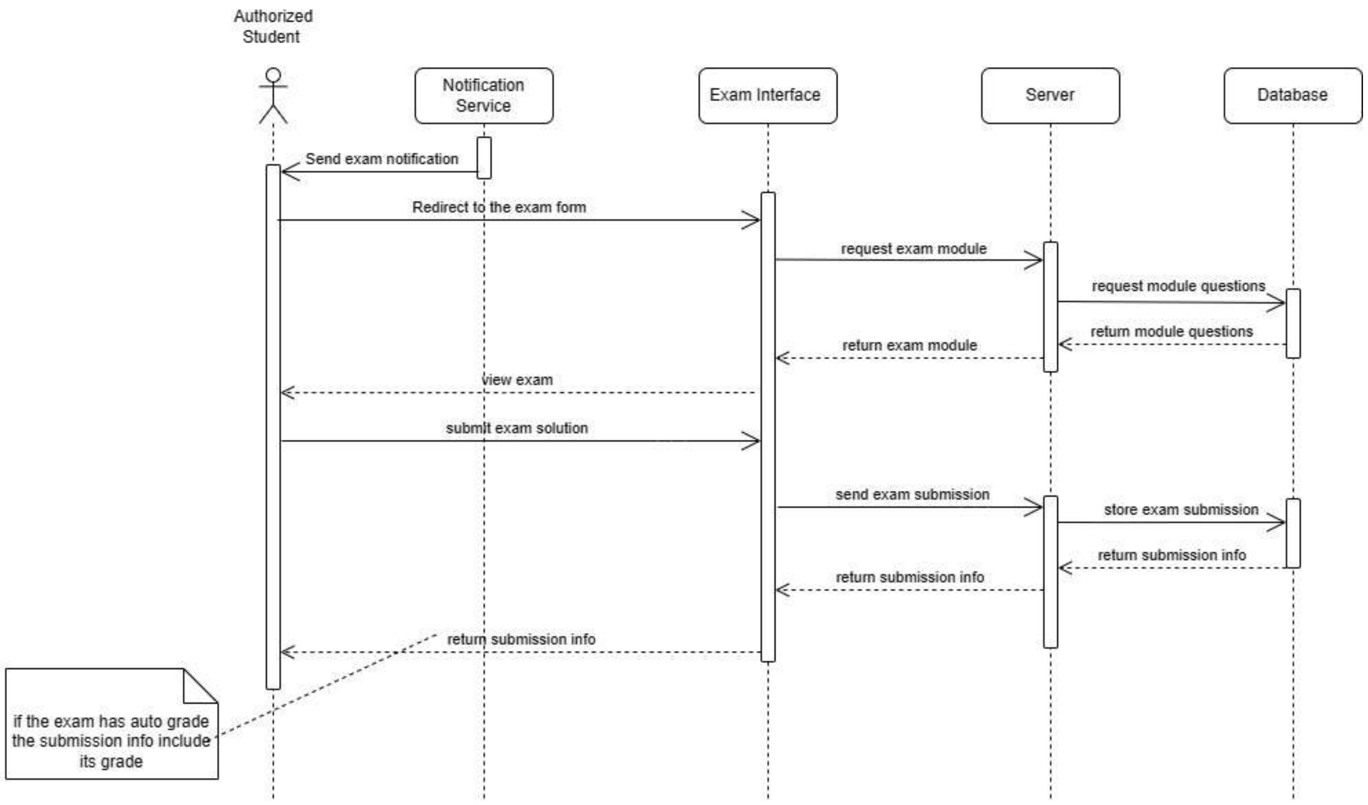


Figure 40 Sequence Diagram Exam Module For Student

3.2.9.4 Review module

3.2.9.4.1 instructor review and evaluate student submissions

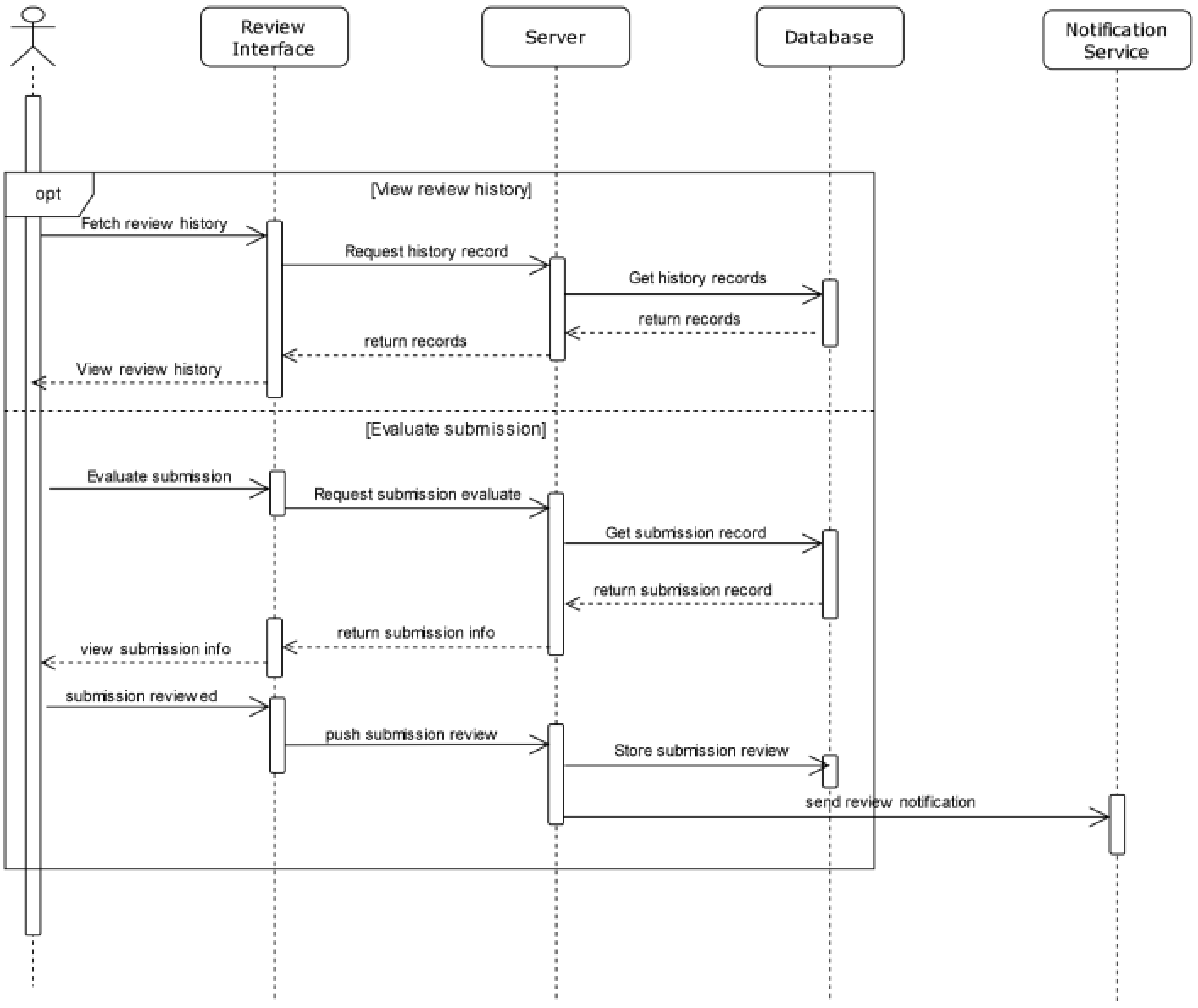


Figure 41 Sequence Diagram Review Module For Professor

3.2.9.4.2 Student Review score

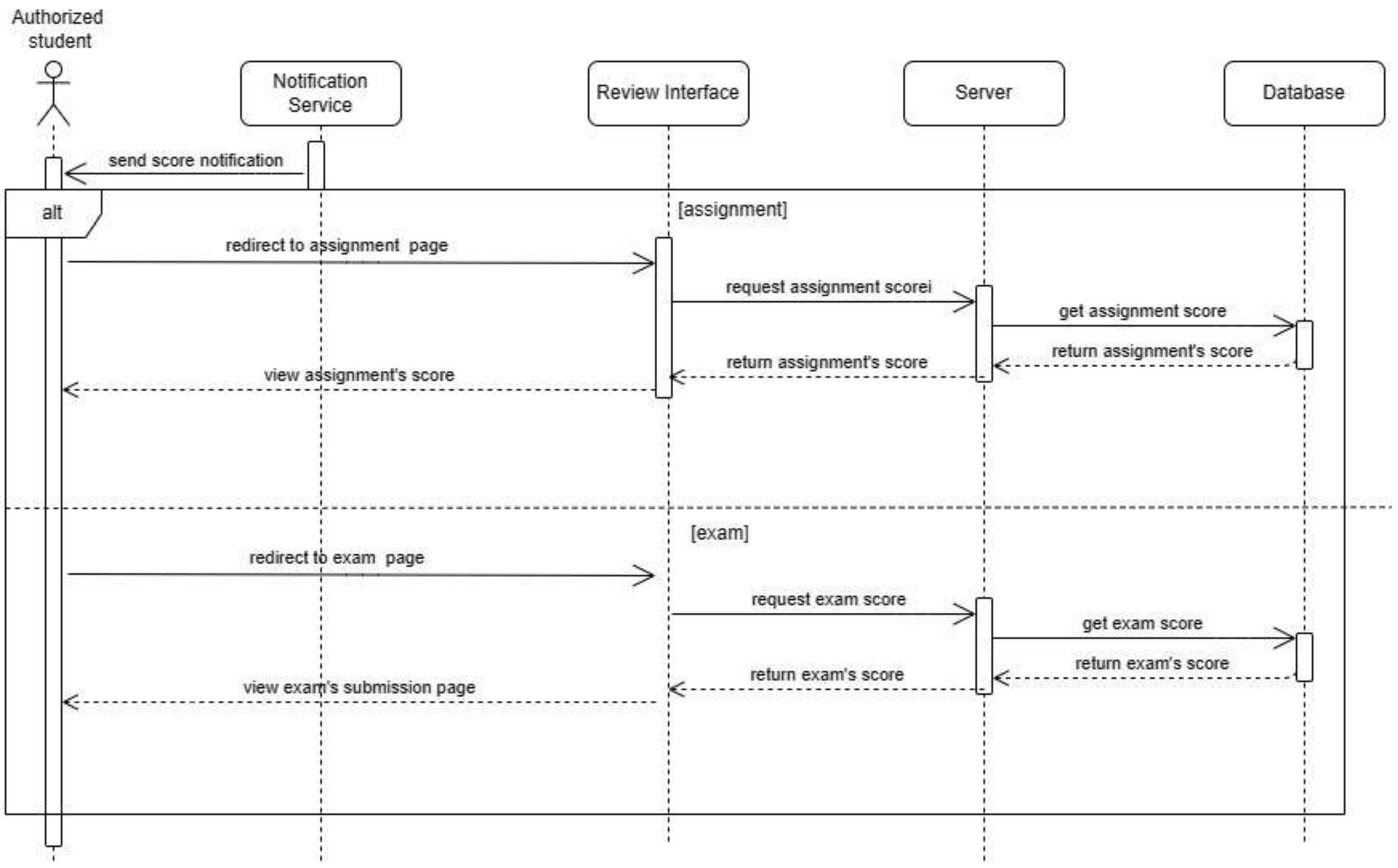


Figure 42 Sequence Diagram Review Module For Student

3.2.9.5 Announcement module

3.2.9.5.1 instructor’s announcement

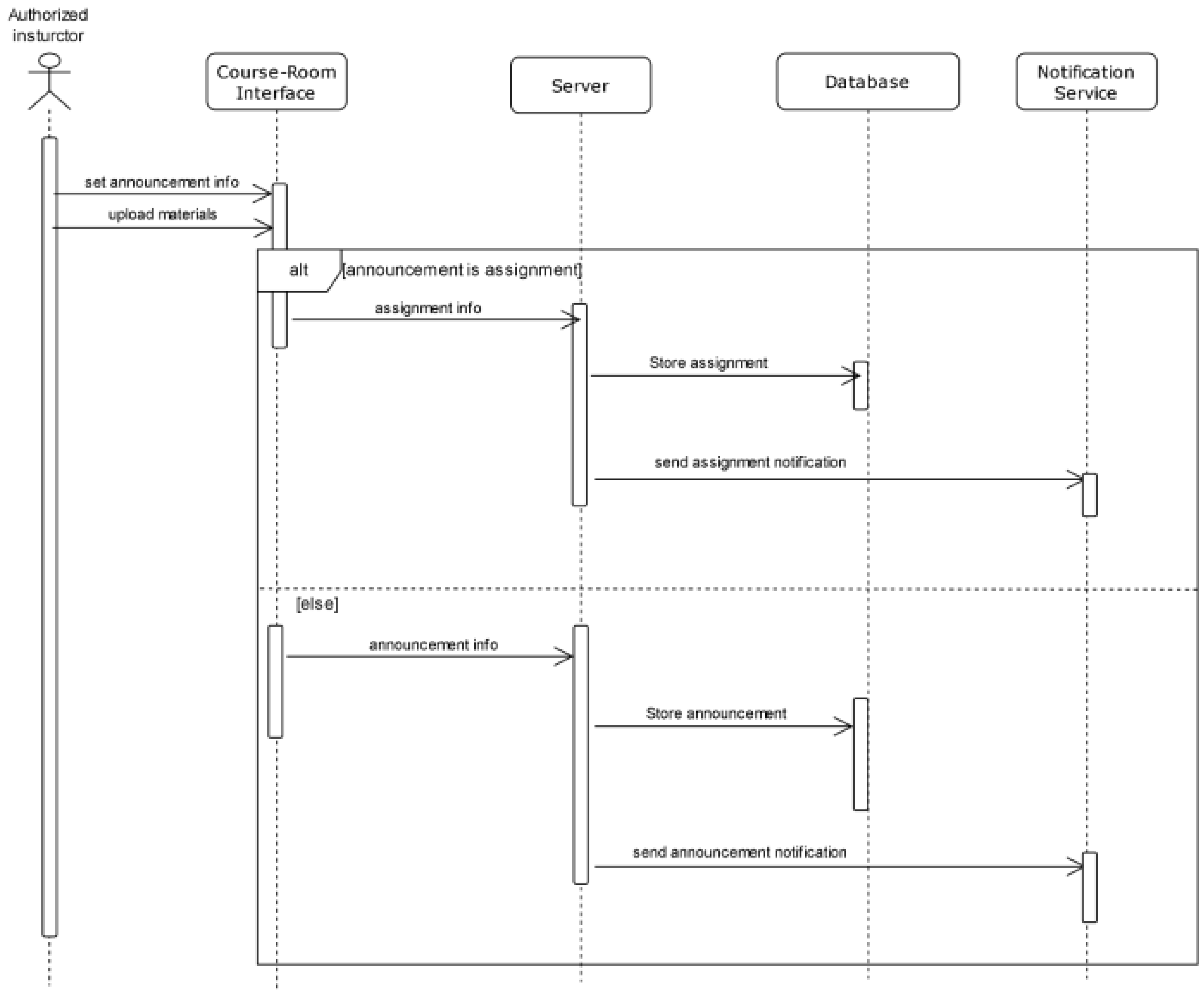


Figure 43 Sequence Diagram Announcement Module For Professor

3.2.9.5.2 Student's announcement

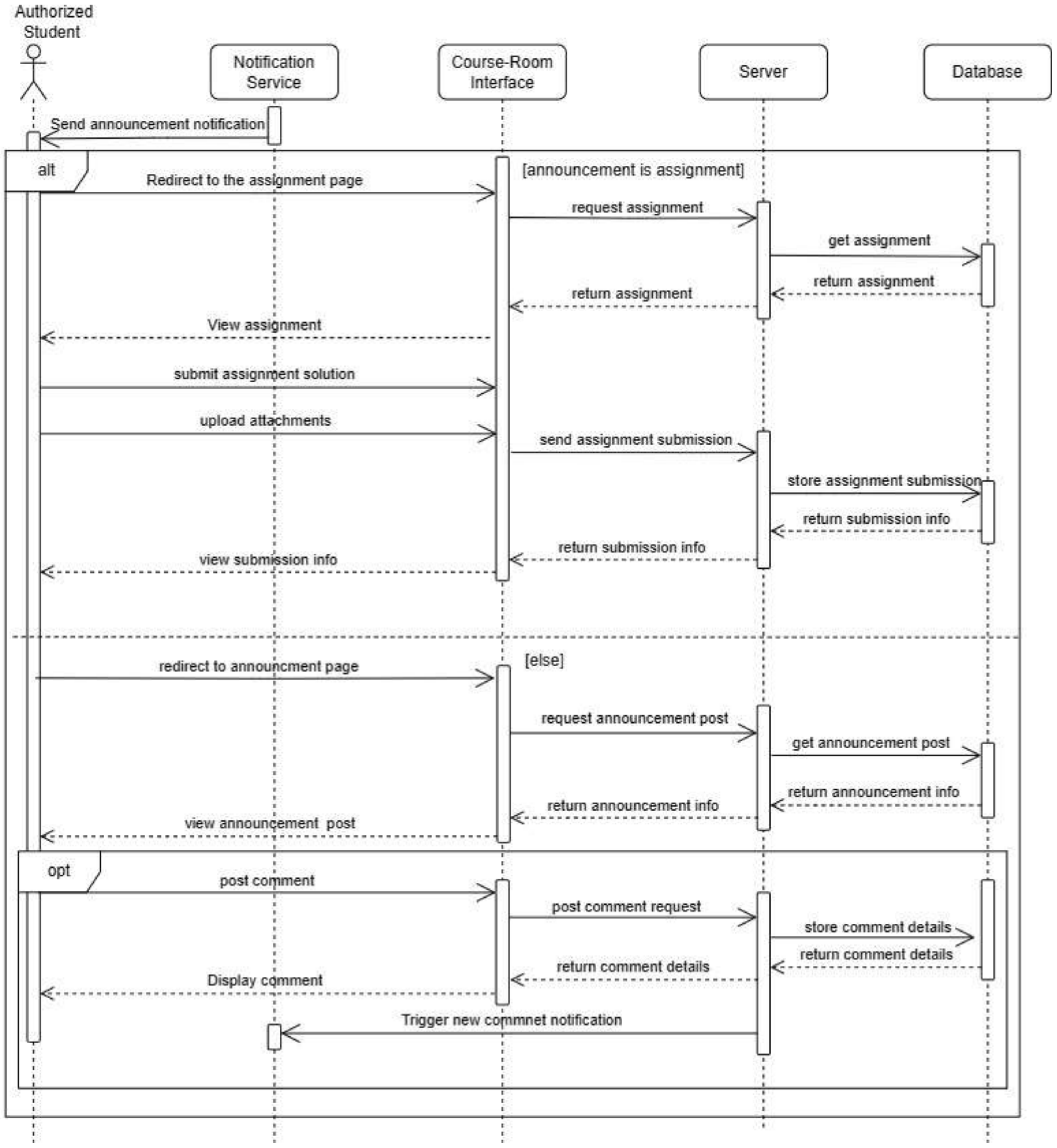


Figure 44 Sequence Diagram Announcement Module For Student

3.2.9.6 CourseWork room settings

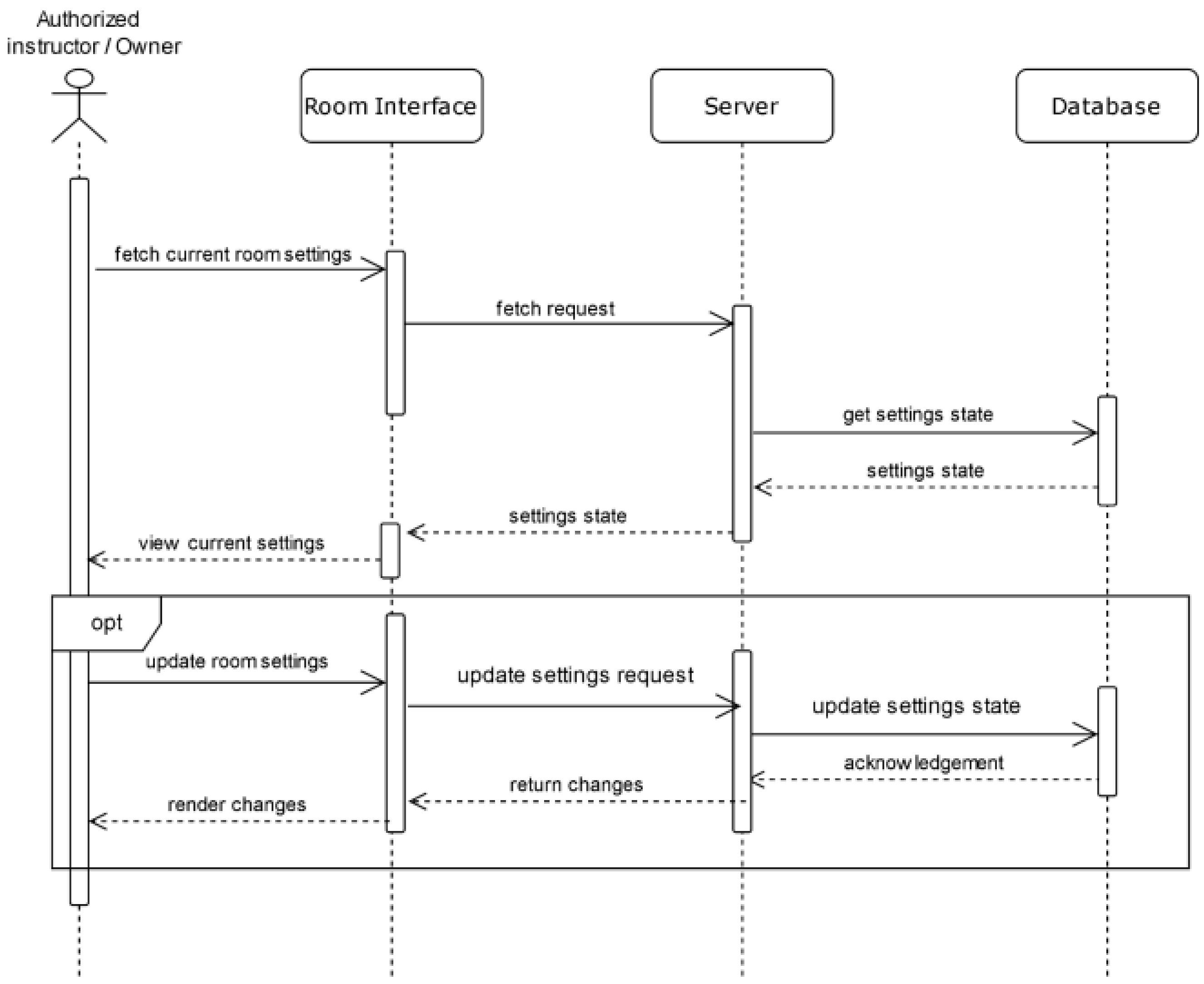


Figure 45 Sequence Diagram CourseWork Room Settings Module

3.2.9.7 Communication

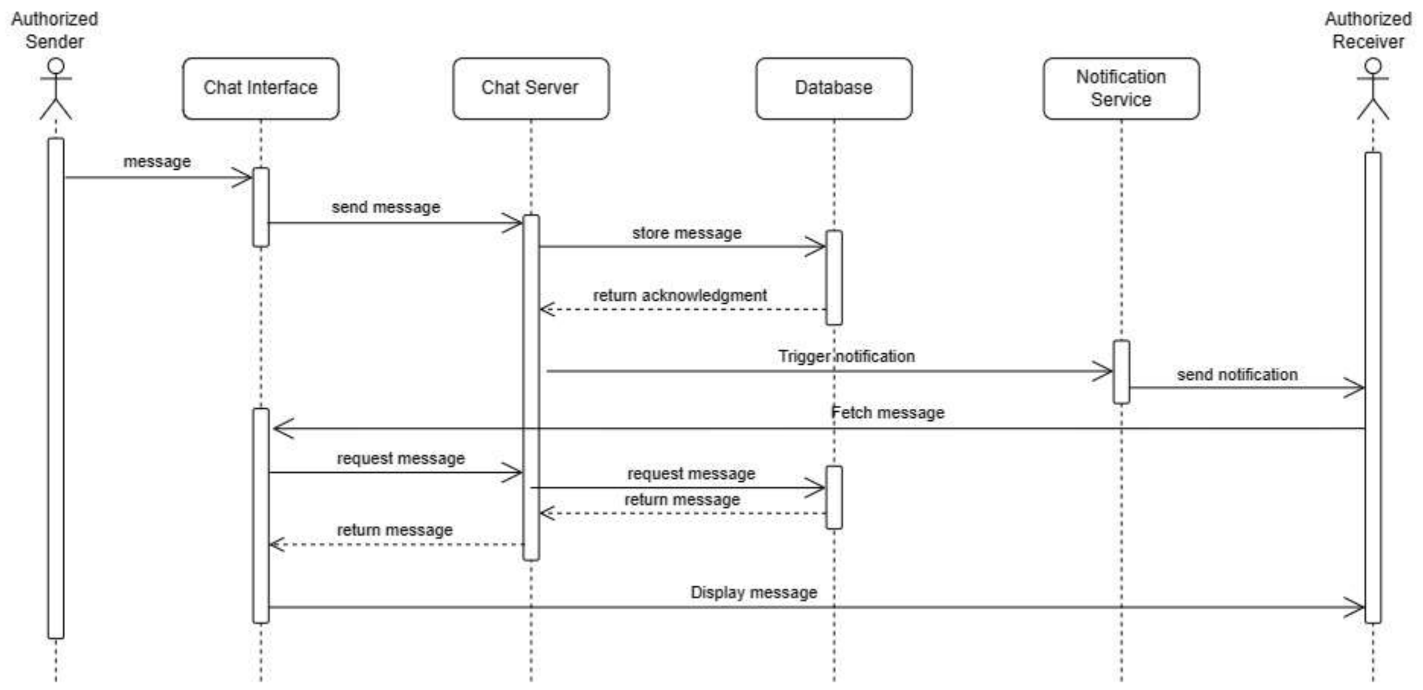


Figure 46 Sequence Diagram Communication Module

3.2.9.8 Feedback

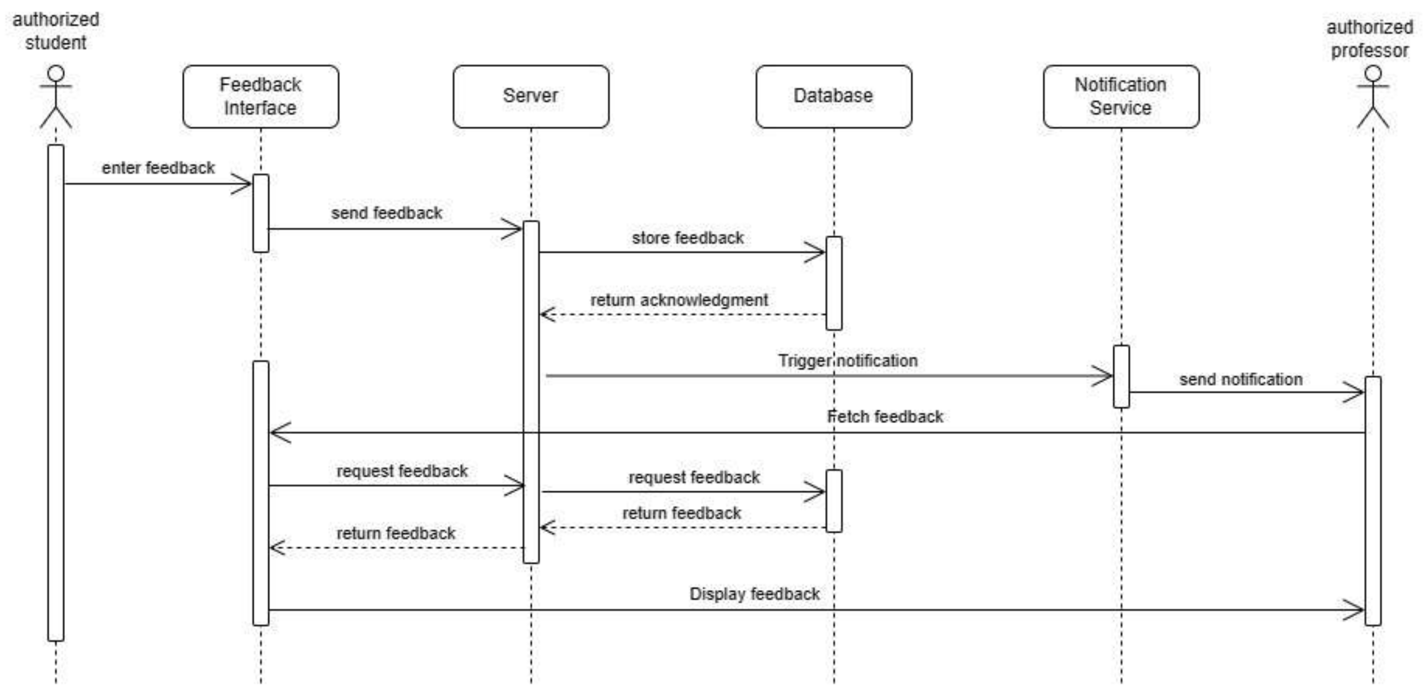


Figure 47 Sequence Diagram Feedback Module

3.2.9.9 Administration

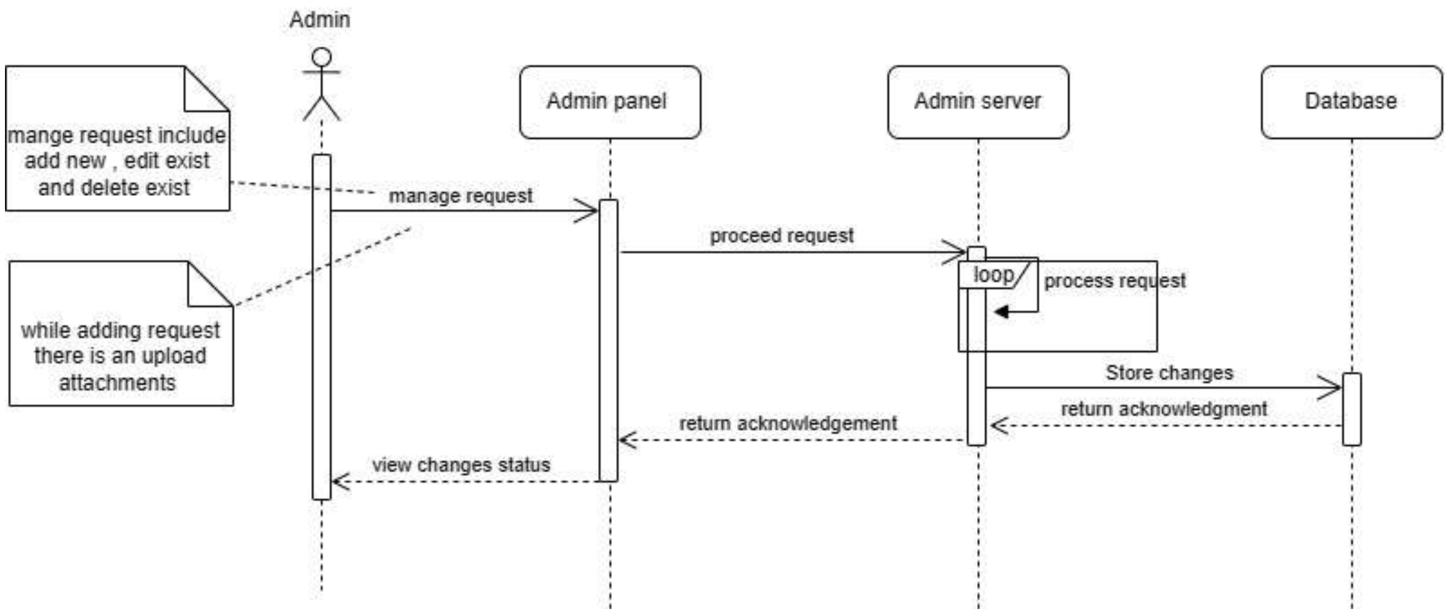


Figure 48 Sequence Diagram Administration Module

4. System Implementation

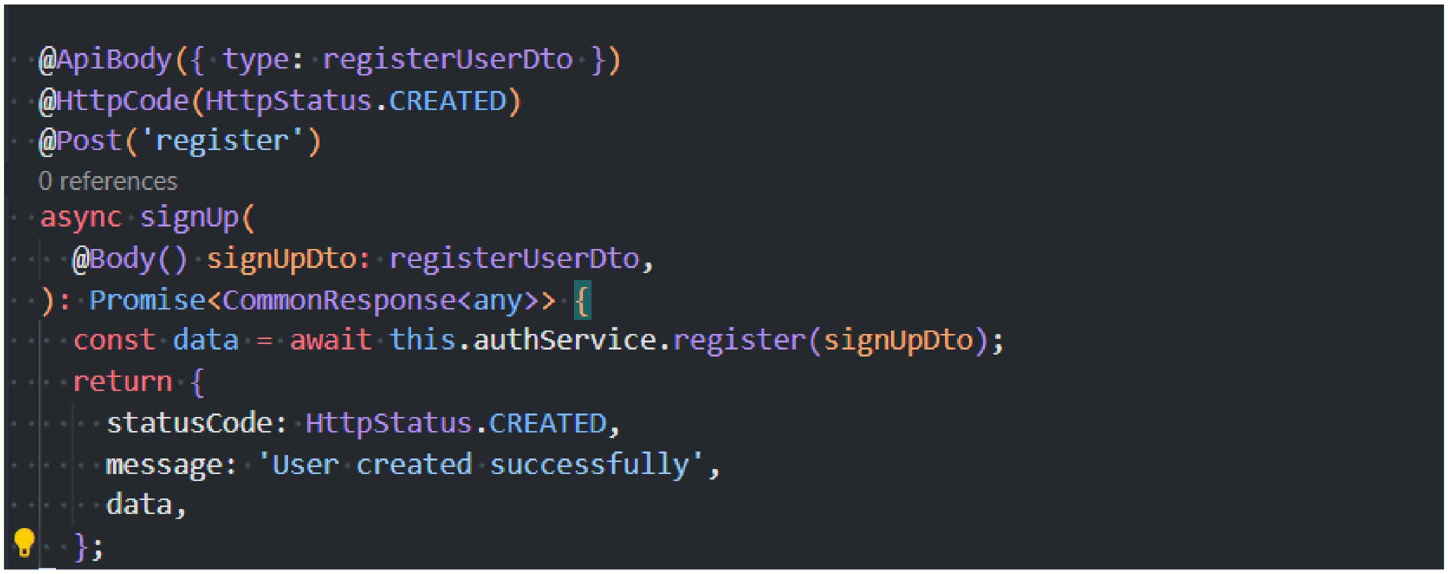
4.1 Tools & Technologies

As mentioned in chapter2 section **Tools & Technologies**

4.2 Implementation of functionalities

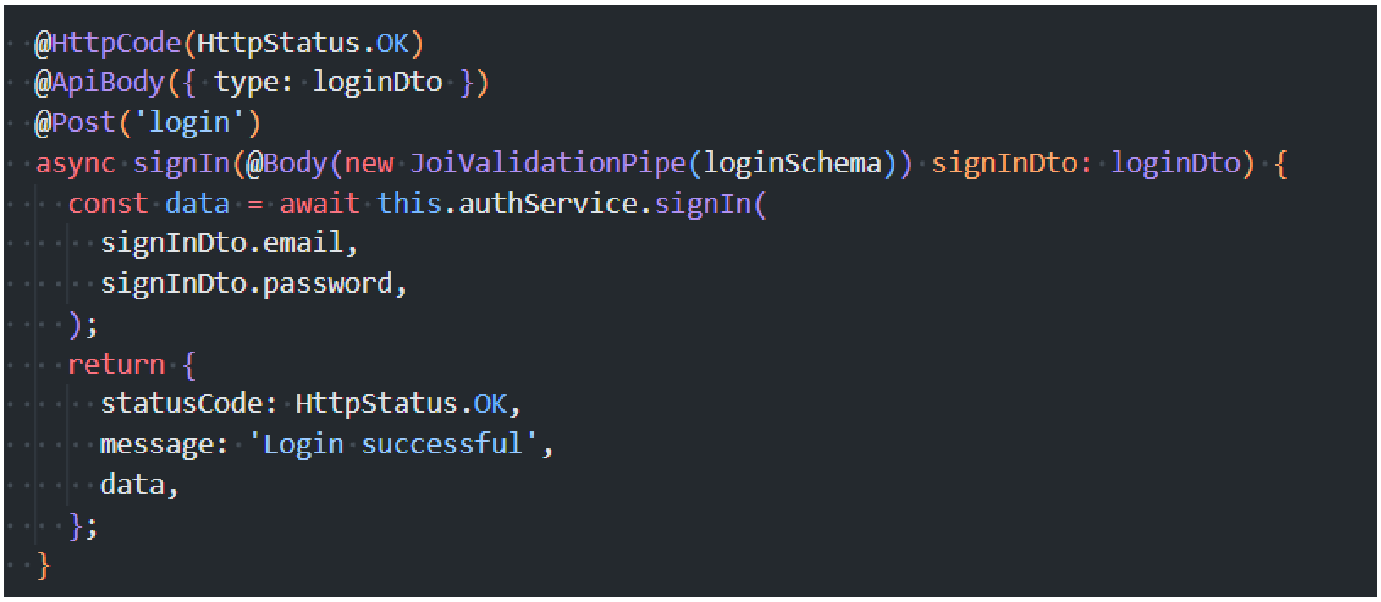
4.2.1 User management

4.2.1.1 User register



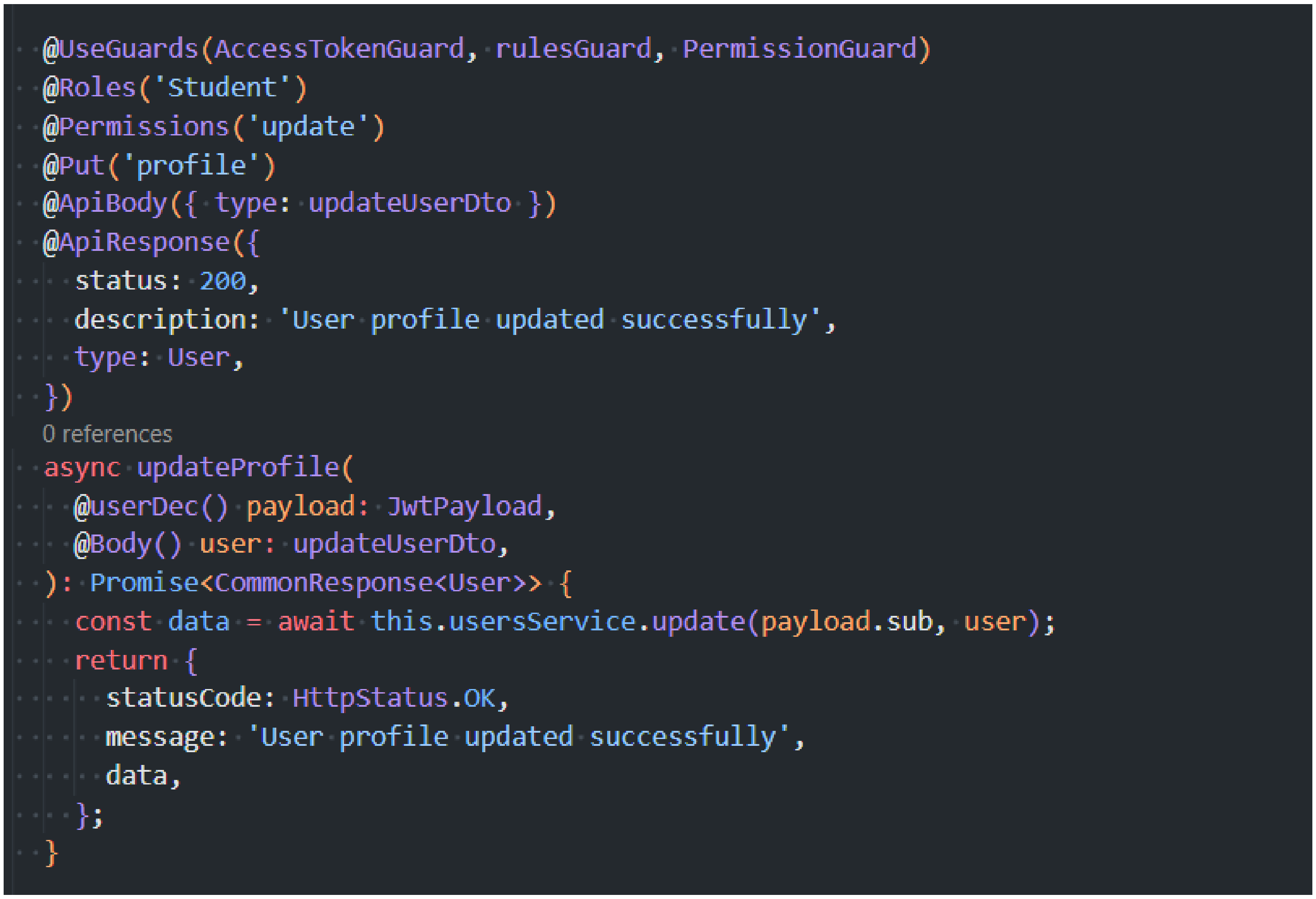
```
@ApiBody({ type: registerUserDto })
@HttpCode(HttpStatus.CREATED)
@Post('register')
0 references
async signUp(
  @Body() signUpDto: registerUserDto,
): Promise<CommonResponse<any>> {
  const data = await this.authService.register(signUpDto);
  return {
    statusCode: HttpStatus.CREATED,
    message: 'User created successfully',
    data,
  };
```

4.2.1.2 User login

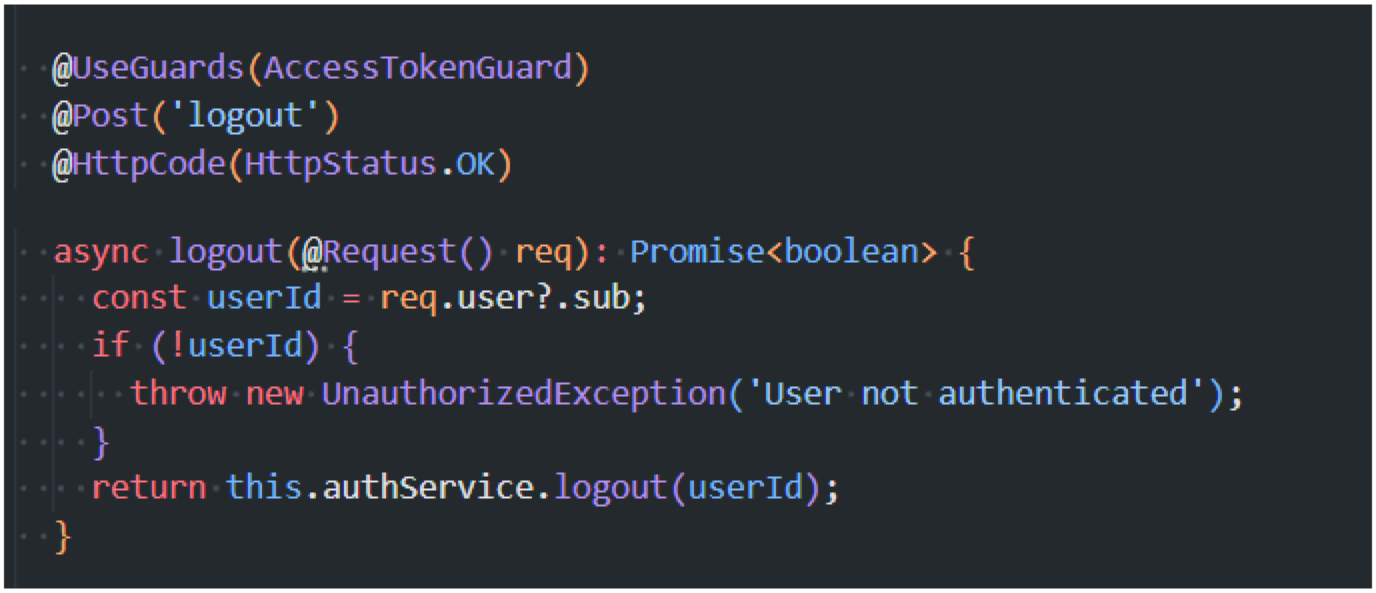


```
@HttpCode(HttpStatus.OK)
@ApiBody({ type: loginDto })
@Post('login')
async signIn(@Body(new JoiValidationPipe(loginSchema)) signInDto: loginDto) {
  const data = await this.authService.signIn(
    signInDto.email,
    signInDto.password,
  );
  return {
    statusCode: HttpStatus.OK,
    message: 'Login successful',
    data,
  };
}
```

4.2.1.3 Account Management & logout

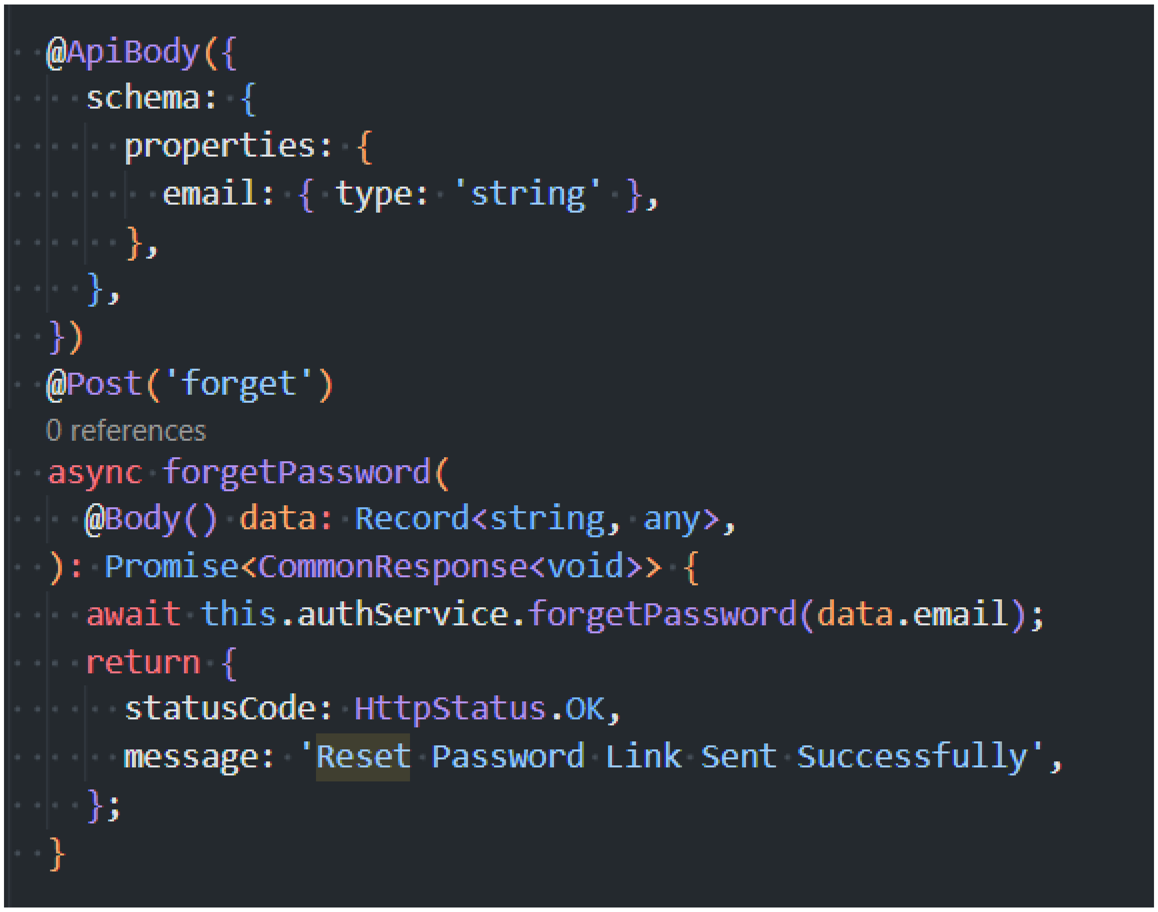


```
@UseGuards(AccessTokenGuard, rulesGuard, PermissionGuard)
@Roles('Student')
@Permissions('update')
@Put('profile')
@ApiBody({ type: updateUserDto })
@ApiResponse({
  status: 200,
  description: 'User profile updated successfully',
  type: User,
})
0 references
async updateProfile(
  @userDec() payload: JwtPayload,
  @Body() user: updateUserDto,
): Promise<CommonResponse<User>> {
  const data = await this.usersService.update(payload.sub, user);
  return {
    statusCode: HttpStatus.OK,
    message: 'User profile updated successfully',
    data,
  };
}
```

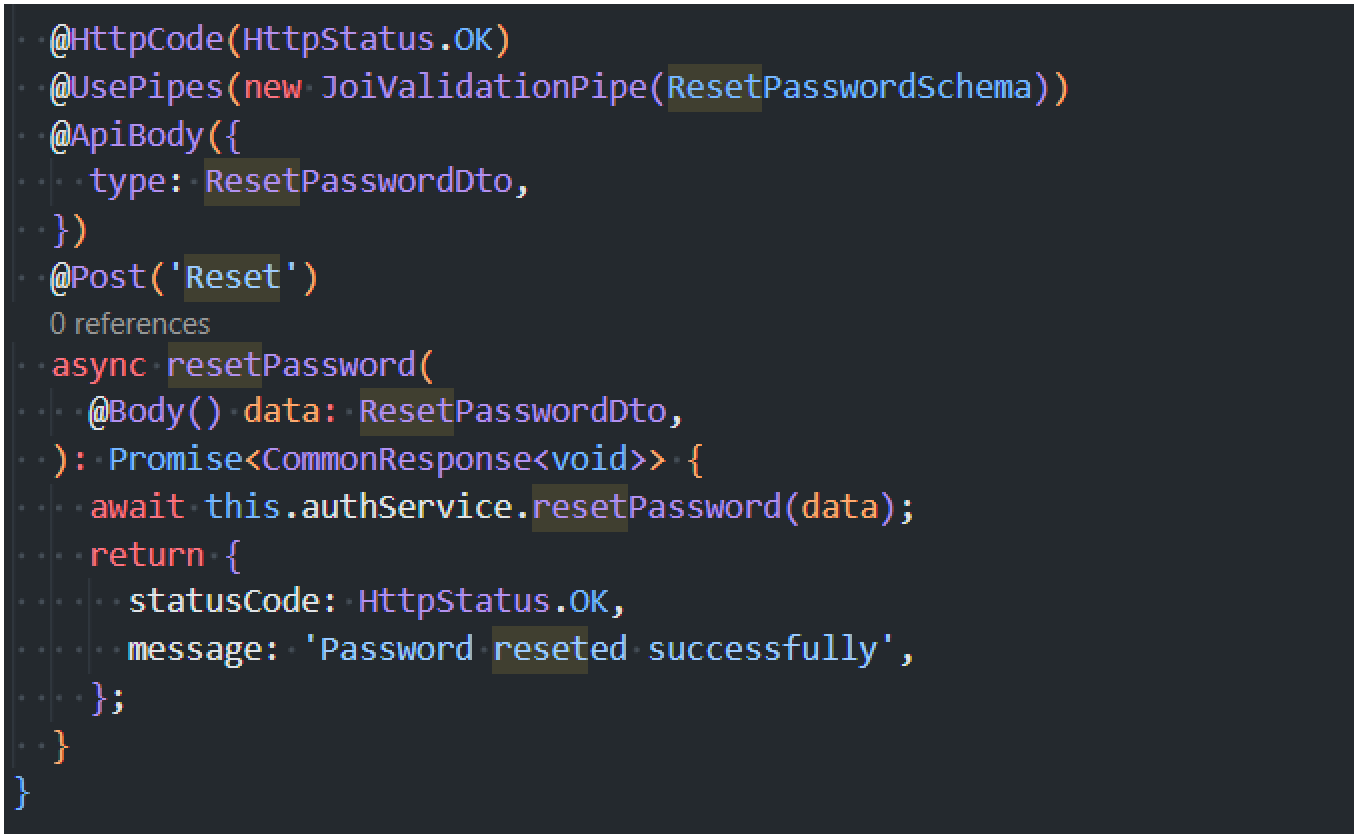


```
@UseGuards(AccessTokenGuard)
@Post('logout')
@HttpCode(HttpStatus.OK)

async logout(@Request() req): Promise<boolean> {
  const userId = req.user?.sub;
  if (!userId) {
    throw new UnauthorizedException('User not authenticated');
  }
  return this.authService.logout(userId);
}
```

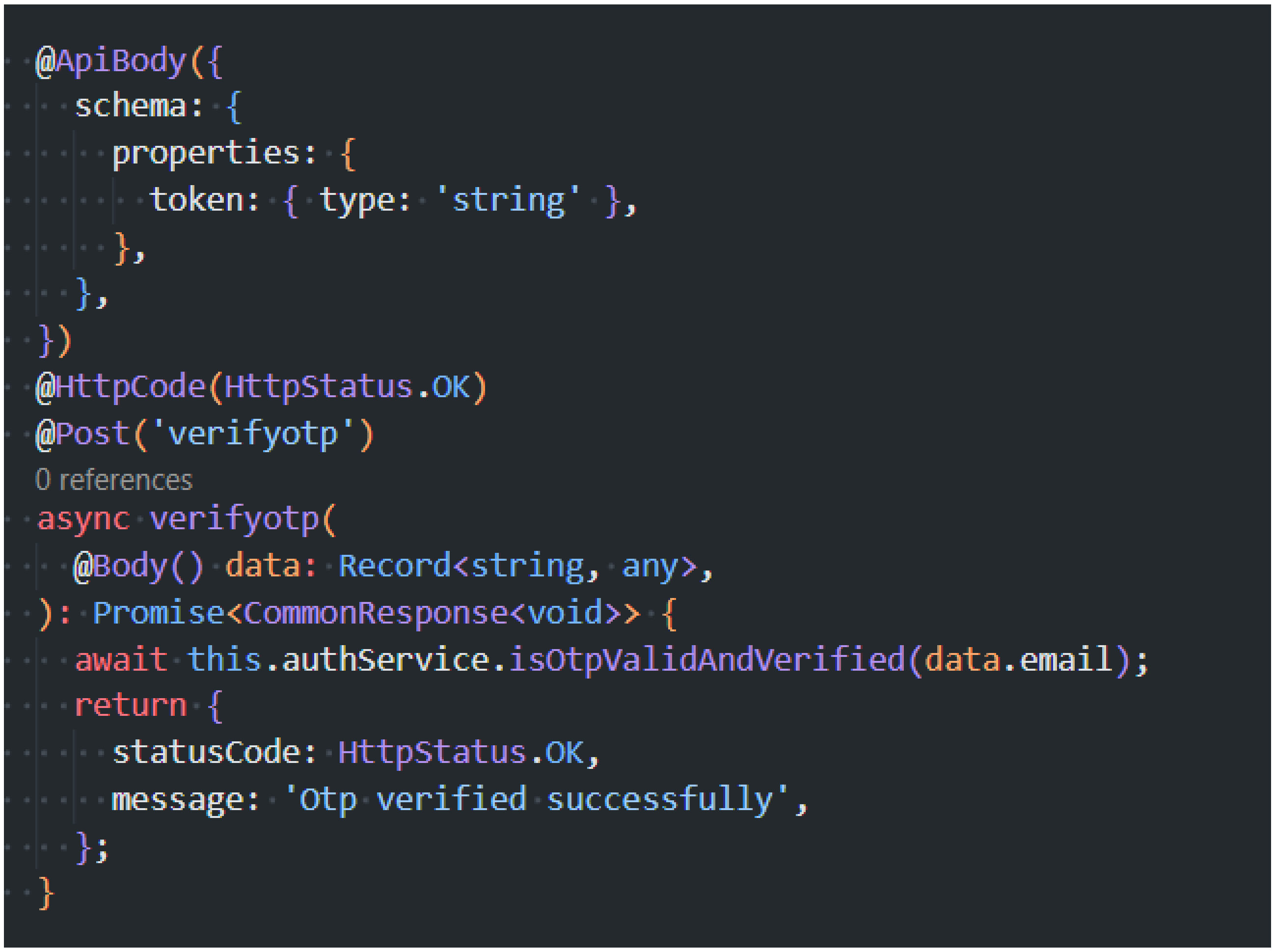
4.2.1.4 Forget Password



```
@ApiBody({
  schema: {
    properties: {
      email: { type: 'string' },
    },
  },
})
@Post('forget')
0 references
async forgetPassword(
  @Body() data: Record<string, any>,
): Promise<CommonResponse<void>> {
  await this.authService.forgetPassword(data.email);
  return {
    statusCode: HttpStatus.OK,
    message: 'Reset Password Link Sent Successfully',
  };
}
```



```
  @HttpCode(HttpStatus.OK)
  @UsePipes(new JoiValidationPipe(ResetPasswordSchema))
  @ApiBody({
    type: ResetPasswordDto,
  })
  @Post('Reset')
  0 references
  async resetPassword(
    @Body() data: ResetPasswordDto,
  ): Promise<CommonResponse<void>> {
    await this.authService.resetPassword(data);
    return {
      statusCode: HttpStatus.OK,
      message: 'Password reseted successfully',
    };
  }
}
```

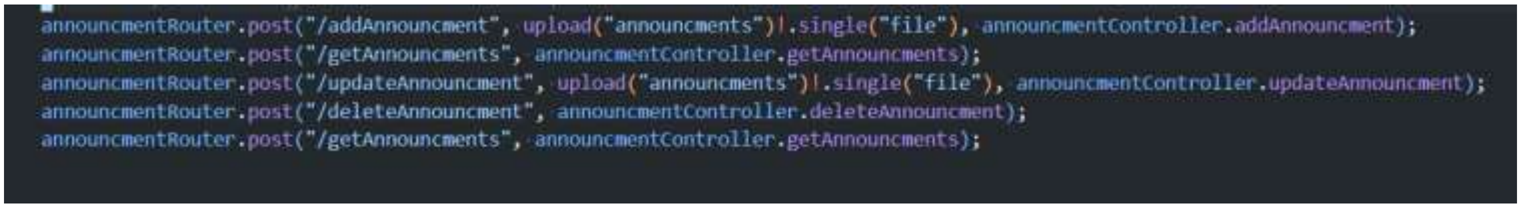


```
  @ApiBody({
    schema: {
      properties: {
        token: { type: 'string' },
      },
    },
  })
  @HttpCode(HttpStatus.OK)
  @Post('verifyotp')
  0 references
  async verifyotp(
    @Body() data: Record<string, any>,
  ): Promise<CommonResponse<void>> {
    await this.authService.isOtpValidAndVerified(data.email);
    return {
      statusCode: HttpStatus.OK,
      message: 'Otp verified successfully',
    };
  }
```

4.2.2 Classroom service

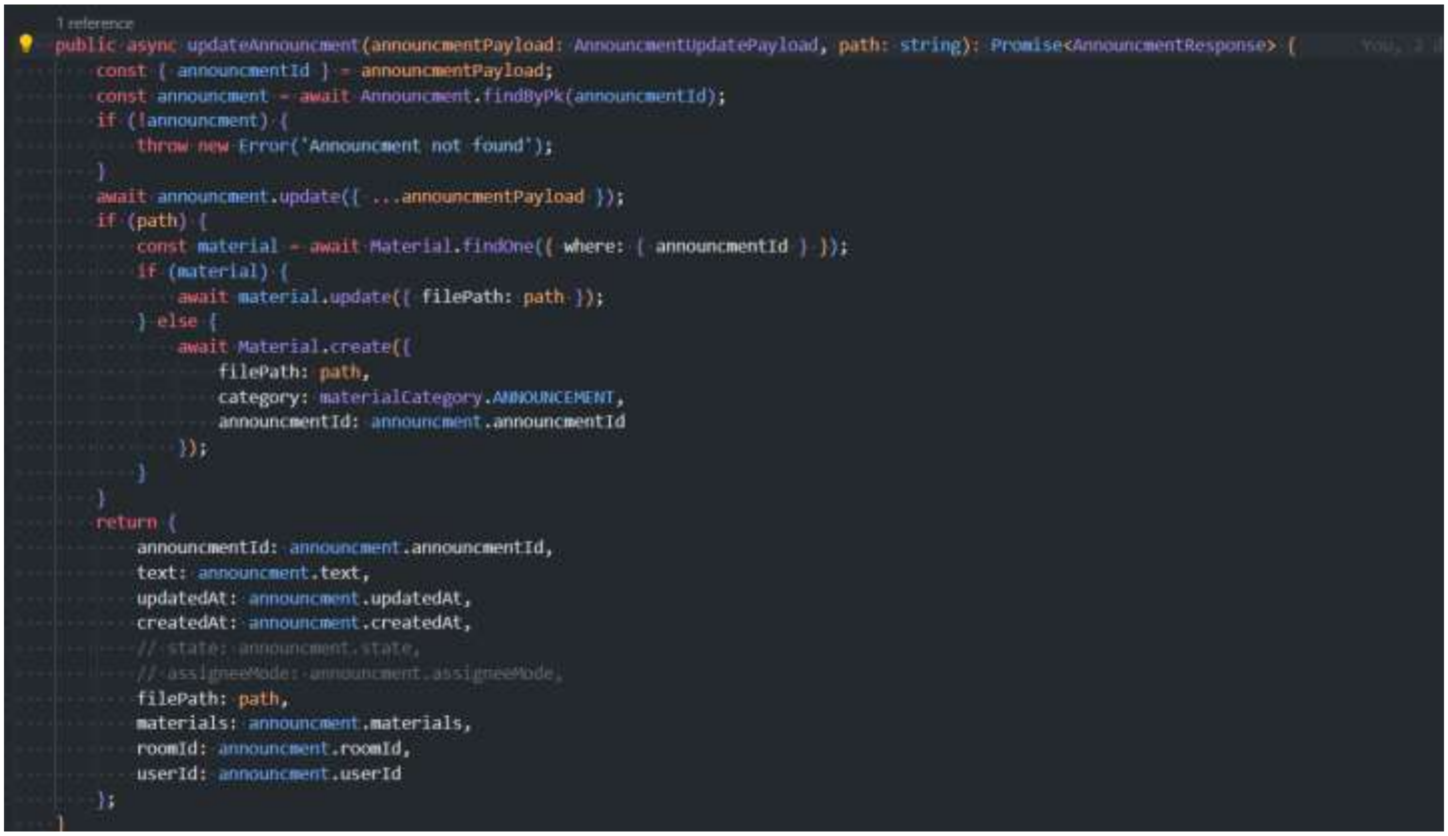
4.2.2.1 Announcements

endpoints:



```
announcmentRouter.post("/addAnnouncment", upload("announcments")!.single("file"), announcmentController.addAnnouncment);
announcmentRouter.post("/getAnnouncments", announcmentController.getAnnouncments);
announcmentRouter.post("/updateAnnouncment", upload("announcments")!.single("file"), announcmentController.updateAnnouncment);
announcmentRouter.post("/deleteAnnouncment", announcmentController.deleteAnnouncment);
announcmentRouter.post("/getAnnouncments", announcmentController.getAnnouncments);
```

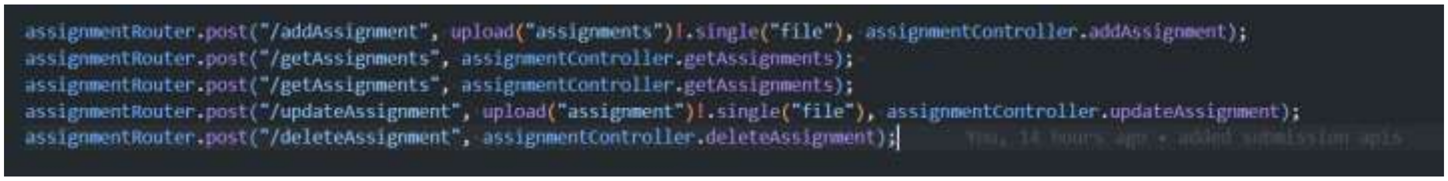
Update service example



```
1 reference
public async updateAnnouncment(announcmentPayload: AnnouncmentUpdatePayload, path: string): Promise<AnnouncmentResponse> {
    const { announcmentId } = announcmentPayload;
    const announcment = await Announcment.findByPk(announcmentId);
    if (!announcment) {
        throw new Error('Announcment not found');
    }
    await announcment.update({ ...announcmentPayload });
    if (path) {
        const material = await Material.findOne({ where: { announcmentId } });
        if (material) {
            await material.update({ filePath: path });
        } else {
            await Material.create({
                filePath: path,
                category: materialCategory.ANNOUNCEMENT,
                announcmentId: announcment.announcmentId
            });
        }
    }
    return {
        announcmentId: announcment.announcmentId,
        text: announcment.text,
        updatedAt: announcment.updatedAt,
        createdAt: announcment.createdAt,
        // state: announcment.state,
        // assigneeMode: announcment.assigneeMode,
        filePath: path,
        materials: announcment.materials,
        roomId: announcment.roomId,
        userId: announcment.userId
    };
}
```

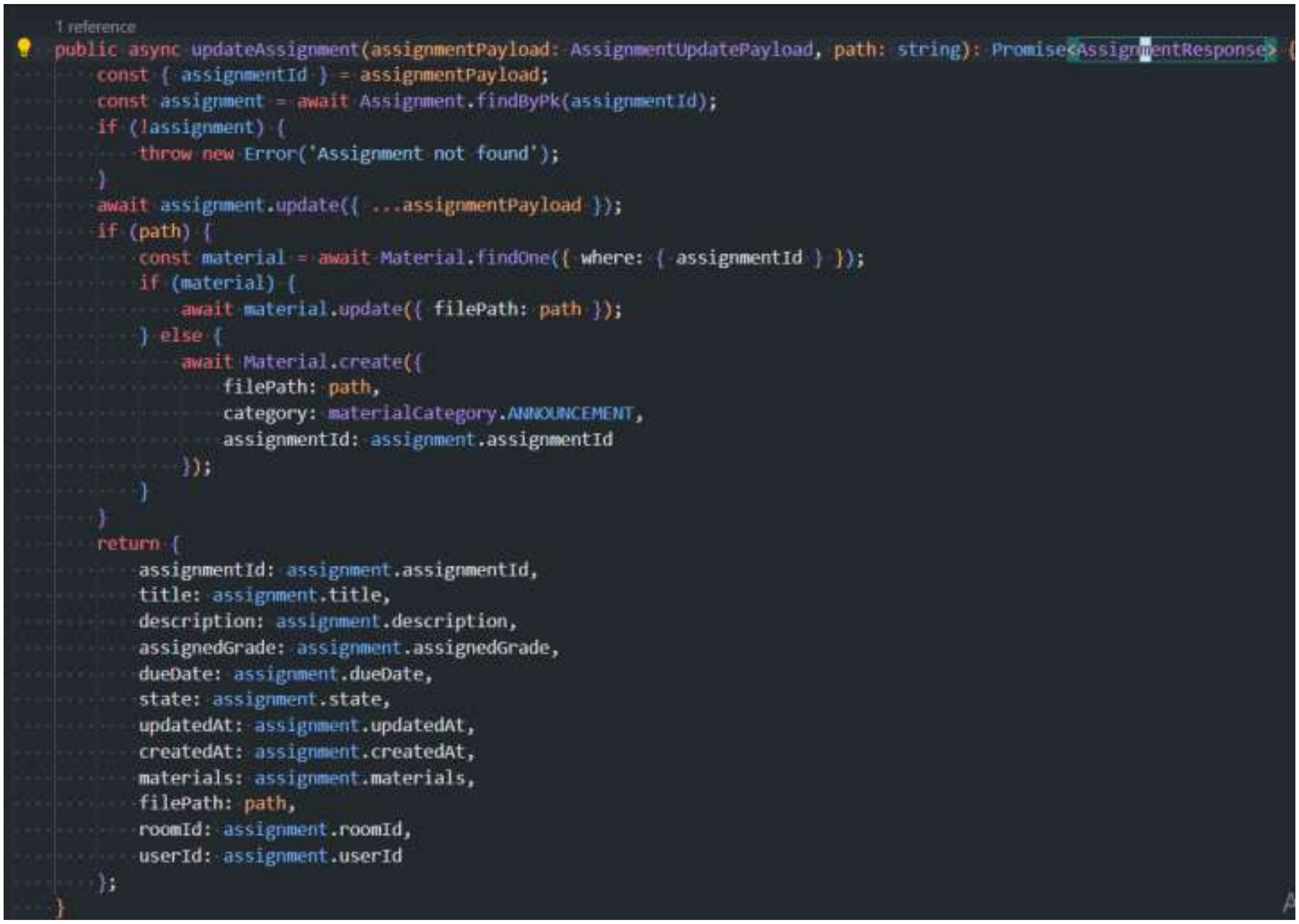
4.2.2.2 Assignments

endpoints:



```
assignmentRouter.post("/addAssignment", upload("assignments")!.single("file"), assignmentController.addAssignment);
assignmentRouter.post("/getAssignments", assignmentController.getAssignments);
assignmentRouter.post("/getAssignments", assignmentController.getAssignments);
assignmentRouter.post("/updateAssignment", upload("assignment")!.single("file"), assignmentController.updateAssignment);
assignmentRouter.post("/deleteAssignment", assignmentController.deleteAssignment);      You, 14 hours ago • added submission apis
```

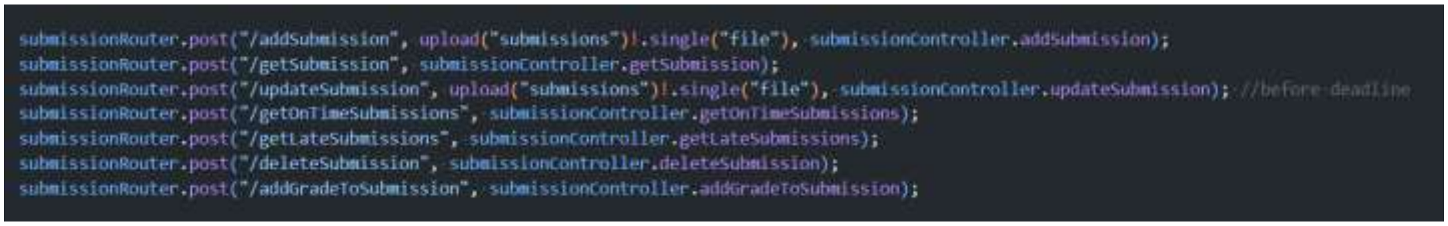
Update service example:



```
1 reference
public async updateAssignment(assignmentPayload: AssignmentUpdatePayload, path: string): Promise<AssignmentResponse> {
    const { assignmentId } = assignmentPayload;
    const assignment = await Assignment.findByPk(assignmentId);
    if (!assignment) {
        throw new Error('Assignment not found');
    }
    await assignment.update({ ...assignmentPayload });
    if (path) {
        const material = await Material.findOne({ where: { assignmentId } });
        if (material) {
            await material.update({ filePath: path });
        } else {
            await Material.create({
                filePath: path,
                category: materialCategory.ANNOUNCEMENT,
                assignmentId: assignment.assignmentId
            });
        }
    }
    return {
        assignmentId: assignment.assignmentId,
        title: assignment.title,
        description: assignment.description,
        assignedGrade: assignment.assignedGrade,
        dueDate: assignment.dueDate,
        state: assignment.state,
        updatedAt: assignment.updatedAt,
        createdAt: assignment.createdAt,
        materials: assignment.materials,
        filePath: path,
        roomId: assignment.roomId,
        userId: assignment.userId
    };
}
```

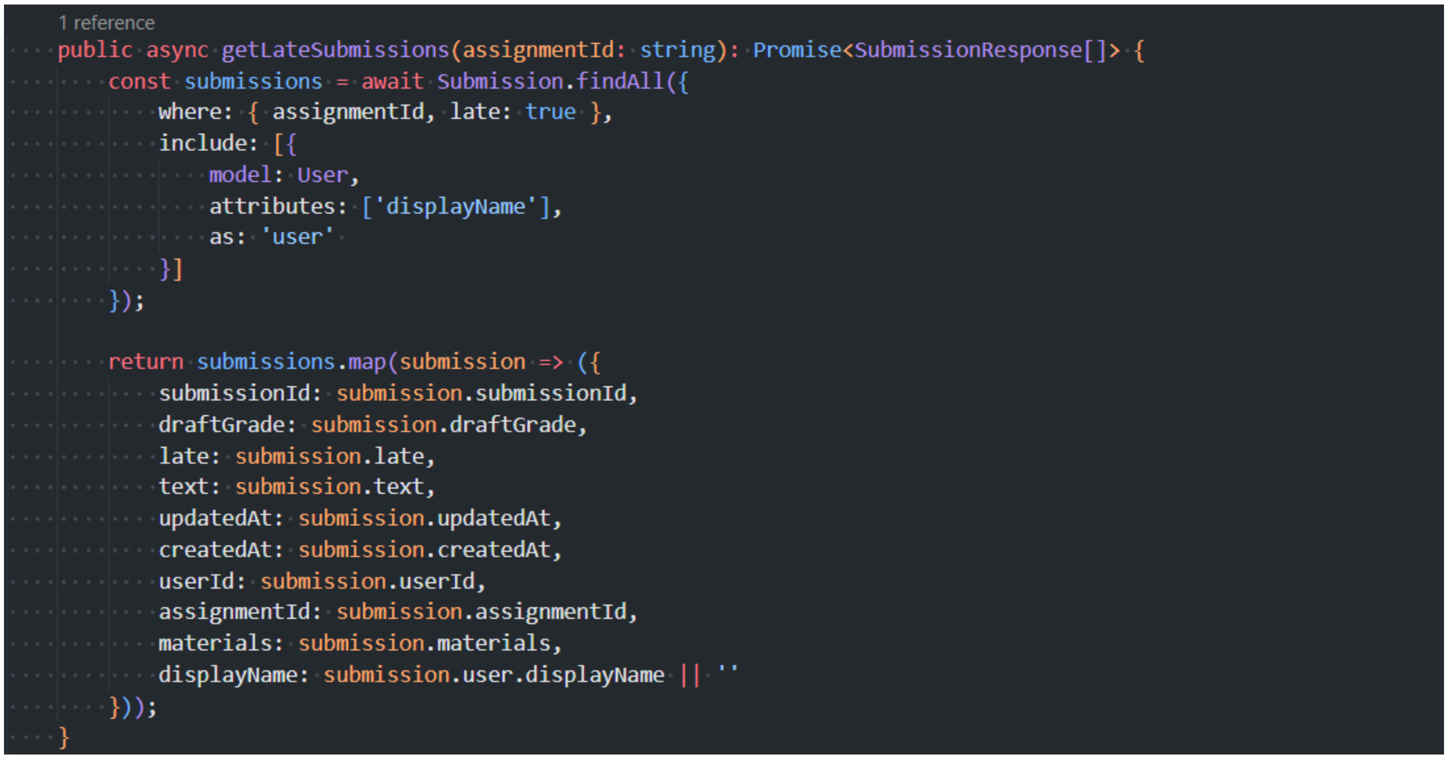
4.2.2.3 Submissions

Apis:



```
submissionRouter.post("/addSubmission", upload("submissions")!.single("file"), submissionController.addSubmission);
submissionRouter.post("/getSubmission", submissionController.getSubmission);
submissionRouter.post("/updateSubmission", upload("submissions")!.single("file"), submissionController.updateSubmission); //before deadline
submissionRouter.post("/getOnTimeSubmissions", submissionController.getOnTimeSubmissions);
submissionRouter.post("/getLateSubmissions", submissionController.getLateSubmissions);
submissionRouter.post("/deleteSubmission", submissionController.deleteSubmission);
submissionRouter.post("/addGradeToSubmission", submissionController.addGradeToSubmission);
```

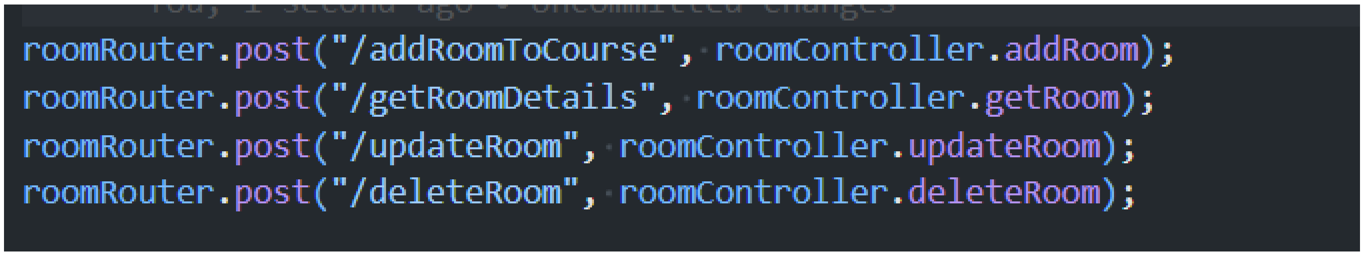
Get Late Submissions service example:



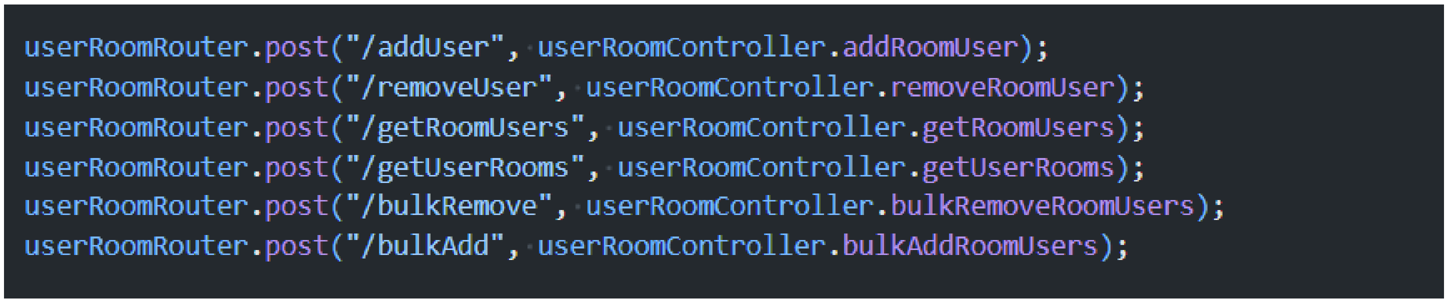
```
    1 reference
    public async getLateSubmissions(assignmentId: string): Promise<SubmissionResponse[]> {
        const submissions = await Submission.findAll({
            where: { assignmentId, late: true },
            include: [{
                model: User,
                attributes: ['displayName'],
                as: 'user'
            }]
        });

        return submissions.map(submission => ({
            submissionId: submission.submissionId,
            draftGrade: submission.draftGrade,
            late: submission.late,
            text: submission.text,
            updatedAt: submission.updatedAt,
            createdAt: submission.createdAt,
            userId: submission.userId,
            assignmentId: submission.assignmentId,
            materials: submission.materials,
            displayName: submission.user.displayName || ''
        }));
    }
```

4.2.2.4 Room management

Apis:

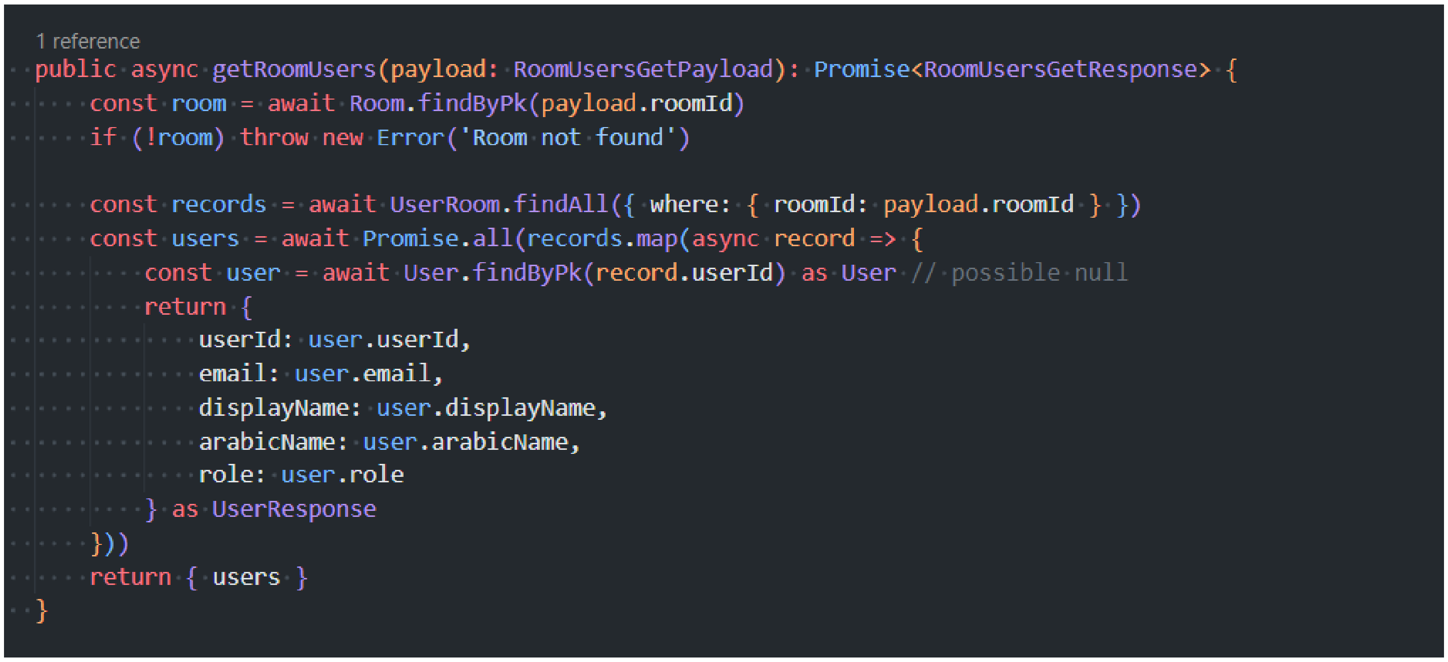


```
        You, 1 second ago • Uncommitted changes
roomRouter.post("/addRoomToCourse", roomController.addRoom);
roomRouter.post("/getRoomDetails", roomController.getRoom);
roomRouter.post("/updateRoom", roomController.updateRoom);
roomRouter.post("/deleteRoom", roomController.deleteRoom);
```



```
userRoomRouter.post("/addUser", userRoomController.addRoomUser);
userRoomRouter.post("/removeUser", userRoomController.removeRoomUser);
userRoomRouter.post("/getRoomUsers", userRoomController.getRoomUsers);
userRoomRouter.post("/getUserRooms", userRoomController.getUserRooms);
userRoomRouter.post("/bulkRemove", userRoomController.bulkRemoveRoomUsers);
userRoomRouter.post("/bulkAdd", userRoomController.bulkAddRoomUsers);
```

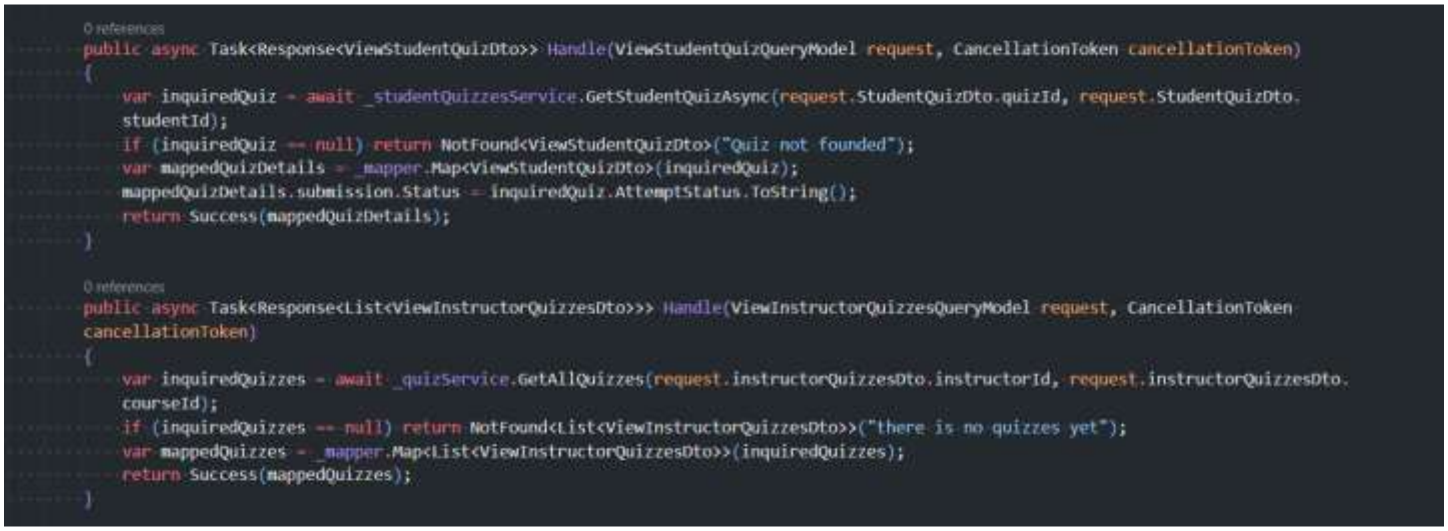
Get Room Users Service example:



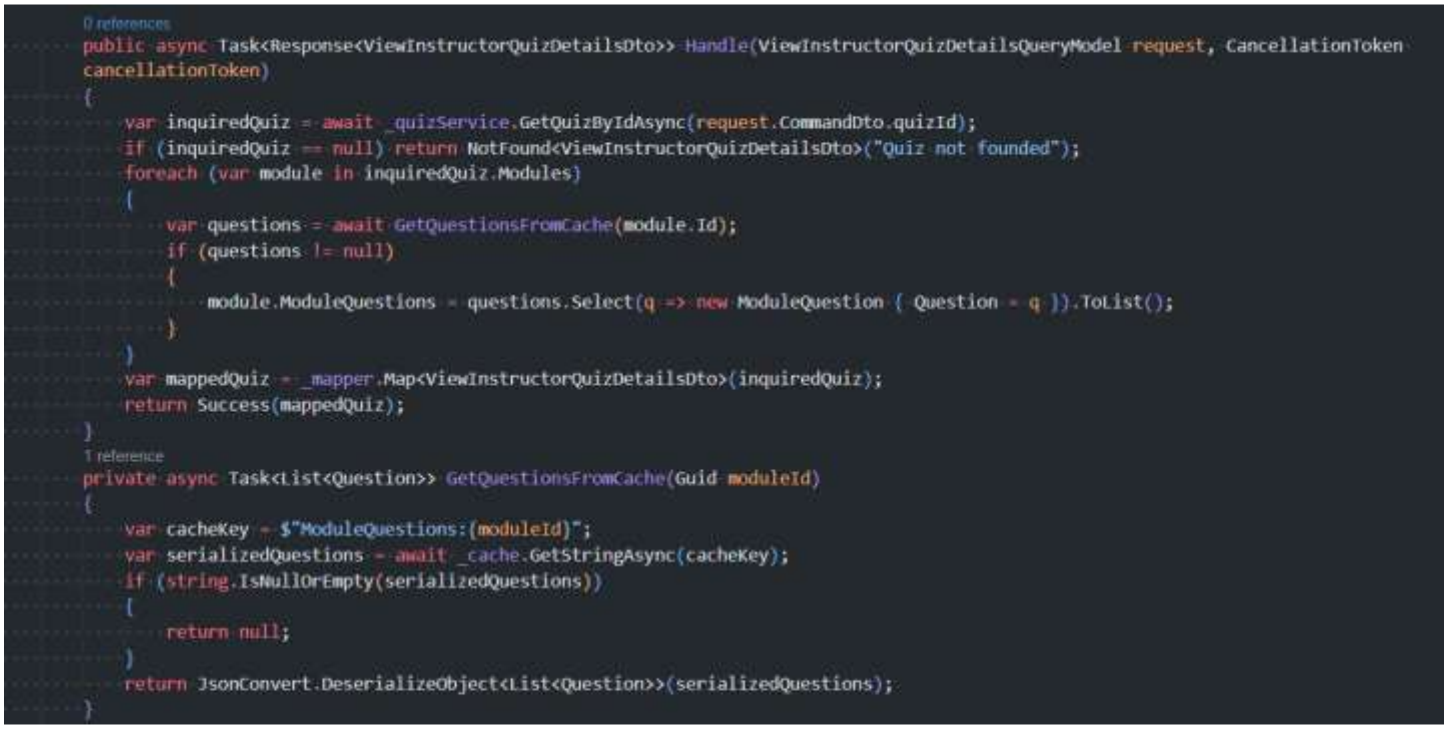
```
1 reference
public async getRoomUsers(payload: RoomUsersGetPayload): Promise<RoomUsersGetResponse> {
    const room = await Room.findByPk(payload.roomId)
    if (!room) throw new Error('Room not found')

    const records = await UserRoom.findAll({ where: { roomId: payload.roomId } })
    const users = await Promise.all(records.map(async record => {
        const user = await User.findByPk(record.userId) as User // possible null
        return {
            userId: user.userId,
            email: user.email,
            displayName: user.displayName,
            arabicName: user.arabicName,
            role: user.role
        } as UserResponse
    }))
    return { users }
}
```

4.2.3 Exam Service

Handlers:

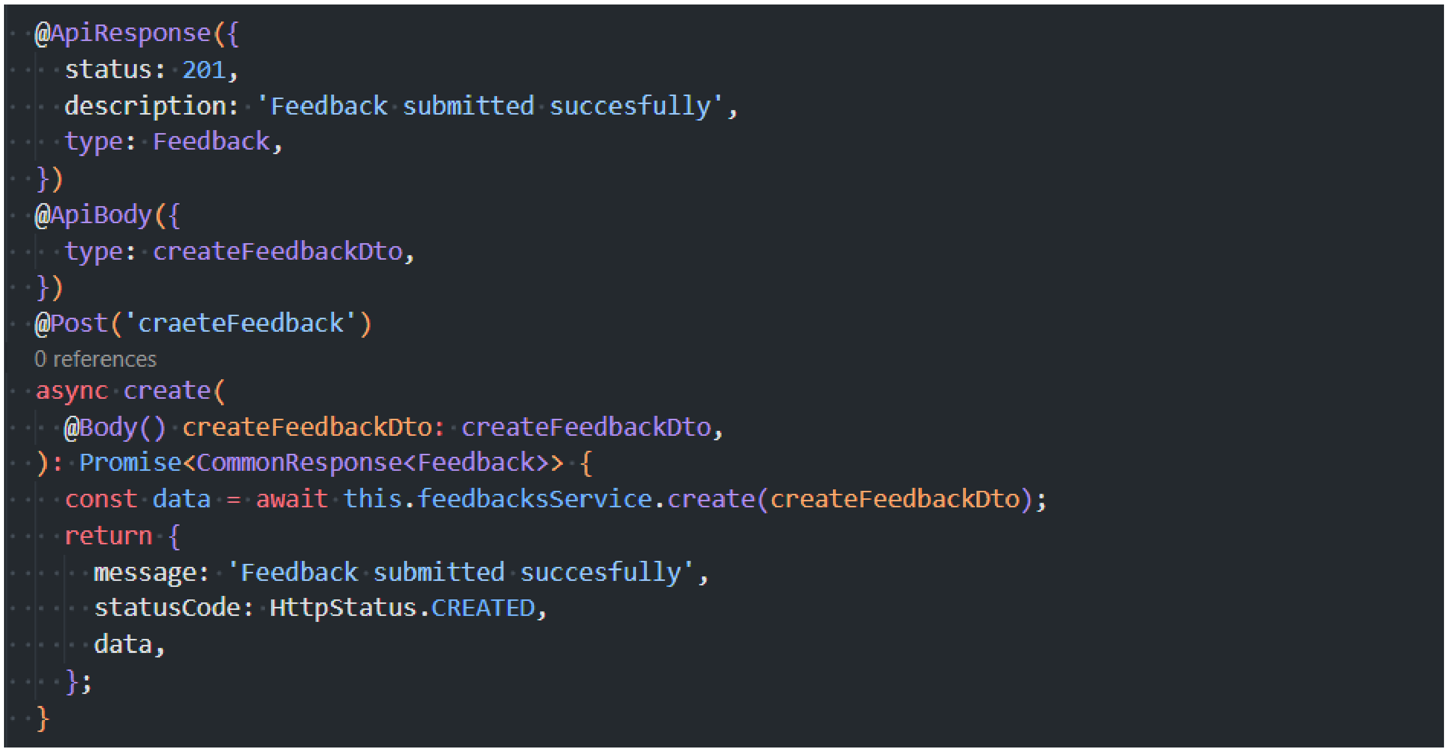


```
0 references
public async Task<Response<ViewStudentQuizDto>> Handle(ViewStudentQuizQueryModel request, CancellationToken cancellationToken)
{
    var inquiredQuiz = await _studentQuizzesService.GetStudentQuizAsync(request.StudentQuizDto.quizId, request.StudentQuizDto.
    studentId);
    if (inquiredQuiz == null) return NotFound<ViewStudentQuizDto>("Quiz not founded");
    var mappedQuizDetails = _mapper.Map<ViewStudentQuizDto>(inquiredQuiz);
    mappedQuizDetails.submission.Status = inquiredQuiz.AttemptStatus.ToString();
    return Success(mappedQuizDetails);
}

0 references
public async Task<Response<List<ViewInstructorQuizzesDto>>> Handle(ViewInstructorQuizzesQueryModel request, CancellationToken
cancellationToken)
{
    var inquiredQuizzes = await _quizService.GetAllQuizzes(request.instructorQuizzesDto.instructorId, request.instructorQuizzesDto.
    courseId);
    if (inquiredQuizzes == null) return NotFound<List<ViewInstructorQuizzesDto>>("there is no quizzes yet");
    var mappedQuizzes = _mapper.Map<List<ViewInstructorQuizzesDto>>(inquiredQuizzes);
    return Success(mappedQuizzes);
}
```

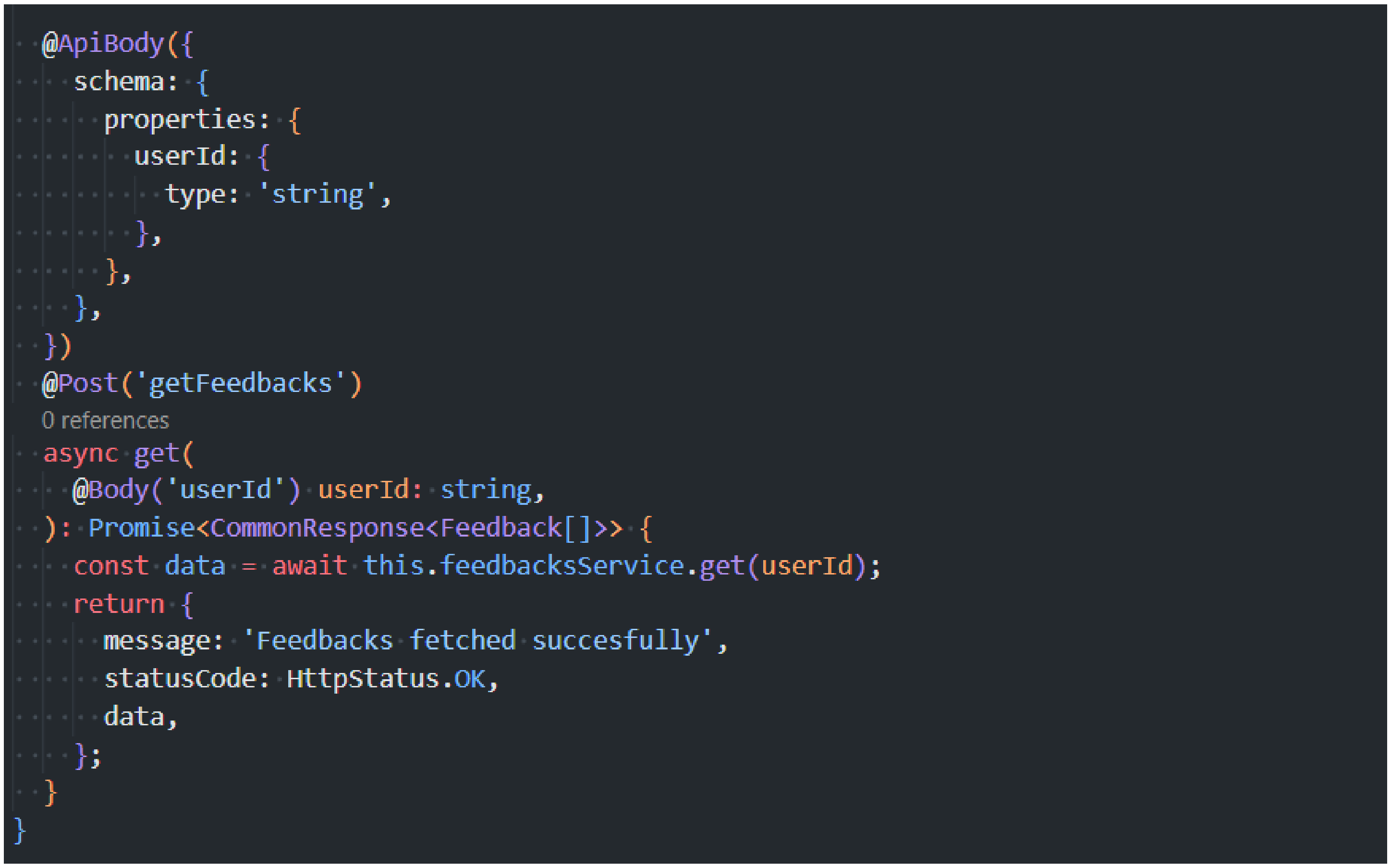


```
0 references
public async Task<Response<ViewInstructorQuizDetailsDto>> Handle(ViewInstructorQuizDetailsQueryModel request, CancellationToken
cancellationToken)
{
    var inquiredQuiz = await _quizService.GetQuizByIdAsync(request.CommandDto.quizId);
    if (inquiredQuiz == null) return NotFound<ViewInstructorQuizDetailsDto>("Quiz not founded");
    foreach (var module in inquiredQuiz.Modules)
    {
        var questions = await GetQuestionsFromCache(module.Id);
        if (questions != null)
        {
            module.ModuleQuestions = questions.Select(q => new ModuleQuestion { Question = q }).ToList();
        }
    }
    var mappedQuiz = _mapper.Map<ViewInstructorQuizDetailsDto>(inquiredQuiz);
    return Success(mappedQuiz);
}
1 reference
private async Task<List<Question>> GetQuestionsFromCache(Guid moduleId)
{
    var cacheKey = $"ModuleQuestions:{moduleId}";
    var serializedQuestions = await _cache.GetStringAsync(cacheKey);
    if (string.IsNullOrEmpty(serializedQuestions))
    {
        return null;
    }
    return JsonConvert.DeserializeObject<List<Question>>(serializedQuestions);
}
```

4.2.4 Feedback Service

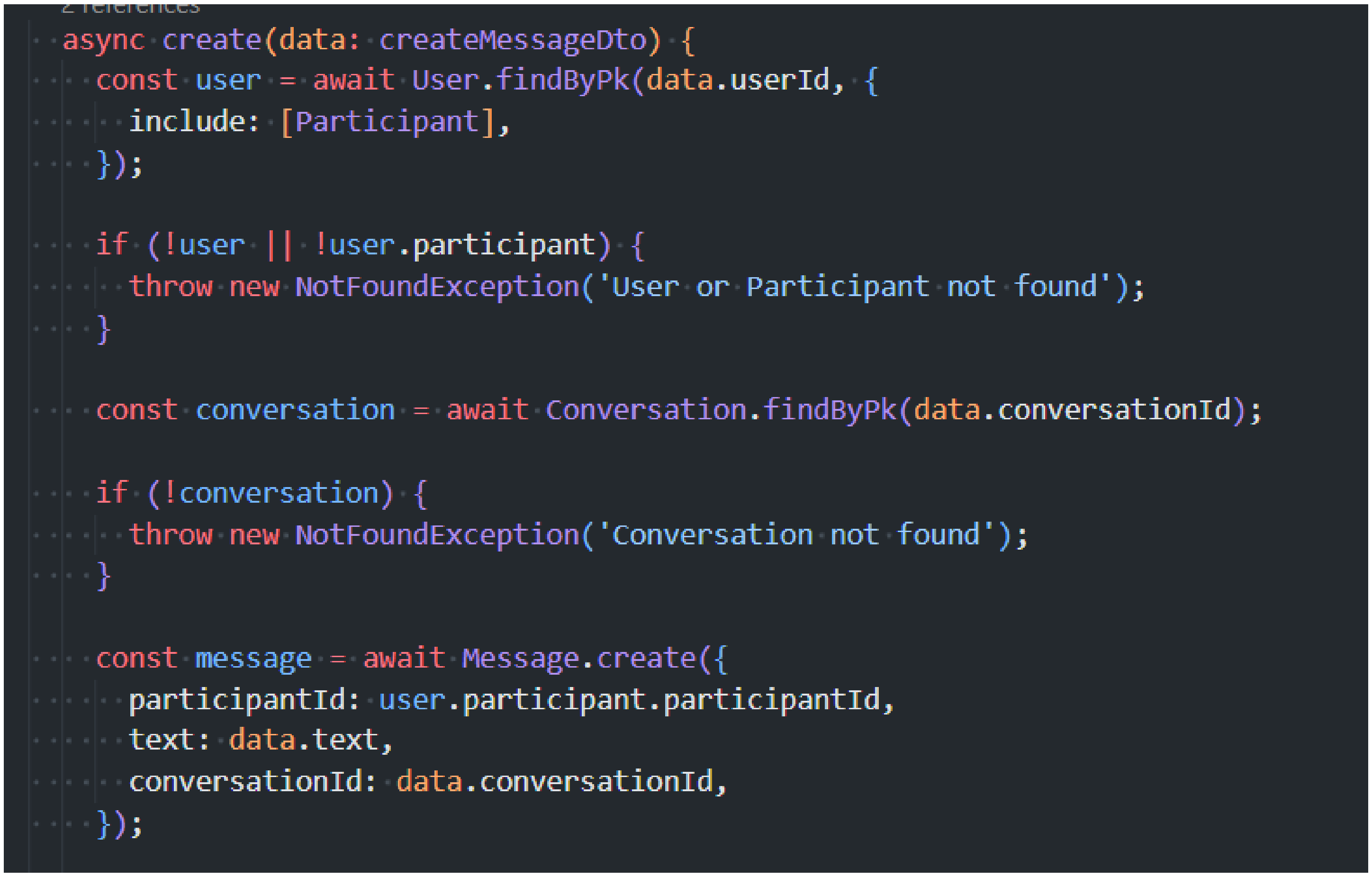


```
  @ApiResponse({
    status: 201,
    description: 'Feedback submitted succesfully',
    type: Feedback,
  })
  @ApiBody({
    type: createFeedbackDto,
  })
  @Post('craeteFeedback')
  0 references
  async create(
    @Body() createFeedbackDto: createFeedbackDto,
  ): Promise<CommonResponse<Feedback>> {
    const data = await this.feedbacksService.create(createFeedbackDto);
    return {
      message: 'Feedback submitted succesfully',
      statusCode: HttpStatus.CREATED,
      data,
    };
  }
```

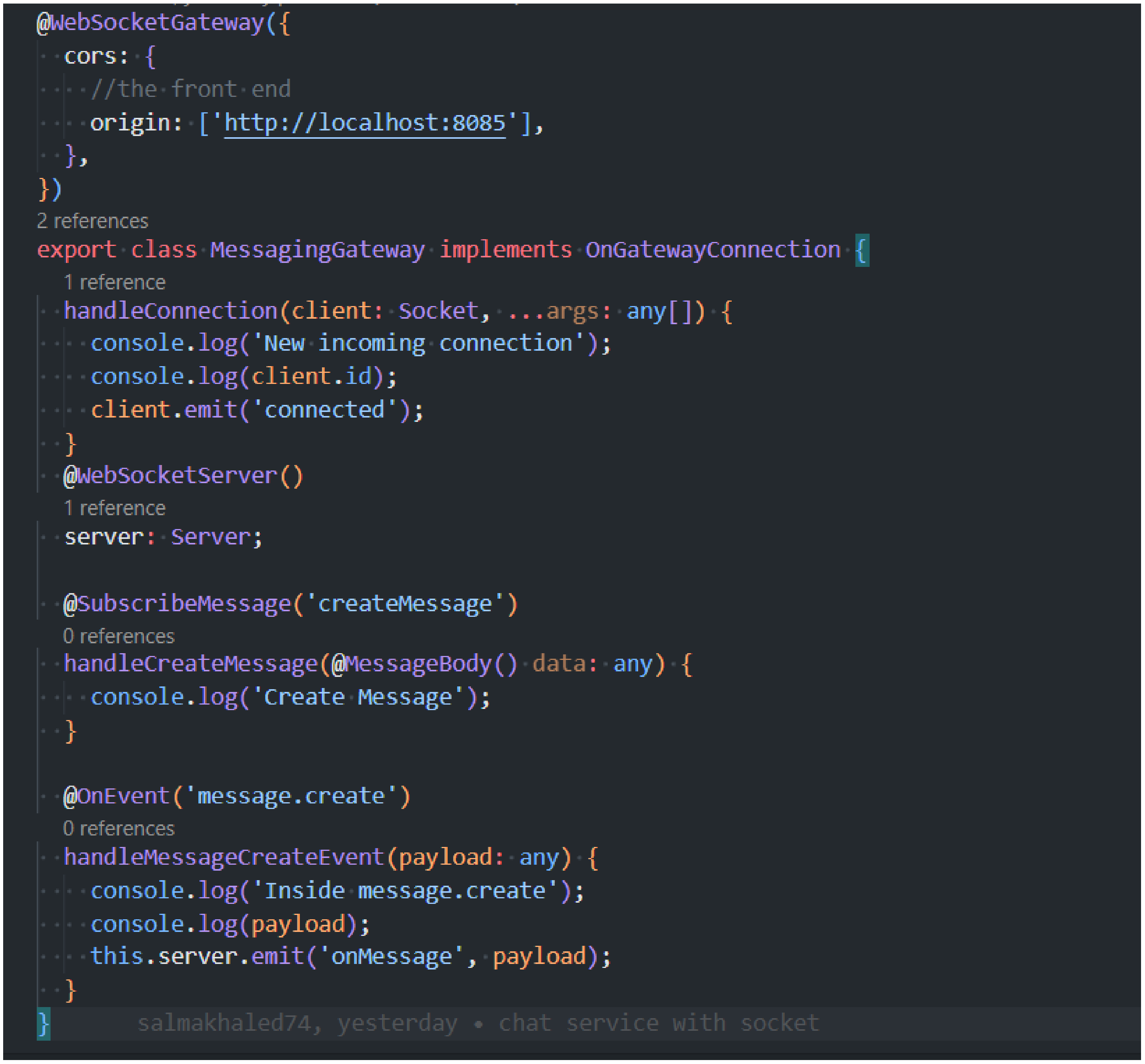


```
  @ApiBody({
    schema: {
      properties: {
        userId: {
          type: 'string',
        },
      },
    },
  })
  @Post('getFeedbacks')
  0 references
  async get(
    @Body('userId') userId: string,
  ): Promise<CommonResponse<Feedback[]>> {
    const data = await this.feedbacksService.get(userId);
    return {
      message: 'Feedbacks fetched succesfully',
      statusCode: HttpStatus.OK,
      data,
    };
  }
}
```

4.2.5 Chat Service

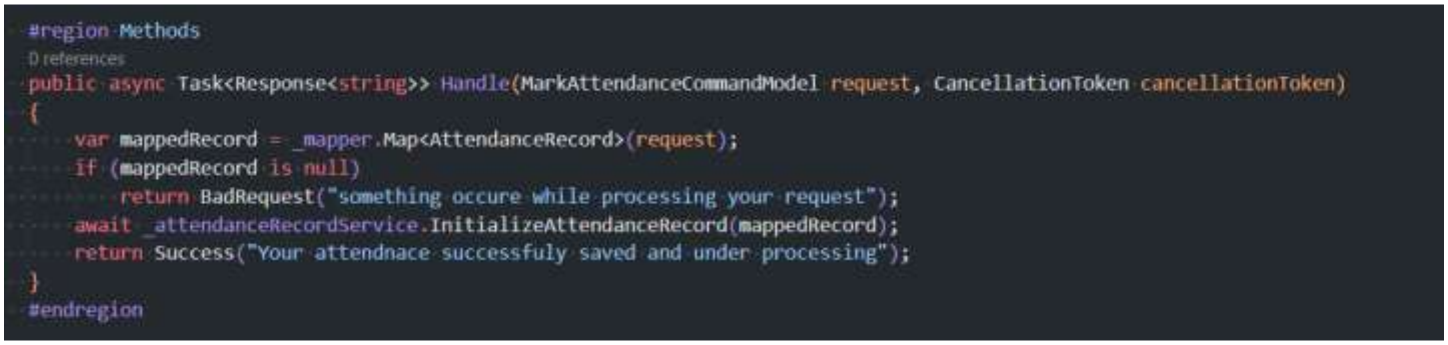


```
async create(data: createMessageDto) {
    const user = await User.findByPk(data.userId, {
      include: [Participant],
    });

    if (!user || !user.participant) {
      throw new NotFoundException('User or Participant not found');
    }

    const conversation = await Conversation.findByPk(data.conversationId);

    if (!conversation) {
      throw new NotFoundException('Conversation not found');
    }

    const message = await Message.create({
      participantId: user.participant.participantId,
      text: data.text,
      conversationId: data.conversationId,
    });
```

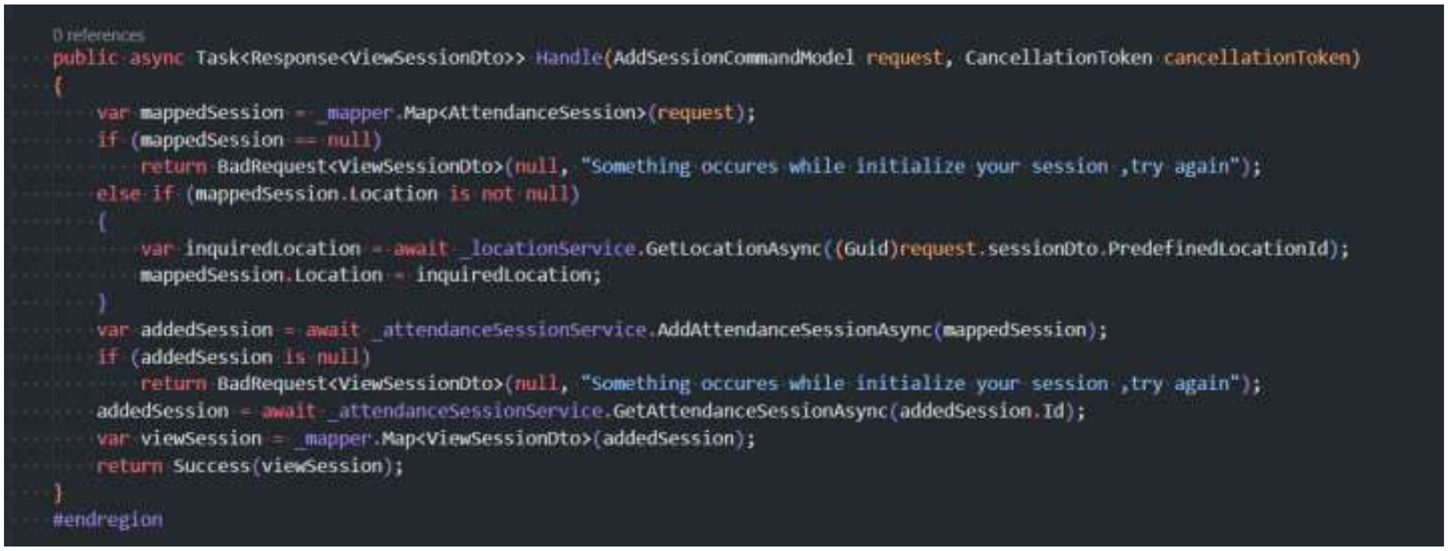


```
@WebSocketGateway({
  cors: {
    //the front end
    origin: ['http://localhost:8085'],
  },
})
2 references
export class MessagingGateway implements OnGatewayConnection {
  1 reference
  handleConnection(client: Socket, ...args: any[]) {
    console.log('New incoming connection');
    console.log(client.id);
    client.emit('connected');
  }
  @WebSocketServer()
  1 reference
  server: Server;

  @SubscribeMessage('createMessage')
  0 references
  handleCreateMessage(@MessageBody() data: any) {
    console.log('Create Message');
  }

  @OnEvent('message.create')
  0 references
  handleMessageCreateEvent(payload: any) {
    console.log('Inside message.create');
    console.log(payload);
    this.server.emit('onMessage', payload);
  }
}       salmakhaled74, yesterday • chat service with socket
```

4.2.6 Attendance Service



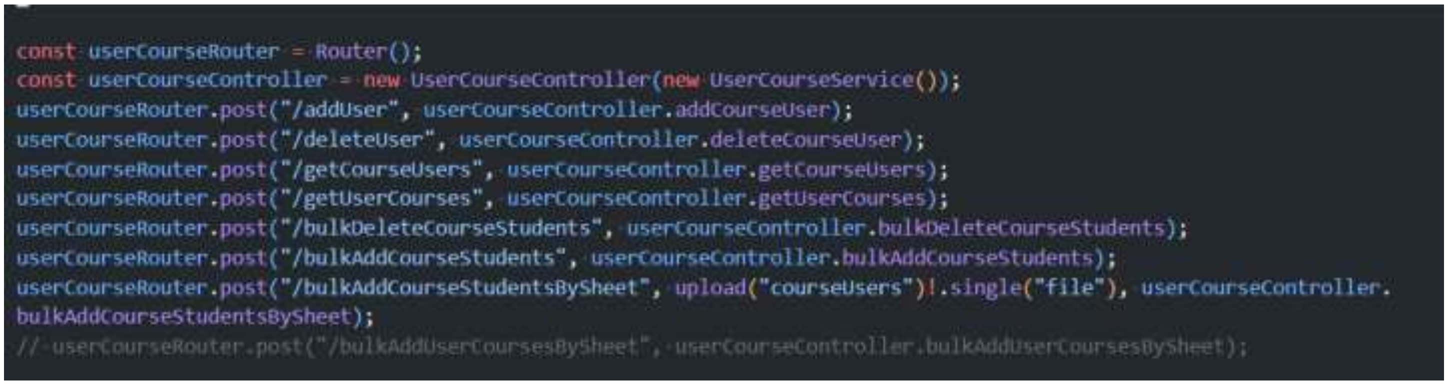
```
#region Methods
0 references
public async Task<Response<string>> Handle(MarkAttendanceCommandModel request, CancellationToken cancellationToken)
{
    var mappedRecord = _mapper.Map<AttendanceRecord>(request);
    if (mappedRecord is null)
        return BadRequest("something occure while processing your request");
    await _attendanceRecordService.InitializeAttendanceRecord(mappedRecord);
    return Success("Your attendnace successfuly saved and under processing");
}
#endregion
```



```
    0 references
    public async Task<Response<ViewSessionDto>> Handle(AddSessionCommandModel request, CancellationToken cancellationToken)
    {
        var mappedSession = _mapper.Map<AttendanceSession>(request);
        if (mappedSession == null)
            return BadRequest<ViewSessionDto>(null, "Something occures while initialize your session ,try again");
        else if (mappedSession.Location is not null)
        {
            var inquiredLocation = await _locationService.GetLocationAsync((Guid)request.sessionDto.PredefinedLocationId);
            mappedSession.Location = inquiredLocation;
        }
        var addedSession = await _attendanceSessionService.AddAttendanceSessionAsync(mappedSession);
        if (addedSession is null)
            return BadRequest<ViewSessionDto>(null, "Something occures while initialize your session ,try again");
        addedSession = await _attendanceSessionService.GetAttendanceSessionAsync(addedSession.Id);
        var viewSession = _mapper.Map<ViewSessionDto>(addedSession);
        return Success(viewSession);
    }
    #endregion
```

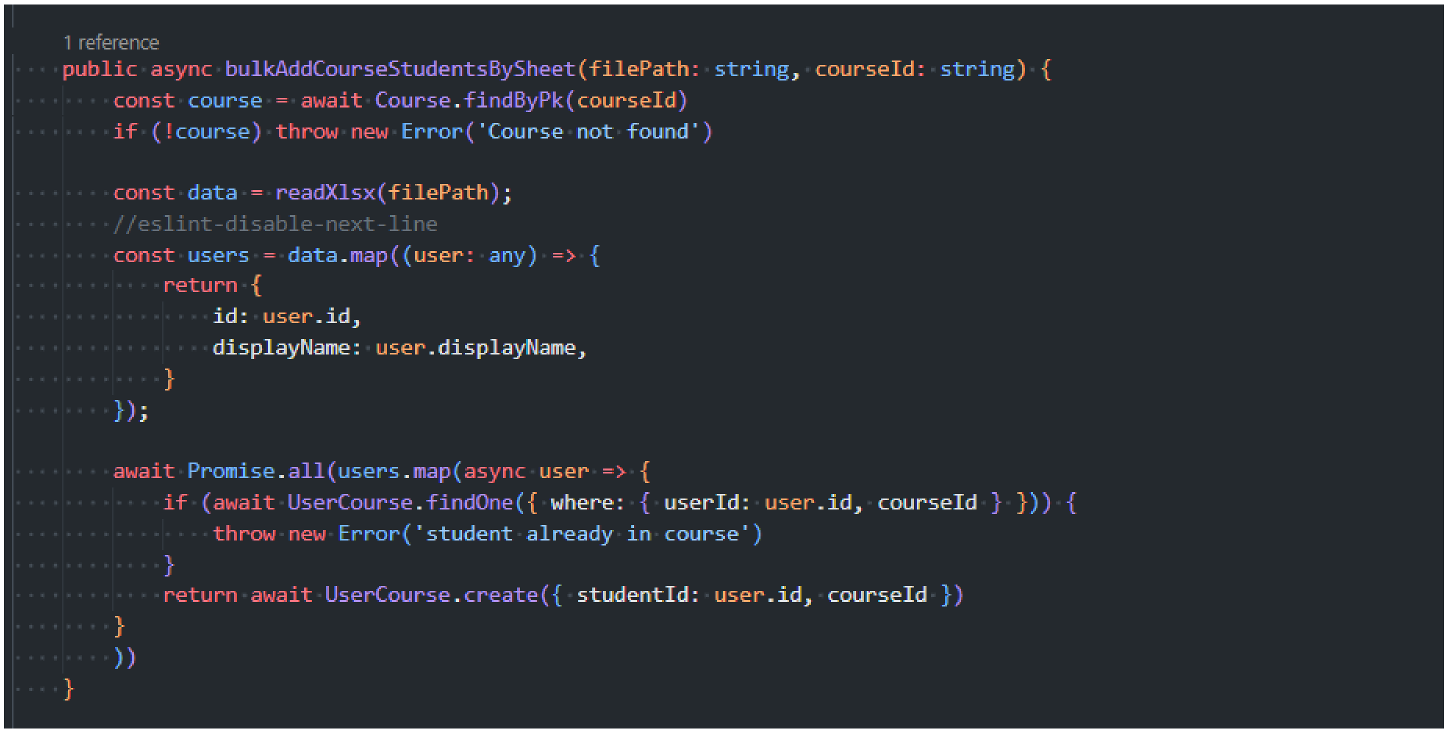
4.2.7 Admin Service

Some endpoints:



```
const userCourseRouter = Router();
const userCourseController = new UserCourseController(new UserCourseService());
userCourseRouter.post("/addUser", userCourseController.addCourseUser);
userCourseRouter.post("/deleteUser", userCourseController.deleteCourseUser);
userCourseRouter.post("/getCourseUsers", userCourseController.getCourseUsers);
userCourseRouter.post("/getUserCourses", userCourseController.getUserCourses);
userCourseRouter.post("/bulkDeleteCourseStudents", userCourseController.bulkDeleteCourseStudents);
userCourseRouter.post("/bulkAddCourseStudents", userCourseController.bulkAddCourseStudents);
userCourseRouter.post("/bulkAddCourseStudentsBySheet", upload("courseUsers")!.single("file"), userCourseController.
bulkAddCourseStudentsBySheet);
// userCourseRouter.post("/bulkAddUserCoursesBySheet", userCourseController.bulkAddUserCoursesBySheet);
```

Bulk Add Course Students By Sheet Service Example:

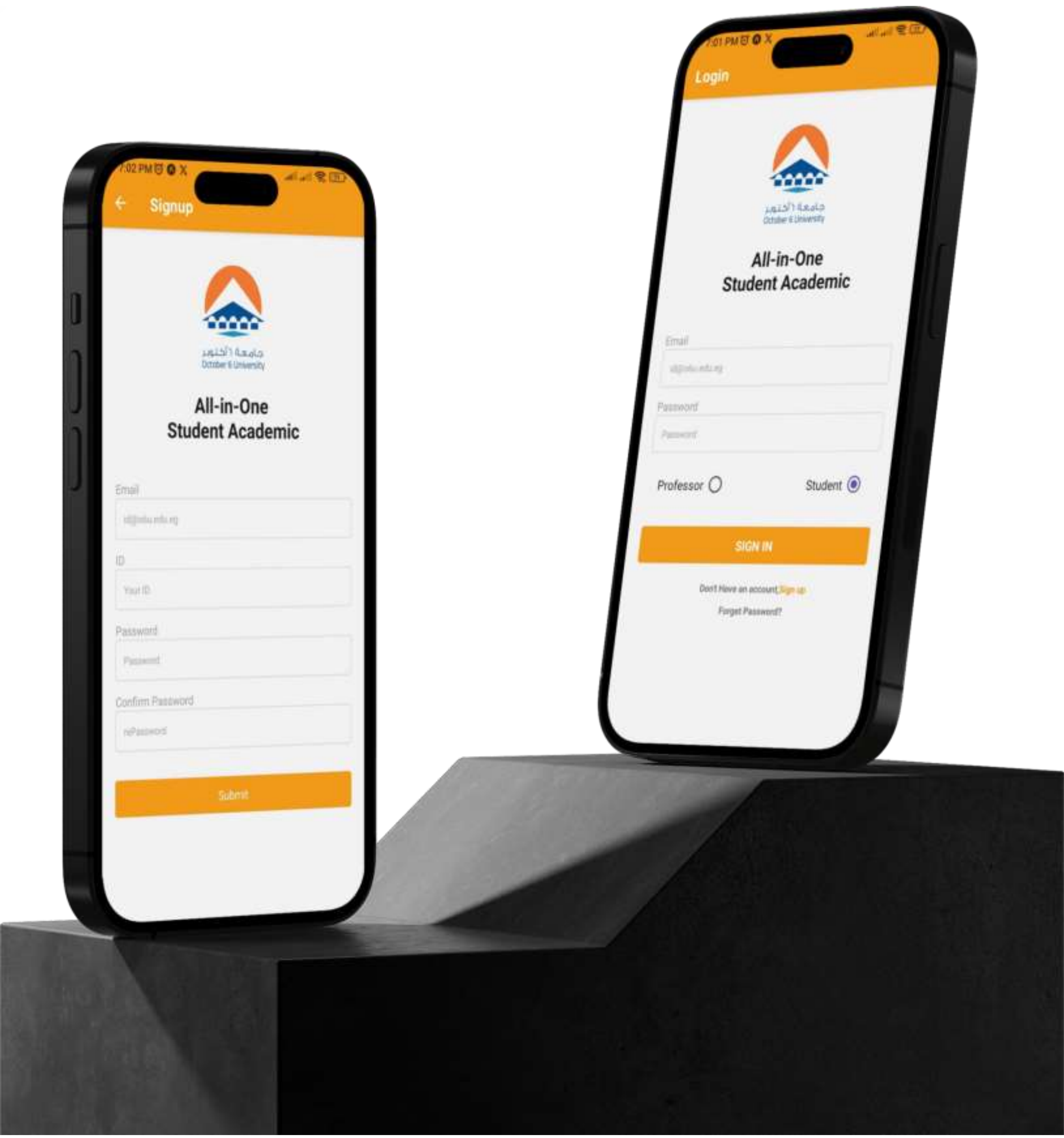


```
1 reference
public async bulkAddCourseStudentsBySheet(filePath: string, courseId: string) {
    const course = await Course.findByPk(courseId)
    if (!course) throw new Error('Course not found')

    const data = readXlsx(filePath);
    //eslint-disable-next-line
    const users = data.map((user: any) => {
        return {
            id: user.id,
            displayName: user.displayName,
        }
    });

    await Promise.all(users.map(async user => {
        if (await UserCourse.findOne({ where: { userId: user.id, courseId } })) {
            throw new Error('student already in course')
        }
        return await UserCourse.create({ studentId: user.id, courseId })
    }
    ))
}
```

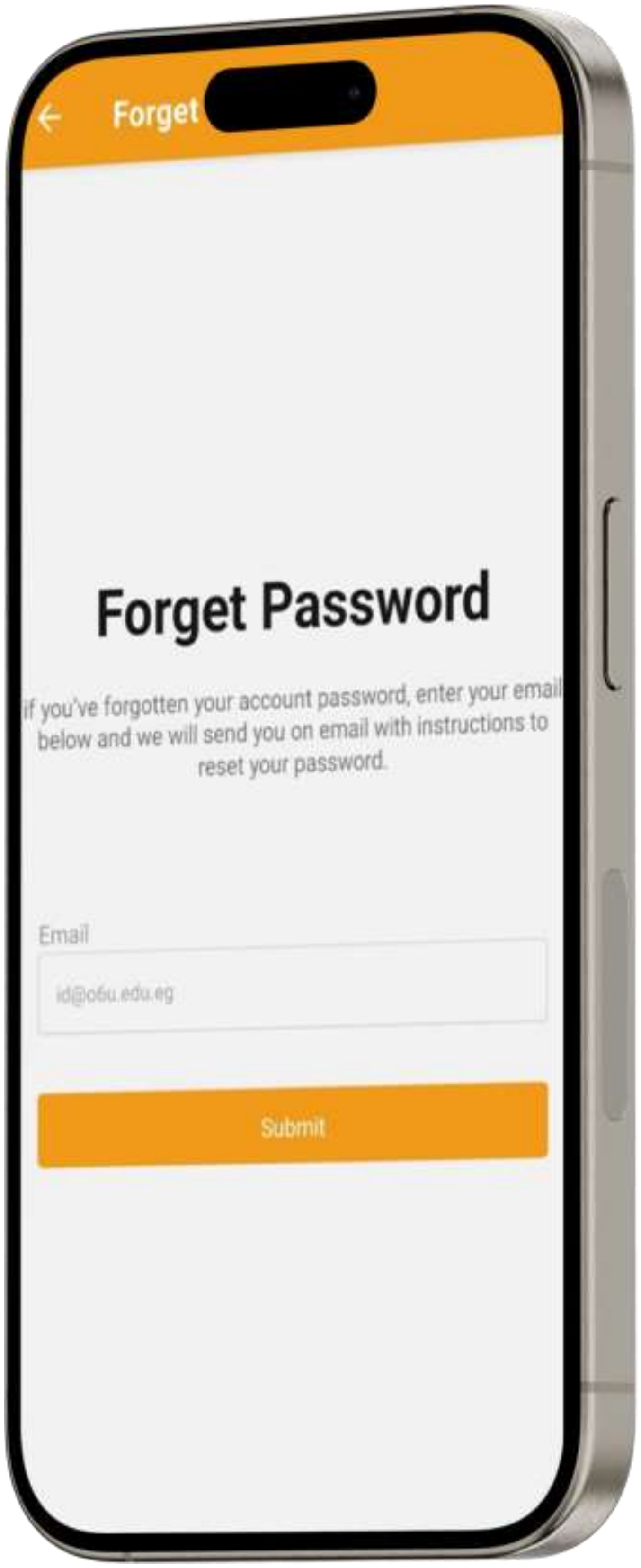
4.3 mobile application screens

4.3.1 student side

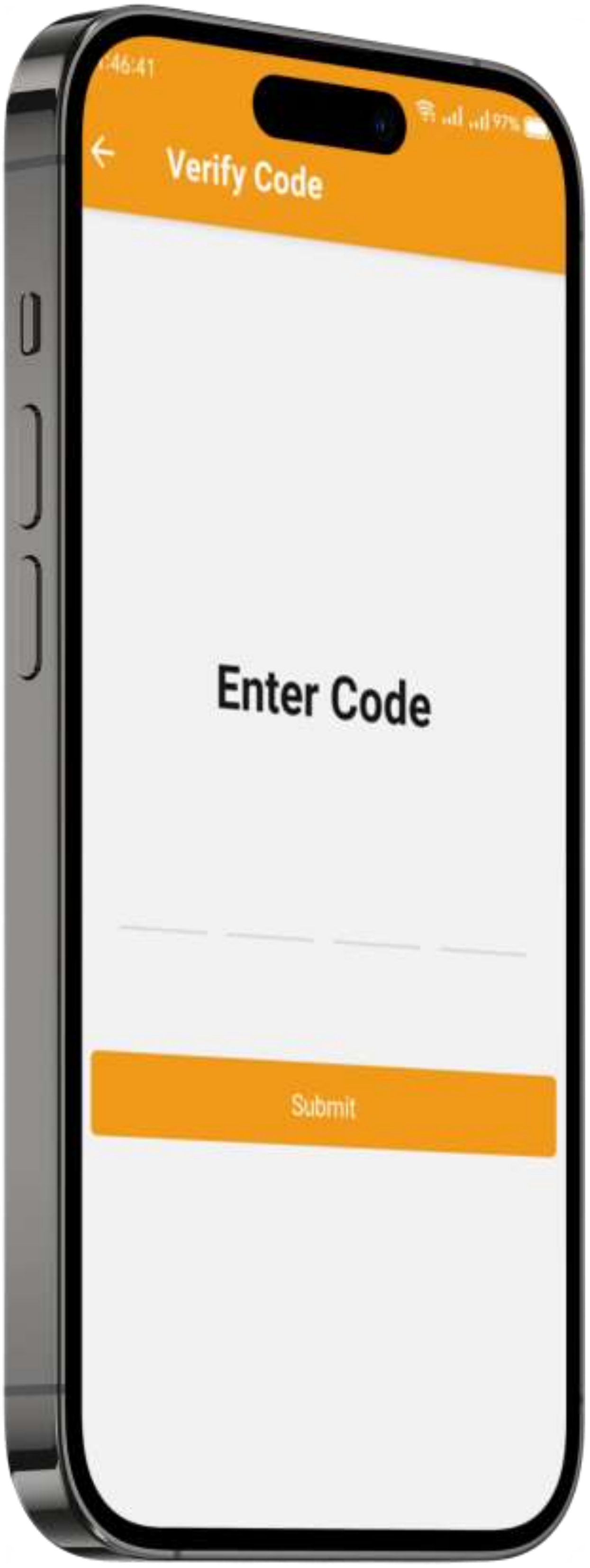
4.3.1.1 Login & registration

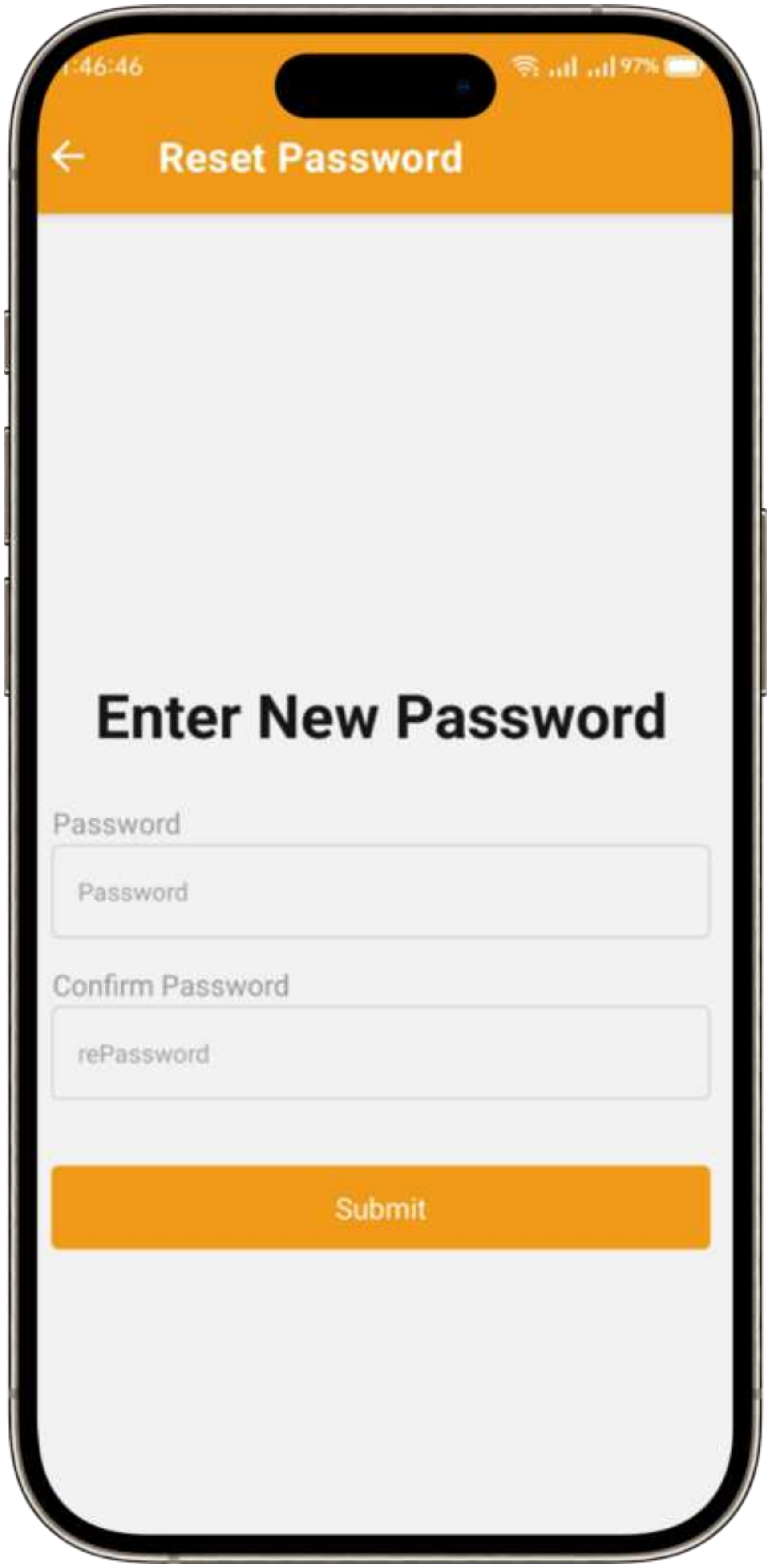


4.3.1.2 credentials restore

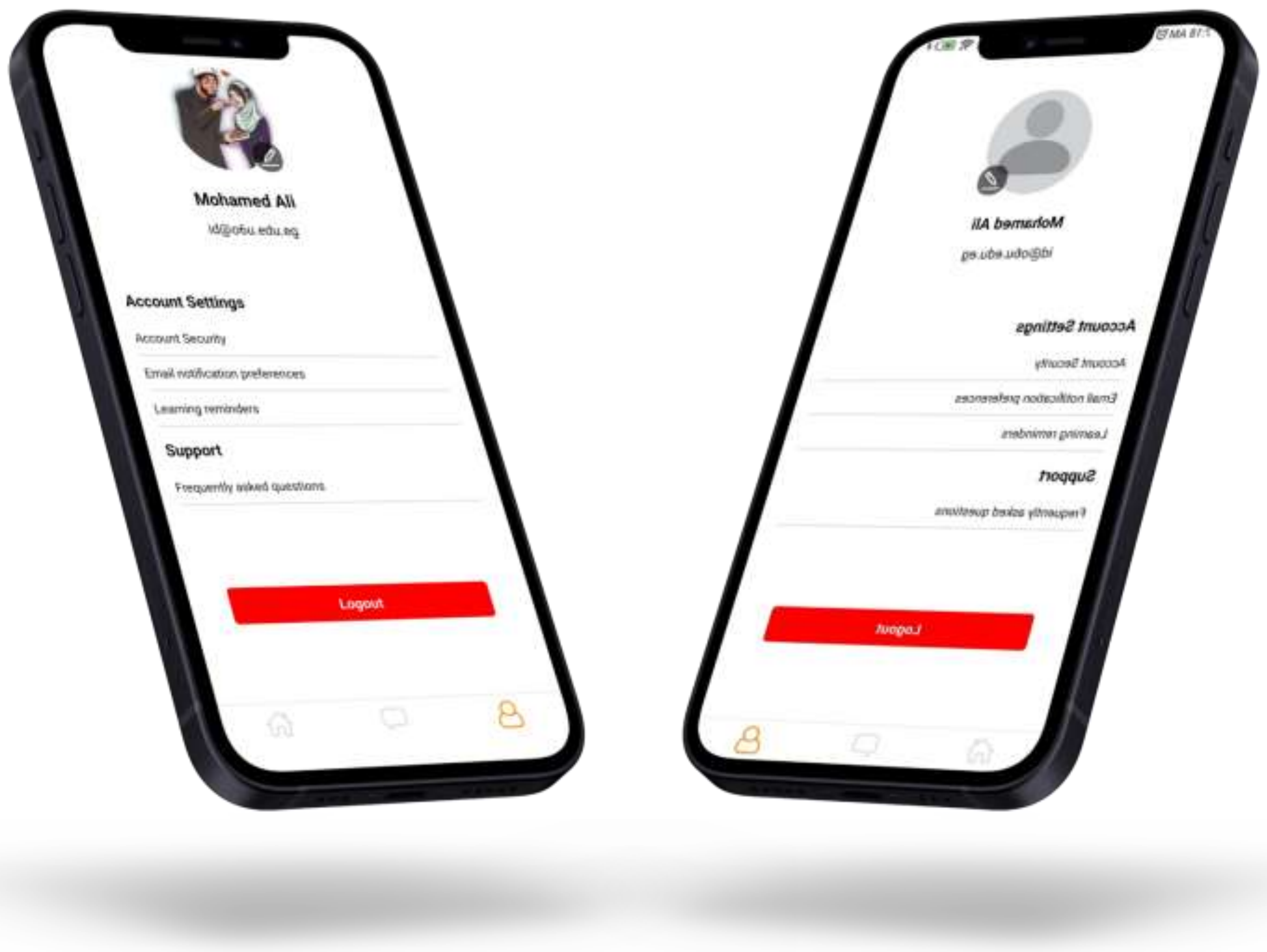


If you've forgotten your account password, enter your email below and we will send you on email with instructions to reset your password.

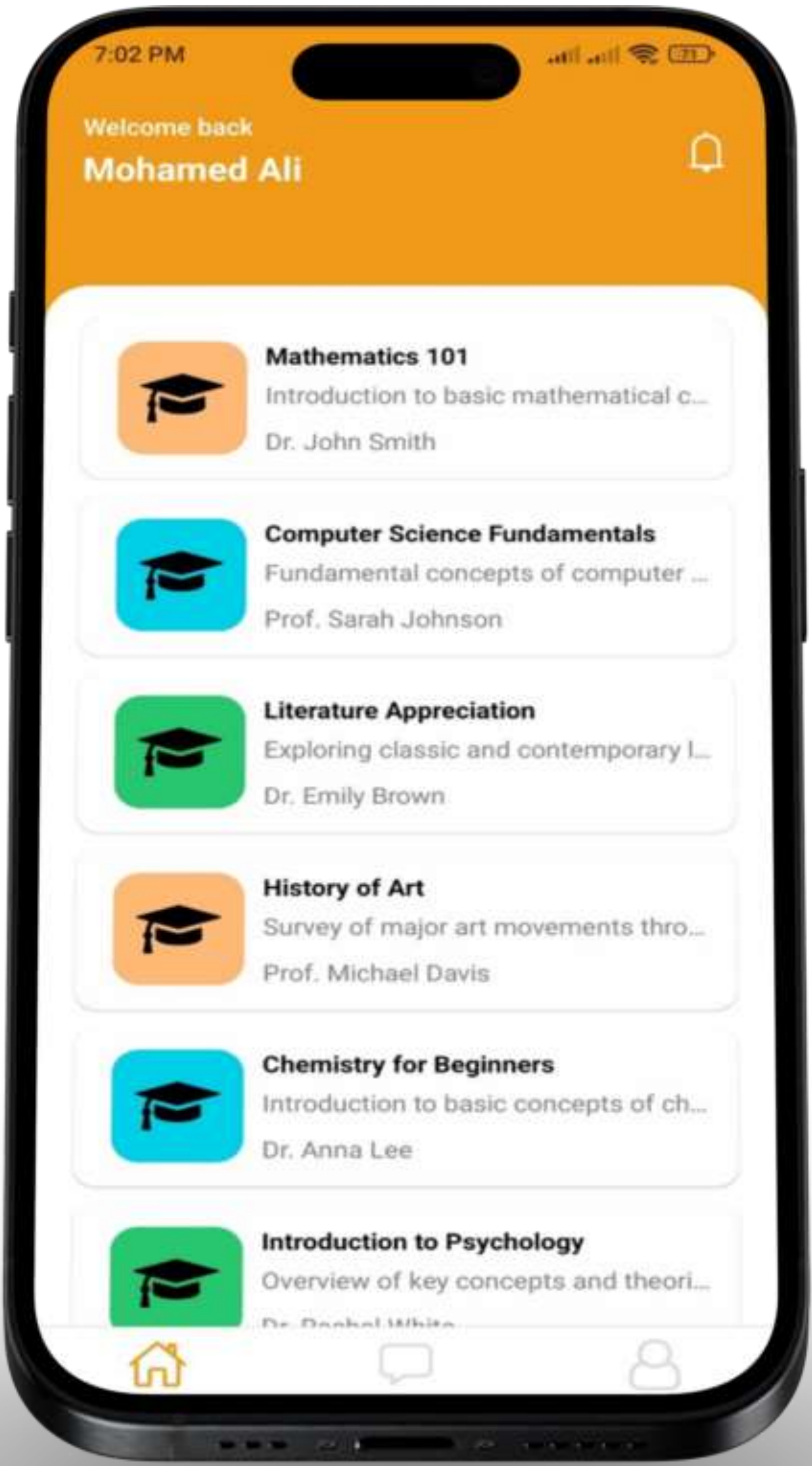



:46:46
97%
Reset Password
Enter New Password
Password
Password
Confirm Password
rePassword
Submit

4.3.1.3 Account Management



4.3.1.4 Home Screen



7:02 PM

Welcome back

Mohamed Ali

Mathematics 101
Introduction to basic mathematical c...
Dr. John Smith

Computer Science Fundamentals
Fundamental concepts of computer ...
Prof. Sarah Johnson

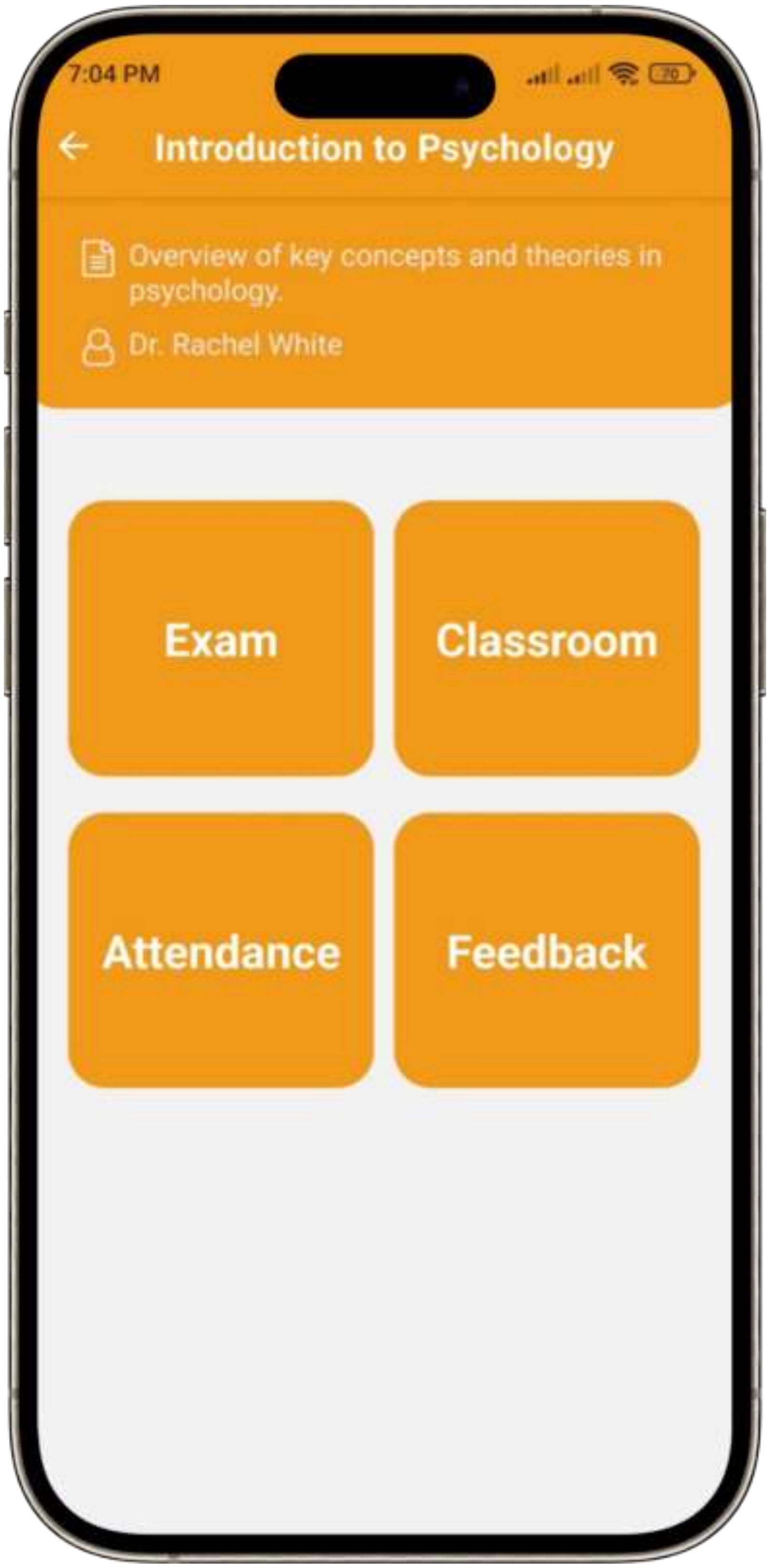
Literature Appreciation
Exploring classic and contemporary l...
Dr. Emily Brown

History of Art
Survey of major art movements thro...
Prof. Michael Davis

Chemistry for Beginners
Introduction to basic concepts of ch...
Dr. Anna Lee

Introduction to Psychology
Overview of key concepts and theori...
Dr. Rachel White

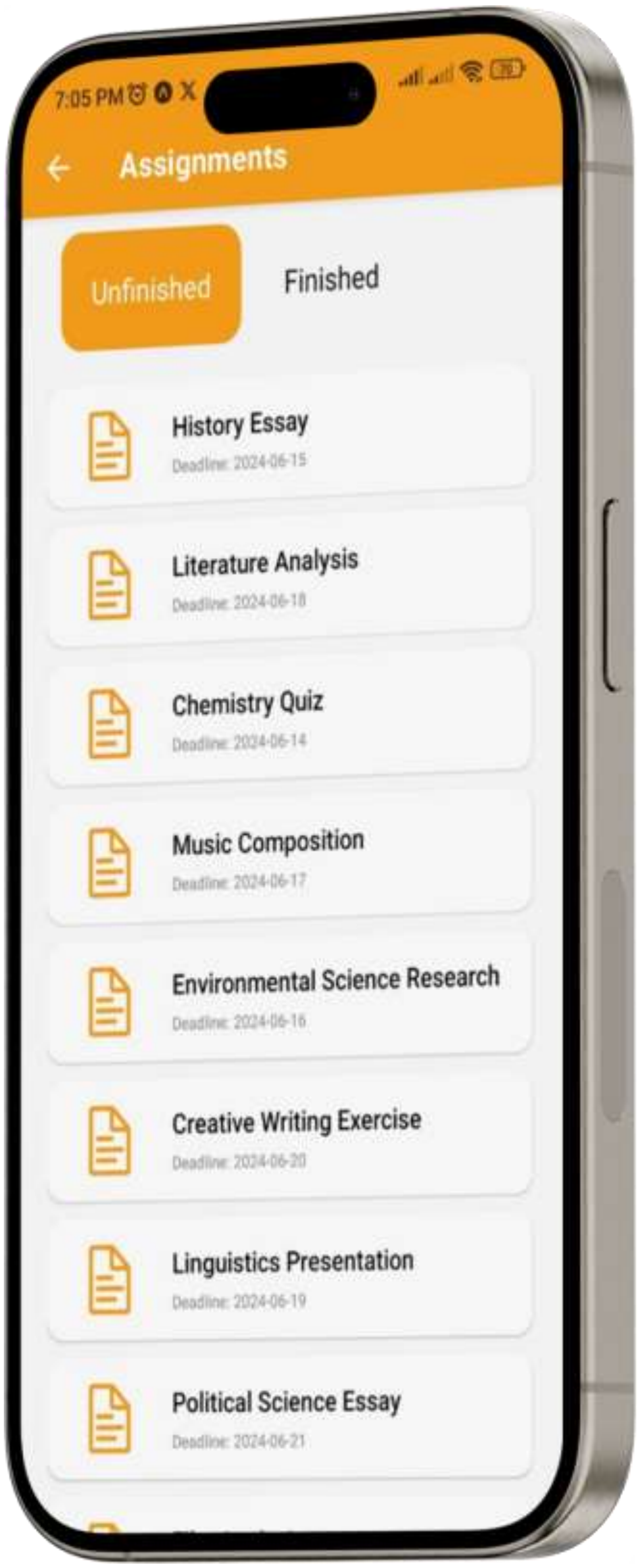
4.3.1.5 Course Home Screen



Overview of key concepts and theories in psychology.

Dr. Rachel White

4.3.1.6 Classroom Module


7:05 PM

Assignments

Unfinished

Finished

History Essay

Deadline: 2024-06-15

Literature Analysis

Deadline: 2024-06-18

Chemistry Quiz

Deadline: 2024-06-14

Music Composition

Deadline: 2024-06-17

Environmental Science Research

Deadline: 2024-06-16

Creative Writing Exercise

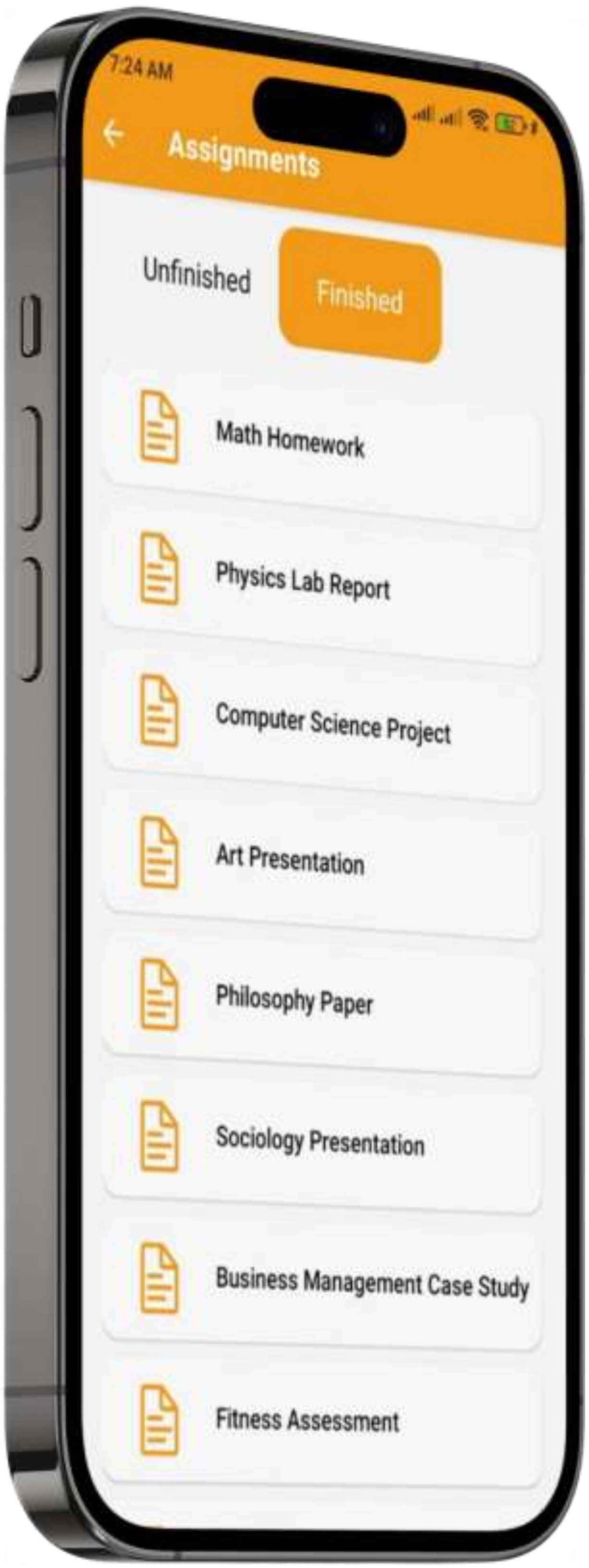
Deadline: 2024-06-20

Linguistics Presentation

Deadline: 2024-06-19

Political Science Essay

Deadline: 2024-06-21


7:24 AM

Assignments

Unfinished

Finished

Math Homework

Physics Lab Report

Computer Science Project

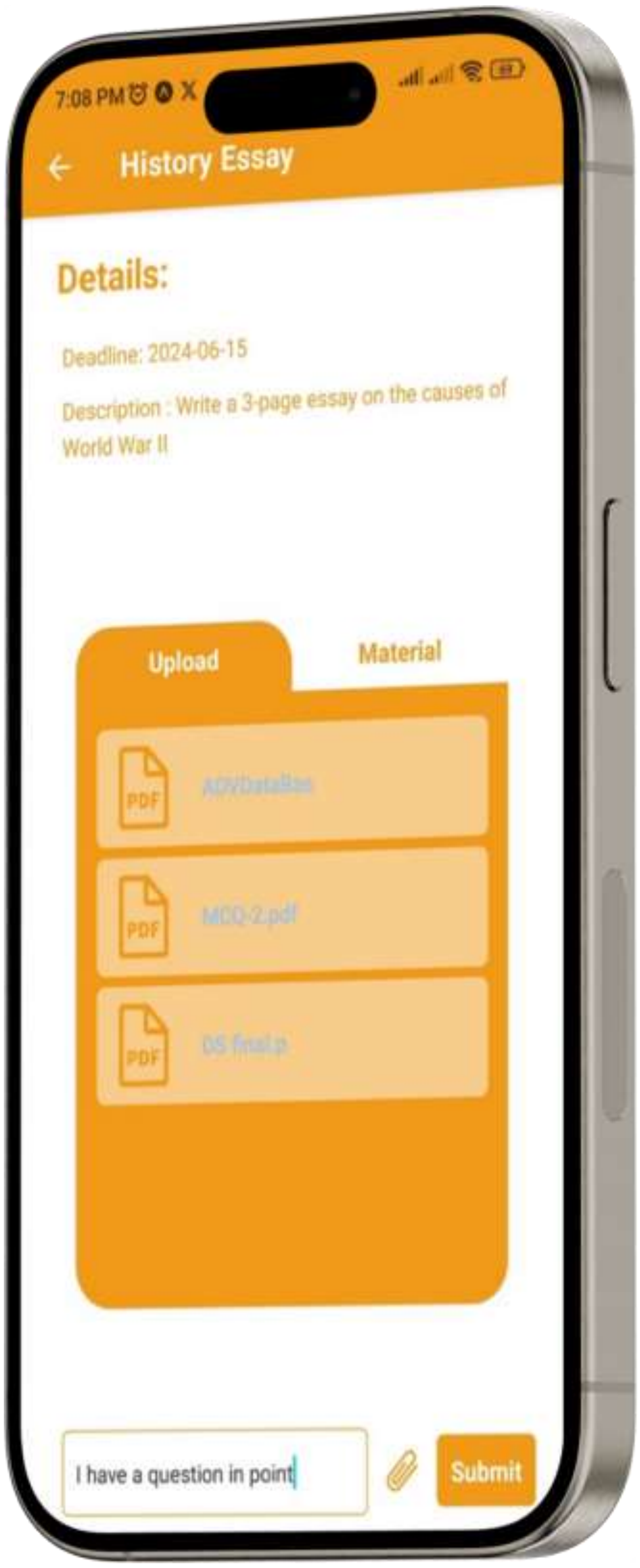
Art Presentation

Philosophy Paper

Sociology Presentation

Business Management Case Study

Fitness Assessment

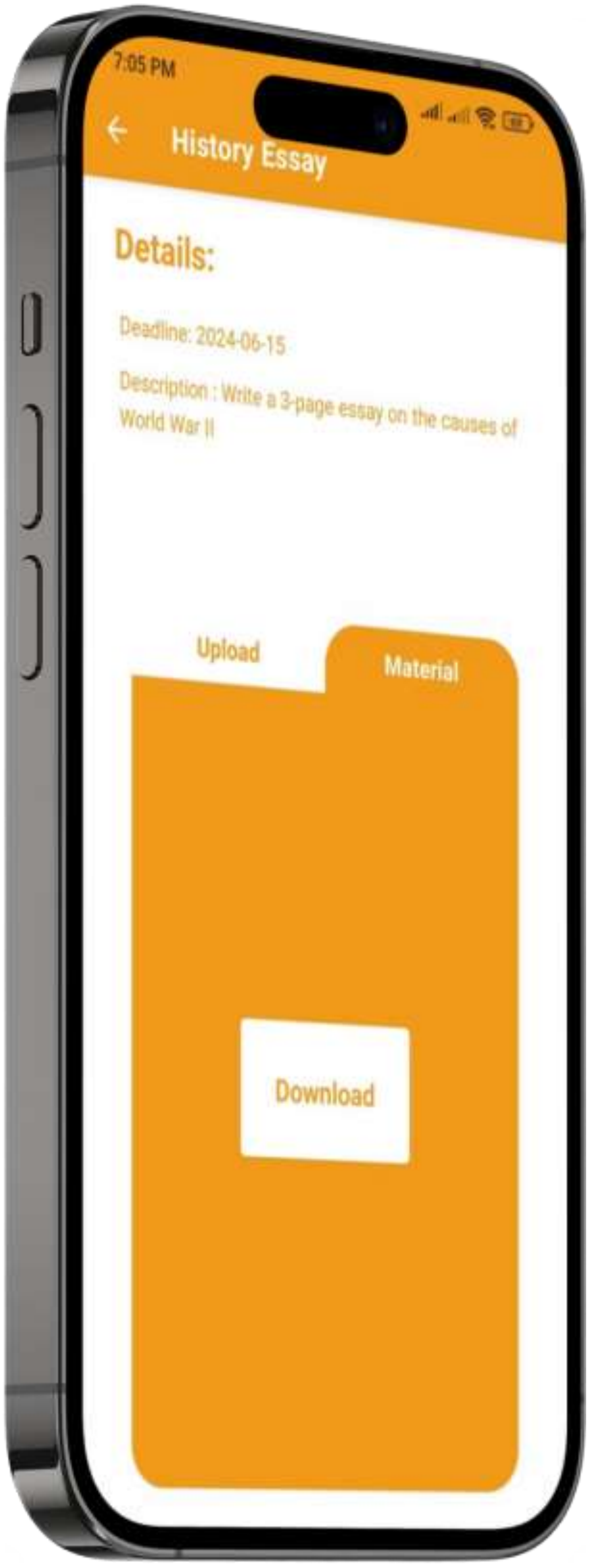


Details:

Deadline: 2024-06-15

Description : Write a 3-page essay on the causes of World War II

I have a question in point

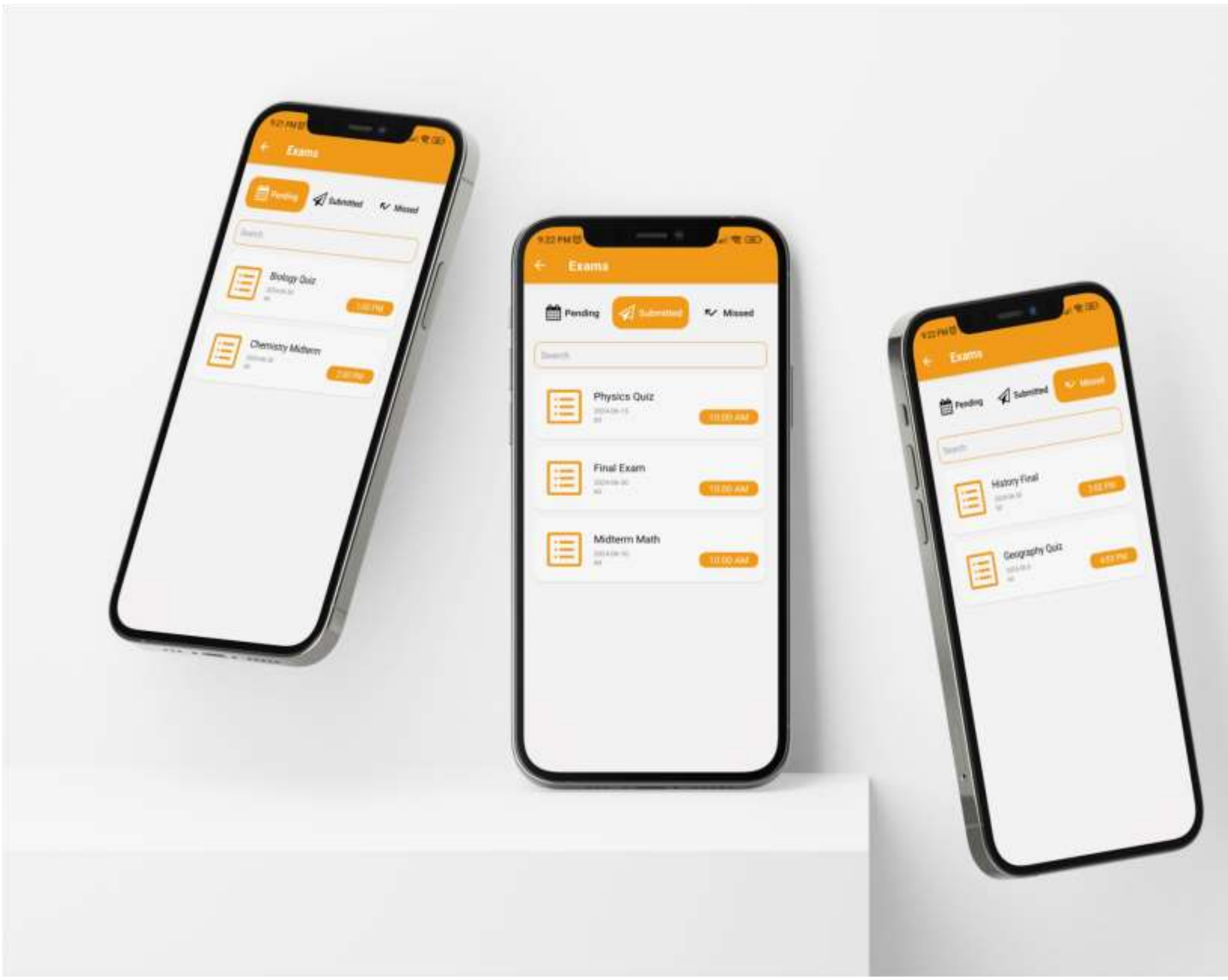


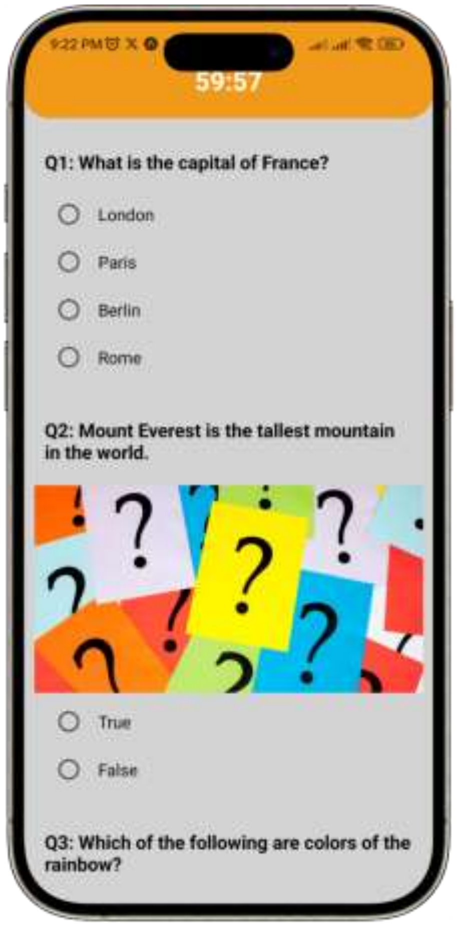
Details:

Deadline: 2024-06-15

Description : Write a 3-page essay on the causes of World War II

4.3.1.7 Exam Module





9:22 PM

59:57

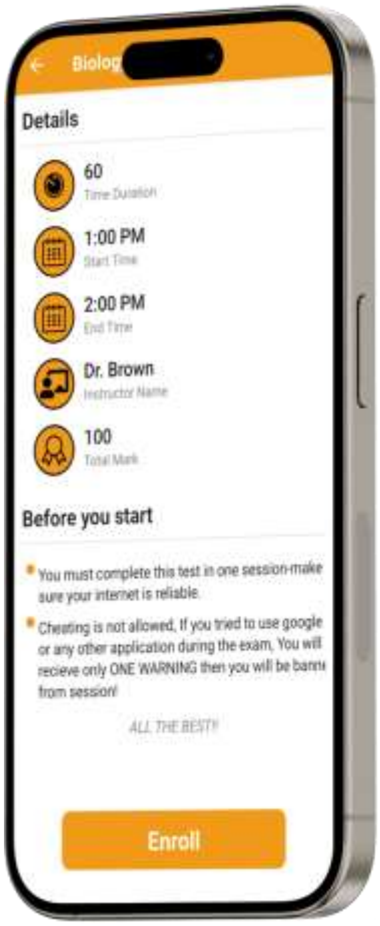
Q1: What is the capital of France?

- ○ London
- ○ Paris
- ○ Berlin
- ○ Rome

Q2: Mount Everest is the tallest mountain in the world.

- ○ True
- ○ False

Q3: Which of the following are colors of the rainbow?



Biolog

Details

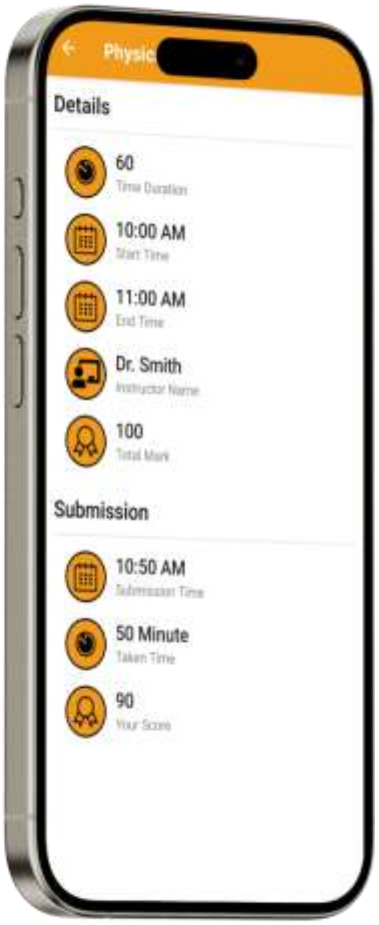
60	Time Duration
1:00 PM	Start Time
2:00 PM	End Time
Dr. Brown	Instructor Name
100	Total Mark

Before you start

- You must complete this test in one session-make sure your internet is reliable.
- Cheating is not allowed, If you tried to use google or any other application during the exam, You will recieve only ONE WARNING then you will be bann from session!

ALL THE BEST!!

Enroll



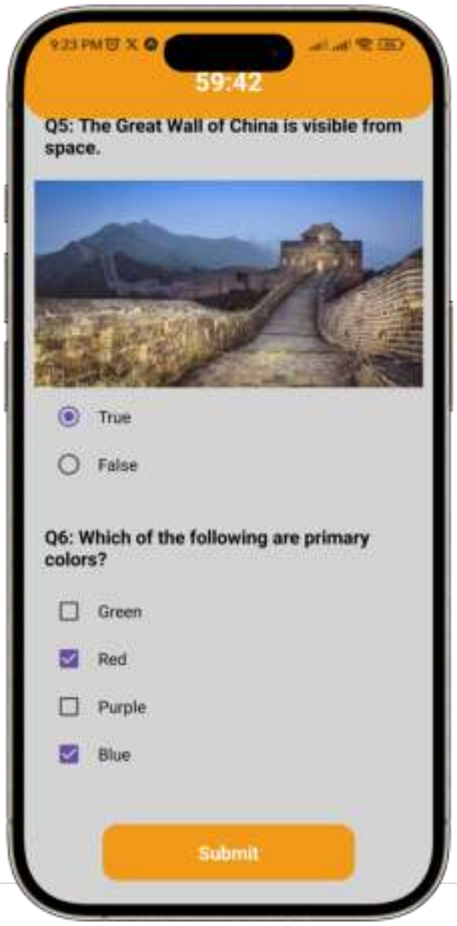
Physic

Details

60	Time Duration
10:00 AM	Start Time
11:00 AM	End Time
Dr. Smith	Instructor Name
100	Total Mark

Submission

10:50 AM	Submission Time
50 Minute	Taken Time
90	Your Score



9:23 PM

59:42

Q5: The Great Wall of China is visible from space.

- [✓ True]
- ○ False

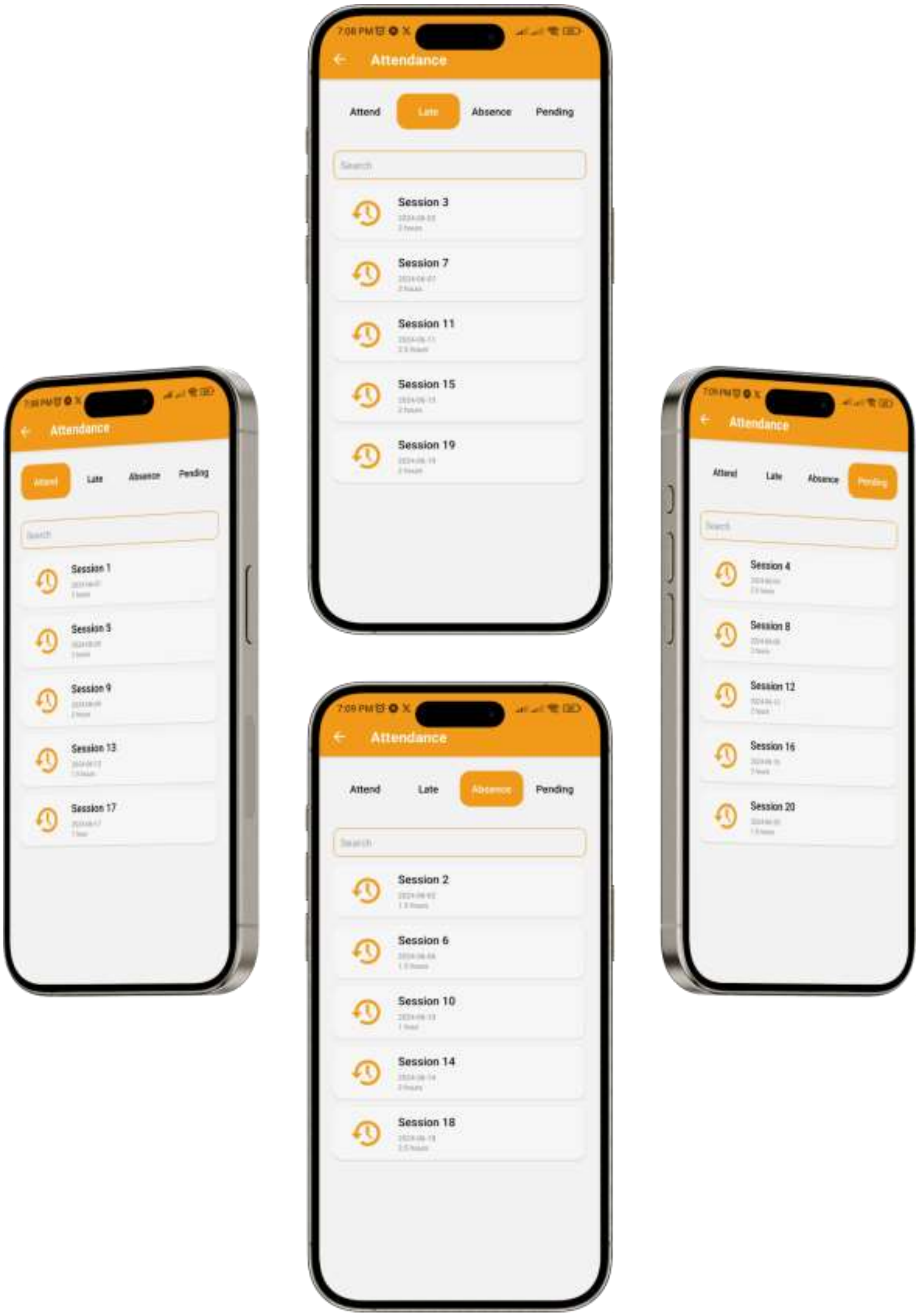
Q6: Which of the following are primary colors?

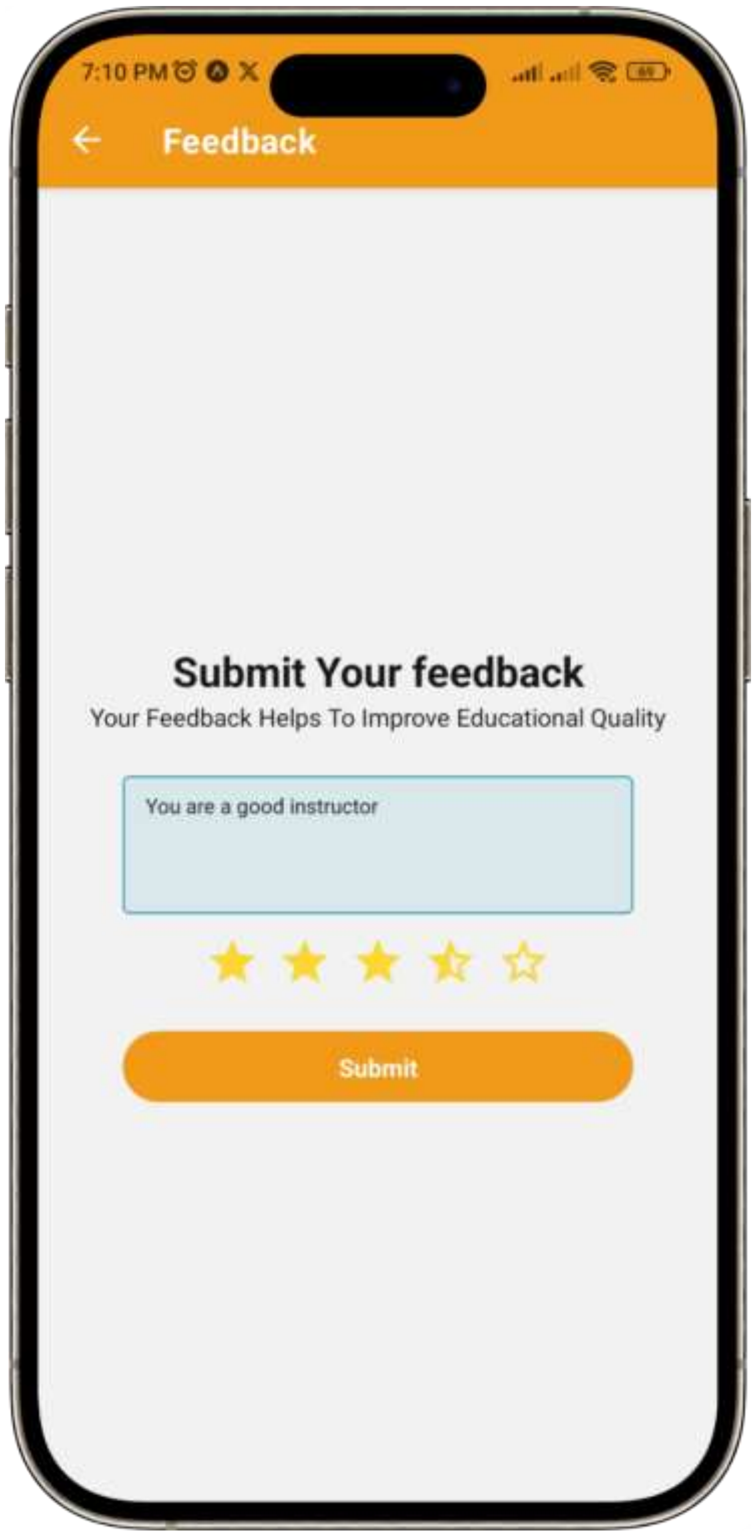
- ☐ Green
- ☑ Red
- ☐ Purple
- ☑ Blue

Submit

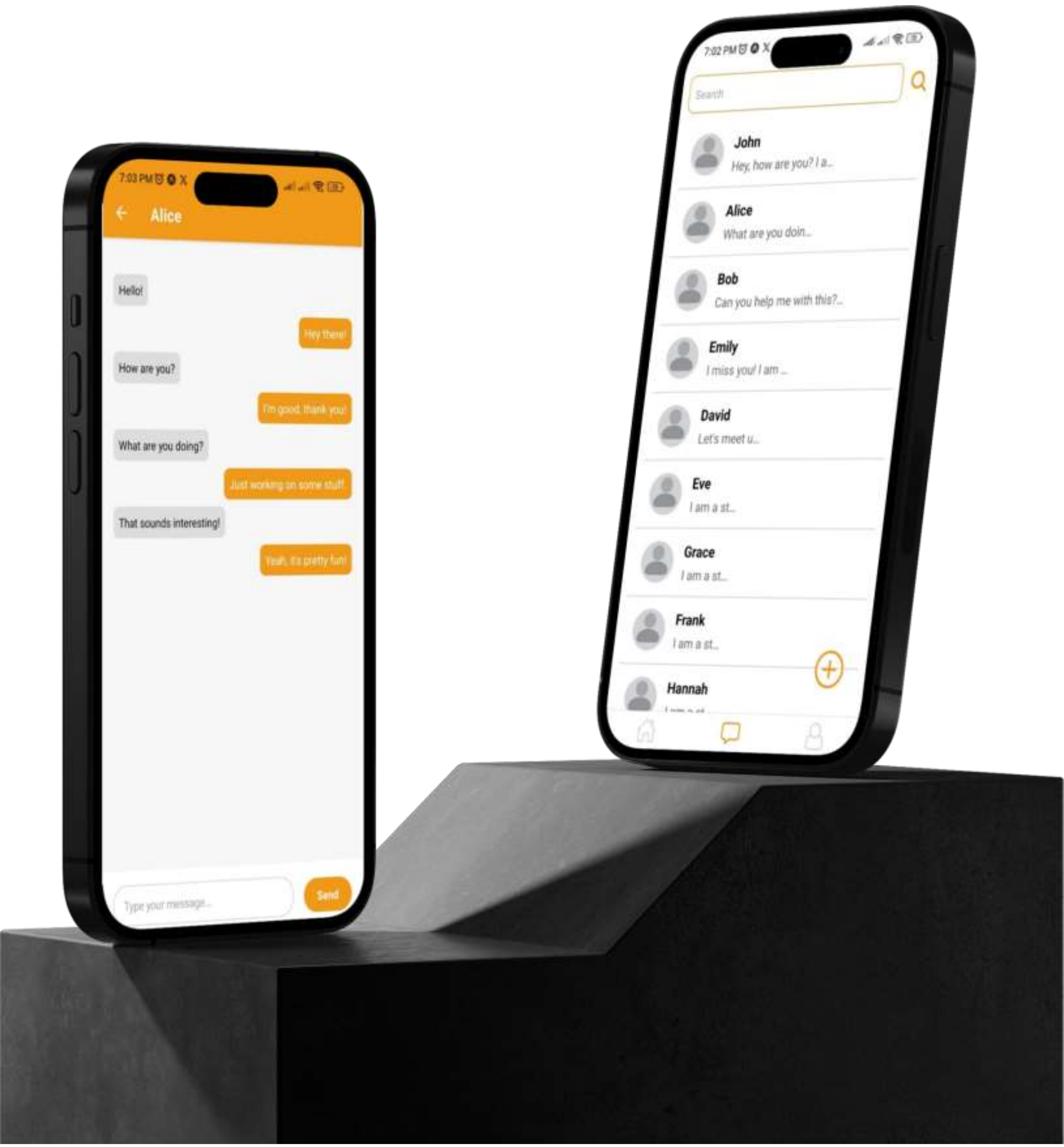
4.3.1.8 Attendance Module

4.3.1.9 Feedback Module




7:10 PM
Feedback
Submit Your feedback
Your Feedback Helps To Improve Educational Quality
You are a good instructor
★ ★ ★ ⯪ ☆
Submit

4.3.1.10 Chat Module



Hello!

Hey there!

How are you?

I'm good, thank you!

What are you doing?

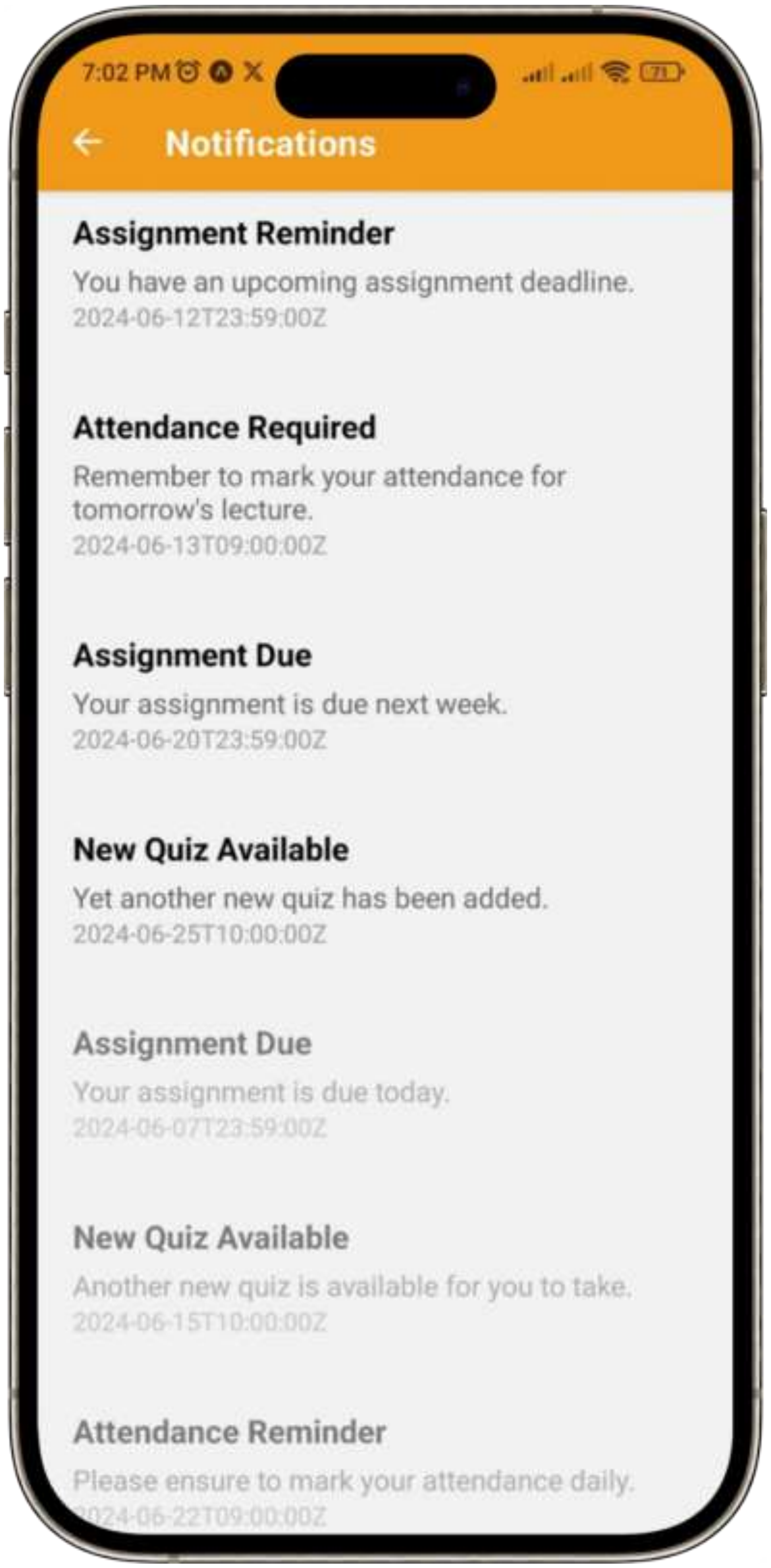
Just working on some stuff

That sounds interesting!

Yeah, it's pretty fun!

John	Hey, how are you? I a...
Alice	What are you doin...
Bob	Can you help me with this?...
Emily	I miss you! I am ...
David	Let's meet u...
Eve	I am a st...
Grace	I am a st...
Frank	I am a st...
Hannah	

4.3.1.11 Notification Service



Assignment Reminder
You have an upcoming assignment deadline.
2024-06-12T23:59:00Z

Attendance Required
Remember to mark your attendance for tomorrow's lecture.
2024-06-13T09:00:00Z

Assignment Due
Your assignment is due next week.
2024-06-20T23:59:00Z

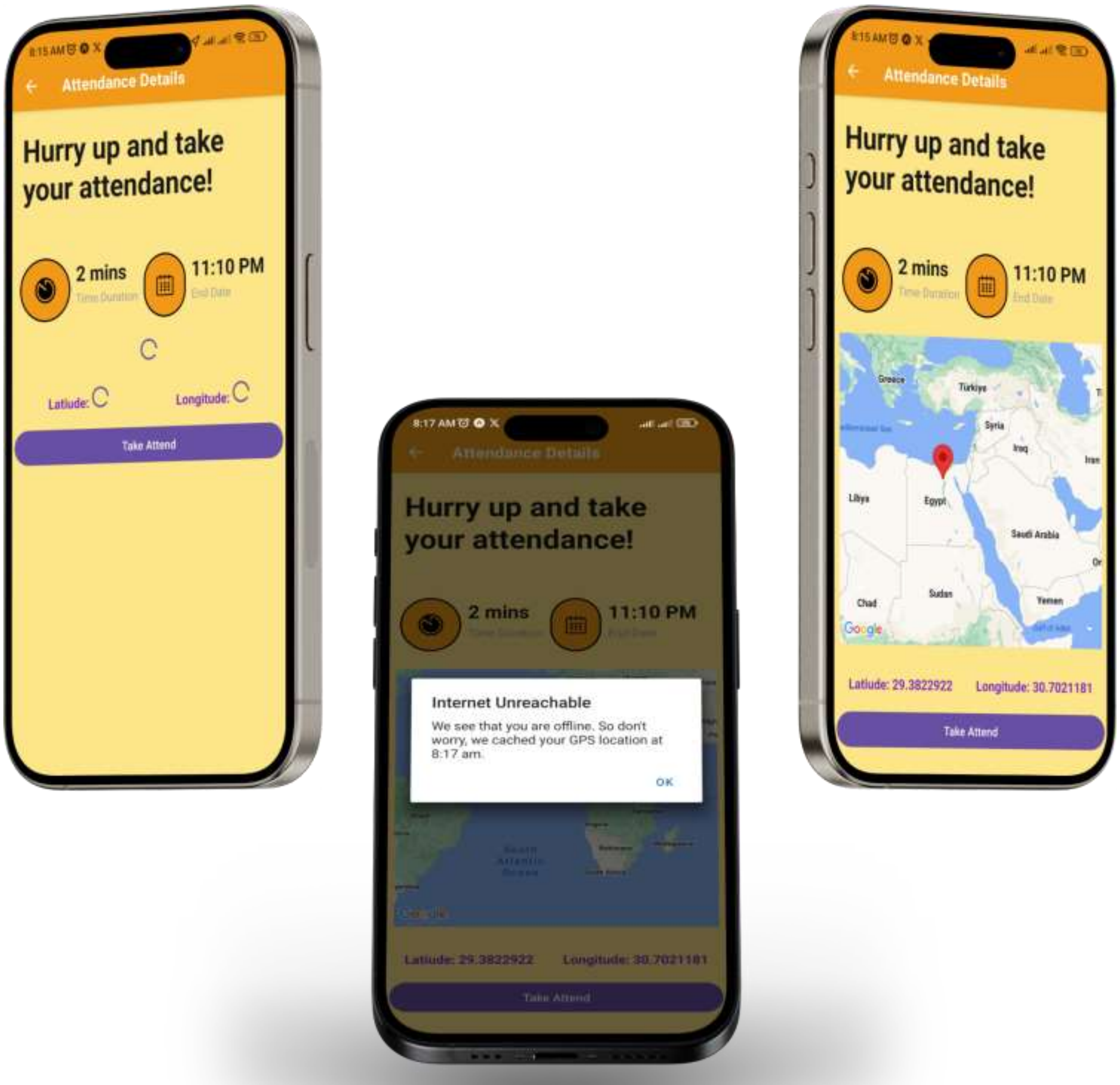
New Quiz Available
Yet another new quiz has been added.
2024-06-25T10:00:00Z

Assignment Due
Your assignment is due today.
2024-06-07T23:59:00Z

New Quiz Available
Another new quiz is available for you to take.
2024-06-15T10:00:00Z

Attendance Reminder
Please ensure to mark your attendance daily.
2024-06-22T09:00:00Z

4.3.1.12 Attendance Module



Internet Unreachable

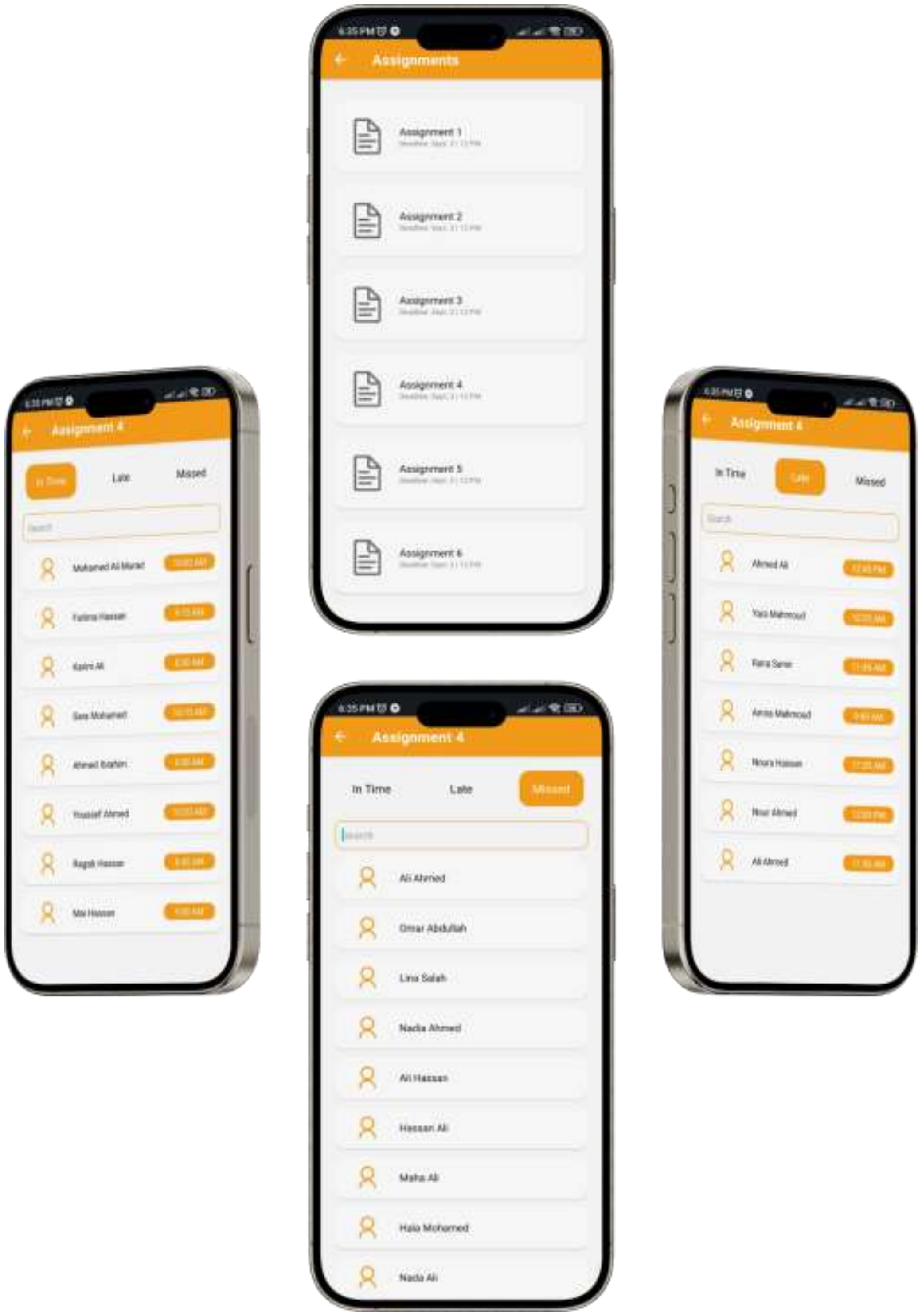
We see that you are offline. So don't worry, we cached your GPS location at 8:17 am.

OK

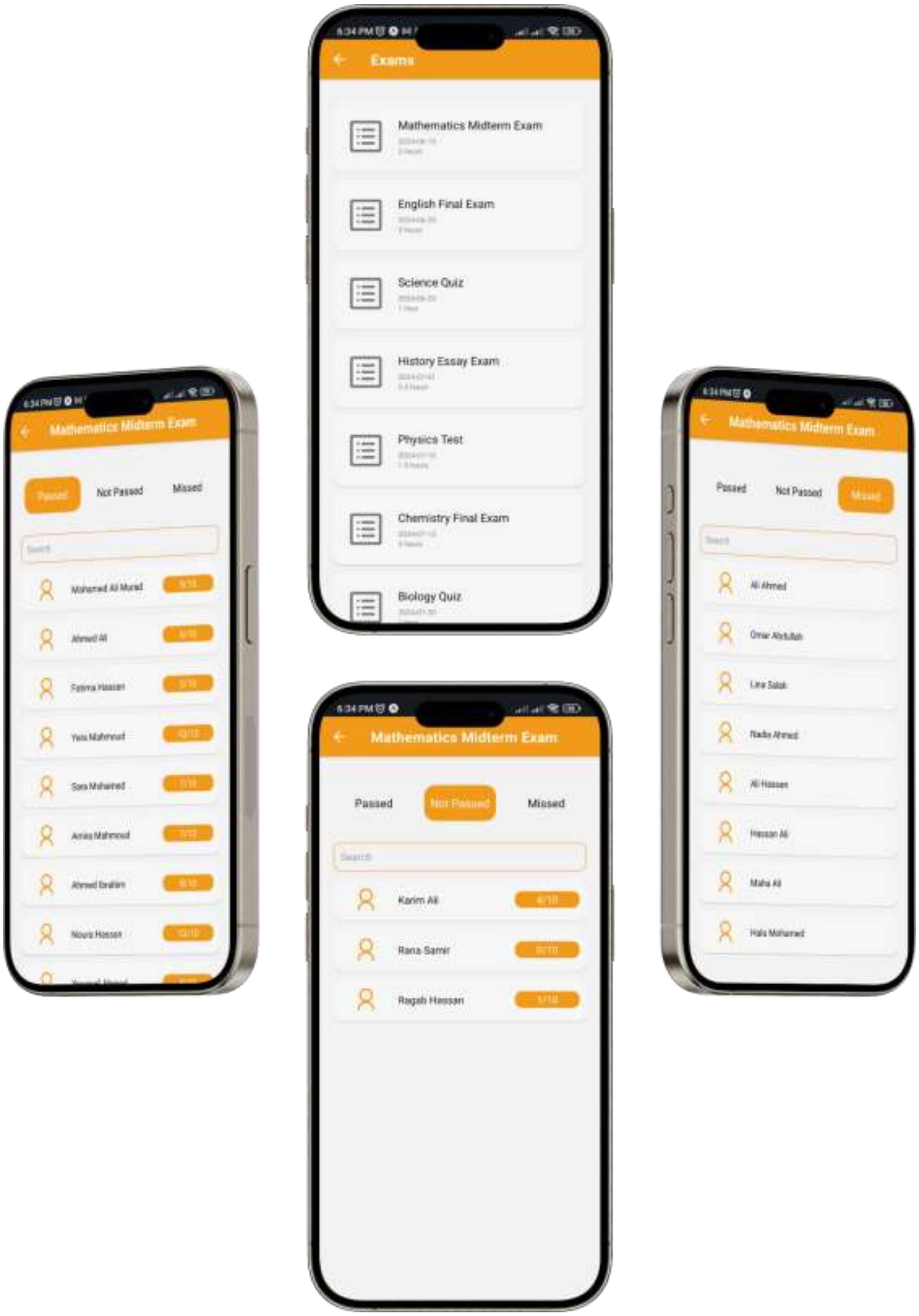
4.3.2 Instructor side

4.3.2.2 login & register (mentioned earlier)

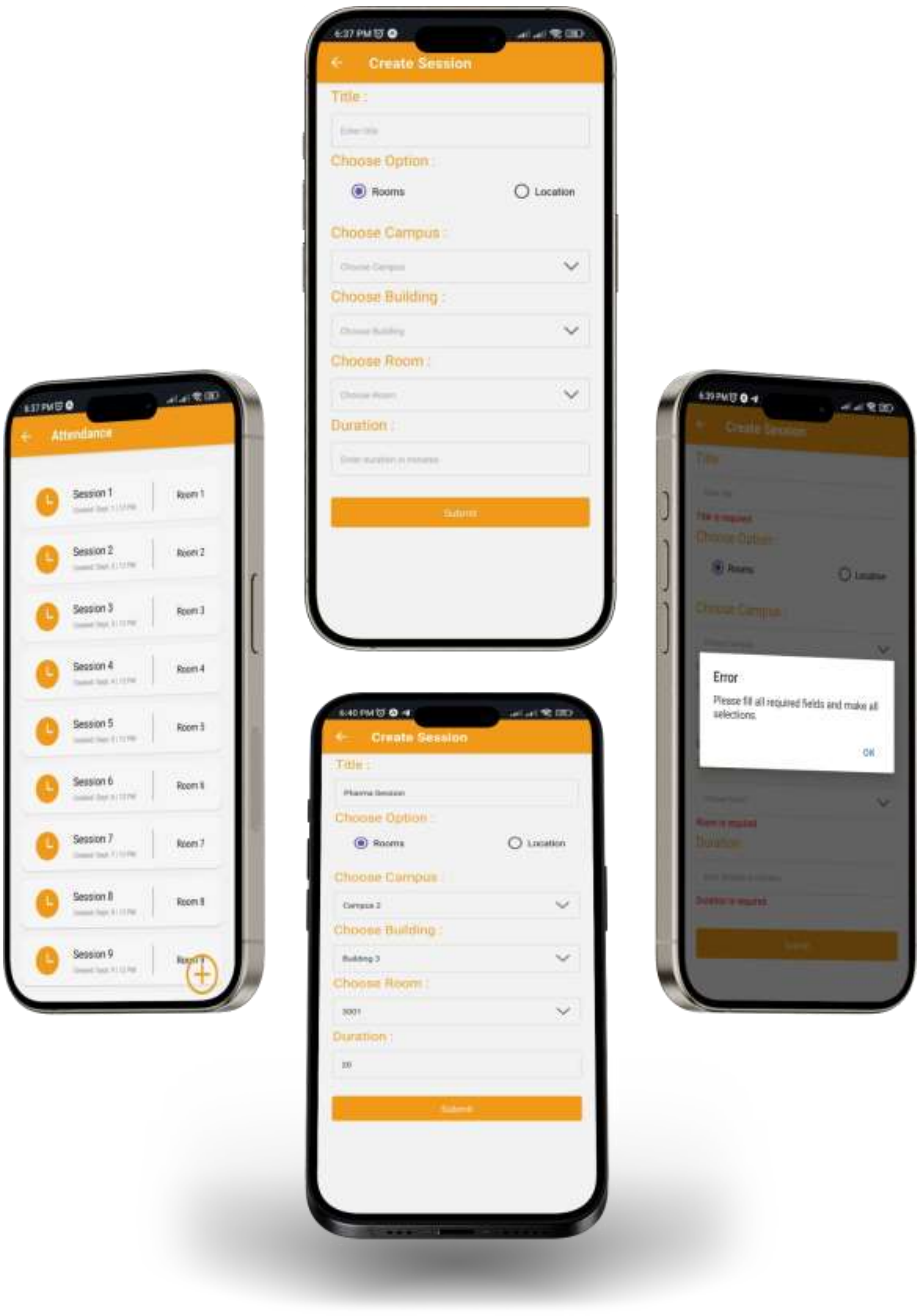
4.3.2.3 Classroom Module



4.3.2.4 Exam Module



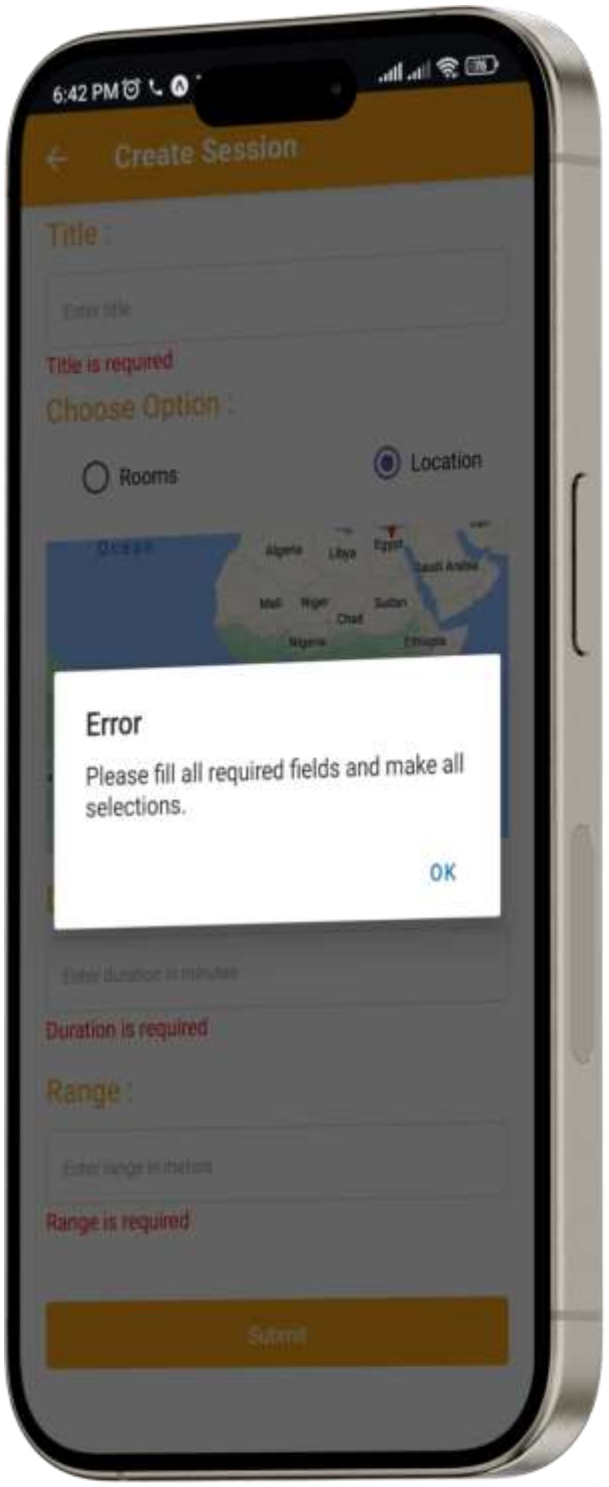
4.3.2.5 Attendance Module



Title : Pharma Session
Choose Option : [✓ Rooms] Location
Choose Campus : Campus 2
Choose Building : Building 3
Choose Room : 3001
Duration : 20

Error

Please fill all required fields and make all selections.



Create Session

Title :

Enter title

Title is required

Choose Option :

Rooms [✓ Location]

Error

Please fill all required fields and make all selections.

OK

Enter duration in minutes

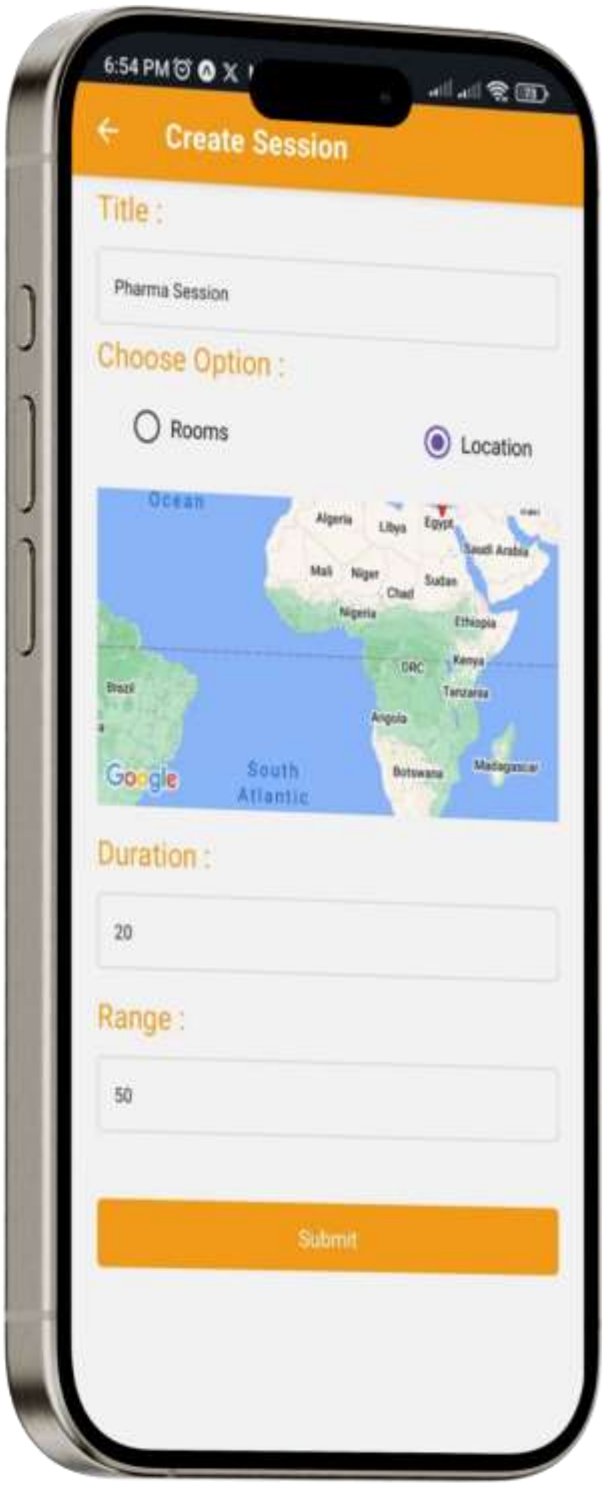
Duration is required

Range :

Enter range in meters

Range is required

Submit



Create Session

Title : Pharma Session

Choose Option :

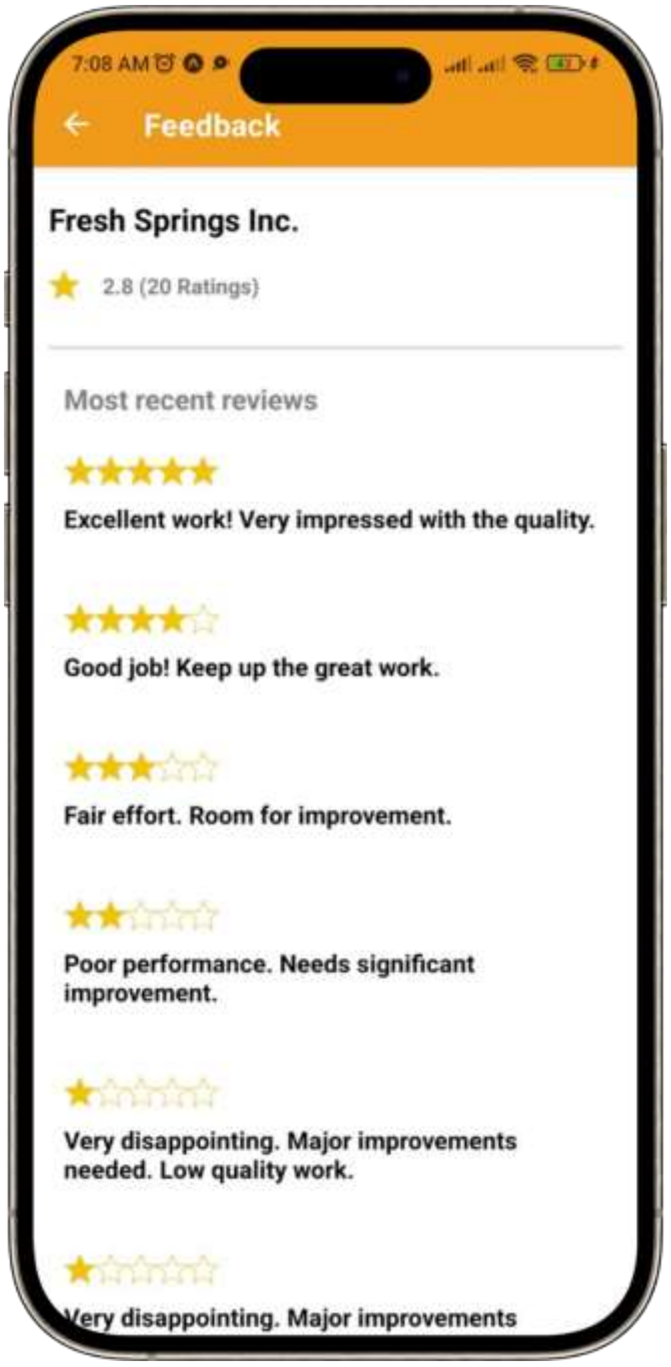
Rooms [✓ Location]

Duration : 20

Range : 50

Submit

4.3.2.6 Feedback Module



Fresh Springs Inc.

★ 2.8 (20 Ratings)

Most recent reviews

★★★★★
Excellent work! Very impressed with the quality.

★★★★☆
Good job! Keep up the great work.

★★★☆☆
Fair effort. Room for improvement.

★★☆☆☆
Poor performance. Needs significant improvement.

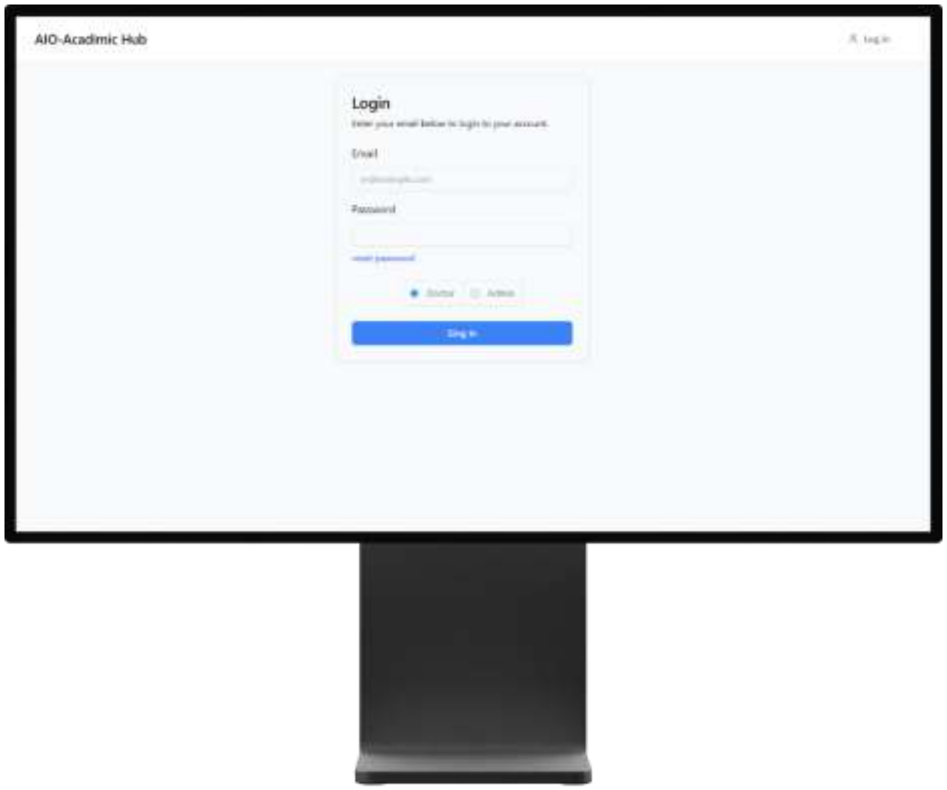
★☆☆☆☆
Very disappointing. Major improvements needed. Low quality work.

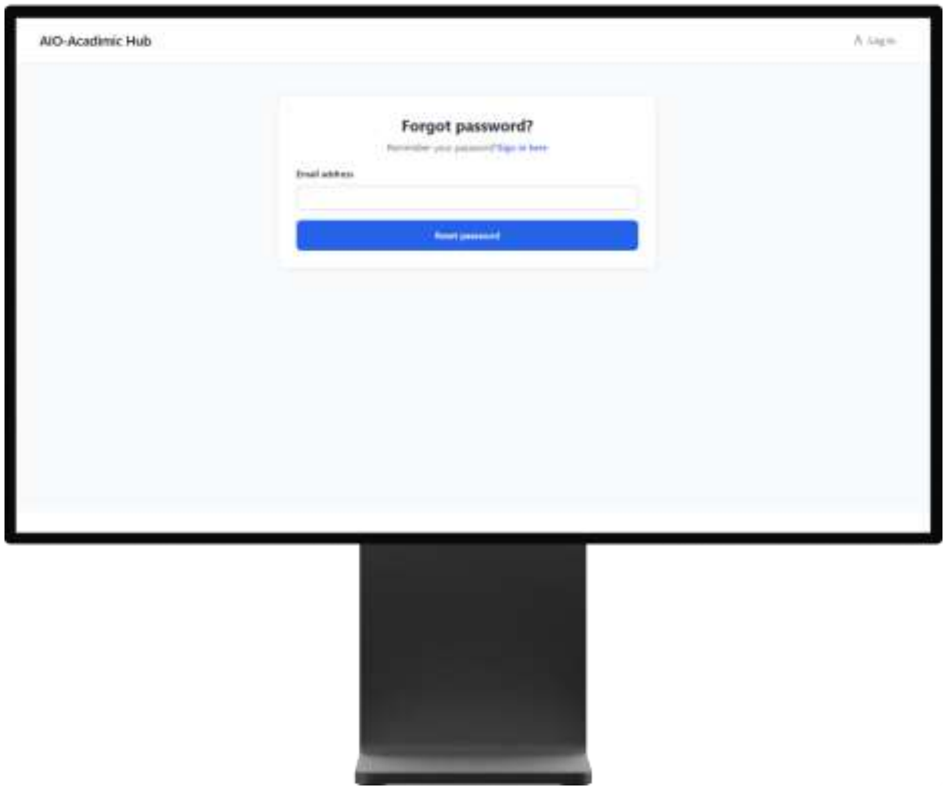
★☆☆☆☆
Very disappointing. Major improvements

4.3.2.7 Chat Module (mentioned earlier)

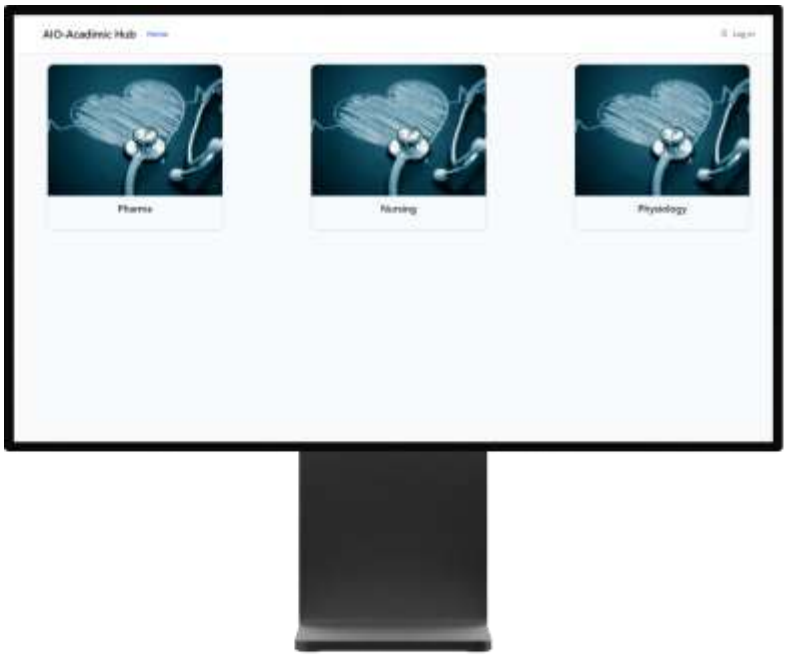
4.4 Dashboard screens

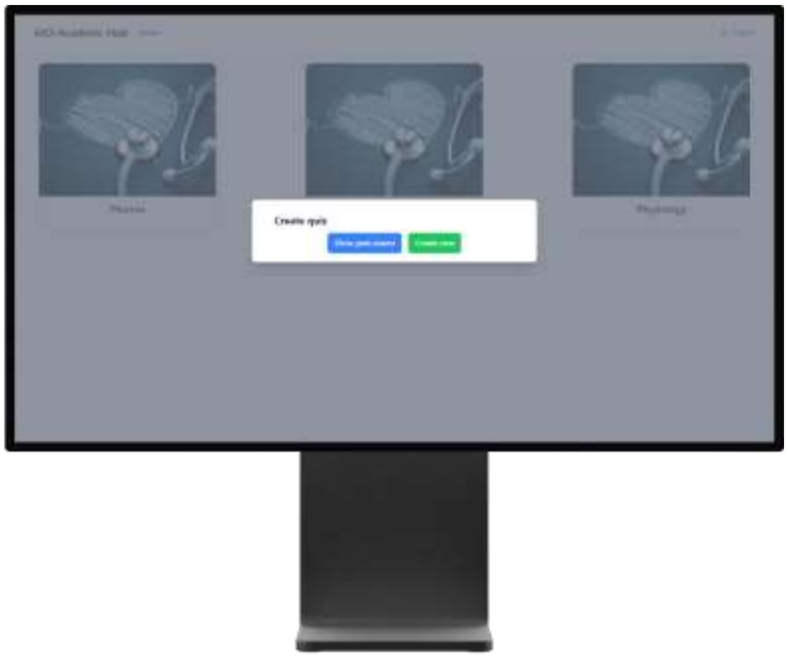
4.4.1 Login & Reset password

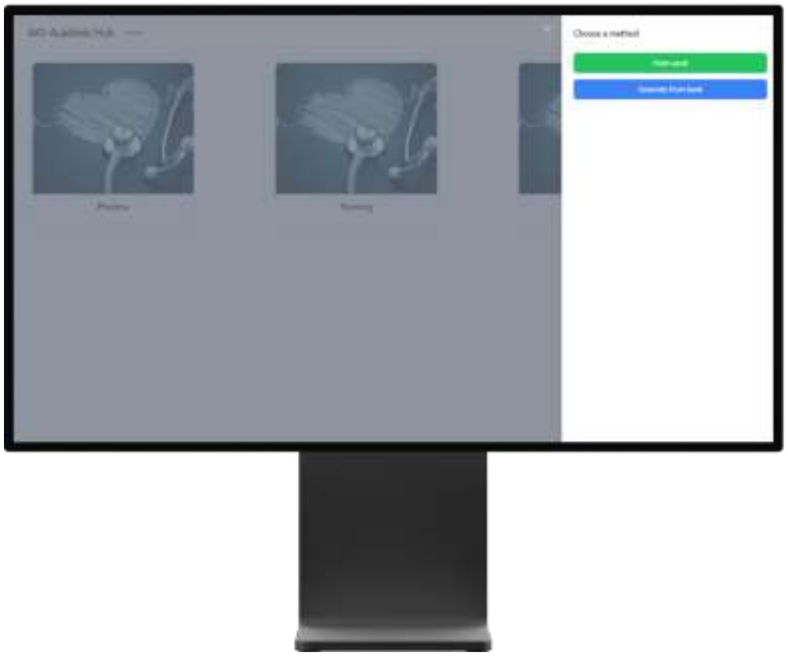


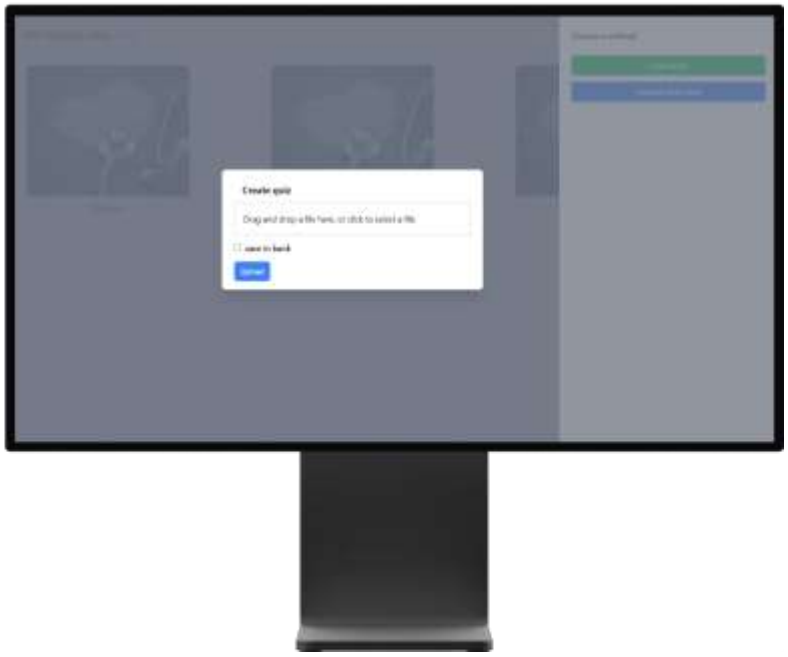


4.4.2 Home page & Method picker with Generate from Excel

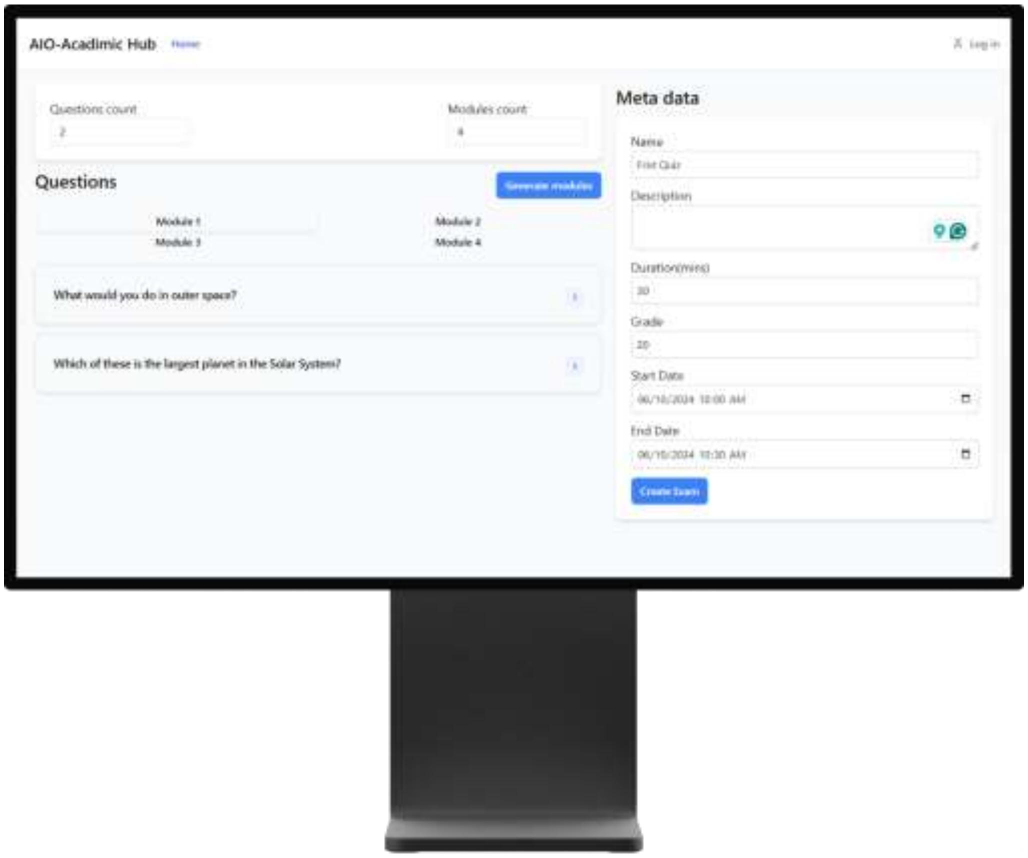








4.4.3 Generate From bank method



Questions count
2

Modules count
4

Questions

Module 1 Module 2
Module 3 Module 4

What would you do in outer space?

Which of these is the largest planet in the Solar System?

Meta data

Name
First Quiz

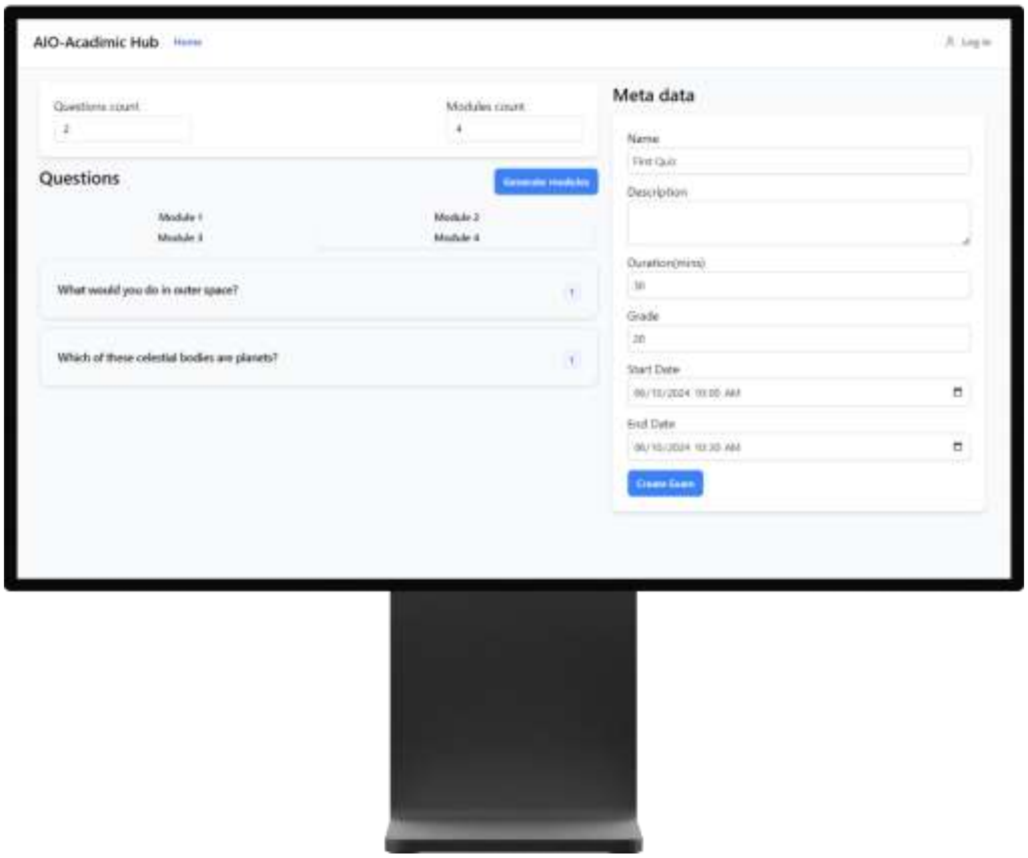
Description

Duration(mins)
30

Grade
20

Start Date
06/10/2024 10:00 AM

End Date
06/10/2024 10:30 AM



Questions count
2

Modules count
4

Questions

Module 1 Module 2
Module 3 Module 4

What would you do in outer space?

Which of these celestial bodies are planets?

Meta data

Name
First Quiz

Description

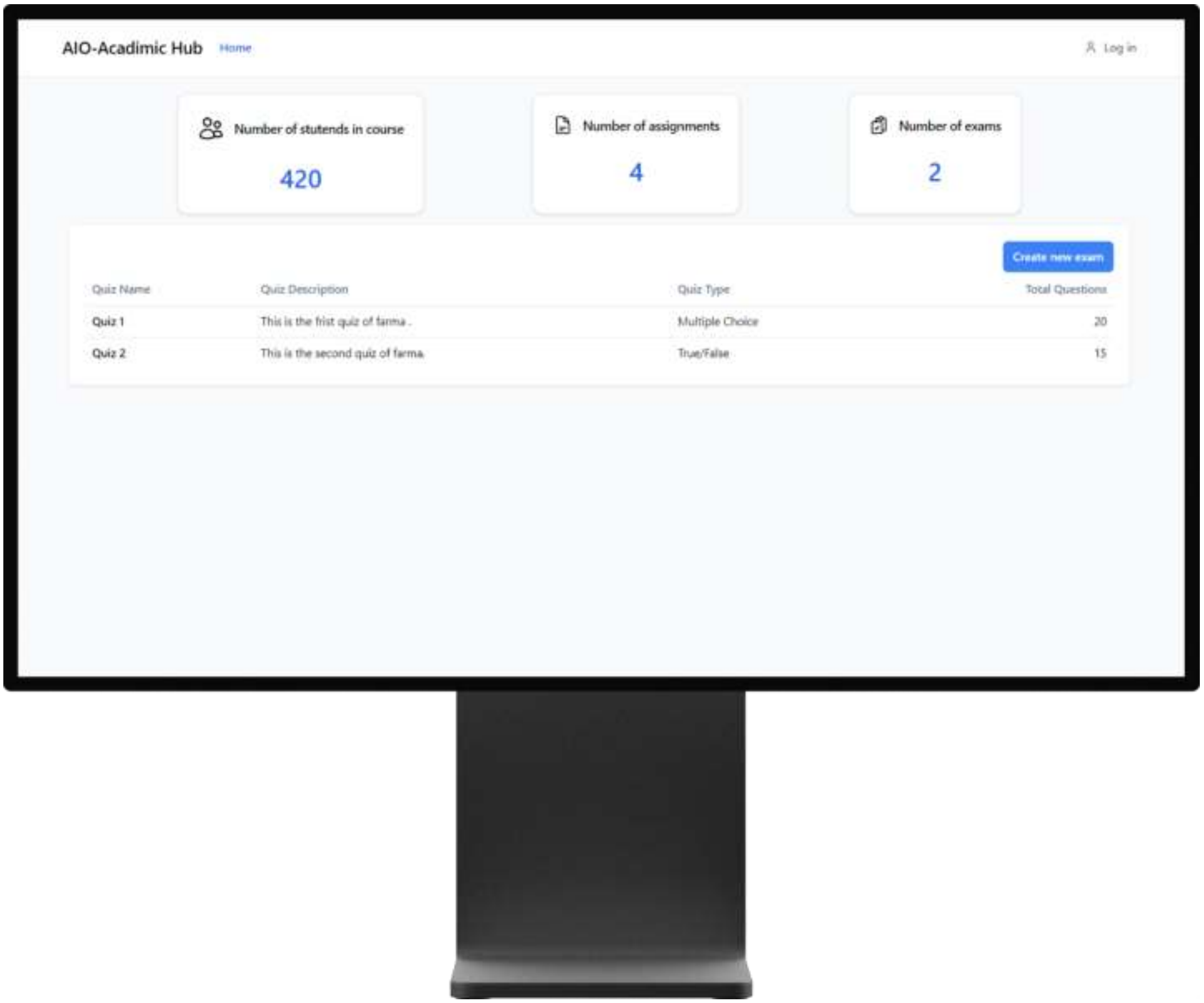
Duration(mins)
30

Grade
20

Start Date
06/10/2024 10:00 AM

End Date
06/10/2024 10:30 AM

4.4.4 Show the past exams details



AIO-Acadimic Hub Home

Log in

Number of stutends in course 420

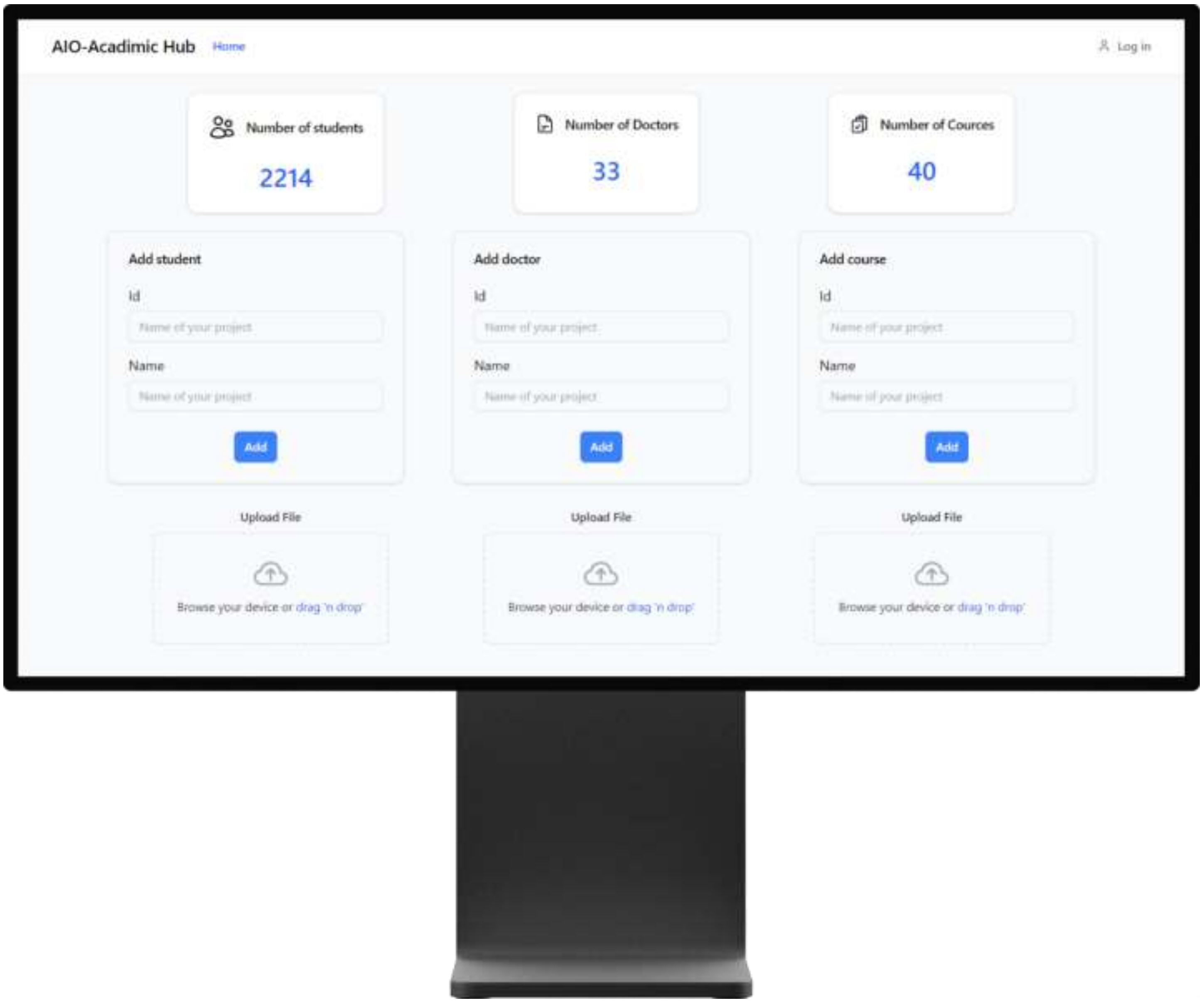
Number of assignments 4

Number of exams 2

Create new exam

Quiz Name	Quiz Description	Quiz Type	Total Questions
Quiz 1	This is the frist quiz of farma .	Multiple Choice	20
Quiz 2	This is the second quiz of farma.	True/False	15

4.4.5 Admin dashboard home page



References

Database of the Year: Postgres. (2021, October 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9520330

Potdar, A. M., Narayan, D. G., Kengond, S., & Mulla, M. M. (2020). *Performance evaluation of docker container and virtual machine*. Procedia Computer Science, 171, 1419–1428. https://doi.org/10.1016/j.procs.2020.04.152

Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis. (2016, August 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/7847138

Pan, C., Luo, Y., Wang, X., & Wang, Z. (2019). *pRedis: Penalty and Locality Aware Memory Allocation in Redis*. Association for Computing Machinery. https://doi.org/10.1145/3357223.3362729

Sharma, N., & Agarwal, R. (2023). *HTTP, WebSocket, and SignalR: A comparison of Real-Time Online Communication Protocols*. In Lecture Notes in Computer Science (pp. 128–135). https://link.springer.com/chapter/10.1007/978-3-031-44084-7_13

Wright, D. (2011). *Software Life Cycle Management Standards: Real-world Scenarios and Solutions for Savings*. IT Governance Publishing.

Shore, J., & Warden, S. (2008). *The art of agile development*. "O'Reilly Media, Inc."

Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. "O'Reilly Media, Inc."

Beighley, L. (2007). *Head first SQL: Your Brain on SQL -- A Learner's Guide*. "O'Reilly Media, Inc."

Oxley, T., Rajlich, N., Holowaychuk, T., & Young, A. (2017). *Node.Js in action*. Simon and Schuster.

Casciaro, M., & Mammino, L. (2020). *Node.js design patterns: Design and implement production-grade Node.js applications using proven patterns and techniques, 3rd Edition*. Packt Publishing Ltd.

Moiseev, A., & Fain, Y. (2020). *TypeScript quickly*. Simon and Schuster.