
# Bahir Dar University

## Bahir Dar Institute of Technology

## Faculty of Computing

## Requirement Analysis Document (RAD)

for

**Fidel AI**: AI-Powered English Learning and Online Tutor System

Submitted to the faculty of computing in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science


| Group Member Name | ID number |
|---|---|
| 1 Sisay Atinkut…… | BDU 1507749 |
| 2 Walelign Enemayehu…… | BDU 1404442 |
| 3 Yohanes Debebe…… | BDU 1508306 |

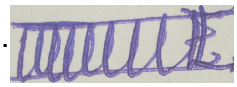
Advisor : Ms Birtukan S.


**2018**

**Bahir Dar University, Bahir Dar Institute of Technology**

# Declaration

The Project is our own and has not been presented for a degree in any other university and all the sources of material used for the project have been duly acknowledged.

| **Name** | **Signature** |
|---|---|
| 1 Sisay Atinkut …….. |  |
| 2 Walelign Enemayehu …….…….. |  |
| 3 Yohanes Debebe …….. |  |

**Faculty:**Computing
**Program**: Regular
**Project Title**:Fidel AI: AI-Powered English Learning and Online Tutor System

This is to certify that I have read this project and that in my supervision and the students' performance, it is fully adequate, in scope and quality, as a project for the degree of Bachelor of Science.

| Ms. Birtukan Shegaw |  |
|---|---|
| **Name of Advisor** | **Signature** |

| Examining Committee Member | **Signature** |
|---|---|
| **1……** | **……** |
| **2……** | **……** |

It is approved that this project has been written in compliance with the formatting rules laid down by the faculty.

# Roles and Responsibilities of the Group Members

| List of Tasks | List members | | |
|---|---|---|---|
| | Sisay Atnkut | Walelign Enemayehu | Yohanes Debebe |
| Data gathering and requirement analysis | √ | √ | √ |
| Figma Design | | √ | |
| Use case drawing | | | √ |
| Sequence diagram | √ | | |
| Class diagram | | | √ |
| State-chart diagram | | | √ |
| Document organization | √ | √ | √ |

# Acknowledgment

We would like to express our heartfelt gratitude to our advisor, Ms. Birtukan Shegaw for her valuable guidance, continued interest, and encouragement throughout our project. Her insightful comments and constructive suggestions paved the way for this work. We would also like to take this opportunity to thank our peers and colleagues in the Information Technology program for the sharing of ideas and collaboration. Last but not least, we would like to extend our thanks to the faculty of computing for their provenance as the best document preparation template.

# List of acronyms

- **AI**: Artificial Intelligence
- **API**: Application Programming Interface
- **BDU**: Bahir Dar University
- **BR**: Business Rule
- **CRC**: Class-Responsibility-Collaborator
- **FREQ**: Functional Requirement
- **IDE**: Integrated Development Environment
- **iOS**: iPhone Operating System
- **LLM**: Large Language Model
- **NFREQ**: Non-Functional Requirement
- **OOAD**: Object-Oriented Analysis and Design
- **RAD**: Requirement Analysis Document
- **UI**: User Interface
- **UML**: Unified Modeling Language

# List of Figures

# List of Tables

# Table of Contentes

# Abstract

English language learning in developing regions faces significant challenges, including limited access to localized digital tools, lack of real-time tutor interaction, and the absence of locally adapted pronunciation support and payment methods. To address these gaps, this project proposes Fidel AI, an AI-powered English learning and online tutor system designed specifically for learners in regions such as Ethiopia.

The platform integrates artificial intelligence for personalized learning paths, grammar correction, alongside a tutor connector that facilitates live, scheduled sessions. It also incorporates local payment gateways Chapa to ensure accessibility. Developed using an Object-Oriented Analysis and Design (OOAD) approach and agile methodology, the system comprises a mobile application (React Native) and a web dashboard (React/FastAPI), supported by UML modeling for system design.

Key features include personalized learning modules, batch management, community forums, and an administrative dashboard for centralized oversight. The system aims to replace unstructured learning methods with a structured, engaging, and locally relevant educational experience. While the project is currently scoped to English language learning, its modular architecture allows for future expansion into additional languages and advanced AI features.

Fidel AI represents a step toward democratizing language education through technology, offering a scalable, affordable, and effective solution for students, tutors, and educational institutions in underserved communities.

# Chapter One: Introduction

## 1.1 Background

English has become the most widely used language worldwide for education, business, science, and international communication. As a result, English language schools are expanding rapidly, especially in developing regions where students aim to improve their language skills for academic and professional advancement. Despite this demand, many schools still use traditional methods of teaching and lack digital platforms that support flexible and personalized learning.

In recent years, Artificial Intelligence (AI) and mobile technologies have transformed the way educational content is delivered. AI-powered learning systems can provide personalized lessons, pronunciation assistance, instant feedback, and continuous practice opportunities. Mobile learning (m-learning) enables students to access content anytime and anywhere, making education more flexible.

However, many global language-learning applications are designed primarily for Western users and do not match the unique needs of local learners. Issues such as inaccessible payment methods, Western-centric pronunciation models, limited real-time tutor interaction, and low dedication to self-study platforms like YouTube create major barriers for students in developing regions.

To address these challenges, this project proposes an AI-Powered English Learning Mobile Application with a Tutor Connector. The system allows students to learn English through AI-generated exercises while being supported by real human tutors assigned by the administrator. The platform integrates English learning packages, local payment options, AI pronunciation tools, structured content delivery, and real-time tutor communication providing a modern and practical solution for language schools.

## 1.2 Statement of the Problem

English language schools and learners face several real and persistent challenges in accessing effective digital learning systems. The key problems include:

**1. Limited Access to Local Payment Methods**

Most global language-learning applications use Western payment methods such as PayPal, Visa, Mastercard, and Apple Pay. These options are not accessible to many local users. Students in developing regions rely more on local payment services,

Chapa, HelloCash, E-Birr, Bank Transfer, etc.The lack of locally integrated payment methods prevents students from subscribing to learning packages and accessing structured digital learning.

**2. Pronunciation Challenges for Local Learners**

Local English learners often struggle with pronunciation due to differences between the sounds in their native language and English. International apps use native-speaker models that do not consider local accents or phonetic limitations. This makes it difficult for learners to practice speaking accurately, leading to frustration and slow progress.

**3. Lack of Real-Time Tutor Connection**

Although many apps offer automated lessons, they do not provide an effective system for real-time communication with tutors. Local schools depend on phone calls or informal messaging apps for communication, which results in missed messages, confusion, lack of follow-up, and no structured learning monitoring. There is no digital platform where tutors, students, and administrators can interact efficiently in real time.

**4. Low Dedication When Learning Through YouTube**

Many language schools try to help students by uploading lessons to YouTube. However, students easily lose focus due to unrelated video recommendations, ads, and lack of teacher follow-up. Without progress tracking, personalized learning paths, or tutor involvement, learners often become inconsistent and unmotivated. This affects learning outcomes and reduces the effectiveness of school-based online learning.

**Summary of the Problem**

These issues create major barriers for English learners, especially in developing countries. Existing apps do not integrate local payment options, provide locally adapted pronunciation guidance, support real-time tutor connections, or offer structured and distraction-free learning. Therefore, there is a need for an integrated solution that solves these problems and improves English learning for students.

## 1.3 Objectives

### 1.3.1 General Objective

To design and develop an AI-Enhanced English Learning Mobile Application that supports local payment systems, addresses pronunciation challenges, enables real-time tutor interaction, and provides structured English learning for language schools.

### 1.3.2 Specific Objectives

- To develop a system where students can select English learning packages, preferred schedules, and pricing.
- To integrate local payment methods suitable for the region, making subscription accessible for all students.
- To provide an administrator dashboard for managing tutors, students, payments, schedules, and learning packages.
- To ensure scalability of the system for future expansion to other languages and advanced features.

## 1.4 Methodology

A strong methodology provides the foundation for the successful design and implementation of the AI-Powered English Learning and Online Tutor System. Since the system includes both a mobile application and a full web platform, and integrates complex components such as AI engines, payment gateway, live teaching, student dashboards, tutor dashboards, and an admin panel the methodology must be multi-layered, systematic, and flexible.

### 1.4.1 Requirement Gathering Methods

Requirements gathering is the systematic process of collecting, identifying, analyzing, and documenting all the needs and expectations of users, stakeholders, and the environment in which the system will operate. It ensures that the final system fully addresses the real problems and delivers the intended solution.

**Purpose of Requirements Gathering:-**

- Identify functional and non-functional requirements for:
  - AI Learning modules
  - Speaking evaluation system

- ➢ Payment integration
- ➢ Live class system
- ➢ User, tutor, and admin interface

- ❖ Analyze user needs.
- ❖ Study local payment systems (Chapa).
- ❖ Define integration needs with online teaching organizations.
- ❖ Define learning modules.

**Requirements Gathering Techniques:-**

**a) Interview**

Detailed conversations with:

- ❖ English learners
- ❖ English tutors
- ❖ Institutions offering English training

Purpose: Understand real learning challenges, system expectations, payment preferences, and tutor needs.

**b) Questionnaire (Survey)**

Distributed online to many students to collect:

- ❖ Common learning gaps
- ❖ Mobile/web usage behavior
- ❖ Preferred learning style

**c) Observation**

Monitoring actual teaching sessions in:

- ❖ Private tutoring
- ❖ Online tutor programs

Purpose: Identify real problems that students face in real-life learning sessions.

**d) Document Analysis**

**Reviewing:**

- ❖ Existing language apps

- ❖ Payment platform documents
- ❖ AI language model documentation

Purpose: Understand features of existing systems and identify what is missing.

**e) Literature Review**

Studying academic and modern industry papers on:

- ❖ Grammar correction systems
- ❖ Personalized language learning
- ❖ Online tutoring platforms

This gives a scientific foundation to the system design.

**Requirement Modeling Approach:-**

We use Object-Oriented Analysis and Design (OOAD) because:The system includes many interacting entities (Student,Tutor,Admin,AIEngine).OOAD simplifies modular design, future updates, and reuses. UML diagrams make system understanding easy. UML Models used: Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram, and ER Diagram.

### 1.4.2 Analysis and Design Methodology

The system we followed Object-Oriented Analysis and Design(OOAD) methodology for analysis and design phases. This methodology focuses on modeling the system based on real-world entities and the interactions between them. This approach helps ensure that the system remains modular, scalable, reusable, and easy to maintain.The system is divided into key objects such as Student, Teachers, AI-Engine, Lesson and Assessment, which have specific attributes and behaviors. This approach supports reusability, encapsulation, and extensibility, allowing the system to easily evolve over time.

**OOAD was chosen for three major reasons:**

**Modularity:-** where the system is broken down into smaller, manageable components, making development and maintenance easier.

**Real-world representation:-** since the platform handles real entities which fit naturally into the OOAD approach.

**Flexibility:-** allowing developers to add new features or adjust existing functionalities without disrupting the entire system, enabling the platform to evolve based on user requirements and feedback.

For the analysis and design stages, several tools were used. UML modeling tools were used to prepare diagrams that provide clear and consistent documentation. In addition, Figma was used for user-interface wireframing and prototyping, which helped the team quickly test interface ideas and gather early feedback.

### 1.4.3 Implementation Methodology

For the development phase, we adopted the Agile methodology to ensure iterative development and quick feedback cycles. Agile allowed us to work in sprints, delivering small increments of functionality, continuously testing, and adjusting the system based on user input. This approach ensures that the final product aligns closely with user expectations and can be adapted to changes in requirements during the development process.

In addition the following tools will be used for the development and support of the platform:

| Tools | Categories | Purpose |
|---|---|---|
| Phyton, FastAPI | Programing Language | Backend |
| React, ReactNative | Frameworks | Web and mobile development |
| Jitsi meet API | Video conferencing tool | To support live tutor–student video classes inside the system. |
| Figma | UI/UX design tool | design the interface prototype for both mobile and web applications before implementation. |
| UML | System modeling tools | To design use case diagrams, class diagrams, sequence diagrams, and other UML artifacts. |
| Chapa | Payment integration tool | To enable secure online payments for tutor |

| | | |
|---|---|---|
| | | sessions and premium features within Ethiopia. |
| LangChain | Natural language processing AI | To perform grammar correction, writing improvement, and AI chat-based learning support. |
| GitHub | Version control system | To manage and track changes in the project's code, ensuring collaboration and safe backups. |
| Vs code | Development IDE | to write code for frontend, backend, database scripts, and AI components. |

Table 1: tools used

## 1.5 Feasibility Study

**Technical Feasibility**

- Uses free and open-source AI models:While cost-effective, this might require more in-house expertise for customization, integration, and maintenance compared to commercial solutions. It also implies careful selection to ensure models meet performance requirements.
- Uses free video conferencing platform:Jitsi is a good open-source option, but its scalability and reliability for a large number of concurrent live classes need careful assessment and potentially self-hosting/managing for better control.
- Mobile App development tools :This is a common and robust tech stack, indicating good technical feasibility.
- Payment integration is possible with chapa.

**Economic Feasibility**

- Low implementation cost due to open source technologies and free APIs: This is a strong advantage, but it's important to factor in the cost of developer time for integration, customization, and ongoing maintenance of these open-source components.

- ❖ Revenue generation through subscription to courses: This outlines a clear multi-faceted business model, which is healthy. Detailed financial projections would be needed here.
- ❖ The system can become self-sustaining.

**Operational Feasibility**

- ❖ Students are familiar with mobile learning applications:This reduces the barrier to entry for users and simplifies user adoption strategies.
- ❖ Organizations can manage students and teachers efficiently through the admin dashboard.
- ❖ The AI system provides extra practice without needing teachers.

**Time Feasibility**

- ❖ Development can be completed within the academic period.
- ❖ Uses agile development:This is a key clarification. Agile methodologies (like Scrum or Kanban) allow for flexibility, continuous feedback, and faster delivery of working software in iterations (sprints), which is well-suited for projects with evolving requirements.

## 1.6 Significance/Beneficiary

**Students**

- ❖ Learn English anytime using AI tools.
- ❖ Book tutors easily for live sessions.
- ❖ Receive grammar correction, and personalized learning.
- ❖ Access affordable learning with local payments..

**Teachers**

- ❖ Create income opportunities through the tutor marketplace.
- ❖ Teach globally using online classes.
- ❖ Receive automated payments and earnings reports.

**Educational institutions**

- ❖ Use the system as blended learning tools.
- ❖ Provide modern English learning solutions.
- ❖ Use as a source of income

**Society**

- ❖ Improves English communication skill in the community.
- ❖ Supports digital learning and technological innovations.
- ❖ Expands access to quality English education at low cost.

## 1.7 Limitation of the project

**Technical limitations**

- ❖ The accuracy of AI speaking evaluation depends on the quality of microphone and environmental noise.
- ❖ Internet connection is required for:
  - ➢ Live video classes
  - ➢ Payment integration
  - ➢ Some AI features
- ❖ Free AI models may not match the accuracy of premium proprietary models such as Google speech or Amazon Transcribe.

**Operational Limitations**

- ❖ The system does not guarantee tutor quality beyond initial verification.
- ❖ Payment services ( Chapa) may face downtime or service interruptions.
- ❖ Live video classes depend on Jitsi's server stability.

**Scope Limitations**

- ❖ The project focuses only on English language learning; other languages are not included.
- ❖ AI grammar correction explains errors but doesn't provide professional academic writing services.

**Resource Limitations**

- ❖ Limited development time may restrict advanced features like:
  - ➢ Offline AI learning
  - ➢ Automatic essay grading
  - ➢ Multi-teacher classrooms
  - ➢ Advanced analytics (deep learning-based recommendations) may not be fully implemented.

## 1.8 Scope of the Project

**In-scope features**

**A. Student features**

- Register, login, and manage profiles.
- English level assessment test
- AI generating learning plan
- Grammar lessons, vocabulary training, quizzes
- AI chatbot
- Join video classes
- Make secure payment

**B. Teachers features**

- Teach live classes
- View earnings and student reviews

**C. Admin features**

- Manage students and teachers
- Approve tutor applications
- Manage courses and content
- Oversee financial transactions
- Handle dispute and issues
- Generate system-wide reports

**Out-of-scope features:**

- Multi-language learning (English only)
- Offline full functionality (AI tools require internet)

## 1.9 Organization of the project

This document is organized into the following main chapters, each addressing a specific phase of the AI-Powered English Learning and Online Tutor System:

**Chapter 1: Introduction**

This chapter establishes the foundation for the FidelAI project.It begins by outlining the challenges in modern language education, such as the lack of personalized feedback

and accessible speaking practice. The chapter defines the project's primary goal: to develop an AI-powered platform that integrates adaptive learning with live online teaching. It details the methodology for requirement gathering and system design, assesses the project's feasibility, and clearly scopes the system's functionalities and limitations, setting the stage for the detailed analysis to follow.

**Chapter 2: System Features and Requirement Analysis**

The conceptual vision from Chapter 1 is transformed into precise technical specifications. The chapter provides a thorough analysis of functional requirements for students, teachers, and administrators. It employs UML modeling including use case, activity, sequence, and state chart diagrams to visualize system interactions and workflows. Business rules governing platform logic, non-functional requirements defining system quality, and hardware/software prerequisites are all documented in detail, forming the complete requirement blueprint for the system.

# Chapter Two: System Features

## 2.1 Existing System

Currently, English language schools rely heavily on manual processes or fragmented digital solutions:

**1. Payment Gaps**

Most learning applications require international payment systems that local students cannot use. Schools often collect payments manually, causing delays and errors.

**2. Weak Pronunciation Support**

Existing apps teach English pronunciation using Western standards. Local students struggle because these models do not match their accent or phonetic background.

**3. No Real-Time Tutor Interaction**

Communication between students and tutors is done through phone calls, Telegram, Messenger, or informal messages. This results in poor scheduling, missed lessons, and weak monitoring.

**4. YouTube-Based Learning Is Ineffective**

Schools upload lessons on YouTube, but students are easily distracted by ads and unrelated content. There is no progress tracking, no evaluation, no tutor feedback, and no structured curriculum.

**5. Lack of Centralized Learning Platform**

There is no unified system combining:

- Student enrollment
- Local payment
- AI learning
- Tutor assignment
- Real-time communication

This makes the overall learning experience inefficient and inconsistent.

## 2.2 Proposed System

The proposed system introduces a complete English learning solution that integrates AI, tutor management, and local payment systems into one mobile application. It is designed specifically to solve the real problems faced by local English schools.

**Key Features and Improvements**

**1. Local Payment Integration**

The app will support Chapa for package subscription, which is easy and accessible.

**2. Real-Time Tutor Connector**

Students select packages and schedule preferences, but tutors are assigned by the admin for proper coordination.
**Tutors can:**

- View assigned students
- Send messages
- Give feedback

Students receive real-time communication and guidance from qualified English tutors.

**3. Structured and Distraction-Free Learning**

Unlike YouTube, the app offers:

- Lesson modules
- AI exercises
- Vocabulary training
- Quizzes
- Tutor live sessions

This increases student dedication and learning outcomes.

**4. Centralized Management for Schools**

Admins can manage:

- Students

- Tutors
- Payments
- Packages
- Schedules
- Reports

All within one administrative dashboard.

## 2.3 Requirement Analysis

### 2.3.1.Functional Requirement

**1. User Management & Authentication**

| ID | Functional requirements | Priority | Description |
|---|---|---|---|
| FREQ-1 | User registration | High | Required for identifying and managing all users in the system. |
| FREQ-2 | Secure user login | High | Core security feature, system access depends on it. |
| FREQ-3 | Password Recovery | Medium | Improves usability but not critical for system operation. |
| FREQ-4 | User profile management | Medium | Needed for personalization but the system can run without full profiles. |

**2. Learning & AI-Based Features**

| ID | Functional requirements | Priority | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| FREQ-5 | English level assessment | High | Essential for personalized learning and AI logic. |
| FREQ-6 | Personalized learning plan | High | Core value of the AI-based learning system. |
| FREQ-7 | AI grammar and writing practice | High | Main self-learning feature replacing constant tutor dependency. |
| FREQ-8 | Learning content access | High | Without content, learning cannot occur |

**3. Tutor & Online Learning Management**

| **ID** | **Functional requirements** | **Priority** | **Description** |
|---|---|---|---|
| FREQ-9 | Tutor registration | High | Required to onboard instructors into the system. |
| FREQ-10 | Tutor profile management | Medium | Improves tutor visibility but not mandatory initially. |
| FREQ-11 | Tutor availability scheduling | High | Required to avoid booking conflicts. |
| FREQ-12 | Live online class | High | Core online teaching feature. |

**4. Payment & Financial Features**

| **ID** | **Functional requirements** | **Priority** | **Description** |
|---|---|---|---|
| FREQ-13 | Online payment | High | Required to access |

| | | | |
|---|---|---|---|
| | processing | | paid services. |
| FREQ-14 | Payment confirmation and receipt | High | Ensures transaction transparency and trust. |

**5. Administration & System Control**

| ID | Functional requirements | Priority | Description |
|---|---|---|---|
| FREQ-15 | User management(admin) | High | Required for system governance and moderation. |
| FREQ-16 | Content management(admin) | High | Ensures learning materials are controlled and updated. |
| FREQ-17 | System monitoring | Medium | Helpful for analytics but not core system functionality. |

**6. Communication & Notification**

| ID | Functional requirements | Priority | Description |
|---|---|---|---|
| FREQ-18 | System notification | Medium | Improves user experience but not system-critical. |
| FREQ-19 | In-App messaging | Low | Optional feature,system can function without it. |

**7. Certification**

| ID | Functional requirements | Priority | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| FREQ-20 | Certificate generation | Medium | Important for course completion recognition. |
| FREQ-21 | Certificate verification | Low | Advanced features mainly for external validation. |

**8. Advanced & Enhancement Features**

| ID | Functional requirements | Priority | Description |
|---|---|---|---|
| FREQ-22 | AI recommendation engine | Medium | Improves personalization over time. |
| FREQ-23 | Offline learning support | Low | Useful in low-connectivity areas but optional. |
| FREQ-24 | Report Export(PDF) | Low | Administrative convenience feature. |

## 2.3.2 System Use Case

**User Management**

**Actors:**

- Admin
- Student
- Tutor

**Description:**

- This group of use cases handles the creation, maintenance, and lifecycle management of user accounts within the learning platform. It encompasses authentication, profile management, and administrative oversight of user data, including soft-delete and restoration capabilities.

**Precondition:**

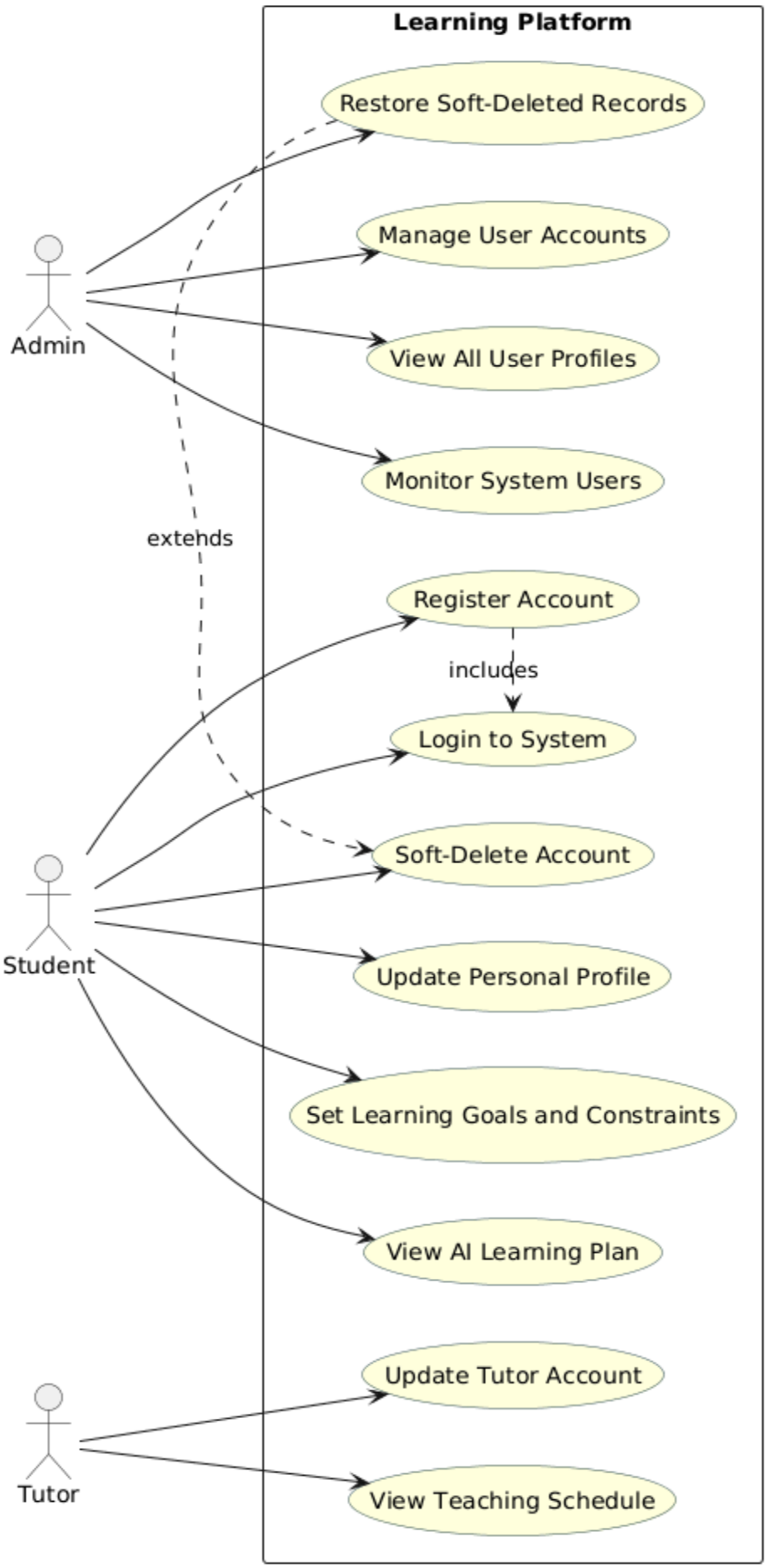
- ❖ The learning platform system is operational and accessible.
- ❖ Administrative functions require the actor to have logged in with Admin privileges.

**Postcondition:**

- ❖ User accounts are created, updated, or managed according to the actor's actions.
- ❖ User profiles contain accurate information.
- ❖ The system's user base is monitored and maintained, with records properly handled even when soft-deleted.

**Basic Steps:**

1. A prospective Student accesses the platform and navigates to the account registration page.
2. The Student provides required details to Register Account, which includes the Login to System process.
3. Upon successful registration and login, the Student can Update Personal Profile and Set Learning Goals and Constraints.
4. Based on the set goals, the system allows the Student to View AI Learning Plan.
5. The Student or Admin can initiate a Soft-Delete Account for a user, moving the account to a deactivated state.
6. An Admin logs in and can Manage User Accounts, View All User Profiles, and Monitor System Users.
7. An Admin can Restore Soft-Deleted Records, reactivating previously deactivated accounts.
8. A Tutor logs in and can Update Tutor Account and View Teaching Schedule.


Learning Platform
Restore Soft-Deleted Records
Manage User Accounts
View All User Profiles
Monitor System Users
extehds
Register Account
includes
Login to System
Soft-Delete Account
Update Personal Profile
Set Learning Goals and Constraints
View AI Learning Plan
Update Tutor Account
View Teaching Schedule
Admin
Student
Tutor

**Batch Management**

**Actors:**

- Admin
- Student
- Tutor

**Description:**

- This group covers the end-to-end management of training batches, from creation and scheduling by administrators to enrollment, participation, and certification for students. It includes community interaction within batches and analytics for administrative oversight.

**Precondition:**

- The User Management subsystem is functional, allowing users (Admin, Student, Tutor) to be authenticated.
- For batch creation, the Admin must have the necessary information available.
- For enrollment, a Student must have an active account.

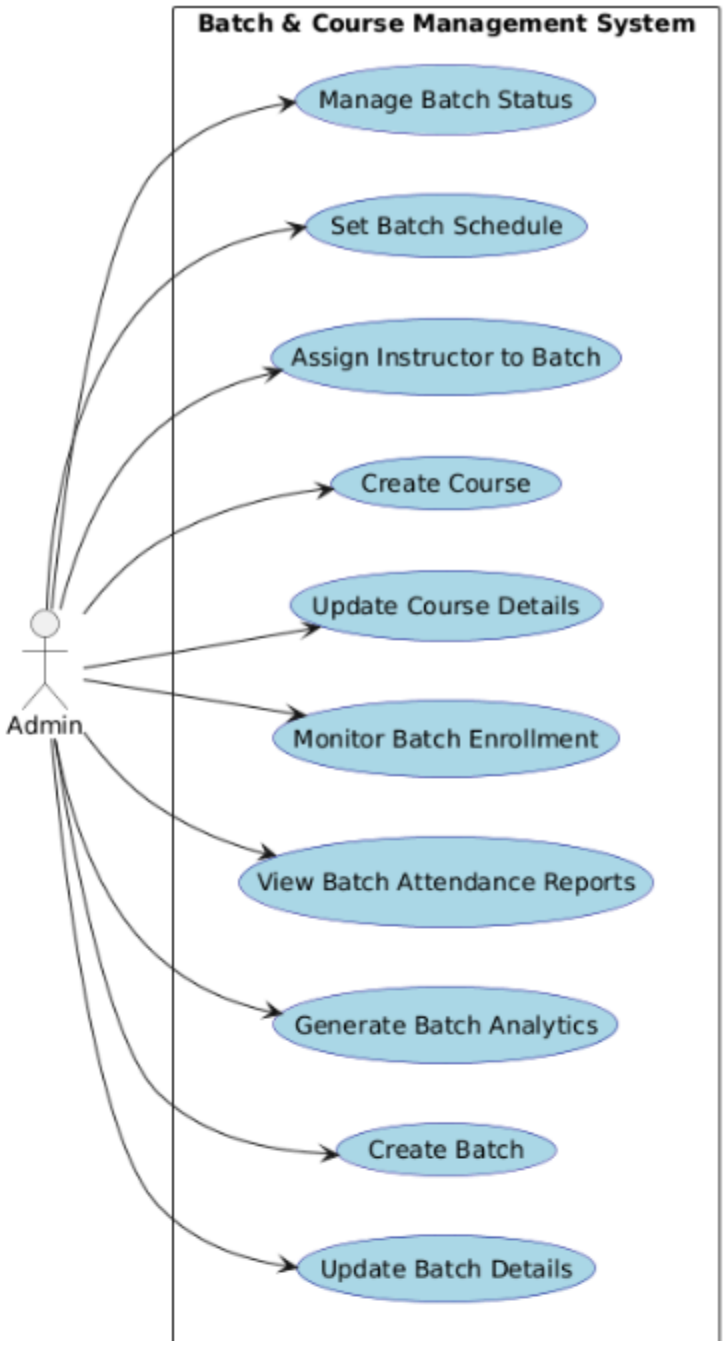
**Postcondition:**

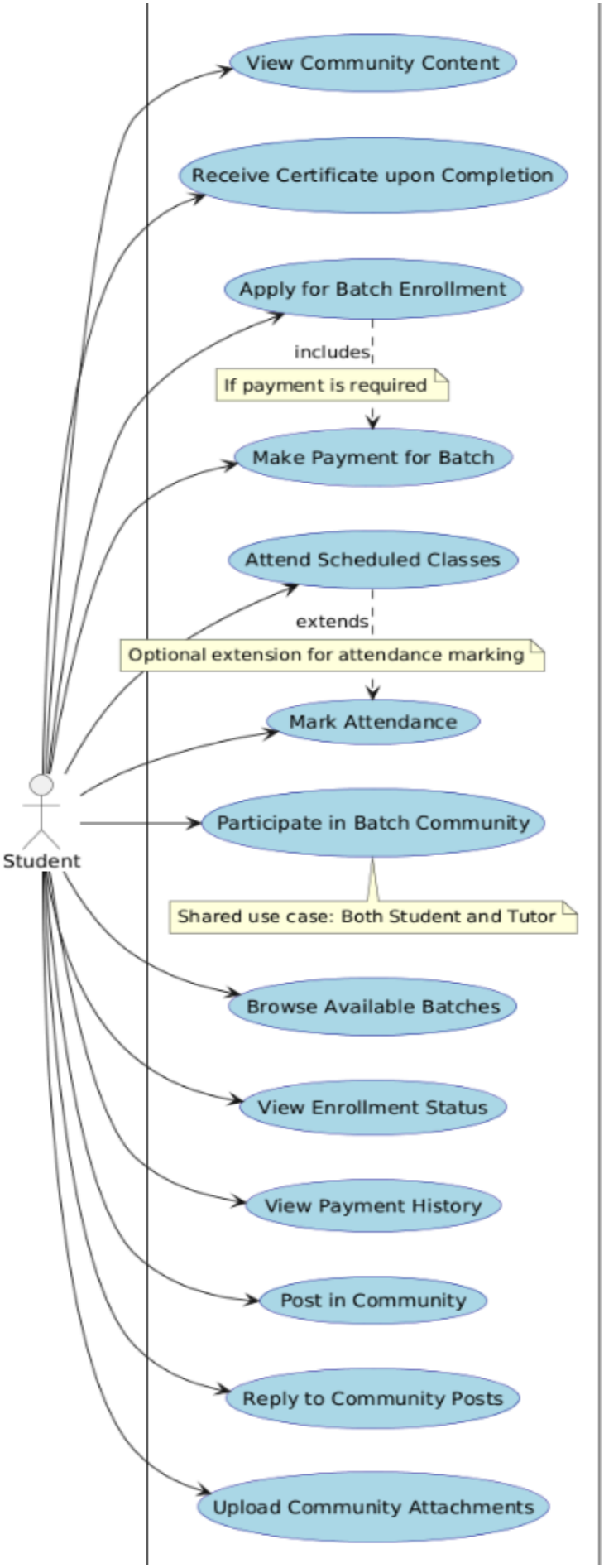
- Batches are created, updated, and scheduled as per administrative actions.
- Students are enrolled in batches, complete payments, attend classes, and can receive certificates.
- Tutors are assigned to batches and can view their schedules.
- A community space exists for each batch, facilitating interaction.
- Administrative reports and analytics are generated based on batch data.

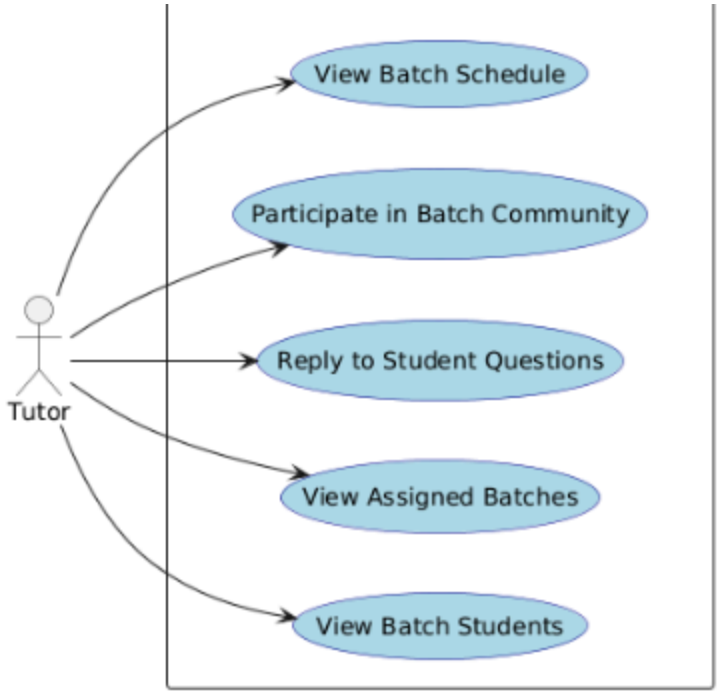
**Basic Steps:**

1. An Admin creates a Course and then a Batch, setting its details, schedule, and status.
2. The Admin Assigns Instructor to Batch.
3. A Student Browses Available Batches and Applies for Batch Enrollment.
4. As part of enrollment, the Student Makes Payment for Batch and can View Payment History.
5. The Admin Monitors Batch Enrollment.
6. Once enrolled, the Student and Tutor can Participate in Batch Community (post, reply, upload, view).

7. The Tutor Views Assigned Batches, Batch Students, and Batch Schedule.
8. The Student Attends Scheduled Classes.
9. The Admin Generates and Views Batch Analytics.
10. Upon batch completion, the Student Receives Certificate upon Completion.




View Community Content
Receive Certificate upon Completion
Apply for Batch Enrollment
includes
If payment is required
Make Payment for Batch
Attend Scheduled Classes
extends
Optional extension for attendance marking
Mark Attendance
Student
Participate in Batch Community
Shared use case: Both Student and Tutor
Browse Available Batches
View Enrollment Status
View Payment History
Post in Community
Reply to Community Posts
Upload Community Attachments




**Notifications & Feedback**

**Actors:**

- Admin
- Student

**Description:**

- This group manages the system's communication and feedback mechanisms. It allows Students to receive and acknowledge system notifications and to submit feedback (optionally anonymous). It enables Admins to view feedback, generate reports, and send notifications to users.

**Precondition:**

- The User Management subsystem is operational, and actors are authenticated.
- The notification system is configured and active.
- For feedback submission, the Student must be using a platform feature or have completed an interaction.

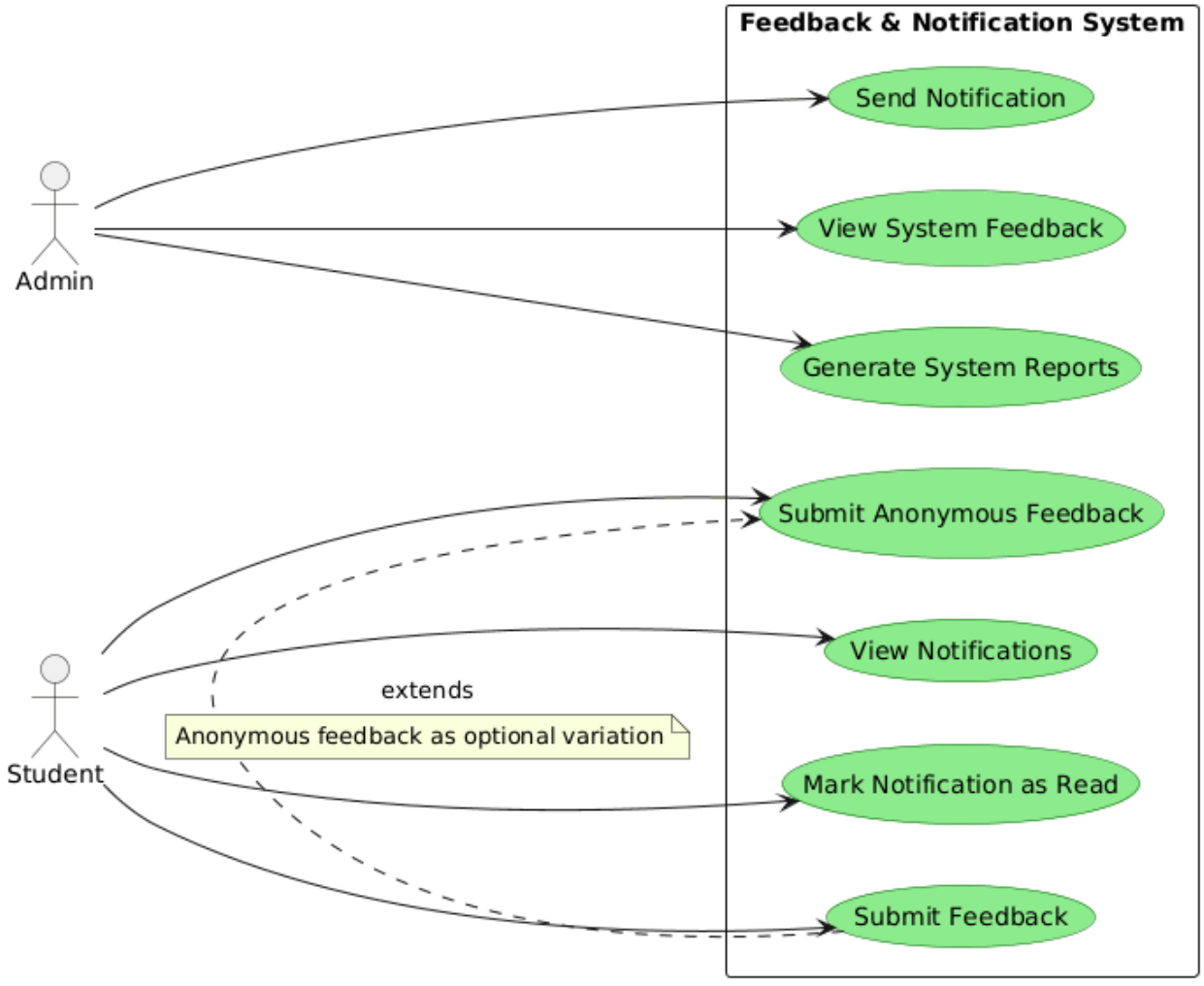
**Postcondition:**

- Notifications are delivered to and acknowledged by Students.

- ❖ Feedback is submitted and stored in the system, available for Admin review.
- ❖ System reports are generated based on feedback and notification data.
- ❖ Notifications are sent to intended recipients as per Admin actions.

**Basic Steps:**

1. The system or an Admin triggers a notification event, causing a notification to be sent to a Student.
2. The Student Views Notifications and Marks Notification as Read.
3. The Student chooses to Submit Feedback, with the option to Submit Anonymous Feedback as an extended variation.
4. The Admin logs in and Views System Feedback.
5. The Admin Generates System Reports based on feedback.
6. The Admin composes and Sends Notification to specific users or groups.




**AI Enhanced Learning System**

**Actors:**

- Student
- AI (as a system actor)

**Description:**

- This group defines the interaction between Students and an AI-driven personalization and support system within the learning platform. It covers the generation of personalized learning paths and content, interactive questions and answers, open-ended conversation, and underlying system optimizations for performance.

**Precondition:**

- The User Management subsystem is functional, and the Student is authenticated.
- The AI subsystem is operational, with necessary models and data available.
- For personalization, the Student has set Learning Goals and Constraints (from User Management group).

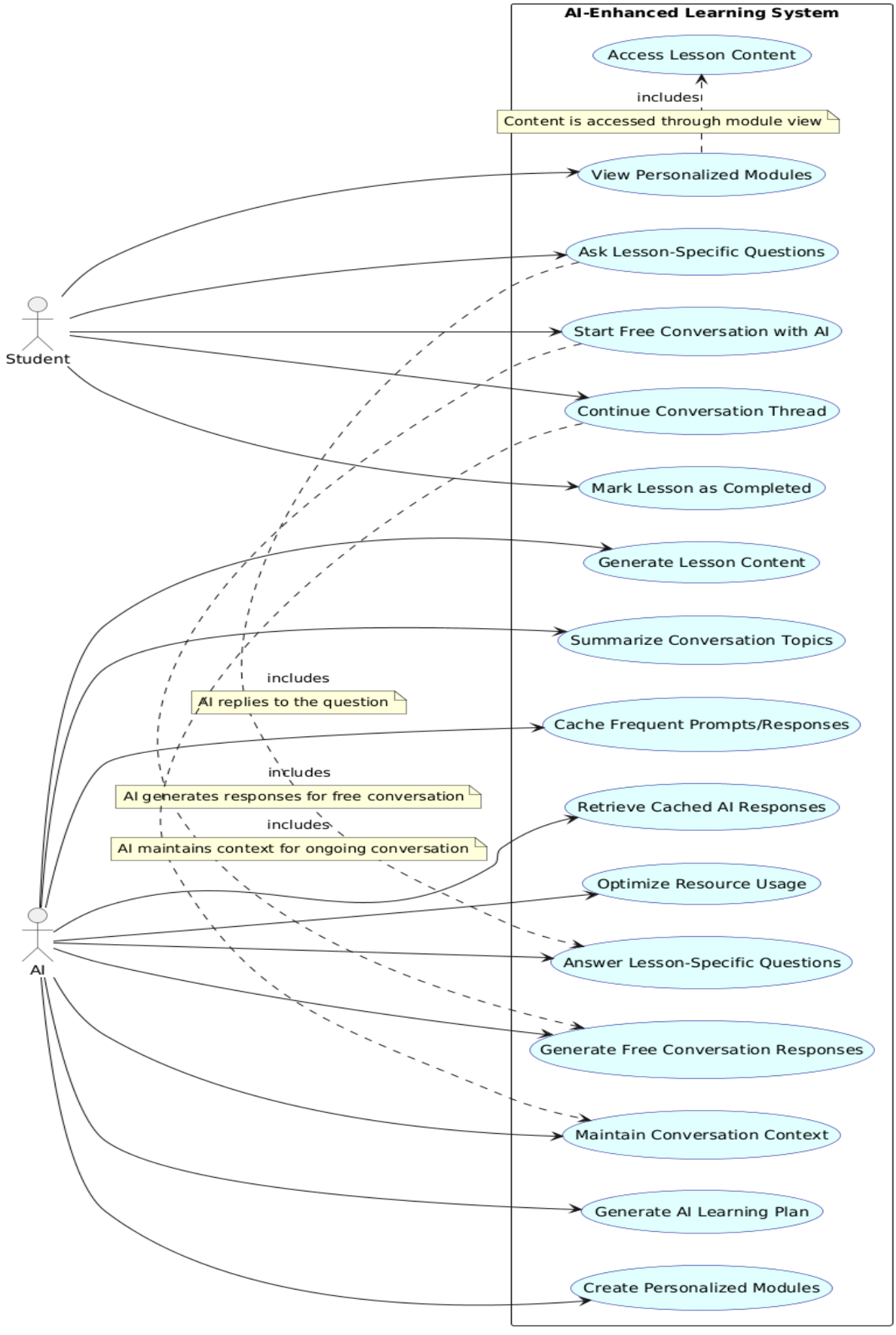
**Postcondition:**

- A personalized learning plan and modules are generated and presented to the Student.
- Lesson content is created and made accessible.
- Student queries and conversations are processed and answered by the AI.
- Conversation context is maintained for ongoing interactions.
- System optimizations are performed to improve response efficiency.

**Basic Steps:**

1. The AI actor Generates an AI Learning Plan based on the student's profile and goals.
2. The AI Creates Personalized Modules and Generates Lesson Content for the plan.
3. The Student Views Personalized Modules and, through that, Accesses Lesson Content.
4. The Student can Ask Lesson-Specific Questions, which triggers the AI to Answer Lesson-Specific Questions.
5. The Student can Start Free Conversation with AI, leading the AI to Generate Free Conversation Responses.

6. The Student can Continue Conversation Thread, and the AI Maintains Conversation Context throughout.
7. In the background, the AI performs system optimizations: Caches Frequent Prompts/Responses, Retrieves Cached AI Responses, and Optimizes Resource Usage.
8. The Student Marks Lesson as Completed.


AI-Enhanced Learning System
Access Lesson Content
includes
Content is accessed through module view
View Personalized Modules
Ask Lesson-Specific Questions
Start Free Conversation with AI
Continue Conversation Thread
Mark Lesson as Completed
Generate Lesson Content
Summarize Conversation Topics
includes
AI replies to the question
Cache Frequent Prompts/Responses
includes
AI generates responses for free conversation
Retrieve Cached AI Responses
includes
AI maintains context for ongoing conversation
Optimize Resource Usage
Answer Lesson-Specific Questions
Generate Free Conversation Responses
Maintain Conversation Context
Generate AI Learning Plan
Create Personalized Modules
Student
AI

## 2.3.3 Business Rule Documentation

Business rules define how the system must operate, what is allowed, what is restricted, and the policies governing all system actions. They ensure consistency, reduce errors, and guide system behavior according to business logic. The business rules for our system are:

**BR-01: User Role Definition Rule**

**Description:-**

- Every registered user shall be assigned exactly one primary role: Student, Tutor, or Administrator.
- A user shall not perform actions outside the permissions of their assigned role.
- Role assignment shall be enforced at both application and database levels.

**Reference:-** FREQ-1, FREQ-2, FREQ-21
**Revision History**:- Initial Definition.
**Purpose:-** Ensures system security and prevents misuse of features.

**BR-02: User Registration Validation Rule**

**Description:-**

- A valid email address or phone number shall be required for registration.
- Duplicate email addresses or phone numbers shall not be allowed.
- Passwords must meet minimum security standards.

**Reference:-** FREQ-1, FREQ-2
**Revision History**:- Initial Definition
**Purpose:-** Prevents duplicate accounts and improves system reliability.

**BR-04: Student Enrollment Rule**

**Description:-**

- A student shall only enroll in courses or batches that are open and active.
- Enrollment shall be confirmed only after successful payment.
- A student shall not be enrolled in the same batch more than once.

**Reference:-** FREQ-1, FREQ-5, FREQ-9

**Revision History**:- Initial Definition
**Purpose:-** Prevents enrollment conflicts and ensures payment accountability.

**BR-05: Payment Processing Rule**

**Description:-**

- All paid services shall require successful payment before access is granted.
- Payment status shall be recorded as Pending, Completed, or Failed.
- Refunds shall be processed according to defined refund policies.

**Reference:-** FREQ-18, FREQ-19
**Revision History**:- Initial Definition
**Purpose:-** Ensures transparent and traceable financial transactions.

**BR-06: AI Usage Limitation Rule**

**Description:-**

- Free users shall have limited access to AI practice features.
- Premium users shall have extended AI usage privileges.
- The system shall track AI usage per user.

**Reference:-** FREQ-7, FREQ-8
**Revision History**:- Initial Definition
**Purpose:-** Controls system resource usage and supports sustainability.

**BR-08: Assessment and Evaluation Rule**

**Description:-**

- Assessment results shall be automatically generated and stored by the system.
- Manual modification of scores shall only be allowed by administrators.
- Assessment history shall be retained permanently.

**Reference:-** FREQ-5, FREQ-10, FREQ-28
**Revision History**:- Initial Definition
**Purpose:-** Ensures fairness and academic integrity.

**BR-09: Tutor Session Scheduling Rule**

**Description:-**

- Tutors shall define their availability in advance.

- ❖ Students shall only book sessions within available time slots.
- ❖ Double booking of tutors shall not be allowed.

**Reference:-** FREQ-14, FREQ-16, FREQ-17
**Revision History**:- Initial Definition
**Purpose:-** Avoids scheduling conflicts and improves service reliability.

**BR-10: Attendance Rule**

**Description:-**

- ❖ Attendance shall be recorded only for live sessions that were actually conducted.
- ❖ A student must attend a minimum percentage of sessions to qualify for certification.
- ❖ Attendance records shall not be editable by students.

**Reference:-** FREQ-16, FREQ-17, FREQ-10
**Revision History**:- Initial Definition
**Purpose:-** Supports accurate certification and reporting.

**BR-11: Certificate Issuance Rule**

**Description:-**

- ❖ Certificates shall be issued only after successful course or batch completion.
- ❖ Each certificate shall have a unique verification ID.
- ❖ Certificates shall be generated digitally.

**Reference:-** FREQ-26
**Revision History**:- Initial Definition
**Purpose:-** Prevents certificate fraud and ensures credibility.

**BR-12: Content Management Rule**

**Description:-**

- ❖ Only administrators shall create, update, or delete official learning content.
- ❖ Tutors may upload supplementary materials subject to admin approval.
- ❖ Deleted content shall be archived, not permanently removed.

**Reference:-** FREQ-9, FREQ-22
**Revision History**:- Initial Definition
**Purpose:-** Maintains content quality and version control.

**BR-13:Communication and Community Rule**

**Description:-**

- ❖ Users shall communicate only within authorized groups or batches.
- ❖ Inappropriate content shall be removable by administrators.
- ❖ All communications shall be logged for moderation purposes.

**Reference:-** FREQ-25, FREQ-24
**Revision History**:- Initial Definition
**Purpose:-** Ensures respectful and safe learning environment.

**BR-14: Notification Rule**

**Description:-**

- ❖ System notifications shall be automatically generated for key events such as enrollment, payments, and schedule changes.
- ❖ Notifications shall not be deleted permanently by users.
- ❖ Read/unread status shall be tracked.

**Reference:-**FREQ-24, FREQ-23
**Revision History**:- Initial Definition
**Purpose:-** Ensures users are informed of important action.

**BR-15: Data Privacy Rule**

**Description:-**

- ❖ User personal data shall not be shared with third parties without consent.
- ❖ Voice recordings shall be used only for learning and evaluation purposes.
- ❖ Users shall have the right to request data deletion.

**Reference:-** FREQ-2, FREQ-21
**Revision History**:- Initial Definition
**Purpose:-** Ensures ethical and legal compliance.

**BR-17: Platform Usage Rule**

**Description:-**

- The system shall be accessible through both web and mobile platforms.
- Core functionality shall be consistent across platforms.
- Feature limitations shall be clearly communicated to users.

**Reference:-** FREQ-30
**Revision History**:- Initial Definition

**BR-18: User Account Status Rule**

**Description:-**

- Every user account shall have a status: Active, Suspended, or Deactivated.
- Suspended users shall not be able to access learning or payment features.
- Only administrators shall change account status.

**Reference:-** FREQ-21
**Revision History**:- Initial Definition
**Purpose:-** Ensures control over misuse and policy violations

**BR-19: Login Attempt Limitation Rule**

**Description:-**

- The system shall limit the number of failed login attempts.
- Accounts shall be temporarily locked after multiple failed attempts.
- Users shall be notified when their account is locked.

**Reference:-** FREQ-2
**Revision History**:- Initial Definition
**Purpose:-** Protects the system against brute-force attacks

**BR-20: AI Feedback Fair-Use Rule**

**Description:-**

- AI feedback shall be generated only for valid learning activities.
- Repeated or automated requests shall be restricted.
- AI-generated feedback shall not be reused across users.

**Reference:-** FREQ-7, FREQ-28
**Revision History**:- Initial Definition
**Purpose:-** Prevents abuse of AI resources and ensures fairness

**BR-22: Session Cancellation Rule**

**Description:-**

- Students may cancel booked sessions within a defined time window.

- ❖ Late cancellations may result in partial or no refunds.
- ❖ Tutors shall be notified immediately upon cancellation.

**Reference:-** FREQ-16, FREQ-20
**Revision History**:- Initial Definition
**Purpose:-** Ensures fairness and minimizes schedule disruption.

**BR-24: Rating Abuse Prevention Rule**

**Description:-**
- ❖ Students shall only rate tutors after attending sessions.
- ❖ Multiple ratings for the same session shall not be allowed.
- ❖ Suspicious rating behavior shall be flagged for review.

**Reference:-** FREQ-10, FREQ-25
**Revision History**:- Initial Definition
**Purpose:-** Maintains credibility of the rating system.

**BR-25: Content Version Control Rule**

**Description:-**

- ❖ All learning content updates shall create a new version.
- ❖ Previous versions shall be archived and retrievable.
- ❖ Students shall continue using the version assigned to their batch.

**Reference:-** FREQ-22
**Revision History**:- Initial Definition
**Purpose:-** Ensures content consistency during learning cycles.

**BR-26: Batch Capacity Rule**

**Description:-**

- ❖ Each batch shall have a maximum enrollment limit.
- ❖ The system shall automatically close enrollment once capacity is reached.
- ❖ Waitlisted students may be notified if slots become available.

**Reference:-** FREQ-4, FREQ-9
**Revision History**:- Initial Definition
**Purpose**:- Prevents overcrowding and ensures quality instruction.

**BR-27: Data Backup Rule**

**Description:-**

- System data shall be backed up regularly.
- Backup data shall be stored securely.
- Only administrators shall restore backups.

**Reference:-** FREQ-23
**Revision History**:- Initial Definition
**Purpose**:- Prevents data loss and supports disaster recovery.

**BR-28: System Maintenance Rule**

**Description:-**

- Scheduled maintenance periods shall be announced in advance.
- Critical system updates may occur during low-usage hours.
- Users shall receive notifications for maintenance windows.

**Reference:-** FREQ-23, FREQ-24
**Revision History**:- Initial Definition
**Purpose:-**Ensures transparency and system reliability.

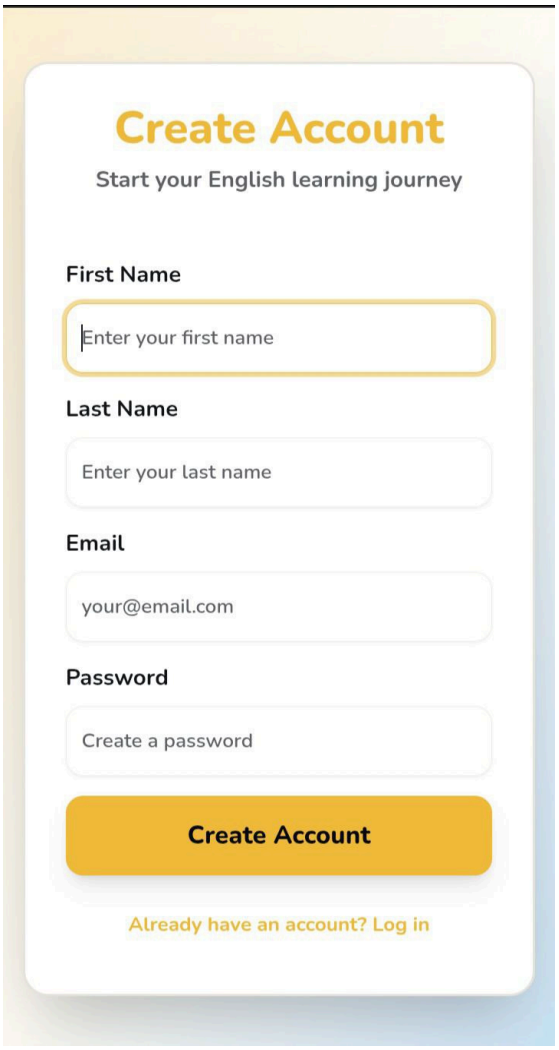
**BR-29 Advertisement & Promotion Rule**

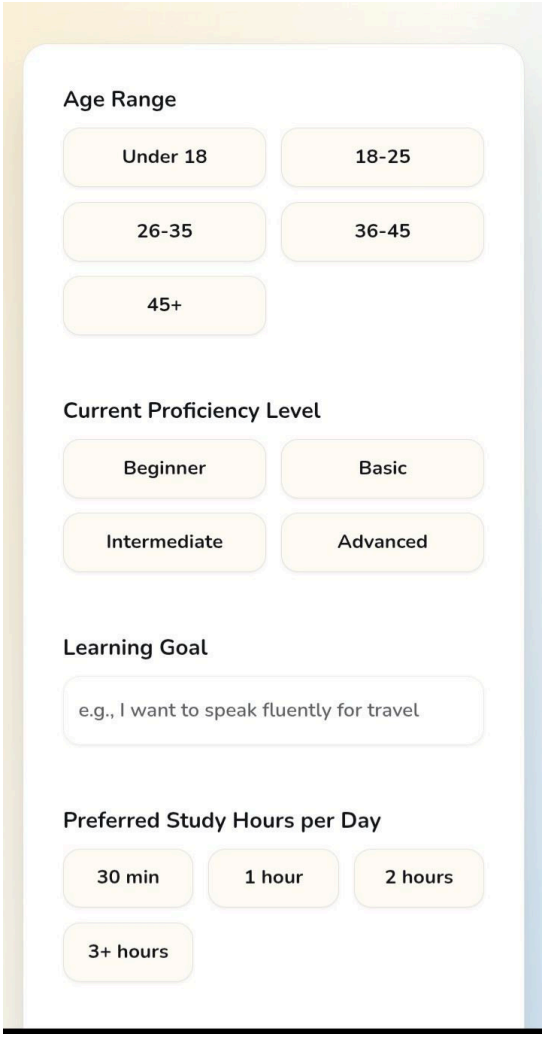
**Description:-**

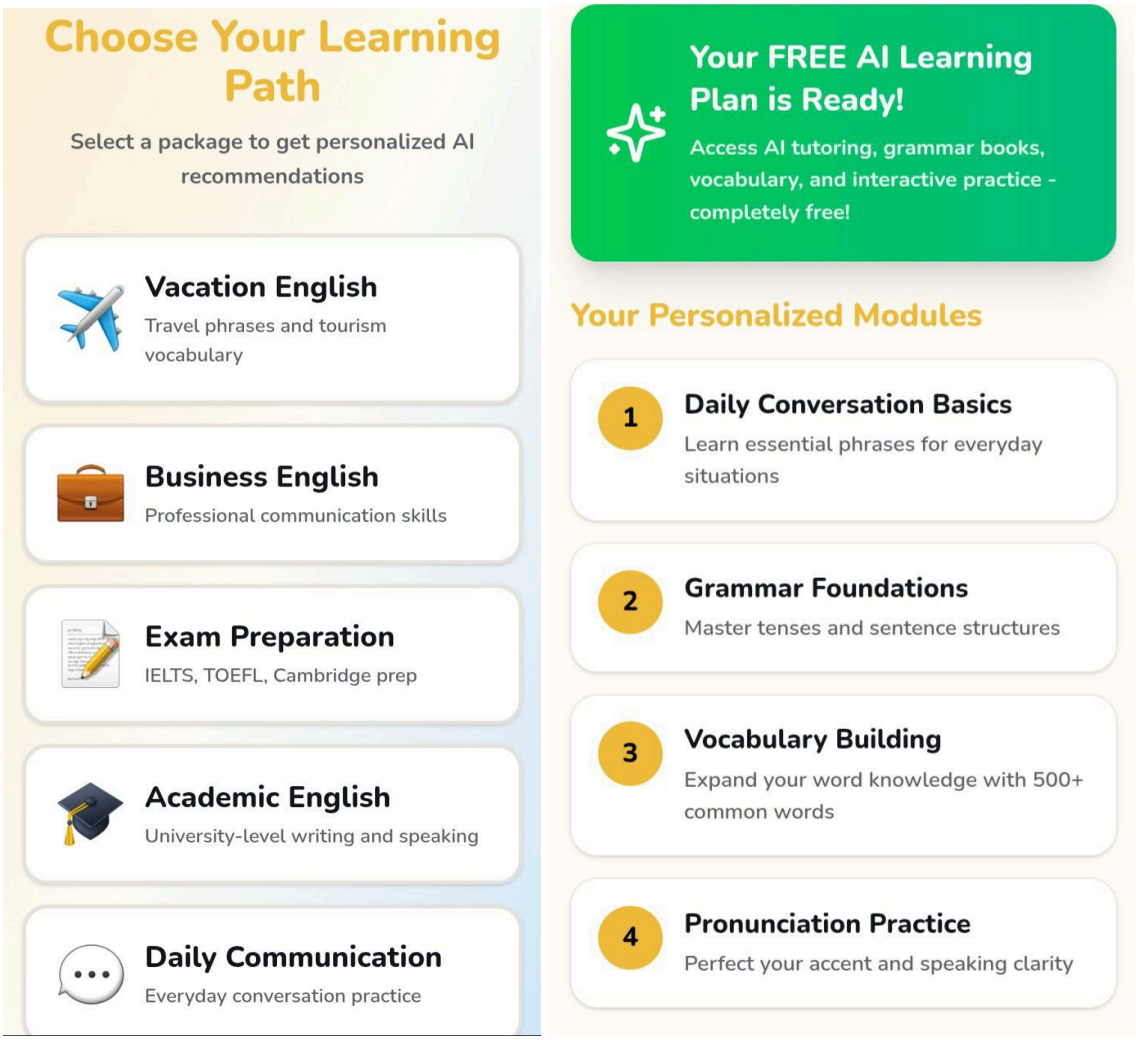
- Promotional content shall not interfere with learning activities.
- Ads shall not appear during live classes or assessments.
- Promotional messages shall be clearly identified.

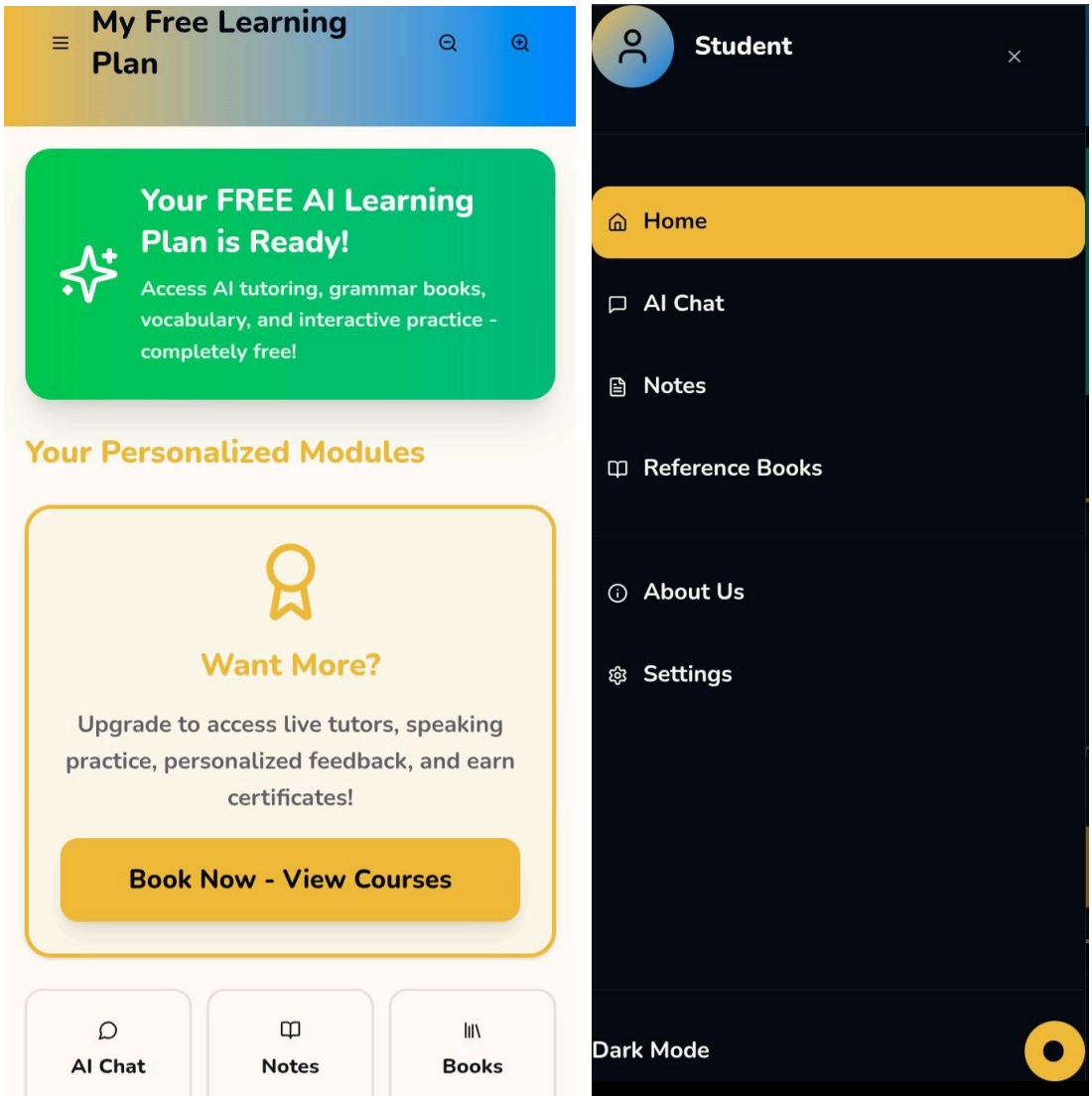
**Reference:-** FREQ-22, FREQ-24
**Revision History**:- Initial Definition
**Purpose**:- Maintains a distraction-free learning environment.

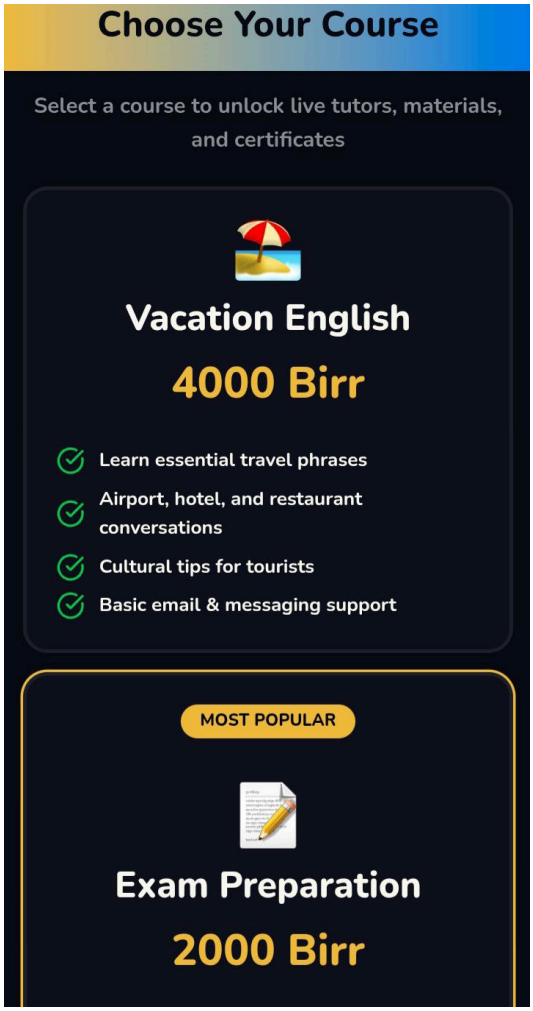
## 2.3.4 User Interface Prototype

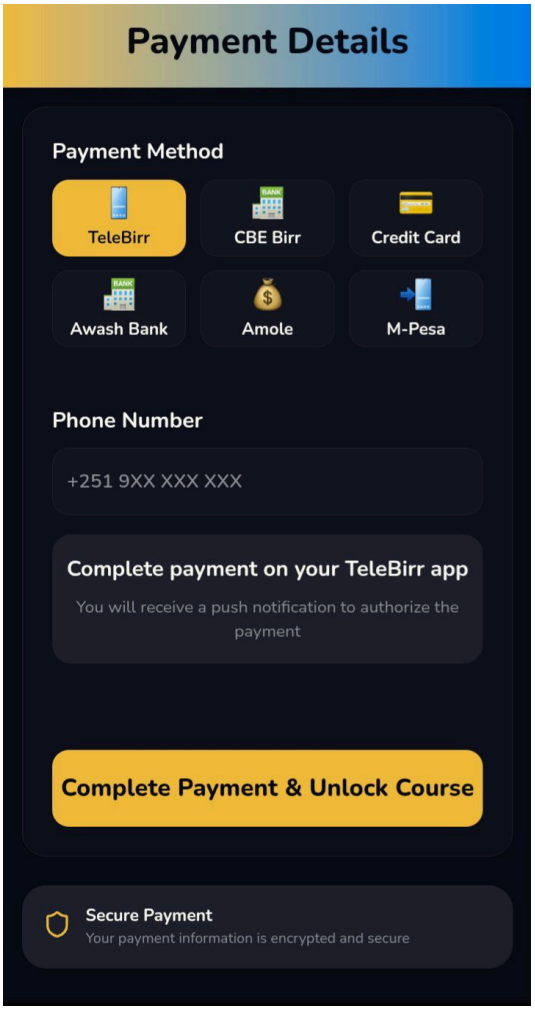

Create Account
Start your English learning journey
First Name
Enter your first name
Last Name
Enter your last name
Email
your@email.com
Password
Create a password
Create Account
Already have an account? Log in

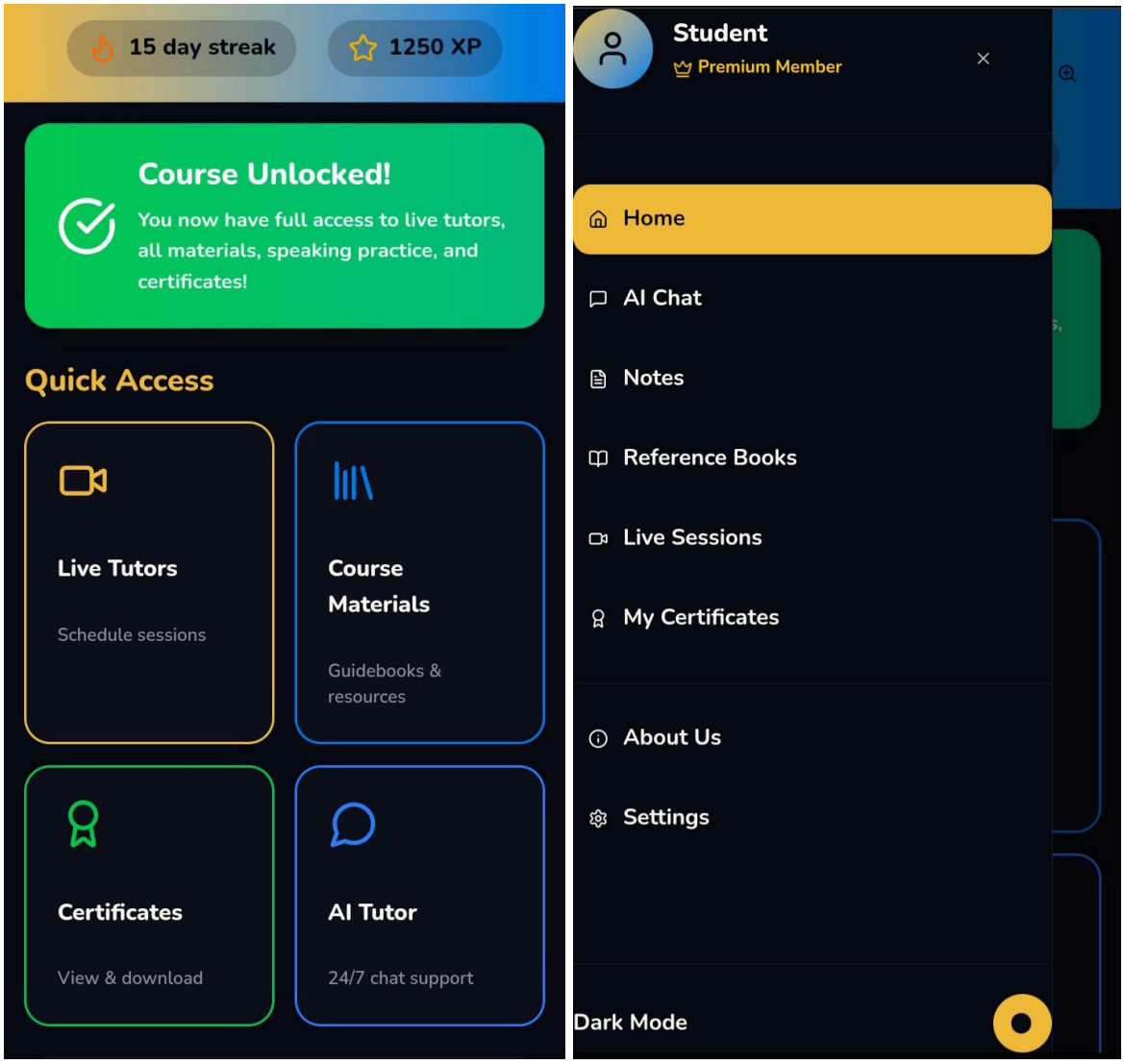

Age Range
Under 18
18-25
26-35
36-45
45+
Current Proficiency Level
Beginner
Basic
Intermediate
Advanced
Learning Goal
e.g., I want to speak fluently for travel
Preferred Study Hours per Day
30 min
1 hour
2 hours
3+ hours


Choose Your Learning Path
Select a package to get personalized AI recommendations
Vacation English
Travel phrases and tourism vocabulary
Business English
Professional communication skills
Exam Preparation
IELTS, TOEFL, Cambridge prep
Academic English
University-level writing and speaking
Daily Communication
Everyday conversation practice
Your FREE AI Learning Plan is Ready!
Access AI tutoring, grammar books, vocabulary, and interactive practice - completely free!
Your Personalized Modules
1
Daily Conversation Basics
Learn essential phrases for everyday situations
2
Grammar Foundations
Master tenses and sentence structures
3
Vocabulary Building
Expand your word knowledge with 500+ common words
4
Pronunciation Practice
Perfect your accent and speaking clarity


My Free Learning Plan
Your FREE AI Learning Plan is Ready!
Access AI tutoring, grammar books, vocabulary, and interactive practice - completely free!
Your Personalized Modules
Want More?
Upgrade to access live tutors, speaking practice, personalized feedback, and earn certificates!
Book Now - View Courses
AI Chat
Notes
Books
Student
Home
AI Chat
Notes
Reference Books
About Us
Settings
Dark Mode


Choose Your Course
Select a course to unlock live tutors, materials, and certificates
Vacation English
4000 Birr
Learn essential travel phrases
Airport, hotel, and restaurant conversations
Cultural tips for tourists
Basic email & messaging support
MOST POPULAR
Exam Preparation
2000 Birr


Payment Details
Payment Method
TeleBirr
CBE Birr
Credit Card
Awash Bank
Amole
M-Pesa
Phone Number
+251 9XX XXX XXX
Complete payment on your TeleBirr app
You will receive a push notification to authorize the payment
Complete Payment & Unlock Course
Secure Payment
Your payment information is encrypted and secure




## 2.3.5 State Chart Diagram

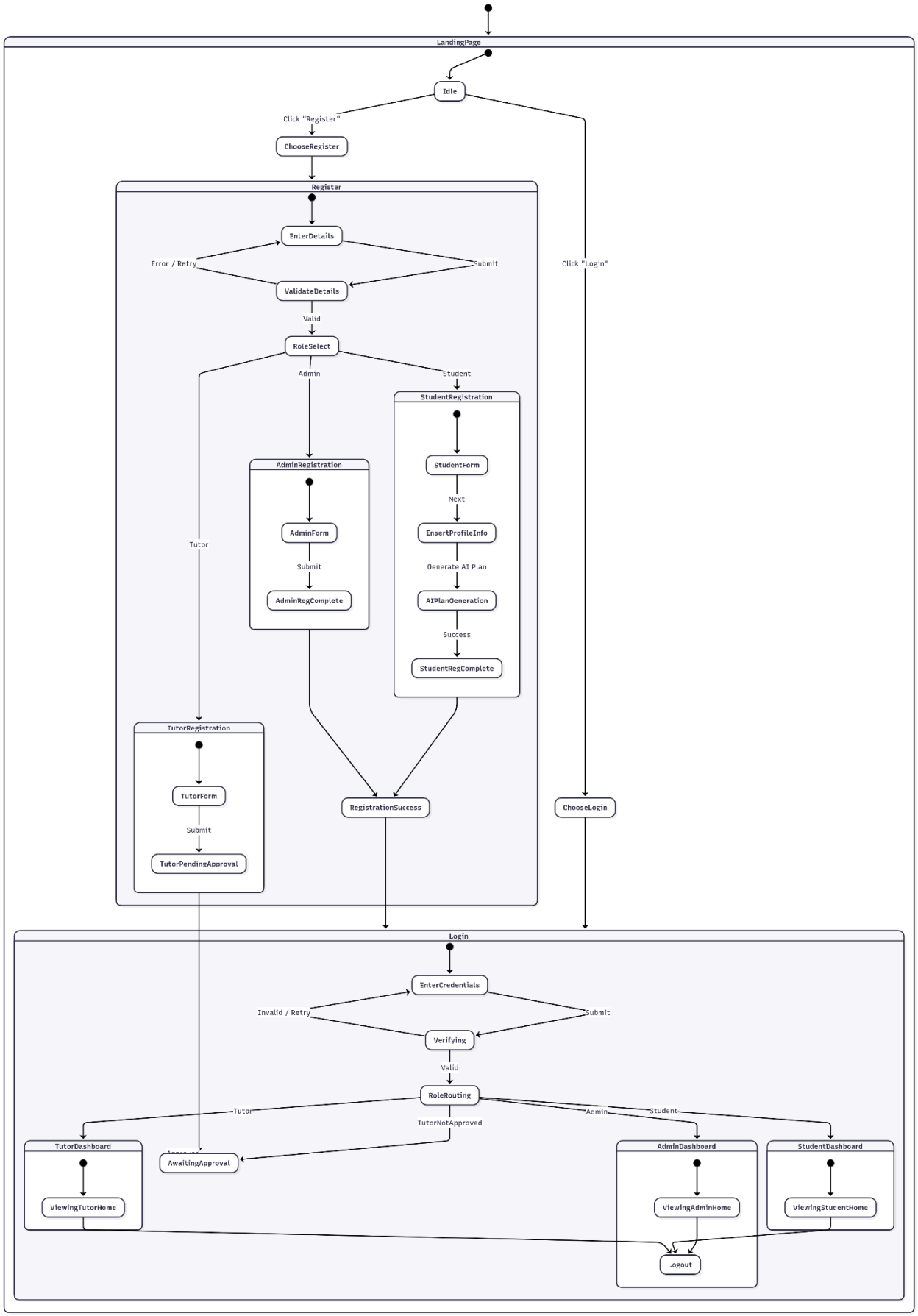
### 1. User Registration & Onboarding

This state chart diagram models a complete authentication and role-based access control flow for the system. It illustrates how users move from the landing page through registration or login and then into role-specific dashboards (Student, Tutor, Admin).

Transitions show both new and existing user flows, including error handling, retries, routing to dashboards, and followed by logout.

Key behavioral characteristics include deterministic role-based routing, error recovery mechanisms, and consistent paths that ensure both new and returning users reach the

same dashboard states. The diagram clearly separates authentication, role determination, and session management, providing a structured view of the system's stateful behavior.


LandingPage
Idle
Click "Register"
ChooseRegister
Register
EnterDetails
Error / Retry
Submit
ValidateDetails
Valid
RoleSelect
Admin
Student
Tutor
AdminRegistration
AdminForm
Submit
AdminRegComplete
StudentRegistration
StudentForm
Next
EnsertProfileInfo
Generate AI Plan
AIPlanGeneration
Success
StudentRegComplete
TutorRegistration
TutorForm
Submit
TutorPendingApproval
RegistrationSuccess
Click "Login"
ChooseLogin
Login
EnterCredentials
Invalid / Retry
Submit
Verifying
Valid
RoleRouting
Tutor
TutorNotApproved
Admin
Student
TutorDashboard
ViewingTutorHome
AwaitingApproval
AdminDashboard
ViewingAdminHome
StudentDashboard
ViewingStudentHome
Logout

**2. Self-Paced Learning Journey**

**This state chart describes a student's end-to-end learning flow on a system.**

It maps how a user moves from logging in, navigating the dashboard, selecting modules, progressing through lessons, receiving AI assistance, and completing learning activities.

**Learning progression follows a structured path:**

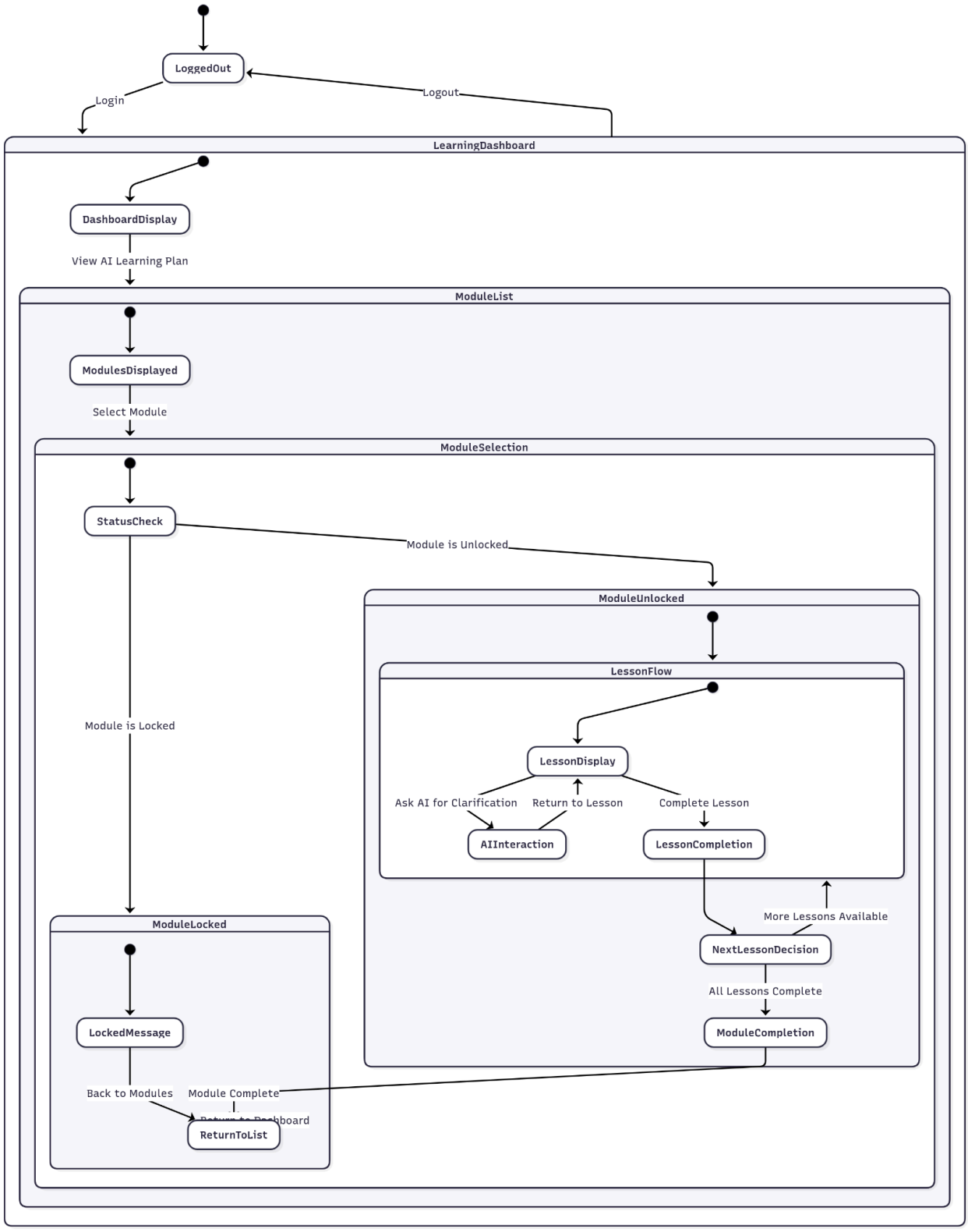
Dashboard ---> Module list ---> Module selection ---> Lessons ---> Module completion.

Modules unlock sequentially as prerequisites are completed.

**AI assistance is integrated within lessons** as an optional, non-disruptive loop that always returns the learner to the content.

**Control flow is deterministic**, with decision states governing locked vs. unlocked modules, next lesson selection, or module completion.

**Overall**, the model provides a scalable, hierarchical structure for guided, AI-supported learning, blending linear lesson progression with flexible navigation and real-time assistance.




## 3. Batch Enrollment Process

This state chart describes the full student batch-enrollment workflow, including batch availability checks, payment processing, and post-payment access.

It shows how a student moves from browsing batches, selecting one, confirming availability, completing payment through an external gateway, and finally gaining access to community and batch resources.

**Core flow:**

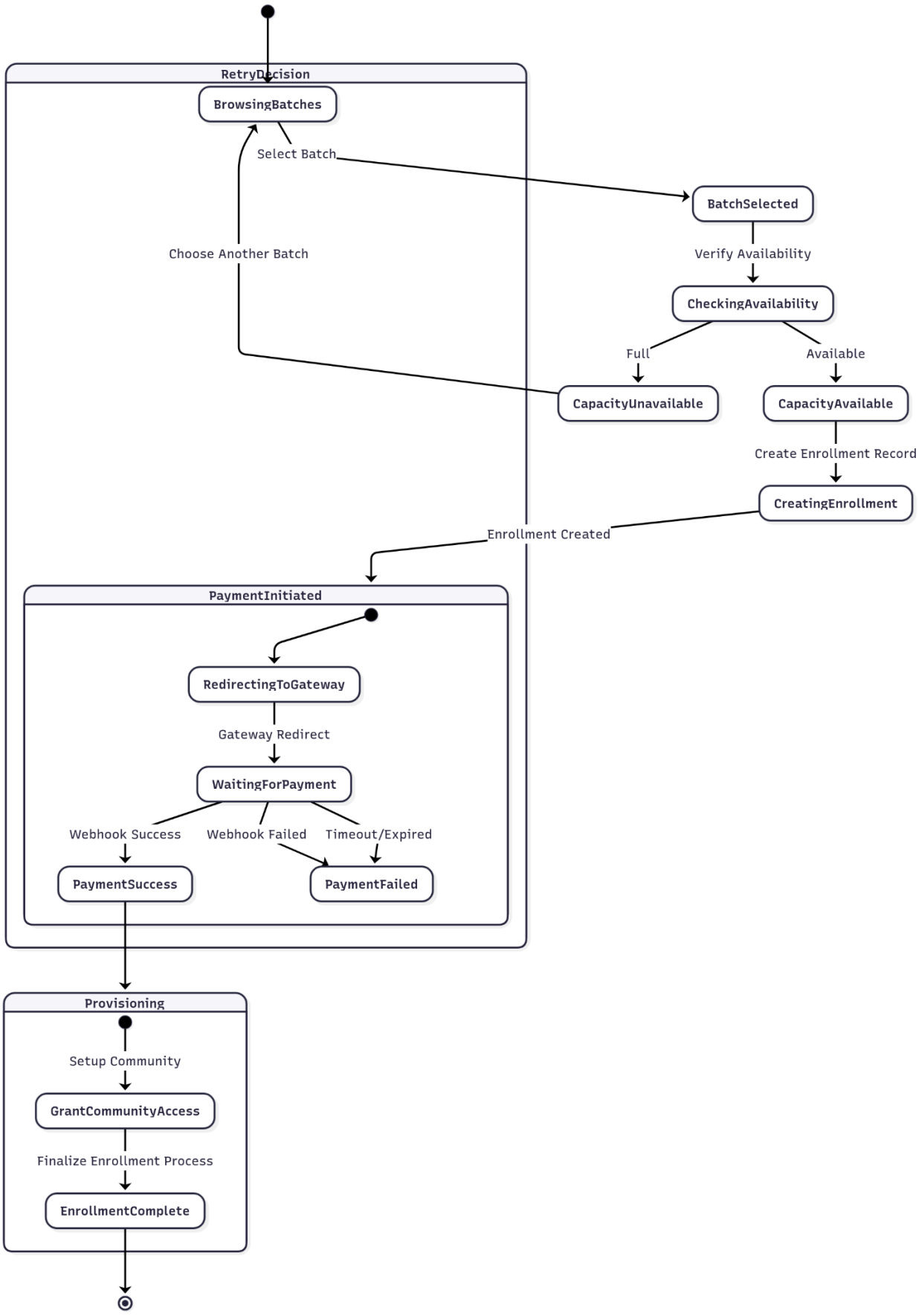
Browse batches ---> Select batch ---> Verify availability ---> Create enrollment ---> Initiate payment ---> Process payment ---> Success ---> Enrollment complete.

**The system integrates external payment gateways**, using redirects and asynchronous webhooks. It must maintain a consistent enrollment state during the payment wait period.

**Key behaviors modeled include:**

- Gated access based on batch capacity and payment confirmation
- Idempotent, safe payment retries
- Automatic community access and resource provisioning after successful payment
- Transactional updates ensuring enrollment and payment stay in sync

**Overall**, the model ensures a guided, reliable enrollment experience, robust payment handling, and strict access control.

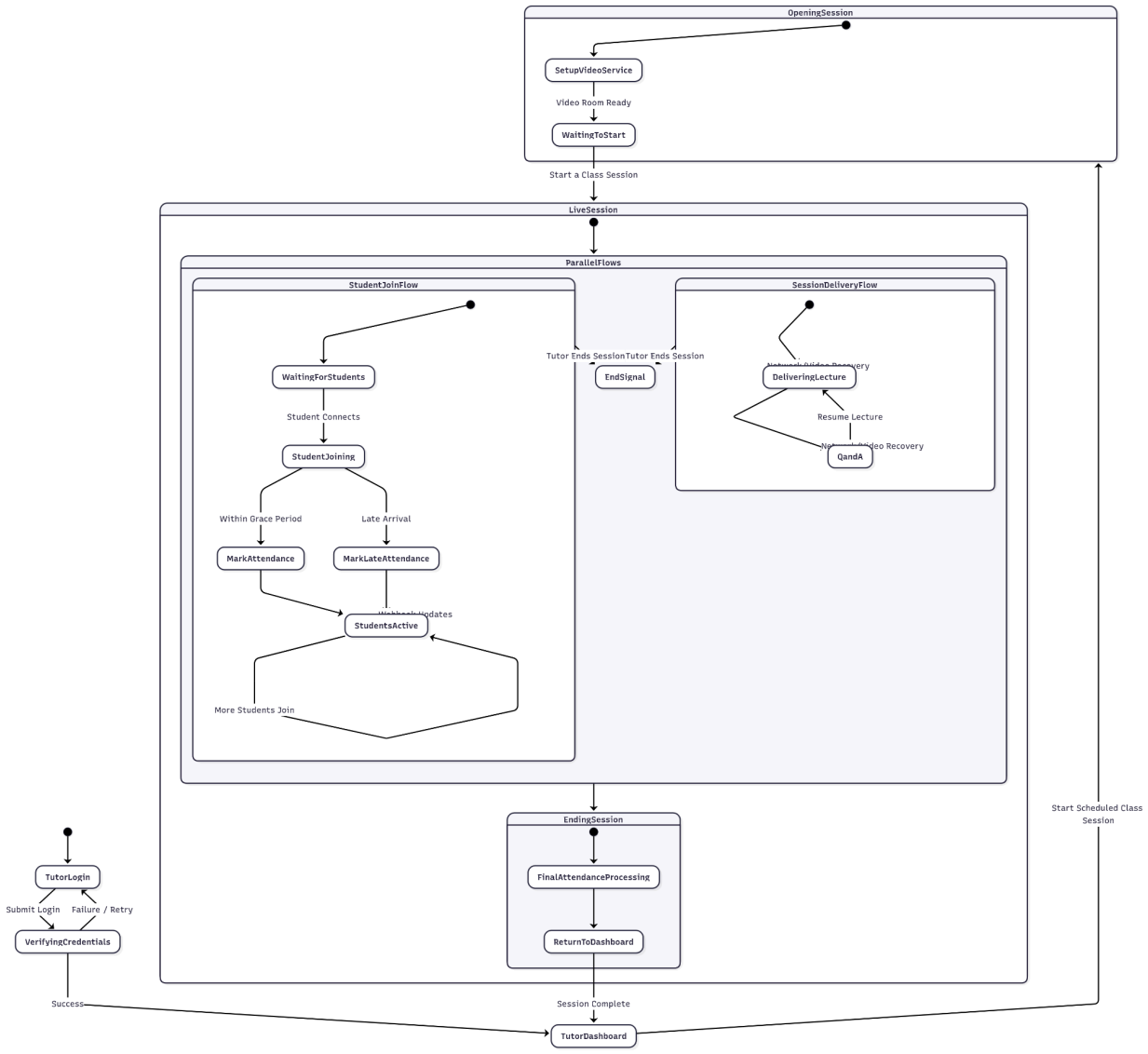

RetryDecision
BrowsingBatches
Select Batch
BatchSelected
Choose Another Batch
Verify Availability
CheckingAvailability
Full
Available
CapacityUnavailable
CapacityAvailable
Create Enrollment Record
CreatingEnrollment
Enrollment Created
PaymentInitiated
RedirectingToGateway
Gateway Redirect
WaitingForPayment
Webhook Success
Webhook Failed
Timeout/Expired
PaymentSuccess
PaymentFailed
Provisioning
Setup Community
GrantCommunityAccess
Finalize Enrollment Process
EnrollmentComplete

**4. Live Class and Attendance Flow**

This state chart models the entire lifecycle of a live tutoring class from the system's perspective.

**Key phases include:**

- Tutor logs in and accesses a dashboard to view and start scheduled classes.
- Class initialization and setting up video conferencing.
- During the live session, two parallel activities occur: students join and attendance tracking while the tutor delivers content (lectures).
- Attendance is automatically marked based on join times, with a configurable grace period for late arrivals.
- Class ends with final attendance processing.




## 5. Community Interaction Flow

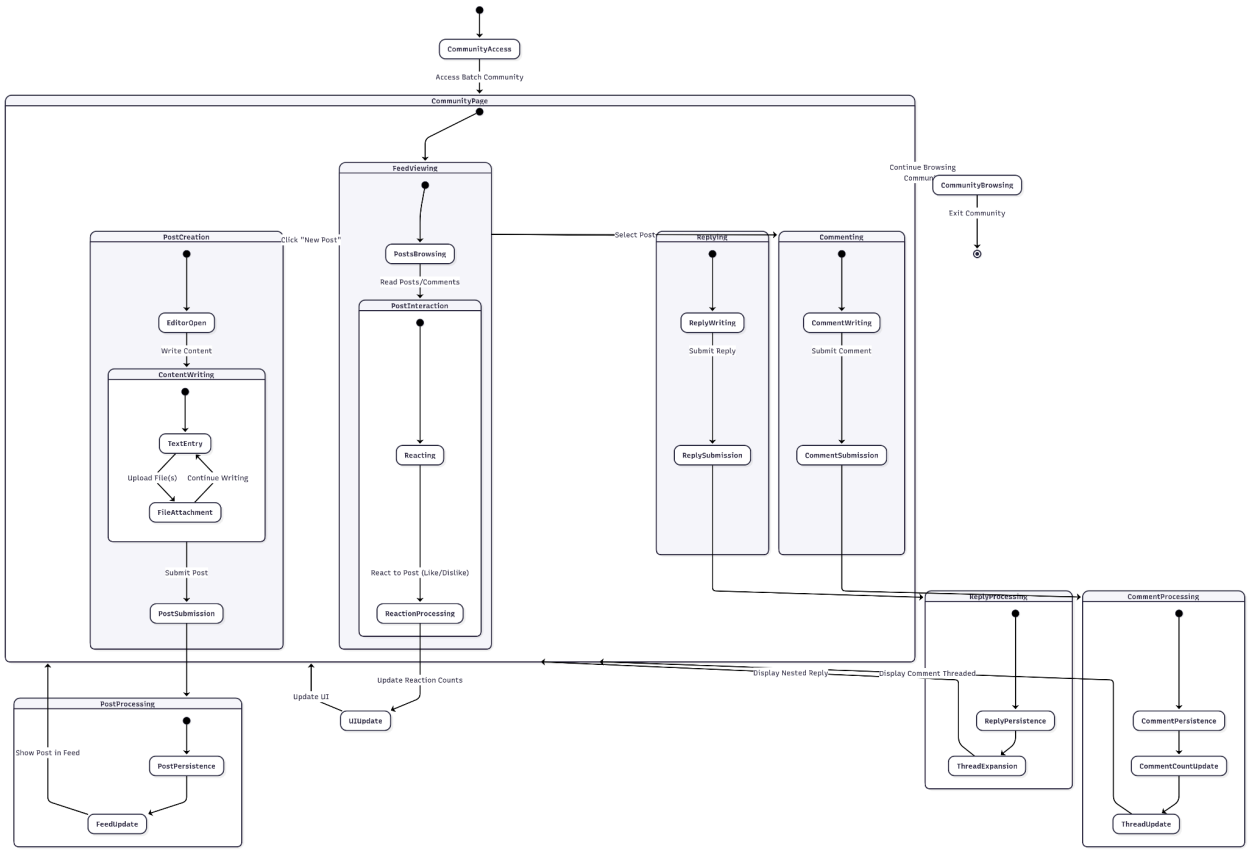
This state chart models a batch community interaction system that supports social learning features like post creation, commenting, replying, and reactions within a learning community.

**Key features:**

- ❖ Multi-modal interaction with hierarchical content: posts ---> comments ---> replies
- ❖ Rich media support with optional file attachments in posts
- ❖ Asynchronous backend processing separated from UI states for smooth experience

**User experience emphasizes:**

- ❖ Easy content discovery and seamless transitions between interaction modes
- ❖ Support for varying levels of engagement from passive reading to active content creation




## 6. AI Interaction Flow

This state chart models an AI-powered educational assistant that supports context-aware, intelligent conversations with caching and follow-up capabilities.

**Key features:**

- ❖ Intelligent caching to speed up responses and reduce redundant AI calls
- ❖ Dual context setup for personalized, relevant answers based on lesson content or user profile
- ❖ Conversation threading to maintain continuity through follow-up questions
- ❖ Hybrid response system balancing fast cached replies with adaptive fresh generation

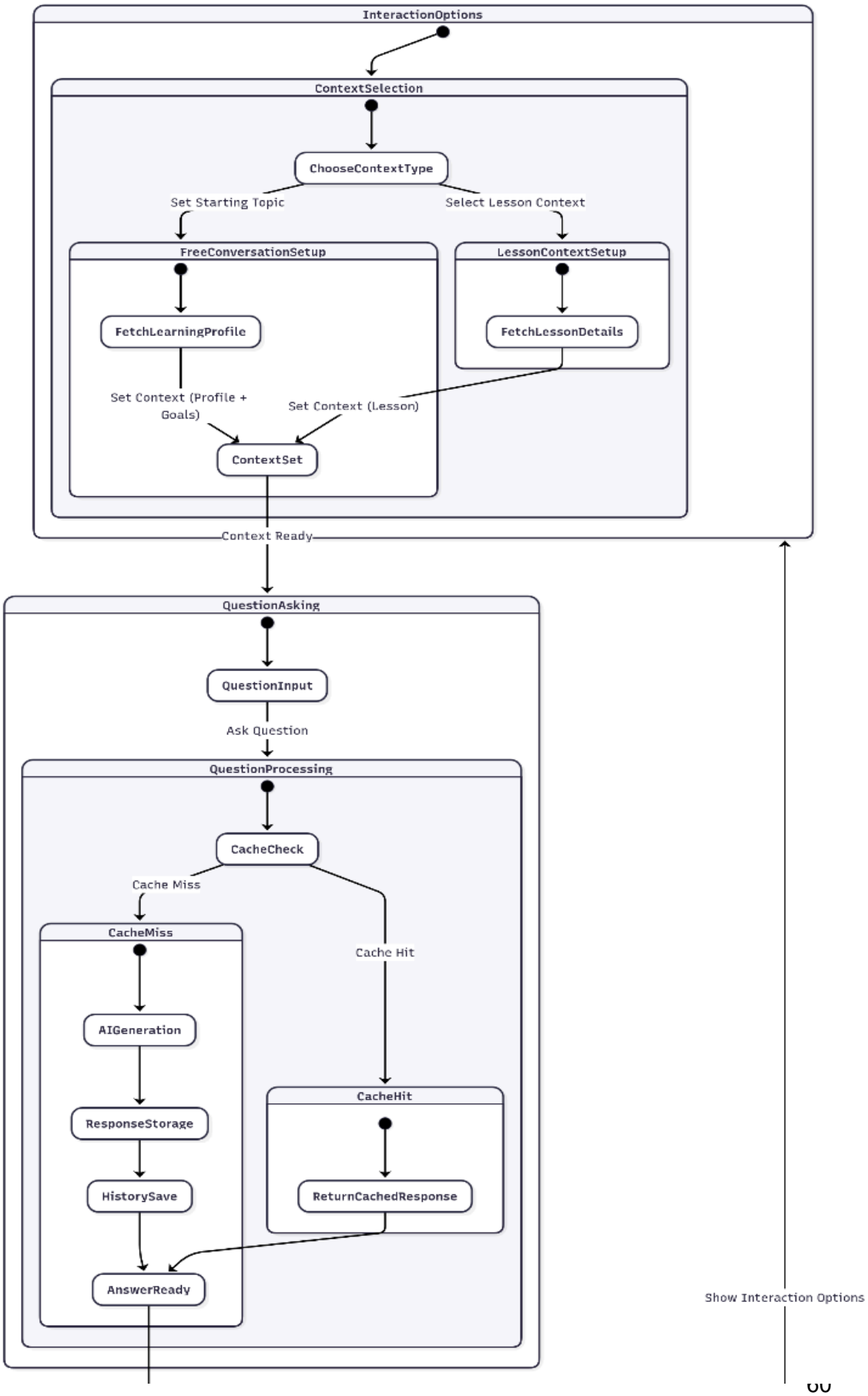
**System behaviors:**

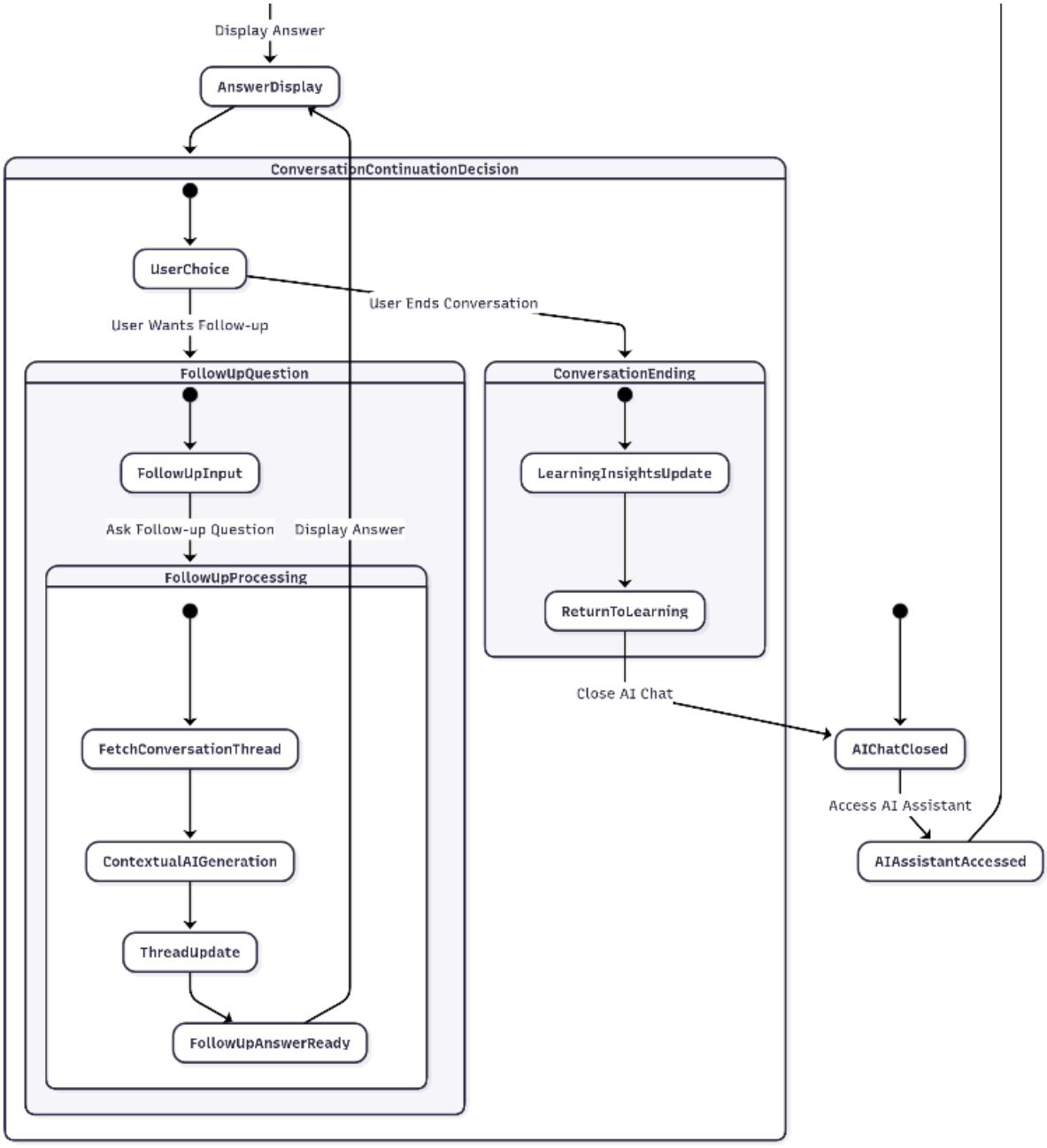
- ❖ Cache keyed on query plus context to ensure relevance

- ❖ Seamless context switching and persistence during conversations

**User experience:**

- ❖ Personalized, natural conversations with instant cached answers
- ❖ Smooth transitions between questions and follow ups
- ❖ Transparent context awareness


InteractionOptions
ContextSelection
ChooseContextType
Set Starting Topic
Select Lesson Context
FreeConversationSetup
LessonContextSetup
FetchLearningProfile
FetchLessonDetails
Set Context (Profile + Goals)
Set Context (Lesson)
ContextSet
Context Ready
QuestionAsking
QuestionInput
Ask Question
QuestionProcessing
CacheCheck
Cache Miss
Cache Hit
CacheMiss
AIGeneration
ResponseStorage
HistorySave
AnswerReady
CacheHit
ReturnCachedResponse
Show Interaction Options

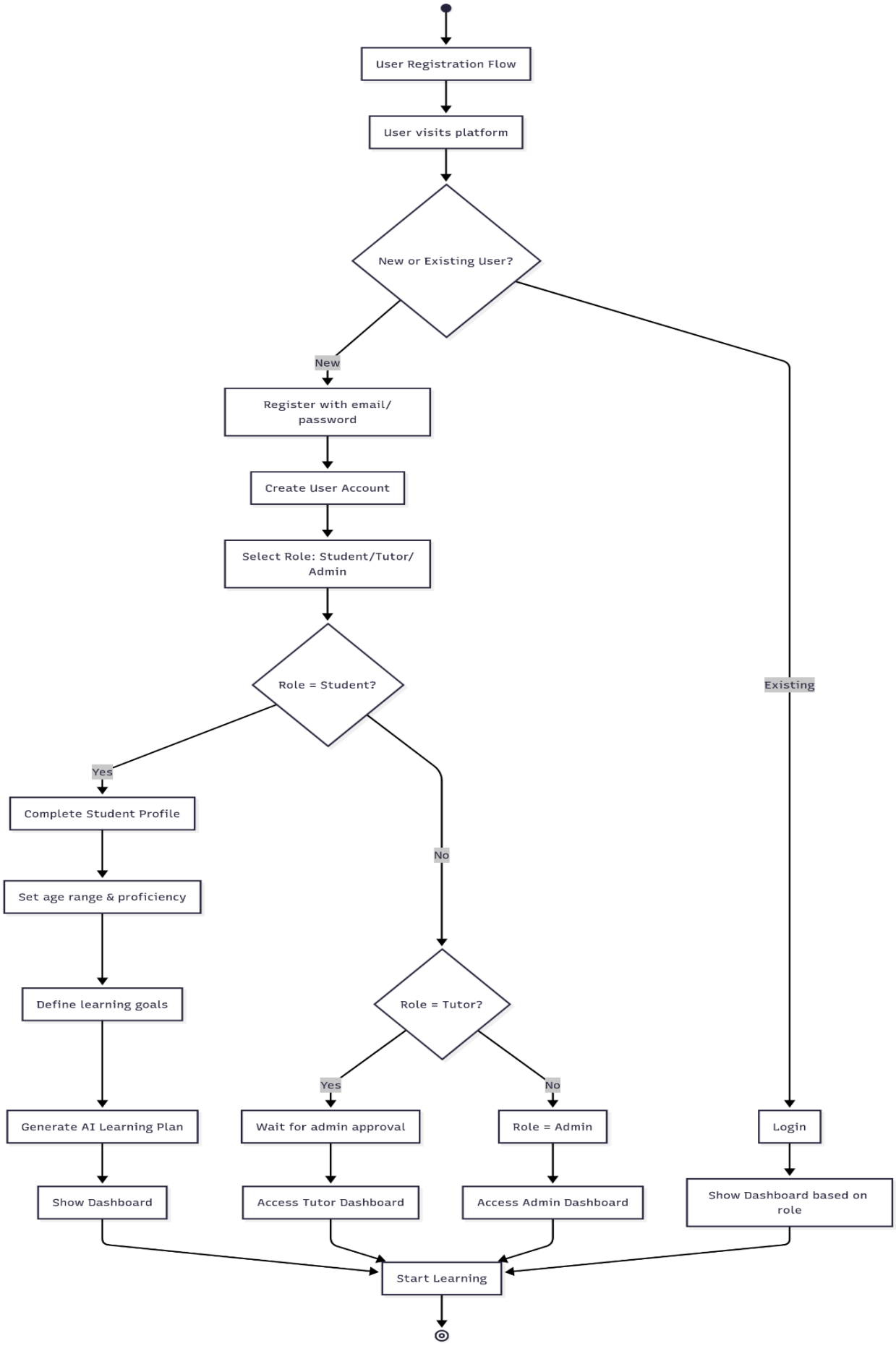



## 2.3.6 Activity Diagram

**User Registration & Onboarding**

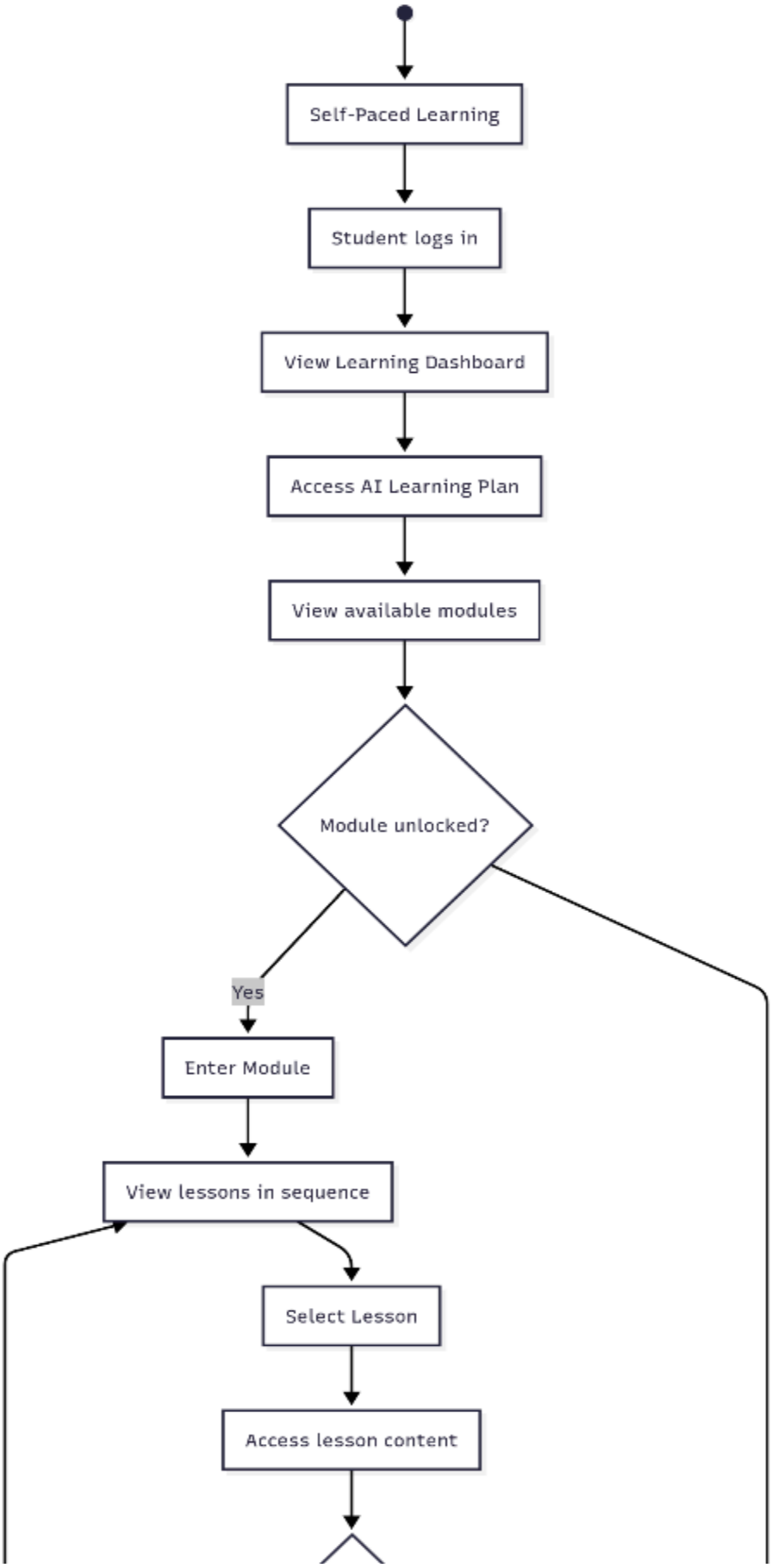
This diagram illustrates the process when a user first visits the platform. It begins with the user arriving and then deciding whether they are a new or existing user. For new users, they register with their email and password, then create a user account. Depending on the role, different paths are taken:

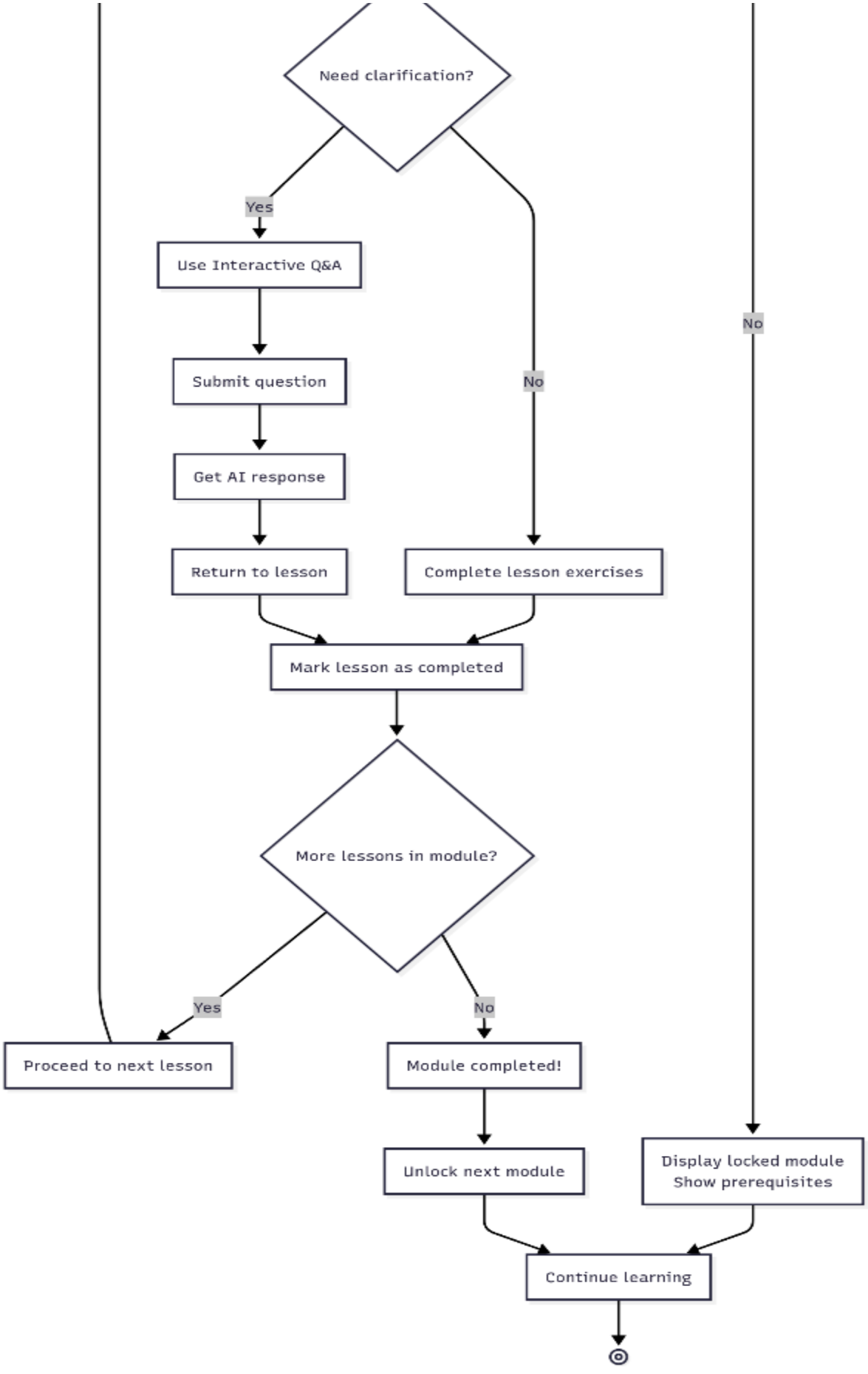
- Students complete their profile, set age range and proficiency, define learning goals, and then an AI learning plan is generated. Then they are shown the dashboard.
- Tutors wait for admin approval before accessing the tutor dashboard.
- Admins go directly to the admin dashboard.
- Existing users simply log in and are shown the dashboard according to their role. The process ends with the user starting to use the platform.


User Registration Flow
User visits platform
New or Existing User?
New
Existing
Register with email/ password
Create User Account
Select Role: Student/Tutor/ Admin
Role = Student?
Yes
No
Complete Student Profile
Set age range & proficiency
Define learning goals
Generate AI Learning Plan
Show Dashboard
Role = Tutor?
Yes
No
Wait for admin approval
Access Tutor Dashboard
Role = Admin
Access Admin Dashboard
Login
Show Dashboard based on role
Start Learning

**Self-Paced Learning Journey**

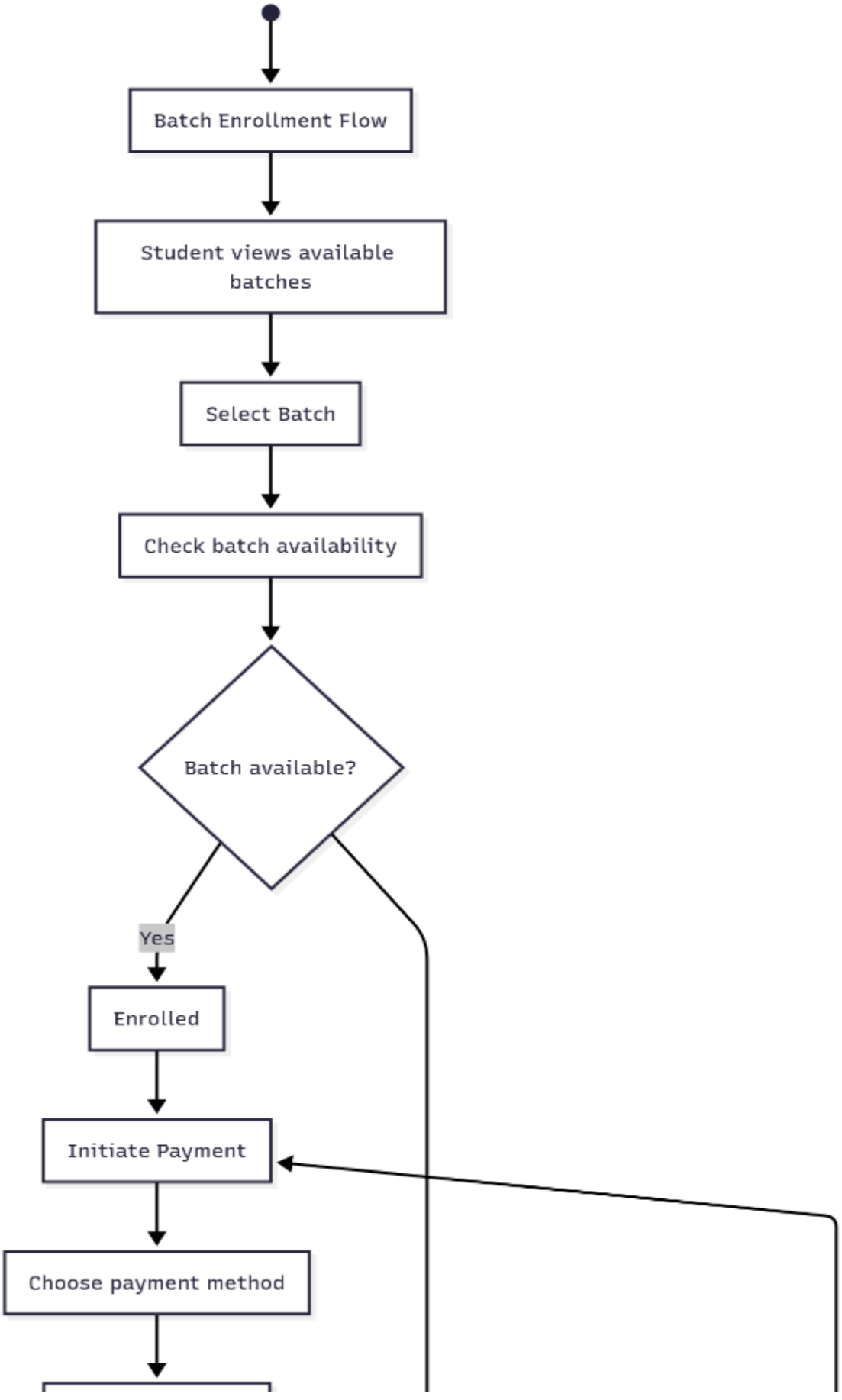
This diagram shows the steps a student takes in self-paced learning. The student logs in and views the learning dashboard, then accesses their AI learning plan. They view available modules and check if a module is unlocked. If not, they see the locked module and prerequisites. If unlocked, they enter the module and view lessons in sequence. They select a lesson and access the content. If they need clarification, they use interactive questions and answers: submit a question, get an AI response and return to the lesson. If no clarification is needed, they complete lesson exercises and mark the lesson as completed. If there are more lessons in the module, they proceed to the next lesson; otherwise, the module is completed, the next module is unlocked, and they continue learning.

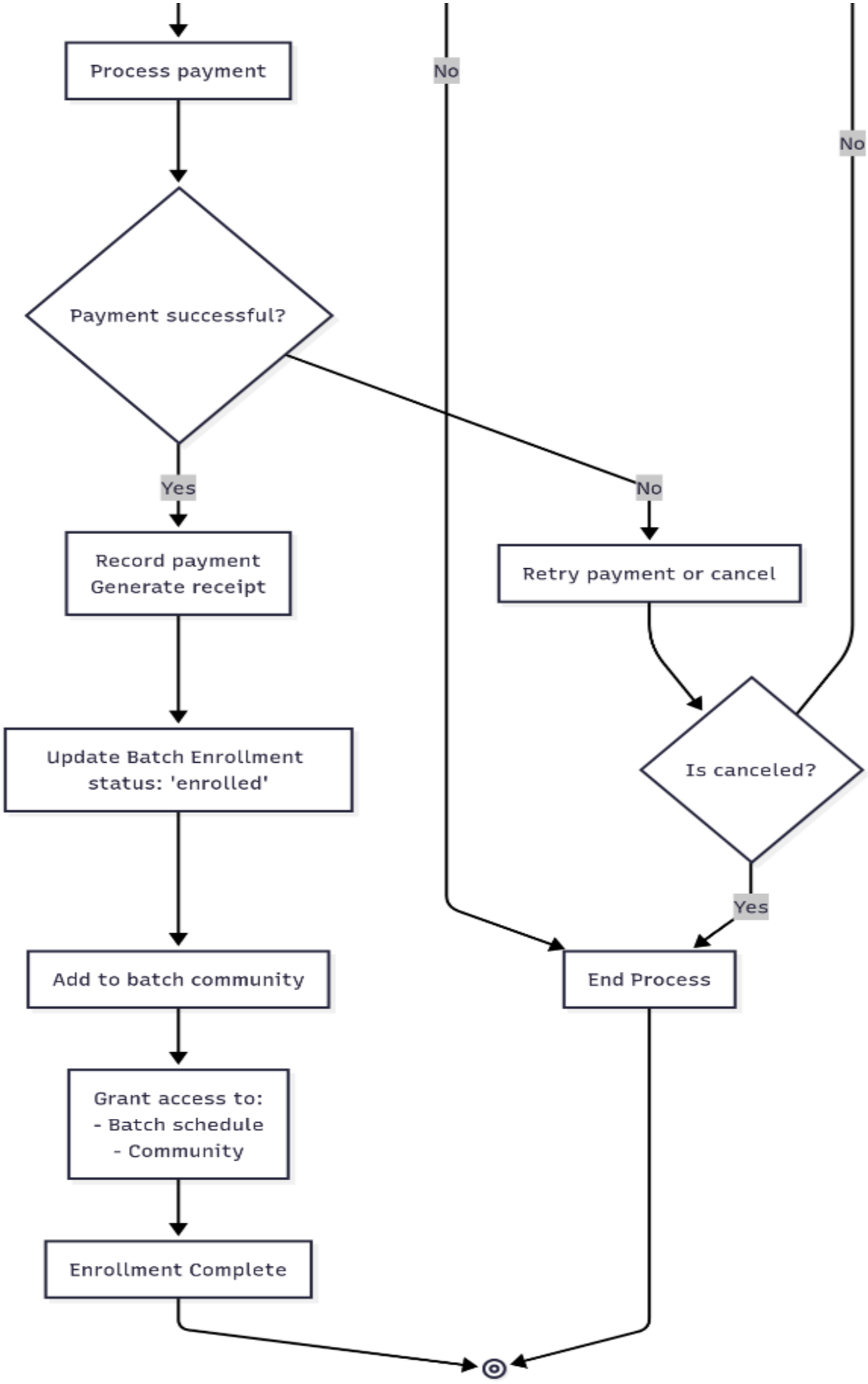

Self-Paced Learning
Student logs in
View Learning Dashboard
Access AI Learning Plan
View available modules
Module unlocked?
Yes
Enter Module
View lessons in sequence
Select Lesson
Access lesson content


Need clarification?
Yes
No
No
Use Interactive Q&A
Submit question
Get AI response
Return to lesson
Complete lesson exercises
Mark lesson as completed
More lessons in module?
Yes
No
Proceed to next lesson
Module completed!
Unlock next module
Display locked module
Show prerequisites
Continue learning

**Batch Enrollment Process**

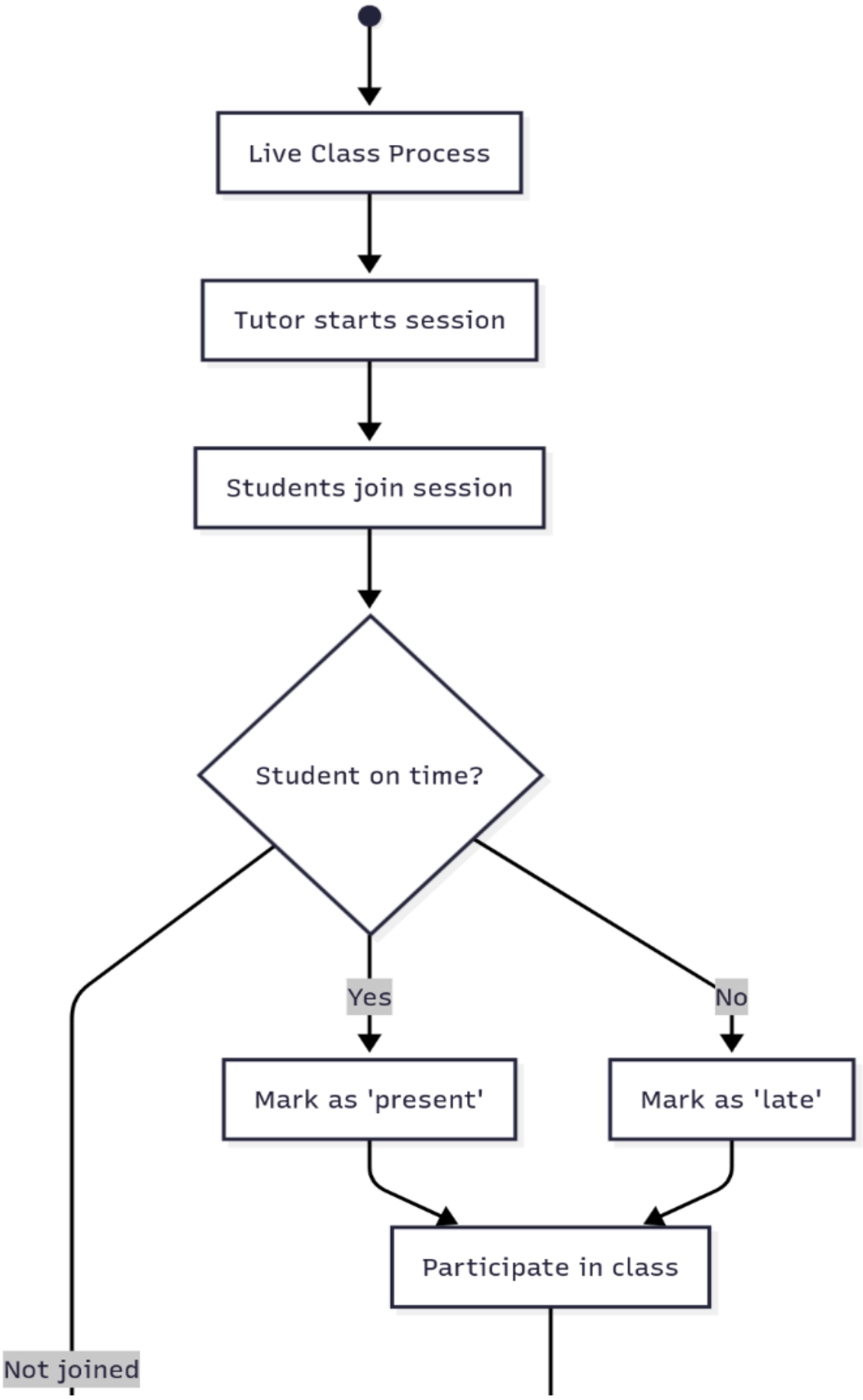
This diagram describes how a student enrolls in a batch. The student views available batches. They select a batch and the system checks the batch availability. If not available, end the process. If available, they enrolled. Then, initiate payment, choose a method, and process the payment. If payment fails, they can retry or cancel. If successful, the payment is recorded and a receipt is generated. Then the batch enrollment status is updated to 'enrolled', they are added to the batch community, and granted access to the batch schedule, and community.

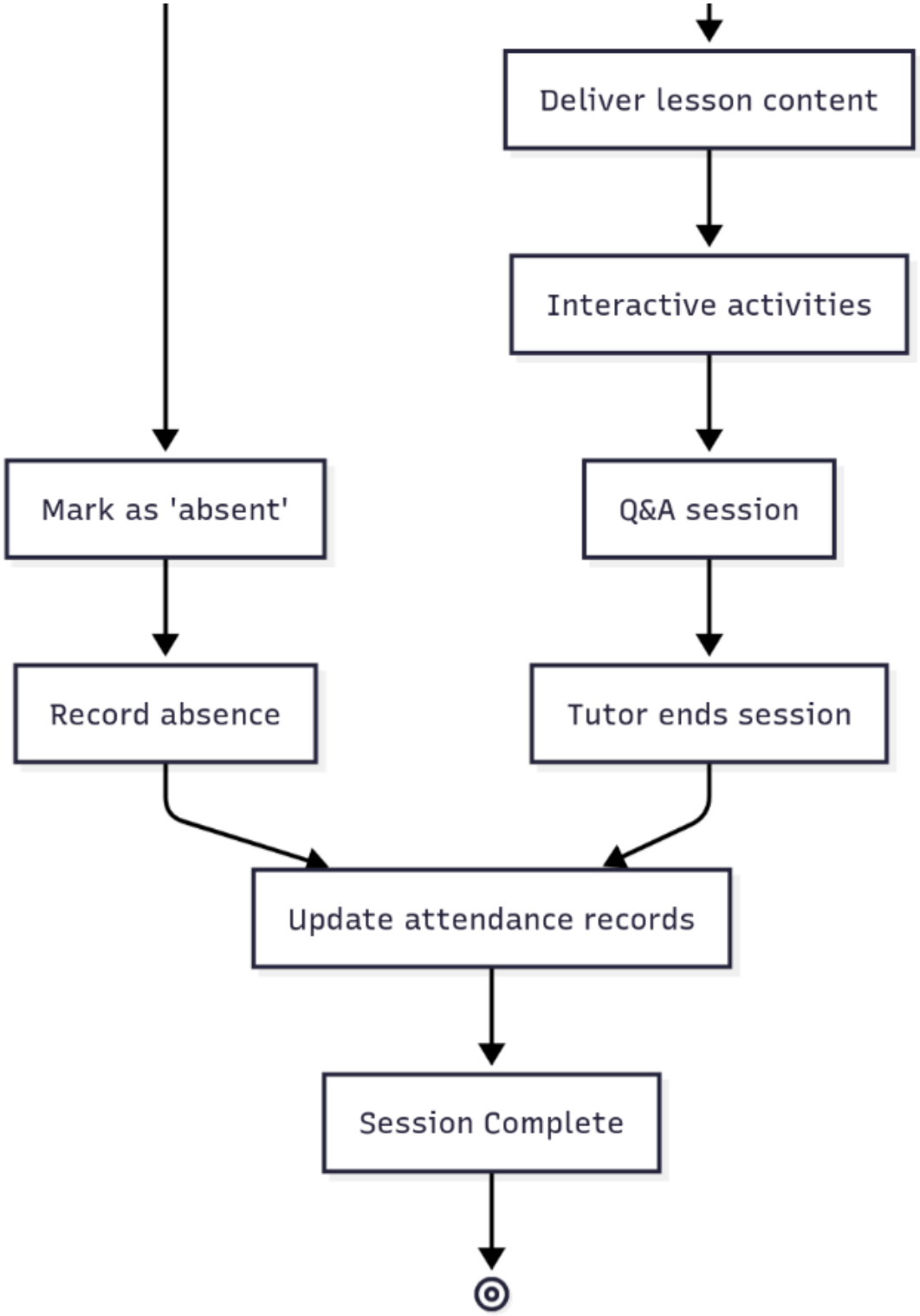

Batch Enrollment Flow
Student views available batches
Select Batch
Check batch availability
Batch available?
Yes
Enrolled
Initiate Payment
Choose payment method


Process payment
Payment successful?
Yes
No
Record payment
Generate receipt
Update Batch Enrollment
status: 'enrolled'
Add to batch community
Grant access to:
- Batch schedule
- Community
Enrollment Complete
No
Retry payment or cancel
Is canceled?
No
Yes
End Process

**Live Class and Attendance Flow**

This diagram outlines the process of a live class. The tutor starts the session. Students join the session and the system checks if they are on time. They are marked as present, late, or absent. Present and late students participate in the class. The tutor delivers content, conducts interactive activities, practice, and holds a questions and answers session. Then the tutor ends the session. Absent students are recorded and included in the attendance update.


Live Class Process
Tutor starts session
Students join session
Student on time?
Yes
No
Mark as 'present'
Mark as 'late'
Participate in class
Not joined

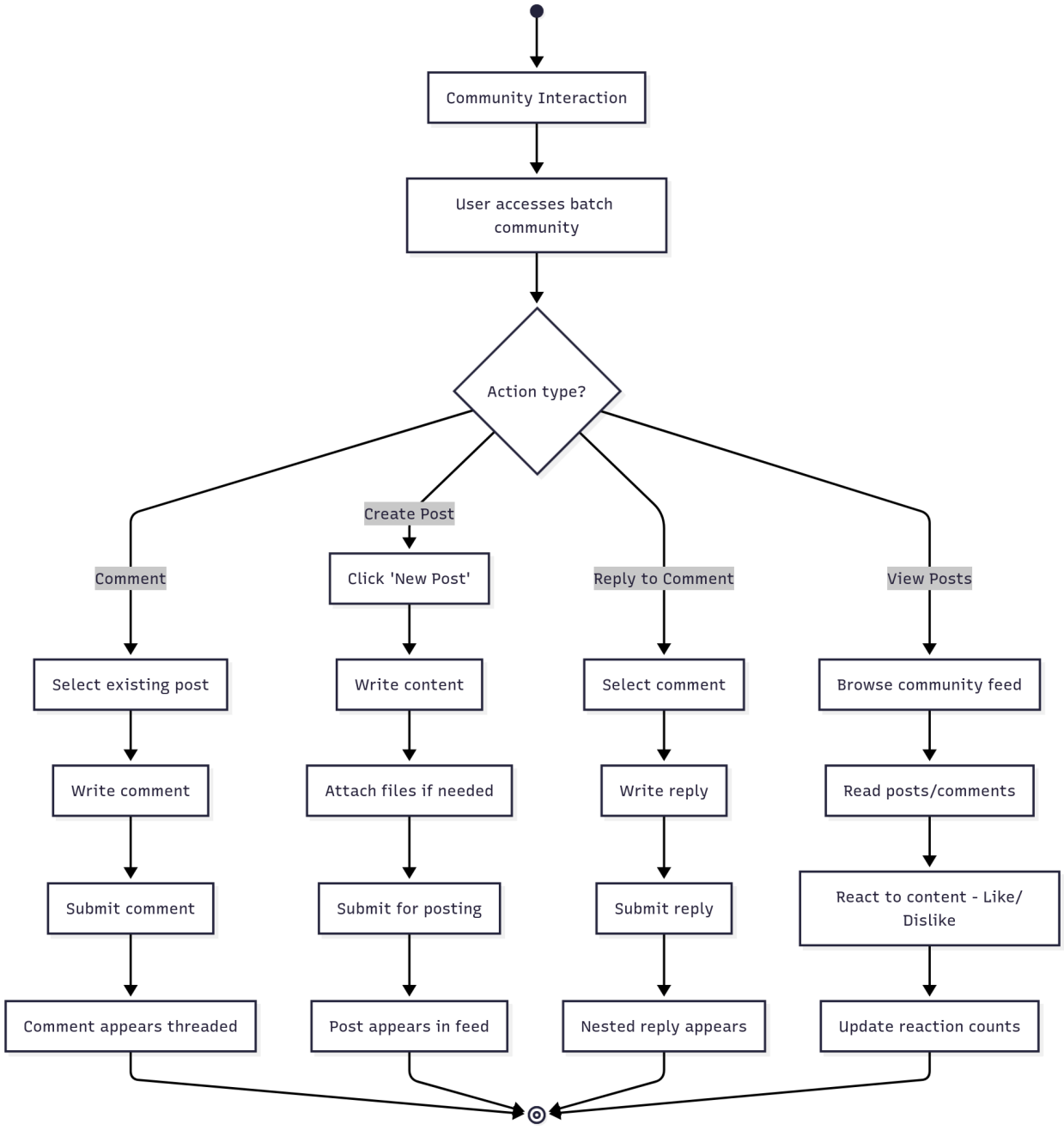



**Community Interaction Flow**

This diagram shows how users interact with the batch community. The user accesses the community and chooses an action: view posts, create a post, comment, or reply to a

comment.

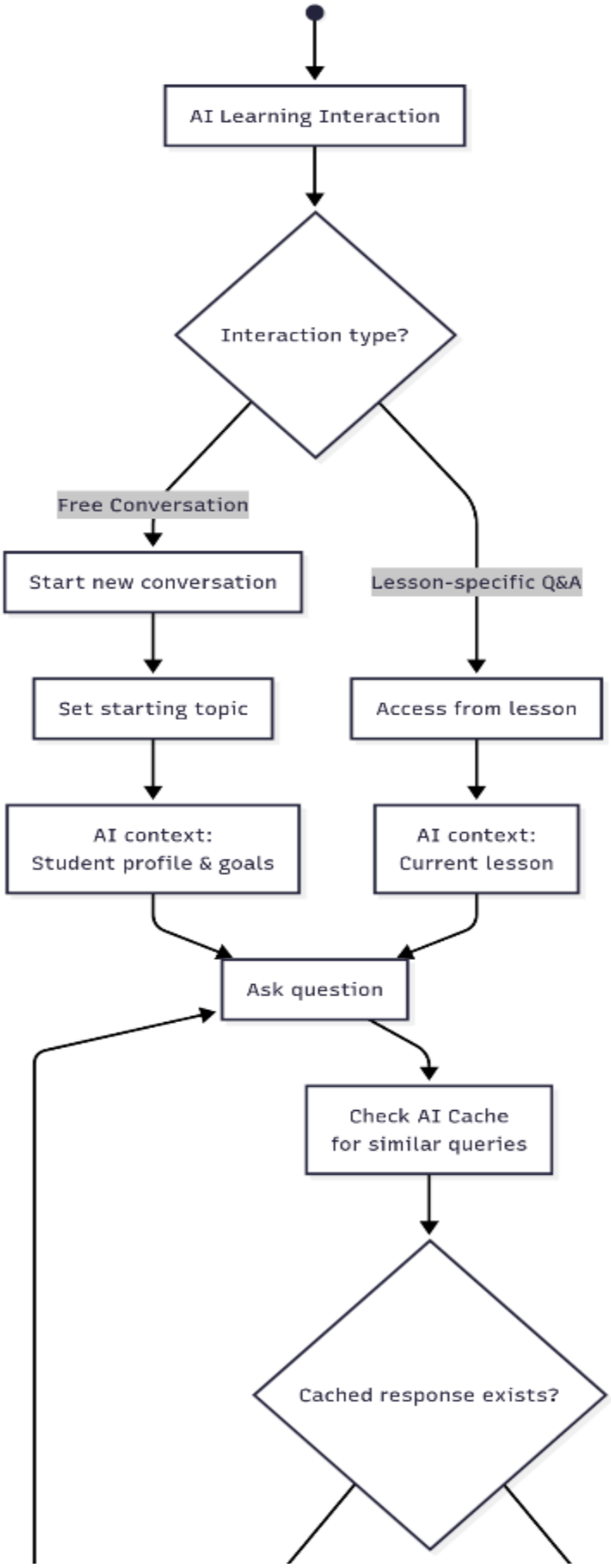
- For viewing posts, they browse the feed, read posts/comments, and react (like/dislike).
- For creating a post, they click 'New Post', write content, attach files if needed, submit, and the post appears in the feed.
- For commenting, they select an existing post, write a comment, submit, and the comment appears threaded.
- For replying to a comment, they select a comment, write a reply, submit, and the nested reply appears.

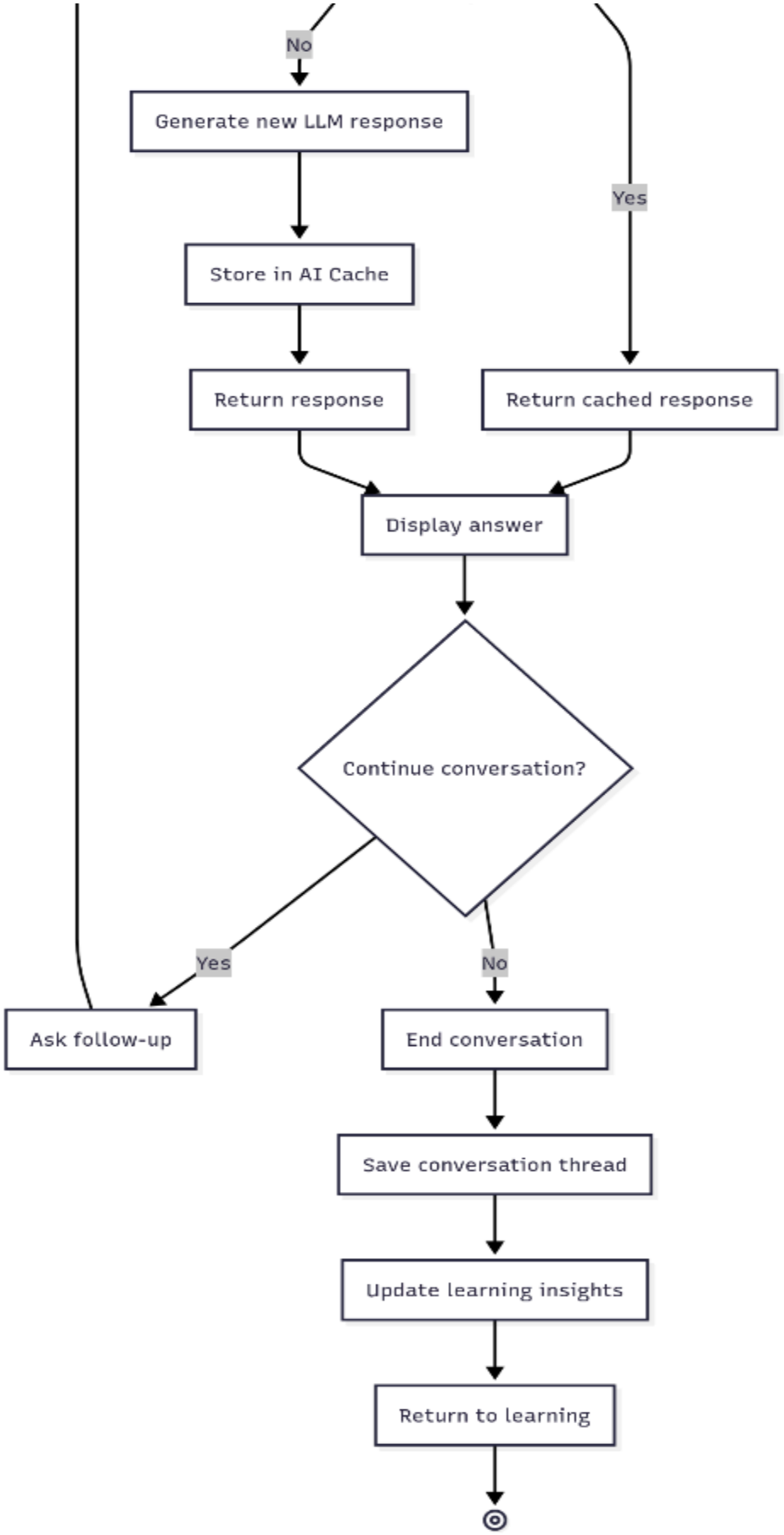



### AI Interaction Flow

This diagram illustrates the AI interaction process. The user chooses between lesson-specific questions and answers or free conversation.

- For lesson-specific questions and answers, the AI context is set to the current lesson.
- For free conversation, the user sets a starting topic and the AI context is set to the student profile and goals.
- Then the user asks a question. The system checks the AI cache for similar queries. If a cached response exists, it is returned; otherwise, a new LLM response is generated, stored in the cache, and returned. The answer is displayed. The user can then continue the conversation (ask follow-up) or end it. The conversation thread is saved and learning insights are updated, then the user returns to learning.


AI Learning Interaction
Interaction type?
Free Conversation
Start new conversation
Set starting topic
AI context:
Student profile & goals
Lesson-specific Q&A
Access from lesson
AI context:
Current lesson
Ask question
Check AI Cache
for similar queries
Cached response exists?

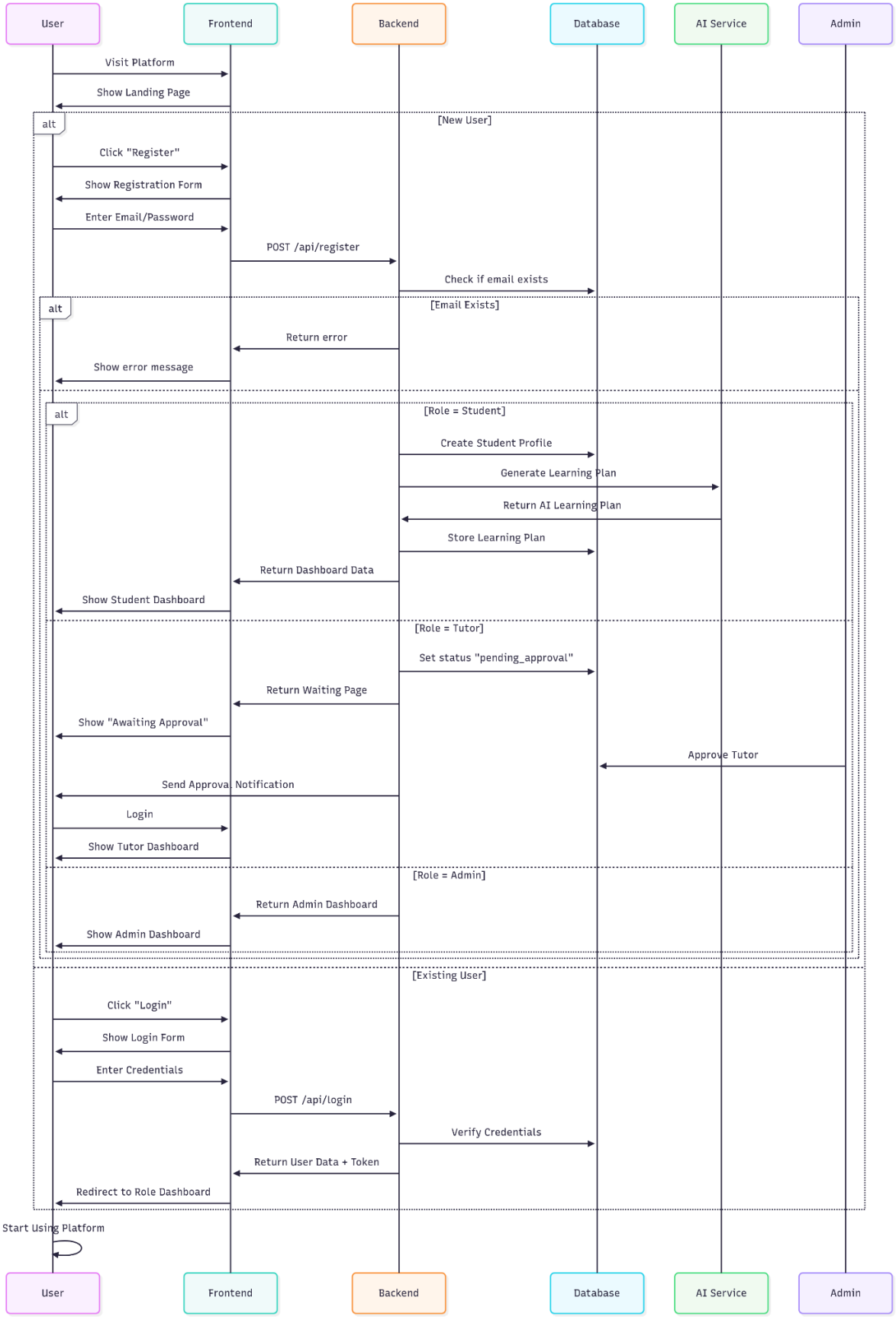

No
Generate new LLM response
Store in AI Cache
Return response
Yes
Return cached response
Display answer
Continue conversation?
Yes
No
Ask follow-up
End conversation
Save conversation thread
Update learning insights
Return to learning

### 2.3.7 Sequence Diagram

**1. User Registration & Onboarding**

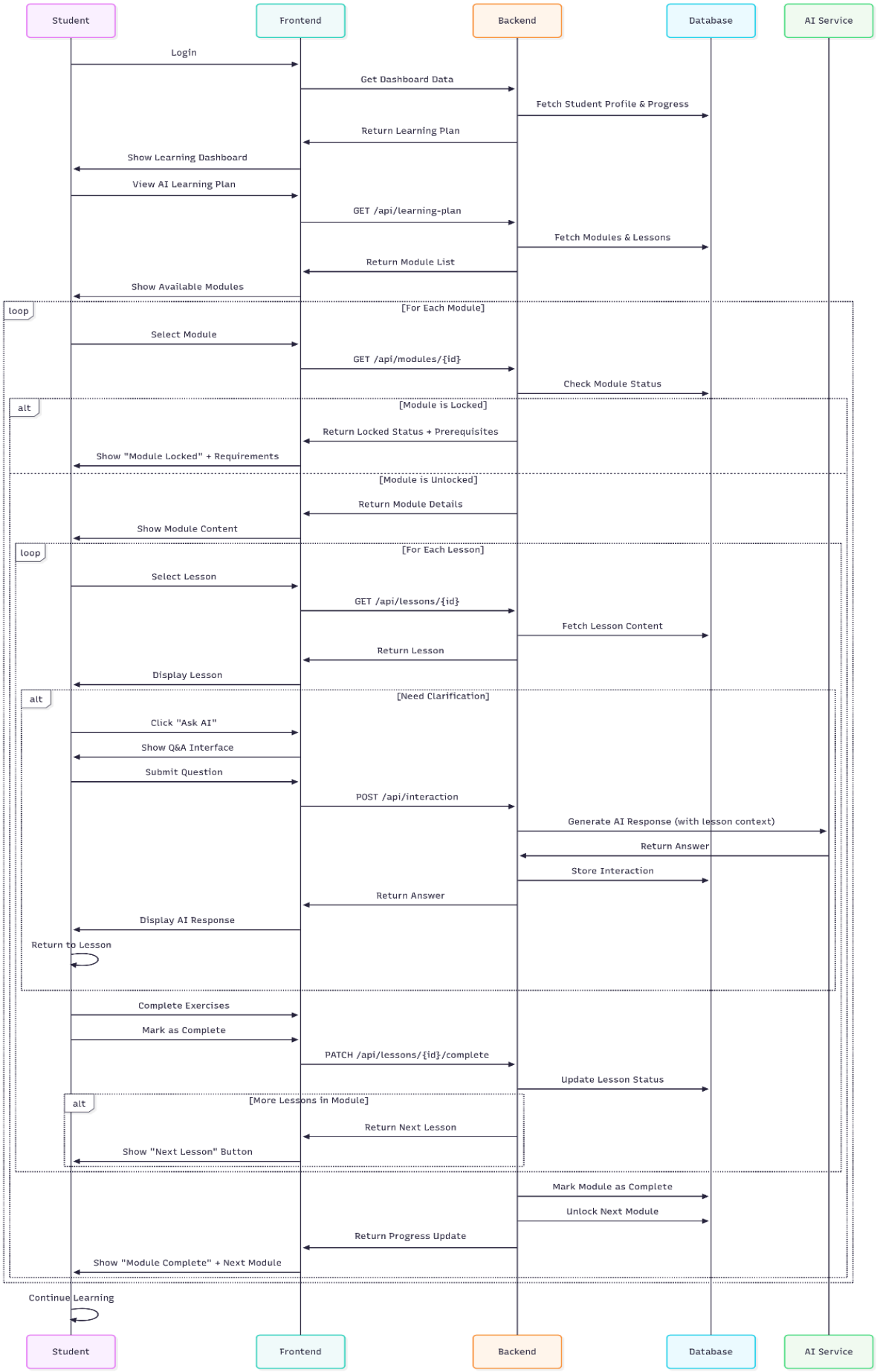
This diagram illustrates the process when a user first visits the platform. It begins with the user arriving and then deciding whether they are a new or existing user. For new users, they register with their email and password, then create a user account. Depending on the role, different paths are taken:

- Students complete their profile, set age range and proficiency, define learning goals, and then an AI learning plan is generated. Then they are shown the dashboard.
- Tutors wait for admin approval before accessing the tutor dashboard.
- Admins go directly to the admin dashboard.
- Existing users simply log in and are shown the dashboard according to their role. The process ends with the user starting to use the platform.


User
Frontend
Backend
Database
AI Service
Admin
Visit Platform
Show Landing Page
alt
[New User]
Click "Register"
Show Registration Form
Enter Email/Password
POST /api/register
Check if email exists
alt
[Email Exists]
Return error
Show error message
alt
[Role = Student]
Create Student Profile
Generate Learning Plan
Return AI Learning Plan
Store Learning Plan
Return Dashboard Data
Show Student Dashboard
[Role = Tutor]
Set status "pending_approval"
Return Waiting Page
Show "Awaiting Approval"
Approve Tutor
Send Approval Notification
Login
Show Tutor Dashboard
[Role = Admin]
Return Admin Dashboard
Show Admin Dashboard
[Existing User]
Click "Login"
Show Login Form
Enter Credentials
POST /api/login
Verify Credentials
Return User Data + Token
Redirect to Role Dashboard
Start Using Platform
User
Frontend
Backend
Database
AI Service
Admin

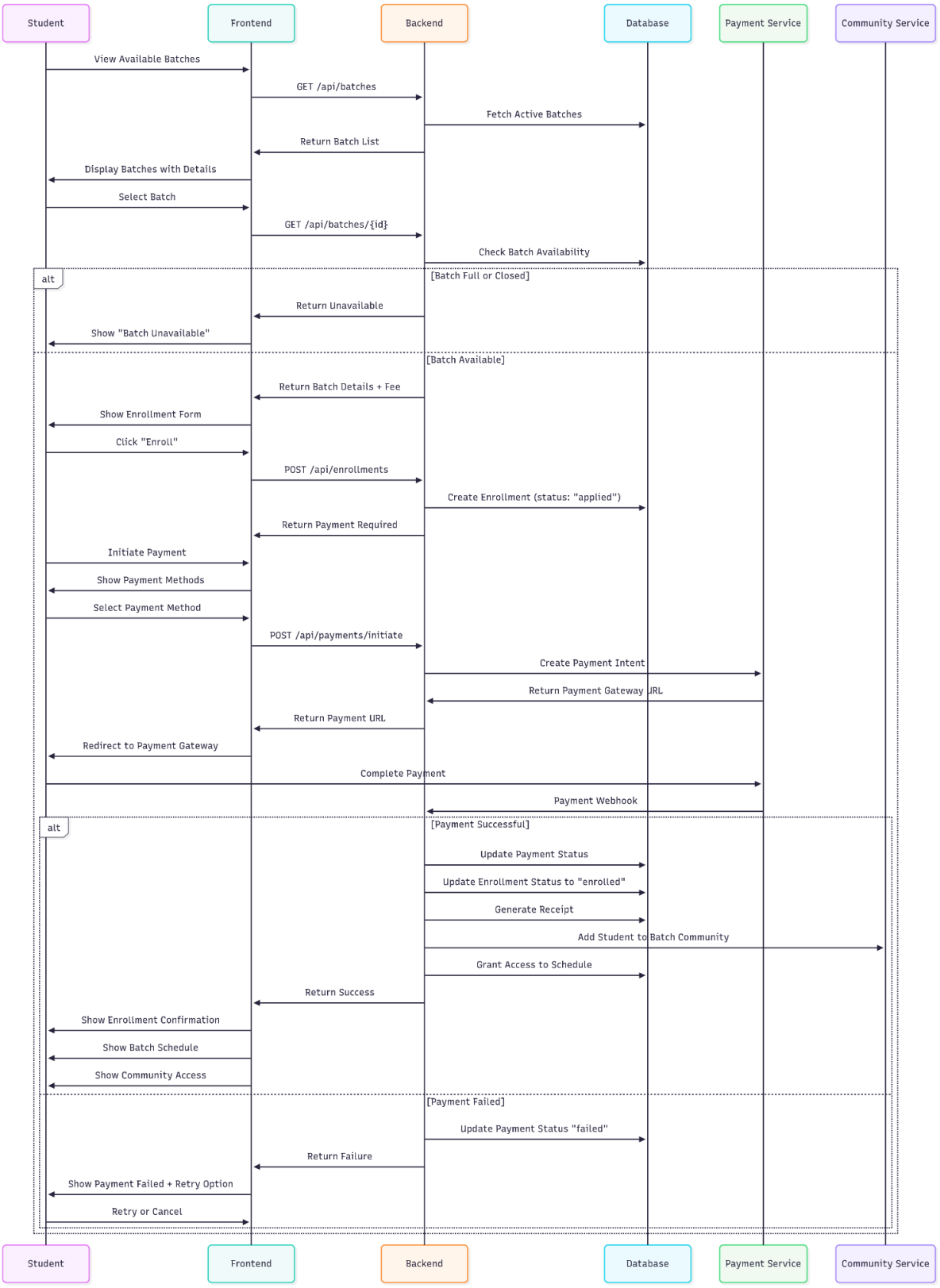
**2. Self-Paced Learning Journey**

This diagram shows the steps a student takes in self-paced learning. The student logs in and views the learning dashboard, then accesses their AI learning plan. They view available modules and check if a module is unlocked. If not, they see the locked module and prerequisites. If unlocked, they enter the module and view lessons in sequence. They select a lesson and access the content. If they need clarification, they use interactive questions and answers: submit a question, get an AI response and return to the lesson. If no clarification is needed, they complete lesson exercises and mark the lesson as completed. If there are more lessons in the module, they proceed to the next lesson; otherwise, the module is completed, the next module is unlocked, and they continue learning.


Student
Frontend
Backend
Database
AI Service
Login
Get Dashboard Data
Fetch Student Profile & Progress
Return Learning Plan
Show Learning Dashboard
View AI Learning Plan
GET /api/learning-plan
Fetch Modules & Lessons
Return Module List
Show Available Modules
loop
[For Each Module]
Select Module
GET /api/modules/{id}
Check Module Status
alt
[Module is Locked]
Return Locked Status + Prerequisites
Show "Module Locked" + Requirements
[Module is Unlocked]
Return Module Details
Show Module Content
loop
[For Each Lesson]
Select Lesson
GET /api/lessons/{id}
Fetch Lesson Content
Return Lesson
Display Lesson
alt
[Need Clarification]
Click "Ask AI"
Show Q&A Interface
Submit Question
POST /api/interaction
Generate AI Response (with lesson context)
Return Answer
Store Interaction
Return Answer
Display AI Response
Return to Lesson
Complete Exercises
Mark as Complete
PATCH /api/lessons/{id}/complete
Update Lesson Status
alt
[More Lessons in Module]
Return Next Lesson
Show "Next Lesson" Button
Mark Module as Complete
Unlock Next Module
Return Progress Update
Show "Module Complete" + Next Module
Continue Learning
Student
Frontend
Backend
Database
AI Service

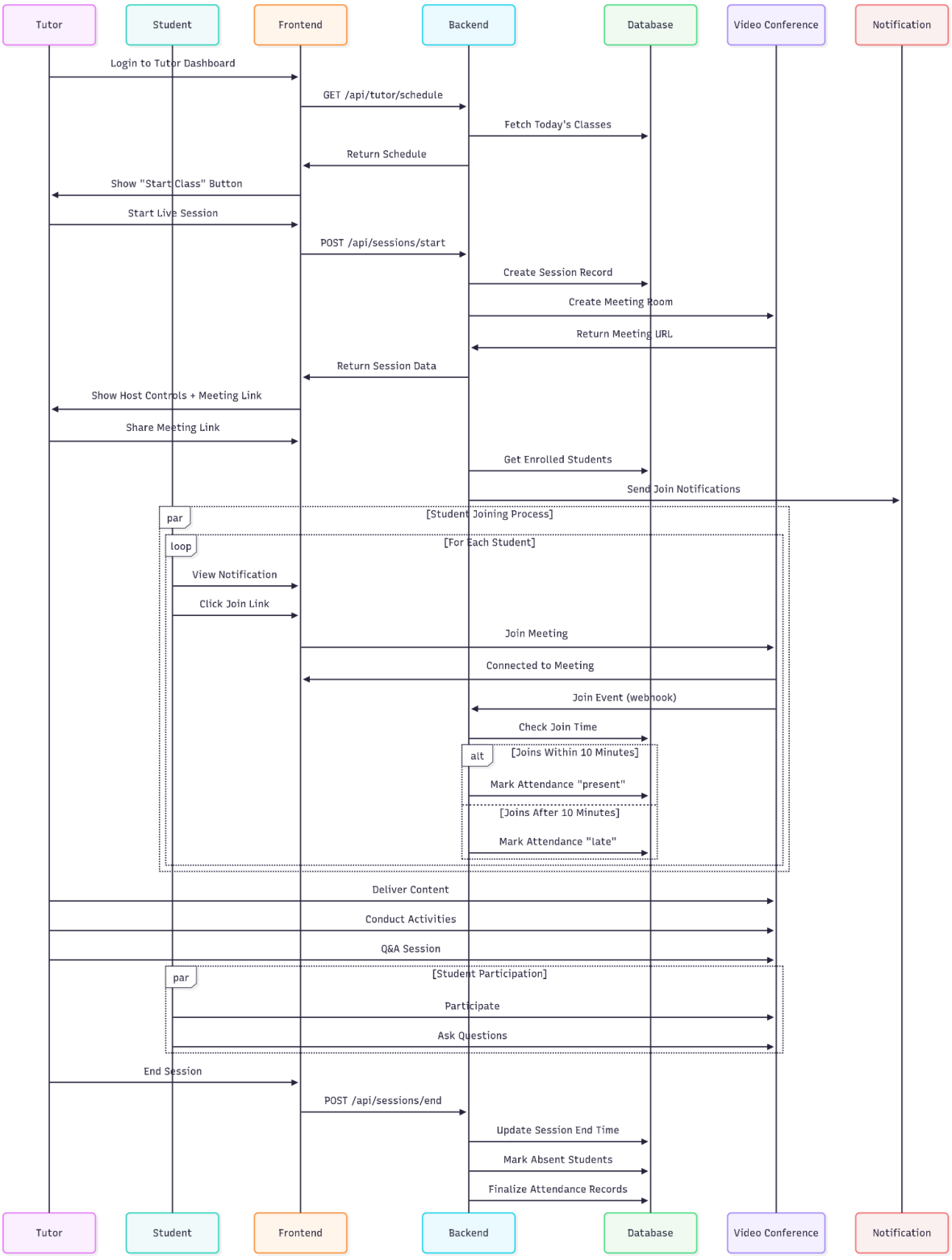
### 3. Batch Enrollment Process

This diagram describes how a student enrolls in a batch. The student views available batches. They select a batch and the system checks the batch availability. If not available, end the process. If available, they enrolled. Then, initiate payment, choose a method, and process the payment. If payment fails, they can retry or cancel. If successful, the payment is recorded and a receipt is generated. Then the batch enrollment status is updated to 'enrolled', they are added to the batch community, and granted access to the batch schedule, and community.




## 4. Live Class and Attendance Flow

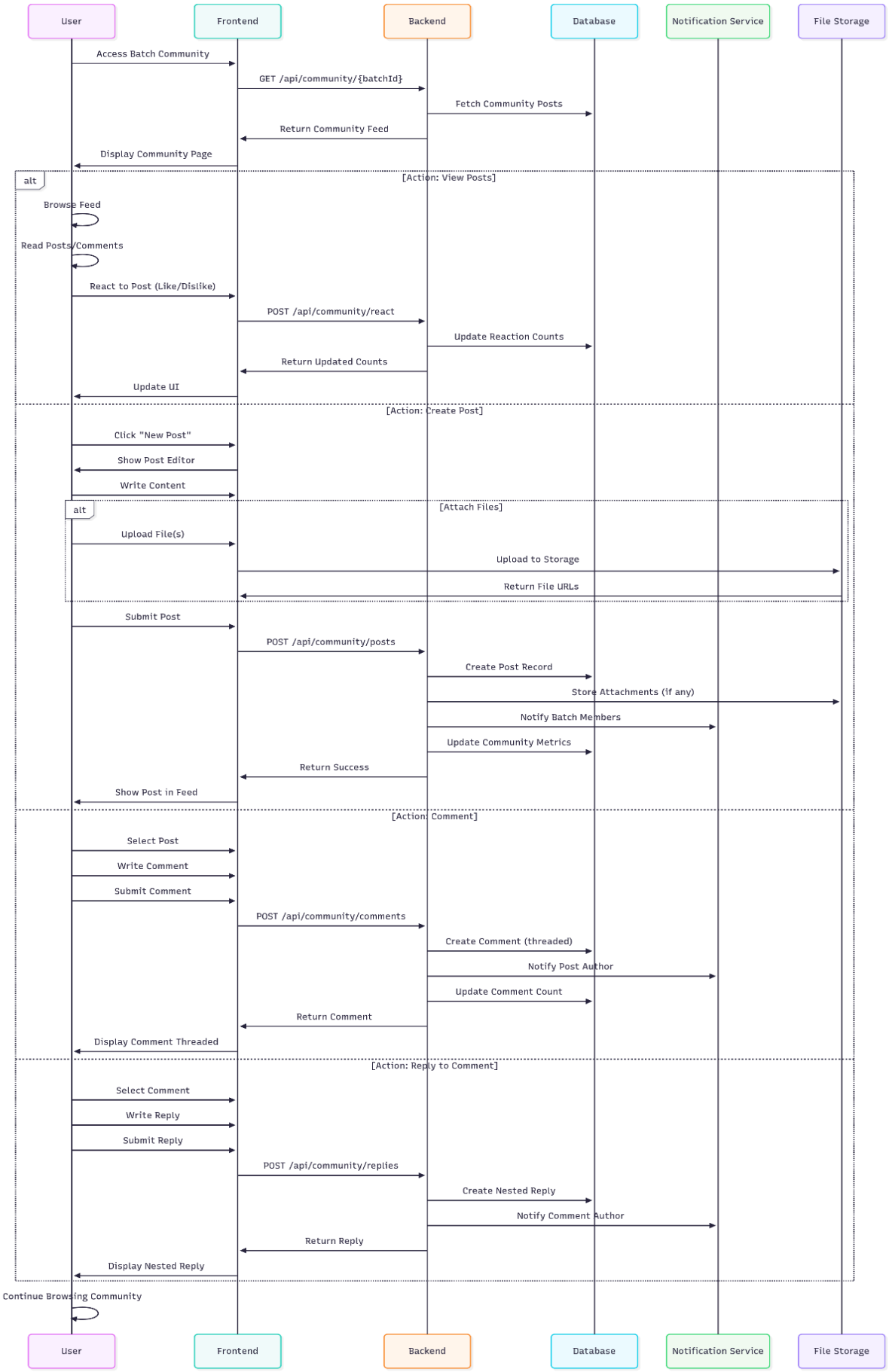
This diagram outlines the process of a live class. The tutor starts the session. Students join the session and the system checks if they are on time. They are marked as present, late, or absent. Present and late students participate in the class. The tutor delivers content, conducts interactive activities, practice, and holds a  questions and answers session. Then the tutor ends the session. Absent students are recorded and included in the attendance update.




## 5. Community Interaction Flow

This diagram shows how users interact with the batch community. The user accesses the community and chooses an action: view posts, create a post, comment, or reply to a comment.

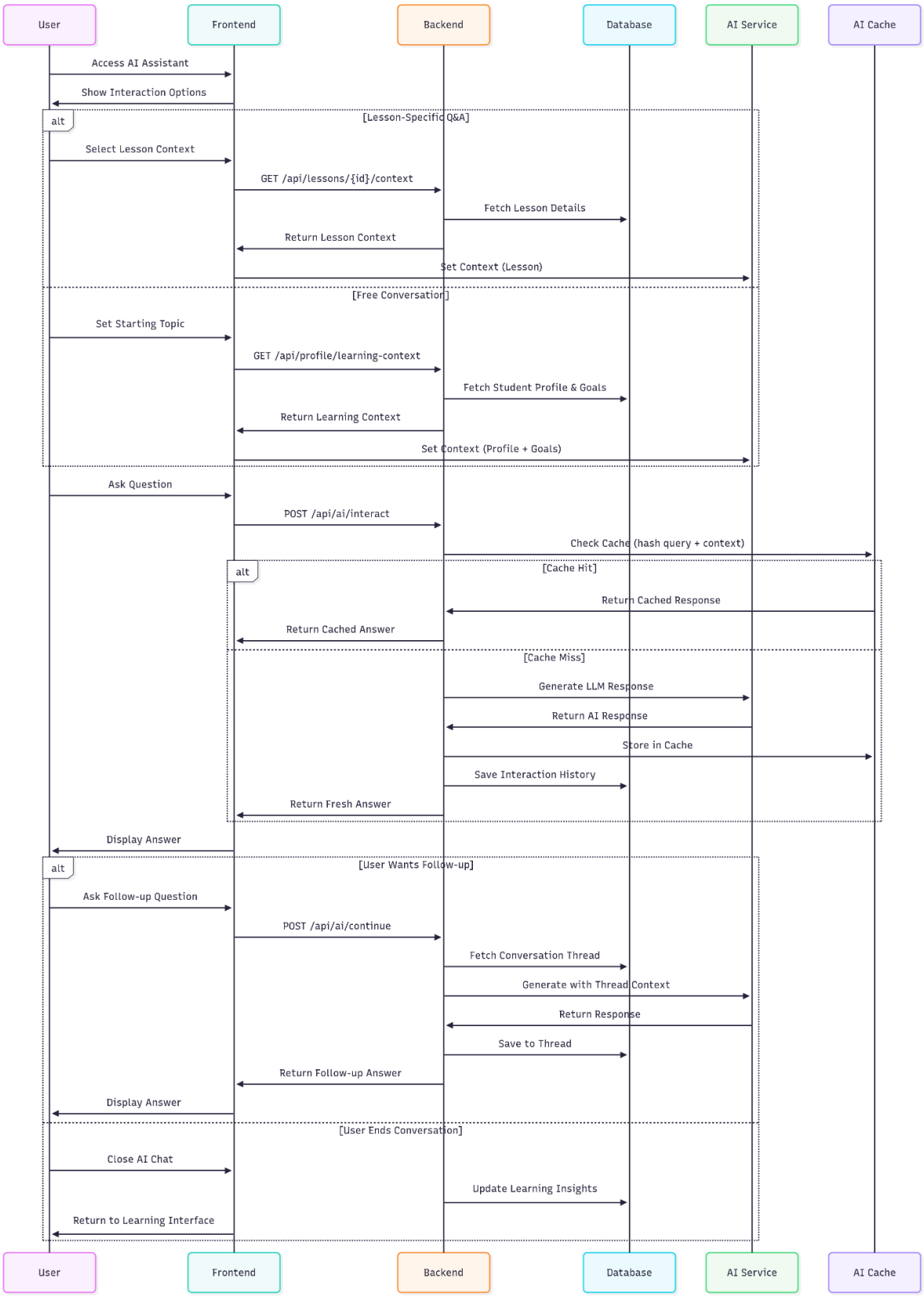
- For viewing posts, they browse the feed, read posts/comments, and react (like/dislike).
- For creating a post, they click 'New Post', write content, attach files if needed, submit, and the post appears in the feed.
- For commenting, they select an existing post, write a comment, submit, and the comment appears threaded.
- For replying to a comment, they select a comment, write a reply, submit, and the nested reply appears.


User
Frontend
Backend
Database
Notification Service
File Storage
Access Batch Community
GET /api/community/{batchId}
Fetch Community Posts
Return Community Feed
Display Community Page
alt
[Action: View Posts]
Browse Feed
Read Posts/Comments
React to Post (Like/Dislike)
POST /api/community/react
Update Reaction Counts
Return Updated Counts
Update UI
[Action: Create Post]
Click "New Post"
Show Post Editor
Write Content
alt
[Attach Files]
Upload File(s)
Upload to Storage
Return File URLs
Submit Post
POST /api/community/posts
Create Post Record
Store Attachments (if any)
Notify Batch Members
Update Community Metrics
Return Success
Show Post in Feed
[Action: Comment]
Select Post
Write Comment
Submit Comment
POST /api/community/comments
Create Comment (threaded)
Notify Post Author
Update Comment Count
Return Comment
Display Comment Threaded
[Action: Reply to Comment]
Select Comment
Write Reply
Submit Reply
POST /api/community/replies
Create Nested Reply
Notify Comment Author
Return Reply
Display Nested Reply
Continue Browsing Community
User
Frontend
Backend
Database
Notification Service
File Storage

### 6. AI Interaction Flow

This diagram illustrates the AI interaction process. The user chooses between lesson-specific questions and answers or free conversation.

- For lesson-specific questions and answers, the AI context is set to the current lesson.
- For free conversation, the user sets a starting topic and the AI context is set to the student profile and goals.
- Then the user asks a question. The system checks the AI cache for similar queries. If a cached response exists, it is returned; otherwise, a new LLM response is generated, stored in the cache, and returned. The answer is displayed. The user can then continue the conversation (ask follow-up) or end it. The conversation thread is saved and learning insights are updated, then the user returns to learning.


User
Frontend
Backend
Database
AI Service
AI Cache
Access AI Assistant
Show Interaction Options
alt
[Lesson-Specific Q&A]
Select Lesson Context
GET /api/lessons/{id}/context
Fetch Lesson Details
Return Lesson Context
Set Context (Lesson)
[Free Conversation]
Set Starting Topic
GET /api/profile/learning-context
Fetch Student Profile & Goals
Return Learning Context
Set Context (Profile + Goals)
Ask Question
POST /api/ai/interact
Check Cache (hash query + context)
alt
[Cache Hit]
Return Cached Response
Return Cached Answer
[Cache Miss]
Generate LLM Response
Return AI Response
Store in Cache
Save Interaction History
Return Fresh Answer
Display Answer
alt
[User Wants Follow-up]
Ask Follow-up Question
POST /api/ai/continue
Fetch Conversation Thread
Generate with Thread Context
Return Response
Save to Thread
Return Follow-up Answer
Display Answer
[User Ends Conversation]
Close AI Chat
Update Learning Insights
Return to Learning Interface
User
Frontend
Backend
Database
AI Service
AI Cache

## 2.3.8 Analysis Class Model

**1. User Management**

**Class: User**

**Description:** Represents a system user with authentication credentials and role-based permissions. Serves as the central identity for all system actions and account management.

**Attributes:**

- email: String - Unique identifier for authentication and communication
- passwordHash: String - Secured authentication credential
- role: Enum(ADMIN, STUDENT, TUTOR) - Defines system permissions and capabilities
- isDeleted: Boolean - Tracks account activation status (soft delete)
- studentProfile: StudentProfile - Associated learner profile (for student role)
- notifications: List<Notification> - Alerts and messages received by this user
- attendanceRecords: List<Attendance> - Session participation history
- batchesInstructed: List<Batch> - Batches where this user serves as tutor
- communityPosts: List<BatchCommunity> - Forum discussions initiated by this user.

**Methods:**

- authenticate(credentials): Boolean - Validates provided credentials against stored values
- hasPermission(action): Boolean - Checks if user role permits the specified action
- deleteAccount(): void - Marks account as inactive without deleting data
- getUnreadNotifications(): List<Notification> - Retrieves notifications with unread status

**Class: StudentProfile**

**Description:** Stores personalized learning information, preferences, and generated AI learning plans for an individual student.

**Attributes:**

- learningPreferences: Map<String, Object> - Personalization settings and preferences
- learningGoals: List<String> - Targeted learning objectives

- constraints: Map<String, Object> - Time, pace, and content limitations
- aiLearningPlan: String - Generated personalized learning pathway
- user: User - Associated user account
- modules: List<Module> - Personalized curriculum modules
- freeConversations: List<FreeConversation> - Unstructured chat history
- batchEnrollments: List<BatchEnrollment> - Batch participation records

**Methods:**

- generateLearningPlan(): void - Creates or updates AI-powered learning pathway
- updatePreferences(newPreferences): void - Modifies learning preferences
- getActiveEnrollments(): List<BatchEnrollment> - Retrieves currently active batch enrollments

**2. Learning Content**

**Class: Module**

**Description:** Organizes lessons into logical curriculum units with controlled and personalization.

**Attributes:**

- title: String - Display name of the module
- description: String - Content overview and objectives
- displayOrder: Integer - Sequence position in curriculum
- isUnlocked: Boolean - Access control based on progression
- studentProfile: StudentProfile - Owning learner profile
- lessons: List<Lesson> - Contained instructional units

**Methods:**

- unlock(): void - Grants access to module content
- getNextLesson(): Lesson - Retrieves next lesson based on display order

**Class: Lesson**

**Description:** Delivers structured instructional content with AI interaction hosting.

**Attributes:**

- title: String - Lesson identifier
- content: String - Instructional material

- ❖ completionStatus: Enum(NOT_STARTED, IN_PROGRESS, COMPLETED) - Progress tracking
- ❖ module: Module - Parent curriculum unit
- ❖ interactions: List<Interaction> - Structured questions and answers dialogues

**Methods:**

- ❖ markComplete(): void - Updates lesson status to completed
- ❖ recordInteraction(question, answer): void Adds structured questions and answers to lesson
- ❖ getCompletionStatus(): Enum - Returns current progress state

**Class: Interaction**

**Description:** Records structured AI question-and-answer dialogues within lesson contexts for learning tracking.

**Attributes:**

- ❖ question: String - Student query or prompt
- ❖ aiResponse: String - AI-generated answer or guidance
- ❖ timestamp: DateTime - When the interaction occurred
- ❖ lesson: Lesson - Containing instructional context

**Methods:**

- ❖ recordResponse(response): void - Stores AI-generated answer
- ❖ getDialogue(): Map<String, String> - Returns question-response pair

**Class: FreeConversation**

**Description:** Manages unstructured AI dialogue threads separate from lesson-specific interactions.

**Attributes:**

- ❖ title: String - Conversation topic or identifier
- ❖ studentProfile: StudentProfile - Owning learner
- ❖ parentConversation: List<FreeConversation> - Threaded responses or continuations
- ❖ messageHistory: List<Map<String, String>> - Sequential chat messages with sender metadata

**Methods:**

- addMessage(sender, content): void - Appends message to conversation history
- threadConversation(newTopic): FreeConversation - Creates new linked conversation

**3. Batch Management**

**Class: Batch**

**Description:** Defines a cohort-based learning group with enrollment limits, scheduling, and community features.

**Attributes:**

- name: String - Batch identifier
- level: String - Proficiency level (beginner, intermediate, advanced)
- startDate: Date - Cohort commencement date
- endDate: Date - Planned conclusion date
- capacity: Integer - Maximum enrollment count
- status: Enum(UPCOMING, ACTIVE, COMPLETED) - Current lifecycle state
- feeAmount: Decimal - Enrollment cost
- batchEnrollments: List<BatchEnrollment> - Student participation records
- instructors: List<User> - Assigned tutors (users with TUTOR role)
- schedules: List<BatchSchedule> - Associated time schedules
- community: BatchCommunity - Dedicated discussion forum

**Methods:**

- checkAvailability(): Boolean - Determines if enrollment slots remain
- updateStatus(newStatus): void - Transitions batch lifecycle state
- calculateEnrollmentPercentage(): Float - Computes current enrollment vs capacity

**Class: BatchSchedule**

**Description:** Links batches with specific time schedules and serves as anchor for attendance recording.

**Attributes:**

- batch: Batch - Associated cohort

- course: Course - Specific course being taught
- schedules: List<Schedule> - Reusable time slot templates
- attendanceRecords: List<Attendance> - Session participation records

**Methods:**

- getUpcomingSessions(): List<DateTime> - Retrieves future session dates
- recordAttendance(sessionDate, attendees): void - Creates attendance records for session

**Class: Schedule**

**Description:** Defines reusable time slot patterns for batch session scheduling.

**Attributes:**

- dayOfWeek: Enum - Scheduled day (MONDAY, TUESDAY, etc.)
- startTime: Time - Session commencement time
- endTime: Time - Session conclusion time
- batchSchedules: List<BatchSchedule> - Batch associations using this template

**Methods:**

- calculateDuration(): Duration - Computes session length.

**Class: Attendance**

**Description:** Records participant presence for specific batch sessions with punctuality tracking.

**Attributes:**

- sessionDate: DateTime - Specific session date and time
- status: Enum(PRESENT, ABSENT, LATE) - Participation state
- checkInTime: DateTime - Actual arrival time
- user: User - Participant (student or tutor)
- batchSchedule: BatchSchedule - Associated scheduled session

**Methods:**

- markPresent(checkInTime): void - Records participant attendance
- markAbsent(): void - Records participant absence

**Class: BatchEnrollment**

**Description:** Manages student enrollment lifecycle within a batch, linking to payment and certification.

**Attributes:**

- enrollmentDate: Date - When student joined batch
- status: Enum(ACTIVE, COMPLETED, DROPPED) - Enrollment state
- studentProfile: StudentProfile - Enrolled learner
- batch: Batch - Joined cohort
- payment: Payment - Associated fee transaction
- certificate: Certificate - Completion after batch completion.

**Methods:**

- enrollStudent(): void - Initiates enrollment process
- completeEnrollment(): void - Marks enrollment as successfully finished
- dropEnrollment(): void - Withdraws student from batch
- isEligibleForCertificate(): Boolean - Determines if completion criteria met.

**Class: Payment**

**Description:** Processes and tracks enrollment fee transactions with receipt management.

**Attributes:**

- amount: Decimal - Transaction value
- transactionDate: DateTime - When payment occurred
- status: Enum(PENDING, COMPLETED, FAILED) - Payment state
- receiptReference: String - Payment confirmation identifier
- batchEnrollment: BatchEnrollment - Associated enrollment

**Methods:**

- processPayment(paymentDetails): Boolean - Attempts to complete transaction
- generateReceipt(): String - Creates payment confirmation document

**Class: Certificate**

**Description:** Generates and stores completion credentials.

**Attributes:**

- ❖ certificateId: String - Unique credential identifier
- ❖ issueDate: Date - When certificate was generated
- ❖ batchEnrollment: BatchEnrollment - Certified enrollment

**Methods:**

- generateCertificate(): void - Creates completion credential

**4. Community & Support**

**Class: BatchCommunity**

**Description:** Facilitates threaded group discussions within a batch cohort with content interaction features.

**Attributes:**

- ❖ batch: Batch - Associated cohort
- ❖ posts: List<CommunityPost> - Discussion threads
- ❖ attachments: List<CommunityAttachment> - Shared files

**Methods:**

- ❖ createPost(author, content): CommunityPost - Initiates new discussion thread

**Class: CommunityAttachment**

**Description:** Manages file attachments within community discussions with metadata tracking.

**Attributes:**

- ❖ filename: String - Original file name
- ❖ fileType: String - MIME type or extension
- ❖ uploadDate: DateTime - When file was shared
- ❖ uploadedBy: User - Sharing user
- ❖ batchCommunity: BatchCommunity - Containing discussion forum

**Methods:**

- ❖ download(): ByteStream - Provides file content

**Class: Notification**

**Description:** Delivers alert messages to users with delivery status tracking.

**Attributes:**

- message: String - Notification content
- timestamp: DateTime - When notification created
- status: Enum(UNREAD, READ) - User interaction state
- user: User - Recipient

**Methods:**

- markAsRead(): void - Updates notification status to read

**5. AI Infrastructure**

**Class: AICache**

**Description:** Caches AI prompt-response pairs to reduce API costs and improve response latency.

**Attributes:**

- prompt: String - Input prompt
- aiResponse: String - Cached AI-generated response

**Methods:**

- storeResponse(prompt, response): void - Creates new cache entry
- retrieveResponse(prompt): String - Returns cached response if available

**6. Additional Supporting Classes**

**Class: Course**

**Description:** Represents a specific instructional course that can be scheduled across multiple batches.

**Attributes:**

- title: String - Course name
- description: String - Course overview and objectives
- batchSchedules: List<BatchSchedule> - Scheduled instances of this course

**Methods:**

- ❖ getScheduledBatches(): List<Batch> - Retrieves batches offering this course

**Class: CommunityPost**

**Description:** Represents an individual discussion message within batch community forums.

**Attributes:**

- ❖ content: String - Message text
- ❖ postDate: DateTime - When message was created
- ❖ author: User - Creating user
- ❖ parentPost: CommunityPost - Thread parent message
- ❖ replies: List<CommunityPost> - Response messages
- ❖ likes: Integer - Positive reaction count
- ❖ dislikes: Integer - Negative reaction count
- ❖ batchCommunity: BatchCommunity - Containing forum

**Methods:**

- ❖ addReply(content, author): CommunityPost - Creates response to this post
- ❖ recordReaction(user, reactionType): void - Updates like/dislike counts

### 2.3.9 Logic Model

**1. Inputs (Resources & Data Structures)**

**Human Resources**

- ❖ Students
- ❖ Tutors
- ❖ Admins

**Digital Resources**

- ❖ User Accounts & Student Profiles
- ❖ Learning Structure: Modules, Lessons
- ❖ AI Interaction Data: Question - answer logs, free conversations, AI cache
- ❖ Batch Management: Courses, Batches, Schedules, Instructors, Enrollments, Attendance
- ❖ Community System: Posts, comments, attachments

- Payment & Certification System
- Notifications & Feedback

**System Resources**

- LLM / AI model
- Scheduling rules
- Payment gateway

**2. Activities (What the System Does)**

**User & Profile Activities**

- Register / login users
- Create and update student learning profiles
- Setting goals and constraints

**AI-Driven Learning Activities**

- Generate personalized learning plans
- Build modules and lessons per profile
- Store interactions with AI (questions, answers, free chat)
- Use AI cache to optimize repeated prompt responses

**Learning Engagement Activities**

- Deliver lessons
- Track lesson completion
- Unlock modules dynamically

**Batch/Class Activities**

- Create courses
- Create batches with start/end dates
- Assign instructors
- Schedule classes
- Enroll students
- Mark attendance

**Community Activities**

- Students and tutors post in batch community
- Threaded replies

- ❖ Upload attachments
- ❖ Like/dislike interactions

**Notifications & Feedback Activities**

- ❖ Send notification alerts
- ❖ Collect user feedback & ratings

**Payments & Certification**

- ❖ Payment processing for batch enrollments
- ❖ Generating receipts
- ❖ Generating certificates after course completion

**3. Outputs (System Deliverables)**

**Learning Outputs**

- ❖ Student learning plans
- ❖ Ordered modules and lessons
- ❖ Stored AI interaction logs

**Operational Outputs**

- ❖ Active batch lists
- ❖ Attendance records
- ❖ Batch schedules
- ❖ Instructor assignments

**Social Outputs**

- ❖ Community posts
- ❖ Comment threads
- ❖ File attachments
- ❖ Interactions (likes, dislikes)

**Administrative Outputs**

- ❖ Notifications delivered
- ❖ User feedback reports

**Business Outputs**

- Payment records
- Completed transactions

## 4. Outcomes (Benefits & Impact)

### Short-Term Outcomes

- Accurate student profiling
- Personalized learning content
- Increased student engagement through lessons and conversations
- Efficient class and batch management
- Improved communication via community
- Transparent payment tracking

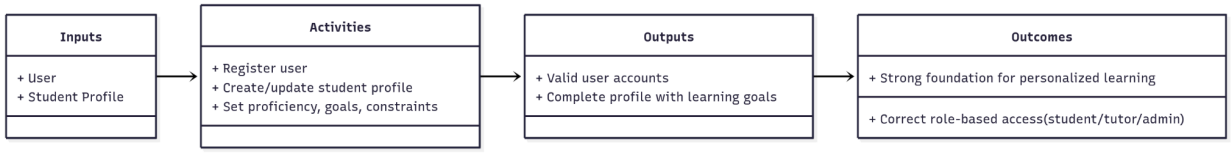
### Intermediate Outcomes

- Better tutor - student interaction
- Faster content creation using AI
- Higher course completion rates

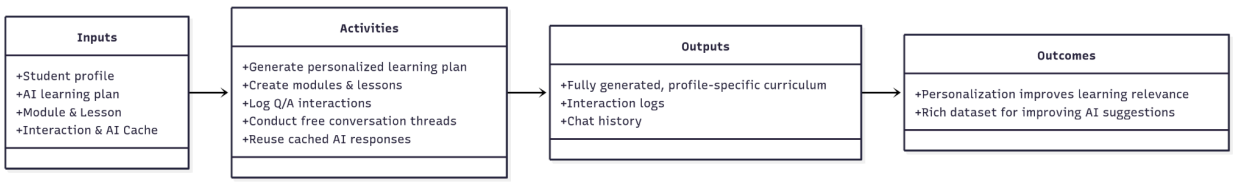
### Long - Term Outcomes

- Scalable, AI-driven language learning ecosystem
- Consistent student language improvement
- Higher retention and revenue
- Strong learning community
- Improved teaching efficiency and quality

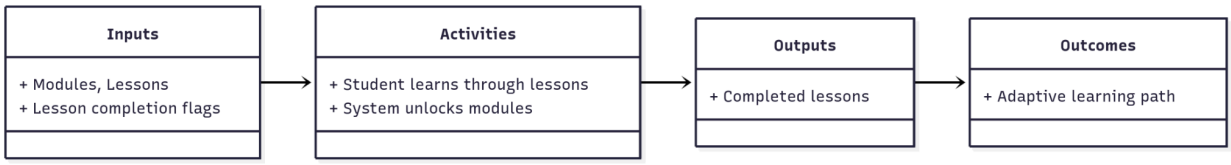
## 1. User & Profile Management Logic Model



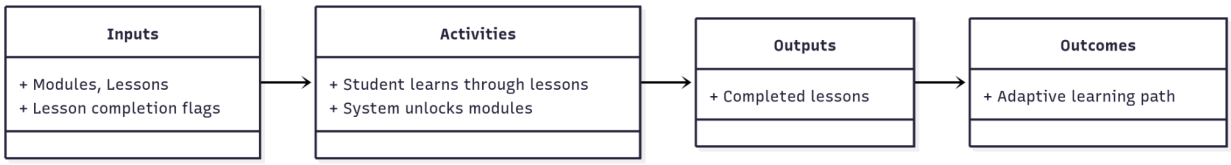

## 2. AI - Driven Learning Plan & Content Logic Model



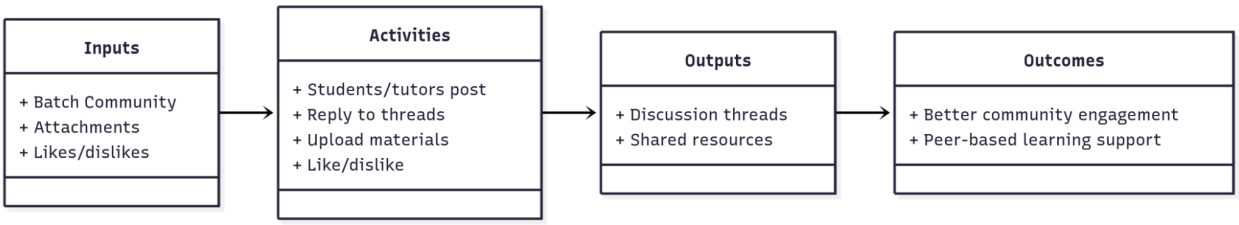

## 3. Learning Engagement Logic Model



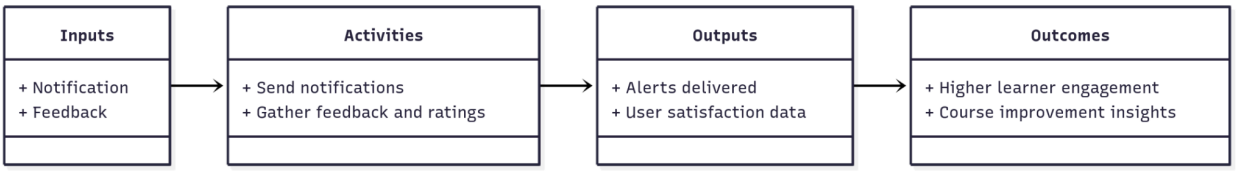

### 4. Batch Management Logic Model




### 5. Community & Social Logic Model




### 6. Notifications & Feedback Logic Model




## 2.4.Non-Functional Requirements

### 1. Reliability & Availability

NFREQ1.1: Data Integrity

- The system shall enforce referential integrity across all database relationships.
- All financial transactions shall be ACID-compliant.
- The system shall prevent data inconsistency during concurrent operations

### 2. Security Requirements

NFREQ2.1: Data Protection

- All user passwords shall be securely stored using industry-standard hashing techniques.

- ❖ Sensitive data shall be encrypted both at rest and in transit.

NFREQ2.2: Access Control

- ❖ Role-based access control shall be enforced at both UI and API levels.
- ❖ Sensitive operations shall require authenticated and authorized requests.

**3. Usability Requirements**

NFREQ3.1: User Experience

- ❖ The user interface shall be intuitive and easy to use for non-technical users.
- ❖ The system shall provide a responsive UI suitable for mobile and web platforms.

**4. Maintainability & Extensibility**

NFREQ4.1: Code Quality

- ❖ API documentation shall be automatically generated and kept up to date.
- ❖ The codebase shall follow standardized coding conventions.

NFREQ4.2: Modularity

- ❖ AI integration components shall be modular and replaceable.
- ❖ Core business logic shall remain independent of AI service providers.

**5. Compatibility Requirements**

NFREQ5.1: Platform & Browser Support

- ❖ The web system shall support modern browsers (Chrome, Firefox, Safari, Edge).
- ❖ The mobile application shall support recent Android and iOS versions.

NFREQ5.2: External Integration

- ❖ The system shall integrate with AI provider APIs for real-time content generation.
- ❖ The system shall integrate with payment gateways using secure APIs.
- ❖ The system shall support translation services for multilingual content.

**6. Data Management & Performance**

NFREQ6.1: Storage & Query Performance

- ❖ Database schemas and indexes shall be optimized for high read performance.

- ❖ Cached AI responses shall reduce repeated processing and improve response time.

**7. Internationalization**

NFREQ7.1: Localization & Language Support

- ❖ The system shall support Unicode for multilingual content.
- ❖ The system shall be designed to support future language localization.
- ❖ Users shall be able to select their preferred language for learning materials.

**8. Ethical AI & Transparency**

NFREQ8.1: AI Explainability

- ❖ The system shall provide simple explanations for AI-generated feedback.
- ❖ Users shall understand why corrections or scores were given.

**9. Auditability & Logging**

NFREQ9.1: Activity Logging

- ❖ The system shall log critical activities such as payments, logins, and admin actions.
- ❖ Logs shall be securely stored for audit purposes.

**10. Performance Under Load**

NFREQ10.1: Load Handling

- ❖ The system shall maintain acceptable performance under peak usage.
- ❖ Background tasks shall not block user interactions.

**11. Sustainability & Cost Efficiency**

NFREQ11.1: Resource Optimization

- ❖ AI requests shall be optimized to minimize computation cost.
- ❖ The system cached results where possible.

## 2.5. System Requirements

System requirements define the minimum and recommended resources needed to develop, deploy, and use the AI-powered web and mobile English learning system.

### 2.5.1.Hardware Requirements

| Component | Specification | Description |
|---|---|---|
| CPU | Intel core i5/AMD Ryzen 5(minimum) | Required for efficient processing of development tools, backend services, and AI operations. |
| RAM | 8 GB(minimum),16 (Recommend ) | Supports smooth multitasking during development, testing, and runtime operations. |
| Storage | 256 GB SSD | Provides fast data access for source code, databases, and system files. |
| Graphics(GPU) | Integrated or dedicated GPU | Enhances performance for AI model processing and emulator execution. |
| Server hardware | 2-4 Virtual core | Supports backend services, AI modules, and multiple concurrent users. |
| Server RAM | 4-8 GB | Ensures stable server-side application and database performance. |
| Internet connectivity | Broadband(3G/4G/Wi-Fi) | Enables online learning, video conferencing, and real-time system interaction. |
| Client Device | Android smartphone PCs | Allows students, tutors, and administrators to access the system. |
| Camera and microphone | Built-in or External | Required for live online classes and AI-based speaking evaluation. |
| Backup storage | Minimum 50 GB | Used for system data backup and recovery. |

Table: Hardware Requirements

### 2.5.2 Software Requirements

| Component | Specification | Description |
|---|---|---|
| Operating system | Window 10+, Linux(Ubuntu) | Provides a development and deployment environment. |
| Mobile development framework | React Native | Used to develop and test the mobile application. |
| Web development technologies | React | Used to build the web-based user interface. |
| Backend framework | Fast API | Implements business logic, APIs, and system services. |
| Database management system | MySQL | Stores user information, learning data, and transactions. |
| AI and Langchain libraries | LLM model | Used for speech recognition, grammar checking, and AI feedback. |
| Web server | Uvicorn | Hosts web application and backend services. |
| Version control system | Git, GitHub | Manages source code and team collaboration. |
| API testing tools | Postman | Used to test and validate backend APIs. |
| UI/UX design tool | Figma | Used for designing system interfaces and user experience. |
| Security tools | Authentication libraries | Ensures secure communication and data protection. |
| Client software | Web Browsers, Android OS | Enables users to access the system via web and mobile platforms. |

Table: Software Requirements

## 2.6 Key Abstraction with CRC Analysis

### 1. User Management Abstractions

**Authentication Principal (User)**

**Class: User Responsibilities:**

- Authenticate users (email/password)
- Manage role-based permissions (admin/student/tutor)
- Track account lifecycle (soft delete)
- Serve as central identity for all system actions

**Collaborators:**

- StudentProfile (for student details)
- Notification (receives alerts)
- BatchInstructor (as tutor role)
- Attendance (participates in batches)

**Learner Profile (StudentProfile)**

**Class: StudentProfile Responsibilities:**

- Store personalized learning preferences
- Generate AI learning plan
- Set learning goals and constraints
- Serve as root for personalized content tree

**Collaborators:**

- Module (has many personalized modules)
- FreeConversation (owns chat history)
- BatchEnrollment (joins batches)
- User (belongs to user)

### 2. Learning Content Abstractions

Curriculum Hierarchy (Module ---> Lesson ---> Interaction)

**Class: Module Responsibilities:**

- ❖ Organize lessons into logical units
- ❖ Control progression (locking/unlocking)
- ❖ Maintain display order
- ❖ Personalize to student profile

**Collaborators:**

- ❖ StudentProfile (owned by)
- ❖ Lesson (contains many)

**Class: Lesson Responsibilities:**

- ❖ Deliver content
- ❖ Track completion status
- ❖ Host structured AI interactions

**Collaborators:**

- ❖ Module (belongs to)
- ❖ Interaction (has many  questions and answers)

**Class: Interaction Responsibilities:**

- ❖ Record structured AI  questions and answers
- ❖ Track learning dialogue
- ❖ Support lesson-specific queries

**Collaborators:**

- ❖ Lesson (belongs to)

**AI Conversation Manager (FreeConversation)**

**Class: FreeConversation Responsibilities:**

- ❖ Manage unstructured AI dialogues
- ❖ Thread conversations
- ❖ Store chat history

**Collaborators:**

- ❖ StudentProfile (owned by)
- ❖ Self-referential (threaded conversations)

**3. Batch Management Abstractions**

**Batch system**

**Class: Batch Responsibilities:**

- Define class cohort (level, dates, capacity)
- Manage enrollment limits
- Track status (upcoming/active/completed)
- Set fee amount

**Collaborators:**

- BatchEnrollment (has many students)
- BatchInstructor (has many tutors)
- BatchSchedule (has many schedules)
- BatchCommunity (has forum)

**Scheduling System** (BatchSchedule ---> Schedule ---> Attendance)

**Class: BatchSchedule Responsibilities:**

- Link batches with time slots
- Coordinate course delivery schedule
- Serve as attendance session anchor

**Collaborators:**

- Batch (belongs to)
- Schedule (uses time template)
- Course (teaches specific course)
- Attendance (records presence)

**Class: Schedule Responsibilities:**

- Define reusable time slots
- Manage day/time patterns

**Collaborators:**

- BatchSchedule (used by)

**Class: Attendance Responsibilities:**

- ❖ Record class participation
- ❖ Track student/tutor presence
- ❖ Monitor punctuality

**Collaborators:**

- ❖ BatchSchedule (for specific session)
- ❖ User (who attended)

**Enrollment Management** (BatchEnrollment ---> Payment ---> Certificate)

**Class: BatchEnrollment Responsibilities:**

- ❖ Manage student enrollment lifecycle
- ❖ Track enrollment
- ❖ Link to payment and certification

**Collaborators:**

- ❖ StudentProfile (who enrolled)
- ❖ Batch (enrolled in)
- ❖ Payment (payment record)
- ❖ Certificate (completion proof)

**Class: Payment Responsibilities:**

- ❖ Process enrollment fees
- ❖ Track transaction status
- ❖ Store payment receipts

**Collaborators:**

- ❖ BatchEnrollment (for enrollment)

**Class: Certificate Responsibilities:**

- ❖ Generate completion certificates
- ❖ Store results and certificates

**Collaborators:**

- ❖ BatchEnrollment (certifies completion)

**4. Community & Support Abstractions**

**BatchCommunity**

**Class: BatchCommunity Responsibilities:**

- ❖ Facilitate group discussions
- ❖ Support threaded conversations
- ❖ Manage likes/dislikes
- ❖ Handle file attachments

**Collaborators:**

- ❖ Batch (belongs to cohort)
- ❖ User (posted by)
- ❖ CommunityAttachment (has attachments)

**Notification System**

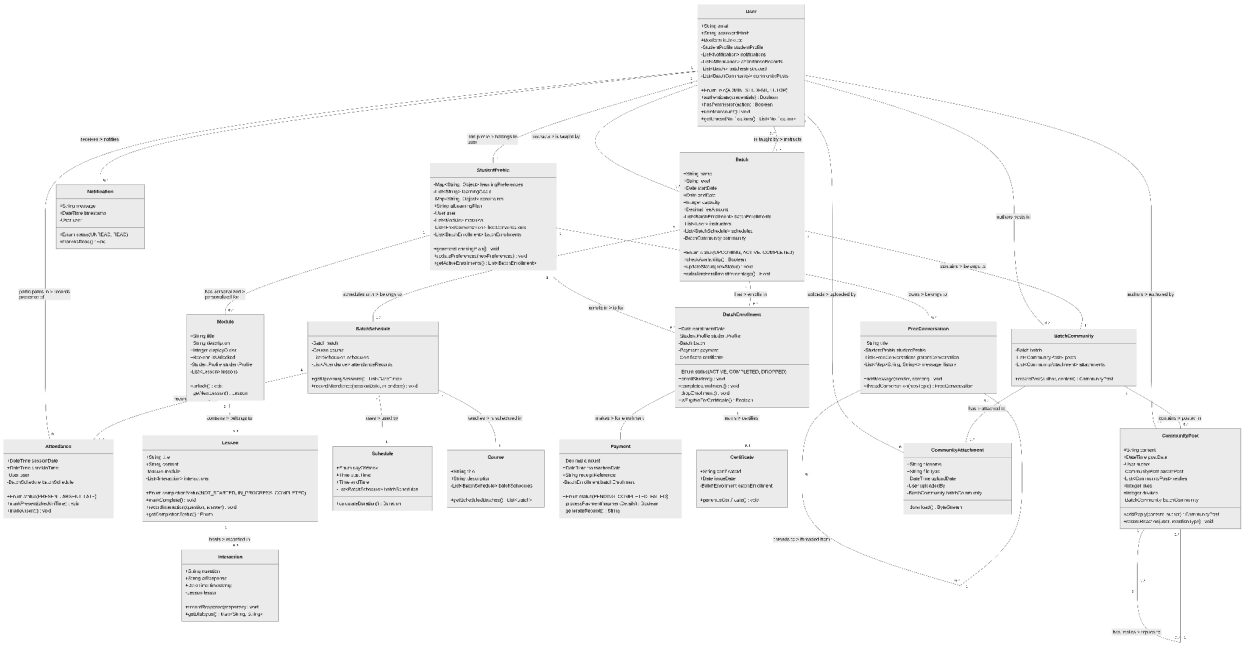
**Class: Notification Responsibilities:**

- ❖ Deliver message alerts
- ❖ Track read/unread status

**Collaborators:**

- ❖ User (notifies)

## 2.6.1 Conceptual Modeling: Class Diagram



**Relationship, Multiplicity and Role Analysis**

**1. User Management Relationships**

**User <---> StudentProfile**

- **Relationship:** One-to-One Association
- **Multiplicity:** User(1) ---> StudentProfile(0..1)
- **Roles:**
  - User ---> StudentProfile: "has profile"
  - StudentProfile ---> User: "belongs to user"

**User <---> Notification**

- **Relationship:** One-to-Many Association
- **Multiplicity:** User(1) ---> Notification(0..*)
- **Roles:**
  - User ---> Notification: "receives"
  - Notification ---> User: "notifies"

**2. Learning Content Relationships**

**StudentProfile <---> Module**

- **Relationship:** One-to-Many Composition

- **Multiplicity:** StudentProfile(1) ---> Module(0..*)
- **Roles:**
    - StudentProfile ---> Module: "has personalized"
    - Module ---> StudentProfile: "personalized for"

**Module <---> Lesson**

- **Relationship:** One-to-Many Composition (Lessons are part of Module)
- **Multiplicity:** Module(1) ---> Lesson(1..*)
- **Roles:**
    - Module ---> Lesson: "contains"
    - Lesson ---> Module: "belongs to"

**Lesson <---> Interaction**

- **Relationship:** One-to-Many Composition
- **Multiplicity:** Lesson(1) ---> Interaction(0..*)
- **Roles:**
    - Lesson ---> Interaction: "hosts"
    - Interaction ---> Lesson: "recorded in"

**StudentProfile <---> FreeConversation**

- **Relationship:** One-to-Many Composition
- **Multiplicity:** StudentProfile(1) ---> FreeConversation(0..*)
- **Roles:**
    - StudentProfile ---> FreeConversation: "owns"
    - FreeConversation ---> StudentProfile: "belongs to"

**FreeConversation <---> FreeConversation**

- **Relationship:** Recursive Association (Self-referential)
- **Multiplicity:** FreeConversation(1) ---> FreeConversation(0..*)
- **Roles:**
    - FreeConversation ---> FreeConversation: "threads to"
    - FreeConversation ---> FreeConversation: "threaded from"

**3. Batch Management Relationships**

**Batch <---> BatchEnrollment**

- **Relationship:** One-to-Many Association
- **Multiplicity:** Batch(1) ---> BatchEnrollment(0..*)

- **Roles:**
    - Batch ---> BatchEnrollment: "has"
    - BatchEnrollment ---> Batch: "enrolls in"

**Batch <---> User (as Instructor)**

- **Relationship:** Many-to-Many Association
- **Multiplicity:** Batch(0..*)* <---> *User(0..*)
- **Roles:**
    - Batch ---> User: "is taught by"
    - User ---> Batch: "instructs"

**Batch <---> BatchSchedule**

- **Relationship:** One-to-Many Association
- **Multiplicity:** Batch(1) ---> BatchSchedule(1..*)
- **Roles:**
    - Batch ---> BatchSchedule: "schedules with"
    - BatchSchedule ---> Batch: "belongs to"

**Batch <---> BatchCommunity**

- **Relationship:** One-to-One Composition
- **Multiplicity:** Batch(1) ---> BatchCommunity(1)
- **Roles:**
    - Batch ---> BatchCommunity: "contains"
    - BatchCommunity ---> Batch: "belongs to"

**BatchSchedule <---> Schedule**

- **Relationship:** Many-to-Many Association
- **Multiplicity:** BatchSchedule(1..*)* <---> *Schedule(1..***)
- **Roles:**
    - BatchSchedule ---> Schedule: "uses"
    - Schedule ---> BatchSchedule: "used by"

**BatchSchedule <---> Course**

- **Relationship:** Many-to-One Association
- **Multiplicity:** BatchSchedule(0..*) ---> Course(1)
- **Roles:**
    - BatchSchedule ---> Course: "teaches"

- Course ---> BatchSchedule: "is scheduled in"

**BatchSchedule <---> Attendance**

- **Relationship:** One-to-Many Association
- **Multiplicity:** BatchSchedule(1) ---> Attendance(0..*)
- **Roles:**
  - BatchSchedule ---> Attendance: "records"
  - Attendance ---> BatchSchedule: "belongs to"

**Attendance <---> User**

- **Relationship:** Many-to-One Association
- **Multiplicity:** Attendance(0..*) ---> User(1)
- **Roles:**
  - Attendance ---> User: "records presence of"
  - User ---> Attendance: "participates in"

**BatchEnrollment <---> StudentProfile**

- **Relationship:** Many-to-One Association
- **Multiplicity:** BatchEnrollment(0..*) ---> StudentProfile(1)
- **Roles:**
  - BatchEnrollment ---> StudentProfile: "is for"
  - StudentProfile ---> BatchEnrollment: "enrolls in"

**BatchEnrollment <---> Payment**

- **Relationship:** One-to-One Association
- **Multiplicity:** BatchEnrollment(1) ---> Payment(1)
- **Roles:**
  - BatchEnrollment ---> Payment: "makes"
  - Payment ---> BatchEnrollment: "for enrollment"

**BatchEnrollment <---> Certificate**

- **Relationship:** One-to-One Association
- **Multiplicity:** BatchEnrollment(1) ---> Certificate(0..1)
- **Roles:**
  - BatchEnrollment ---> Certificate: "earns"
  - Certificate ---> BatchEnrollment: "certifies"

**4. Community & Support Relationships**

**BatchCommunity <---> CommunityPost**

- **Relationship:** One-to-Many Composition (Posts are part of Community)
- **Multiplicity:** BatchCommunity(1) ---> CommunityPost(0..*)
- **Roles:**
    - BatchCommunity ---> CommunityPost: "contains"
    - CommunityPost ---> BatchCommunity: "posted in"

**BatchCommunity <---> CommunityAttachment**

- **Relationship:** One-to-Many Composition (Attachments are part of Community)
- **Multiplicity:** BatchCommunity(1) ---> CommunityAttachment(0..*)
- **Roles:**
    - BatchCommunity ---> CommunityAttachment: "has"
    - CommunityAttachment ---> BatchCommunity: "attached to"

**CommunityAttachment <---> User**

- **Relationship:** Many-to-One Association
- **Multiplicity:** CommunityAttachment(0..*) ---> User(1)
- **Roles:**
    - CommunityAttachment ---> User: "uploaded by"
    - User ---> CommunityAttachment: "uploads"

**CommunityPost <---> User**

- **Relationship:** Many-to-One Association
- **Multiplicity:** CommunityPost(0..*) ---> User(1)
- **Roles:**
    - CommunityPost ---> User: "authored by"
    - User ---> CommunityPost: "authors"

**CommunityPost <---> CommunityPost**

- **Relationship:** Recursive Association (Self-referential)
- **Multiplicity:**
    - Parent: CommunityPost(1) ---> Child: CommunityPost(0..*)
    - Child: CommunityPost(0..1) ---> Parent: CommunityPost(1)
- **Roles:**
    - Parent ---> Child: "has replies"
    - Child ---> Parent: "replies to"

**Class Relationship Summary Table**

| Relationship | Multiplicity | Role Description |
|---|---|---|
| User ---> StudentProfile | 1:1 | User has one StudentProfile |
| User ---> Notification | 1:* | User receives many Notifications |
| StudentProfile ---> Module | 1:* | StudentProfile has many Modules |
| Module ---> Lesson | 1:* | Module contains many Lessons |
| Lesson ---> Interaction | 1:* | Lesson hosts many Interactions |
| Batch ---> BatchEnrollment | 1:* | Batch has many Enrollments |
| Batch <---> User (Instructor) | *:* | Batch taught by many Users |
| BatchSchedule ---> Course | *:1 | BatchSchedule teaches one Course |
| BatchEnrollment ---> Payment | 1:1 | BatchEnrollment makes one Payment |
| BatchEnrollment ---> Certificate | 1:0..1 | BatchEnrollment may earn one Certificate |
| BatchCommunity ---> CommunityPost | 1:* | BatchCommunity contains many Posts |
| CommunityPost <---> CommunityPost | 1:* | Post can have many Replies |

## 2.6.2 Identifying Change Cases

The following outlines several potential change cases for the platform. High-impact opportunities include expanding to additional languages, integrating advanced AI capabilities (like emotion detection and predictive analytics), implementing adaptive learning paths using real-time performance data, and adopting premium speech-recognition APIs for improved accuracy. Medium-impact improvements involve

enabling third-party school management system integrations, adding gamification features to boost engagement, and supporting multi-teacher classroom environments. Most items have a medium likelihood of implementation, except multi-teacher support and premium speech APIs, which are considered less likely.

**1. Expansion to Other Languages**

- Extend the platform beyond English to support additional languages.
- **Likelihood:** Medium
- **Impact:** High

**2. Advanced AI Features**

- Integrate deeper AI capabilities such as emotion detection, predictive analytics, and deep learning-based recommendations.
- **Likelihood:** Medium
- **Impact:** High

**3. Adaptive Learning Paths based on real-time performance analytics**

- Dynamically adjust learning content and recommendations based on continuous performance data.
- **Likelihood:** Medium
- **Impact:** High

**4. Third-Party School Management System (SMS) Integration**

- Enable data sync with external institutional platforms (e.g., SMS, LMS).
- **Likelihood:** Medium
- **Impact:** Medium

**5. Multi - Teacher Classrooms**

- Support team-teaching models where multiple tutors co-host a live session.
- **Likelihood:** Low
- **Impact:** Medium

**6. Gamification**

- Incorporate game-like elements to boost engagement and motivation.
- **Likelihood:** Medium
- **Impact:** Medium

### 7. Premium Speech Recognition APIs (Google Speech, Amazon Transcribe)

- Replace or supplement free speech recognition with commercial APIs (Google Speech, Amazon Transcribe) for higher accuracy.
- **Likelihood:** Low
- **Impact:** High

# Chapter Three: System Design

## 3.1 Architectural Design

### 3.1.1 Component modeling

Component modeling describes the major functional units of the proposed system and how they interact with one another. The **FidelAI (Learning Language with AI)** platform, integrated with an Online English Learning System, is designed using a **modular, service-oriented architecture** to ensure scalability, maintainability, and ease of integration.

The system is composed of the following main components:

#### a) Web Application Component

This component provides browser-based access for students, instructors, and administrators. It supports:

- Course management
- Lesson delivery
- Assessments and evaluations
- Communication and collaboration
- Progress monitoring
- Administrative operations

#### b) Mobile Application Component

The mobile application enables learners to:

- Access learning materials on smartphones and tablets
- Perform AI-based speaking and writing practice
- Receive instant feedback
- Track learning progress

It enhances learning flexibility and supports **offline content caching**.

#### c) Application Server Component

This component implements the core business logic of the system. Its responsibilities include:

- User authentication and authorization
- Course enrollment management
- Assessment processing
- Progress tracking
- Notification management
- Secure communication with the AI engine via APIs

### d) AI Engine (FidelAI) Component

The AI engine delivers intelligent language-processing services, including:

- Grammar correction
- Vocabulary evaluation
- Personalized learning recommendations

This component enables autonomous learning without continuous tutor involvement.

### e) Database Component

The database stores and manages:

- User profiles
- Learning content
- Assessment results
- Progress records
- Business rules
- Transaction data

It ensures **data integrity, consistency, and reliable retrieval**.

### f) Communication and Notification Component

This component manages both real-time and asynchronous communication, including:

- In-app messaging
- Email notifications
- SMS alerts
- System announcements

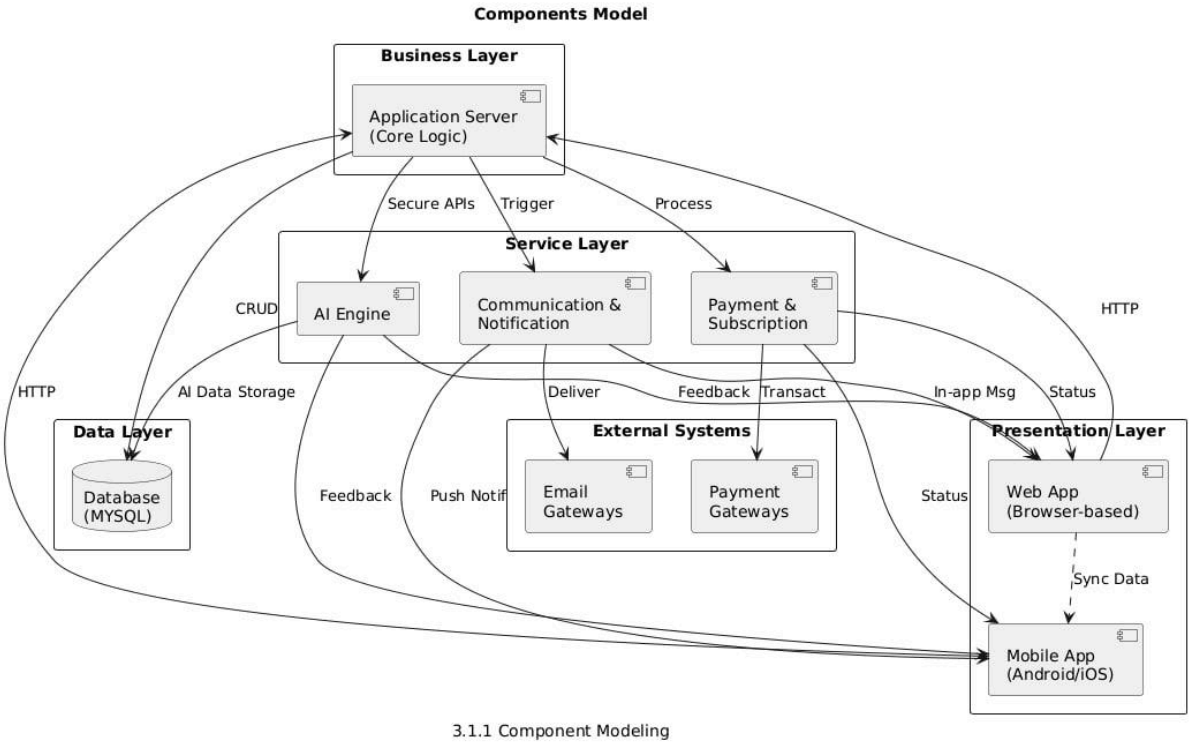
### g) Payment and Subscription Component

This component handles:

- Course payments

- ❖ Subscription management
- ❖ Transaction validation
- ❖ Receipt generation

All financial operations are conducted using **secure online payment gateways**.




3.1.1 Component Modeling

## 3.1.2 Deployment Modeling

Deployment modeling describes the physical distribution of system components across the hardware and network infrastructure. The proposed system adopts a **multi-tier deployment architecture** to ensure high availability, optimal performance, and robust security.

### Client Tier

This tier represents end-user access points and includes:

- ❖ Web browsers running on desktops and laptops
- ❖ Mobile devices operating on Android or iOS

All client devices communicate with backend services using **secure HTTPS connections**.

### Web Server Tier

The web server is responsible for:

- Hosting the web-based user interface
- Serving static resources (e.g., HTML, CSS, JavaScript, media files)
- Forwarding service requests to the application server

### Application Server Tier

The application server executes the core business logic of the system. Its responsibilities include:

- Handling API requests
- Enforcing business rules
- Managing user authentication and authorization
- Coordinating communication between users, the database, and the AI engine

### AI Processing Server Tier

This tier hosts the **FidelAI** models and natural language processing services. It performs:

- Speech recognition and pronunciation analysis
- Text evaluation and grammar assessment
- Feedback generation and learning recommendations

### Database Server Tier

The database server stores all persistent system data, including:

- User profiles and authentication data
- Learning content and course materials
- Assessment results and progress records

It is secured using access control mechanisms and regular backup processes to ensure data integrity and availability.

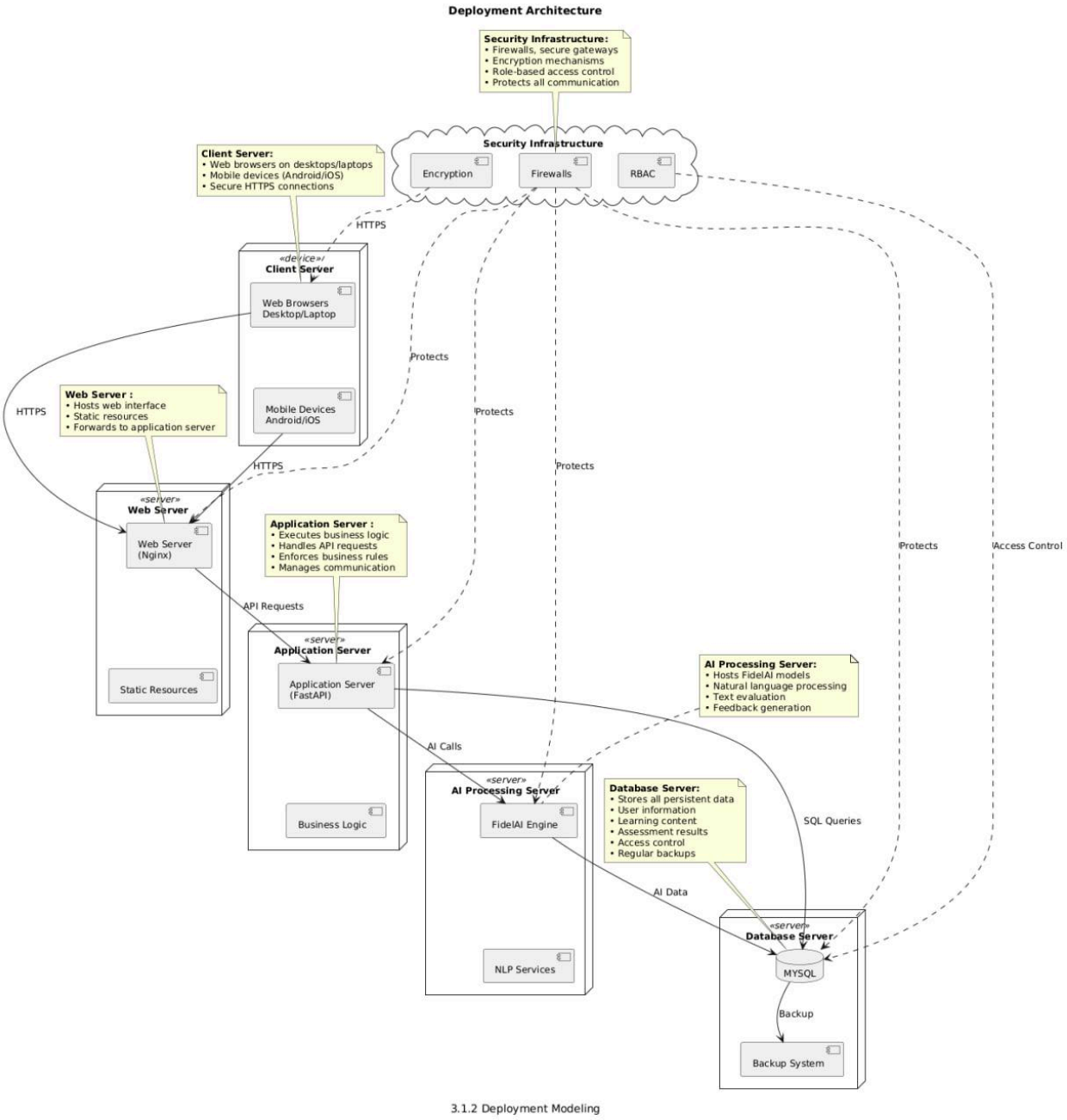
### Network and Security Infrastructure

This layer provides system-wide protection and includes:

- Firewalls and secure gateways
- Encryption mechanisms for data in transit

- ❖ Role-based access control (RBAC) to restrict access to sensitive resources

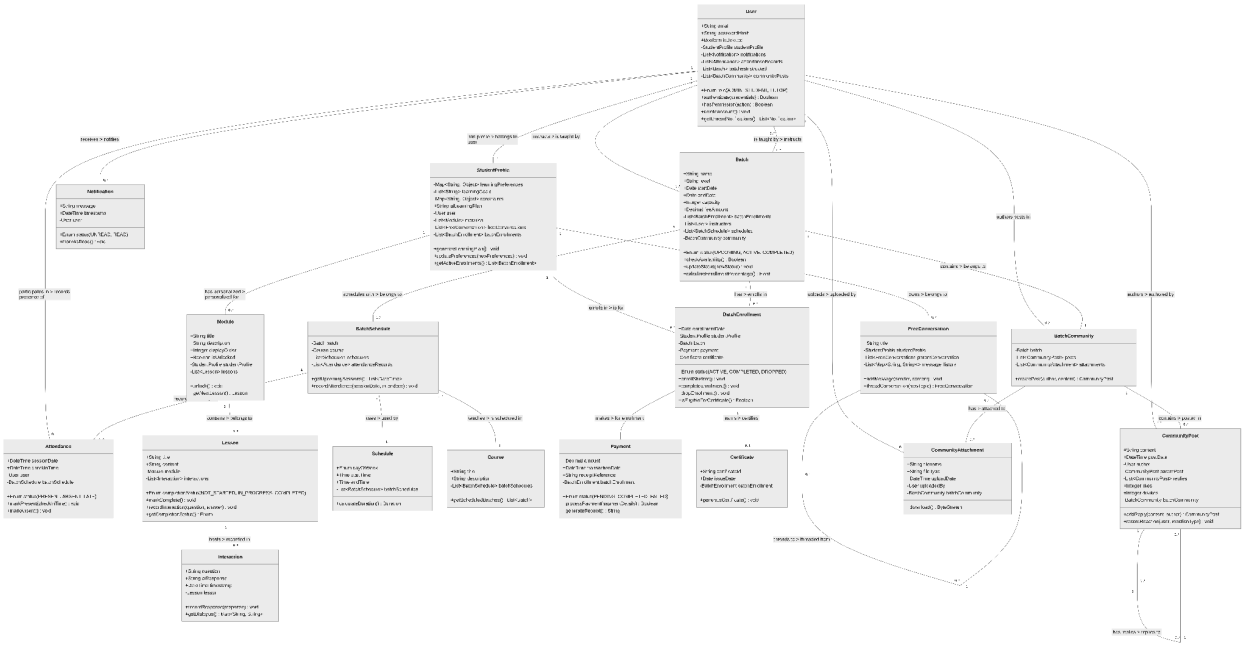
This deployment architecture ensures scalability, reliability, and secure operation of the AI-integrated online English learning system.




3.1.2 Deployment Modeling

## 3.2 Detail Design

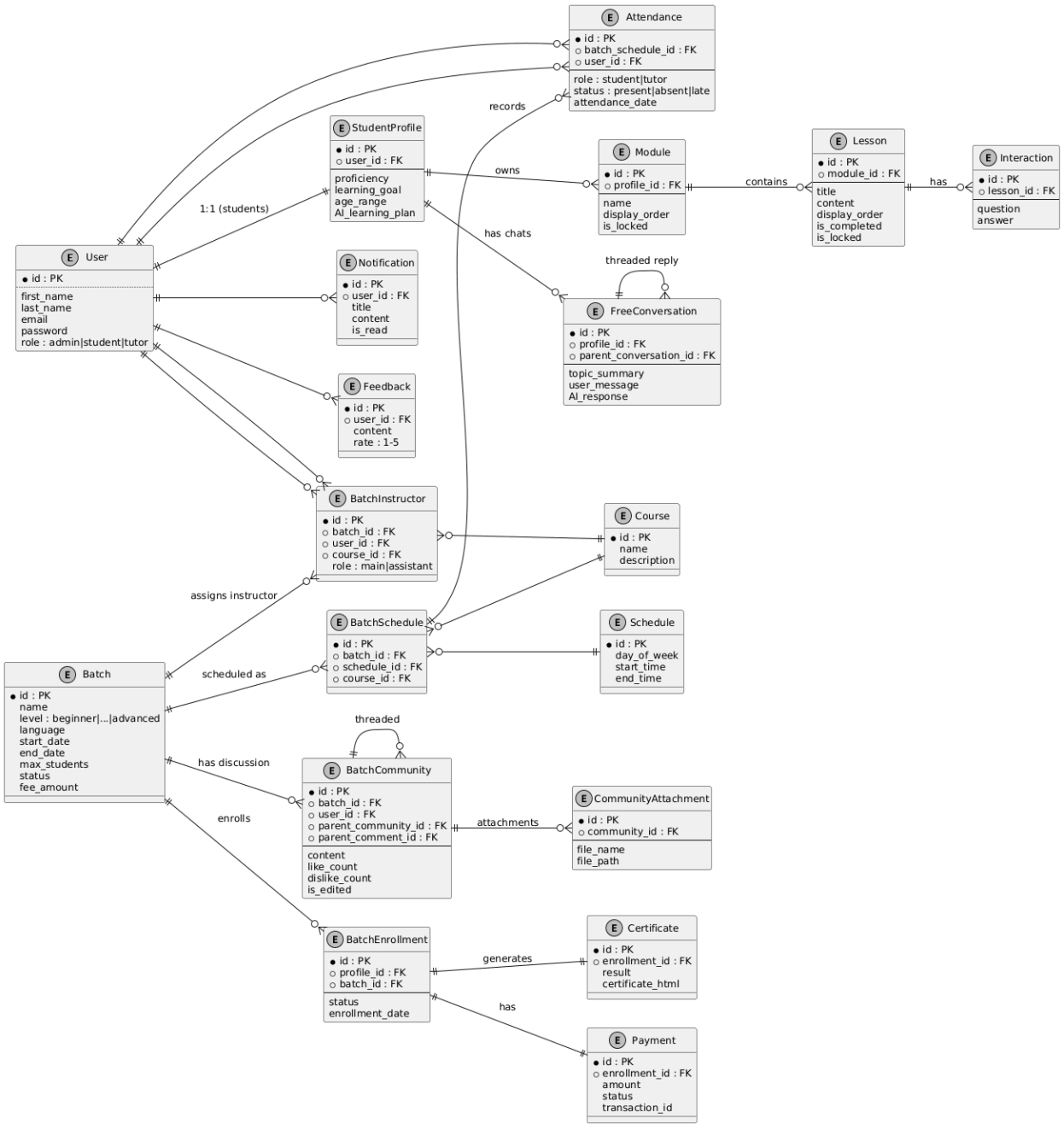
### 3.2.1 Design class model



- **User:** Represents a system account that can act as an admin, student, or tutor. Handles authentication, authorization, account lifecycle, and serves as the central link to profiles, notifications, attendance, teaching responsibilities, and community participation.
- **StudentProfile:** Stores all student-specific learning data and personalization. Manages learning preferences, goals, constraints, AI-generated learning plans, enrolled batches, modules, and free-form AI conversations tied to a single user.
- **Notification:** Represents system-generated messages sent to users. Tracks message content, delivery time, and read/unread status, allowing users to manage and acknowledge notifications.
- **Attendance:** Captures a user's attendance for a scheduled batch session. Tracks presence status, session date, and check-in time, and is linked to both the user and the batch schedule.
- **Module:** Defines a structured learning unit within a student's personalized curriculum. Controls lesson sequencing, unlock logic, and progression through associated lessons.
- **Lesson:** Represents an individual instructional unit within a module. Stores lesson content, completion status, and learning interactions between the student and the AI.

- **Interaction:** Records a single learning exchange within a lesson. Stores learner questions, AI responses, timestamps, and allows reconstruction of the dialogue history.
- **FreeConversation:** Supports open-ended, unstructured conversations between a student and the AI. Allows threaded discussions, message history tracking, and topic branching independent of formal lessons.
- **Batch:** Represents a scheduled cohort-based course offering. Manages enrollment capacity, lifecycle status, instructors, schedules, fees, and its associated community space.
- **BatchSchedule:** Defines how a batch is delivered over time. Links courses to concrete session schedules and maintains attendance records for each session.
- **Schedule:** Represents a reusable time slot definition. Defines recurring day-of-week and time ranges used by batch schedules.
- **Course:** Defines the academic or instructional content offered across batches. Acts as a template that can be scheduled multiple times via different batch schedules.
- **BatchEnrollment:** Represents a student's participation in a specific batch. Tracks enrollment status, payment, completion, and eligibility for certification.
- **Payment:** Handles financial transactions related to batch enrollment. Tracks payment amount, status, transaction details, and receipt generation.
- **Certificate:** Represents proof of successful batch completion. Issued upon meeting completion criteria and linked directly to a batch enrollment.
- **BatchCommunity:** Provides a collaborative discussion space for a batch. Manages posts and shared attachments associated with the batch.
- **CommunityPost:** Represents a discussion post within a batch community. Supports threaded replies, reactions (likes/dislikes), and author tracking.
- **CommunityAttachment:** Represents files shared within a batch community. Stores metadata about uploaded resources and provides download access.

## 3.2.2 Persistent model




### Core Identity & Access

**User:** Represents any system actor (student, instructor, admin).

- ❖ Stores authentication and authorization data (email, password hash, role).
- ❖ Can optionally own a StudentProfile.
- ❖ Can instruct multiple Batches, receive Notifications, participate in Attendance, and author CommunityPosts.

## Personalized Learning

**StudentProfile:** Represents a learner's personalized AI-driven learning context.

- ❖ Contains learning preferences, goals, constraints, and an AI-generated learning plan.
- ❖ Owns:
  - ➢ Modules (structured learning paths),
  - ➢ FreeConversations (open AI chat sessions),
  - ➢ BatchEnrollments (formal course participation).
- ❖ Linked one-to-one with a User.

**Module:** A structured learning unit within a student's personalized plan.

- ❖ Ordered and unlockable.
- ❖ Contains multiple Lessons.
- ❖ Scoped to a single StudentProfile.

**Lesson:** Represents a single learning unit.

- ❖ Tracks completion status.
- ❖ Contains multiple Interactions with the AI.

**Interaction:** Stores AI-student exchanges within a lesson.

- ❖ Includes question, AI response, and timestamp.
- ❖ Provides conversational learning traceability.

**FreeConversation:** Unstructured AI chat outside formal lessons.

- ❖ Belongs to a StudentProfile.
- ❖ Supports hierarchical (parent/child) conversations.
- ❖ Stores full message history.

## Batch-Based Learning

**Batch:** Represents a cohort-based course offering.

- ❖ Defined by level, duration, capacity, fee, and status.
- ❖ Taught by one or more Users (instructors).
- ❖ Owns:
  - ➢ BatchSchedules
  - ➢ BatchEnrollments

- ➢ A dedicated BatchCommunity

**Course:** Defines reusable academic content.

- ❖ Can be scheduled across multiple batches via BatchSchedule.

**BatchSchedule:** Connects a Batch to a Course and time schedules.

- ❖ Tracks attendance per session.
- ❖ Uses one or more Schedule definitions.

**Schedule:** Defines recurring time slots (day and time).

- ❖ Reusable across multiple batch schedules.

**Attendance:** Tracks user participation in scheduled sessions.

- ❖ Linked to a User and a BatchSchedule.
- ❖ Includes session date, check-in time, and attendance status.

## Enrollment, Payments & Certification

**BatchEnrollment:** Represents a student's enrollment in a batch.

- ❖ Links StudentProfile and Batch.
- ❖ Tracks enrollment status and date.
- ❖ Owns:
  - ➢ Payment
  - ➢ Optional Certificate

**Payment:** Stores transaction details for a batch enrollment.

- ❖ Includes amount, status, date, and receipt reference.

**Certificate:** Issued upon successful batch completion.

- ❖ Tied one-to-one with a BatchEnrollment.

## Communication & Community

**Notification:** System messages sent to users.

- ❖ Tracks message content, timestamp, and read status.

**BatchCommunity:** Dedicated community space for a batch.

- ❖ Hosts CommunityPosts and CommunityAttachments.

**CommunityPost:** Represents discussion posts and replies.

- ❖ Supports nested replies (threaded discussions).
- ❖ Tracks likes and dislikes.
- ❖ Authored by a User and scoped to a BatchCommunity.

**CommunityAttachment:** Files shared within a batch community.

- ❖ Uploaded by a User.
- ❖ Linked to a BatchCommunity.

## 3.3 User Interface Design

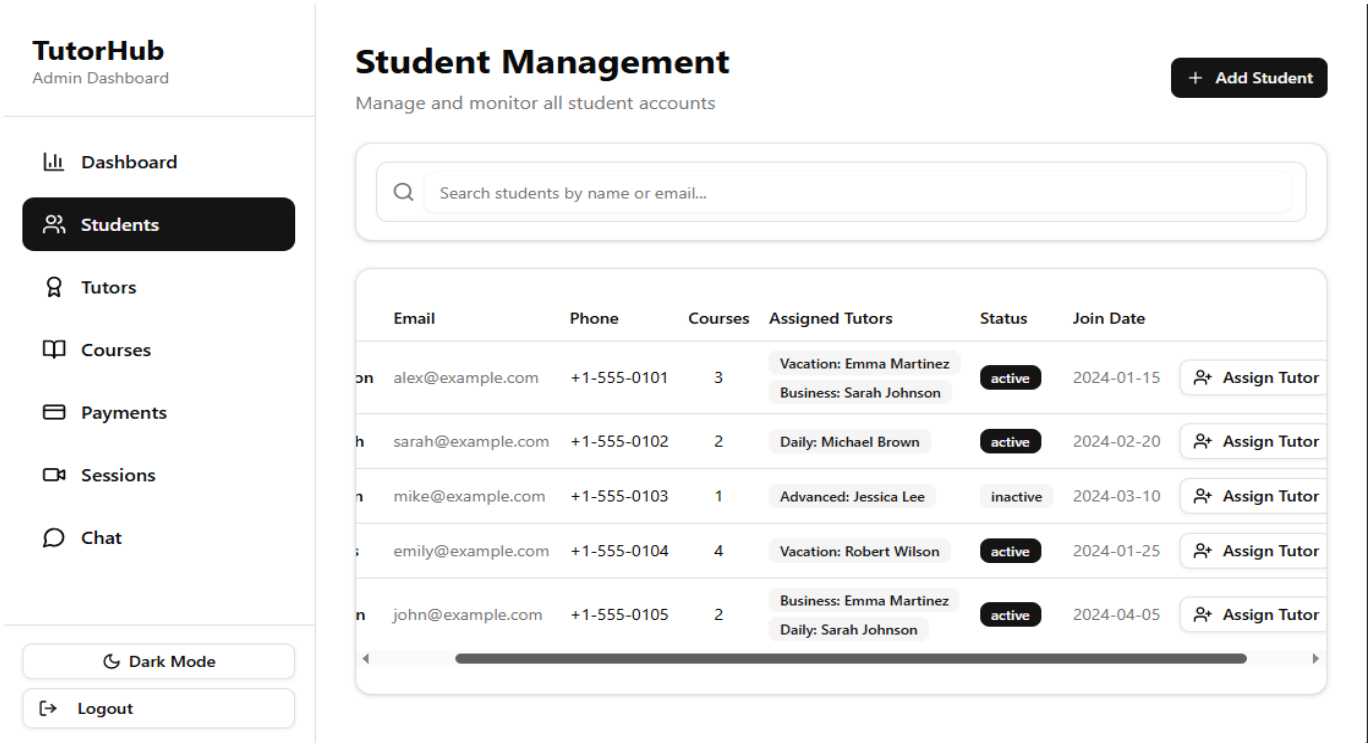



Figure 27 Admin dashboard

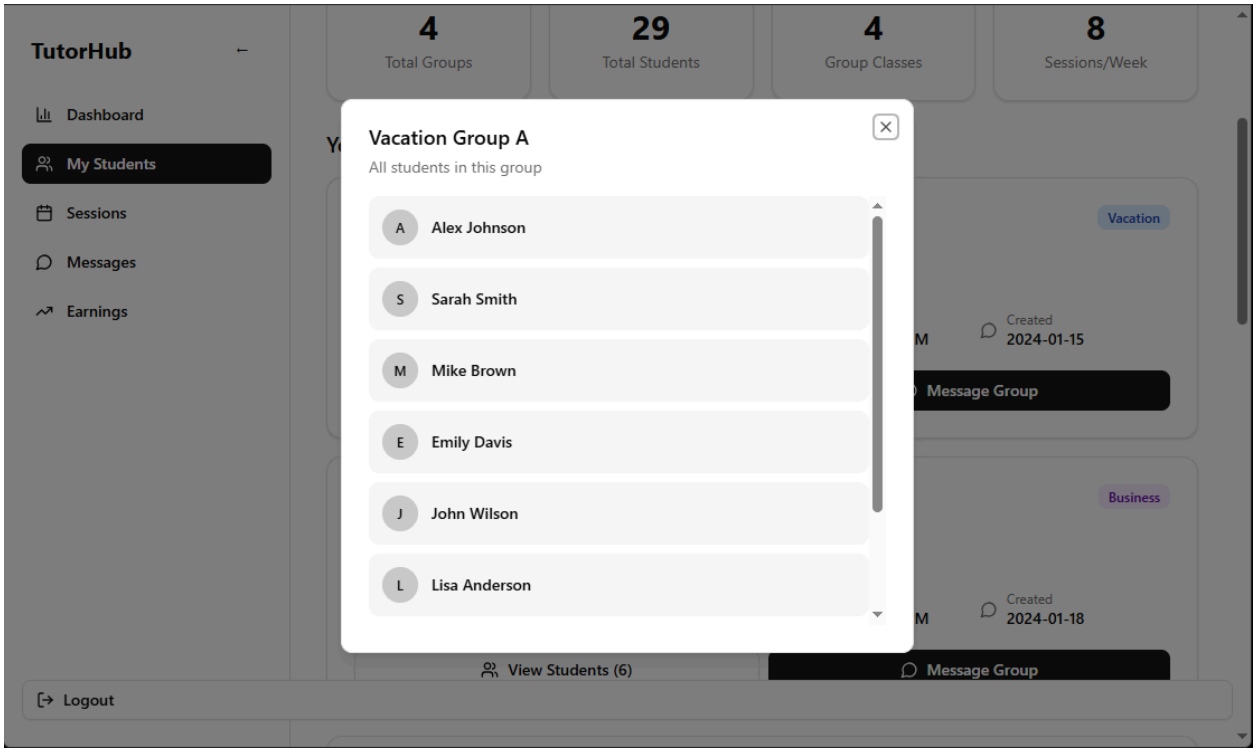



Figure 28 Tutor dashboard

## 3.4 Access control and security

The system enforces Role-Based Access Control (RBAC**)** to ensure that each actor can access only the functionality and data necessary for their responsibilities. This approach enhances security, prevents misuse, and ensures data privacy across the platform.

### User Roles and Access Levels

The system defines three primary user roles:

#### a) Student

Students are end users who consume learning services.

#### Permitted Access

- Register, log in, and manage their own profile
- View and update personal learning goals
- Access AI-generated learning plans, lessons, quizzes, and exercises

- Participate in assigned batches and live classes
- Communicate within authorized batch communities
- Make payments and view their own payment history
- View certificates issued to them
- Submit feedback and receive system notifications

**Restricted Access**

- Cannot access other students' data
- Cannot modify learning content, batch configurations, or system settings
- Cannot access administrative reports or tutor earnings

**b) Tutor**

Tutors provide instructional services and interact with assigned students.

**Permitted Access**

- Register and manage their own tutor profile
- Define availability and view assigned schedules
- Access only the batches and students assigned to them
- Conduct live classes and mark attendance
- Upload supplementary learning materials (subject to administrator approval)
- View their own earnings and reviews

**Restricted Access**

- Cannot manage payments, users, or system configurations
- Cannot access student data outside their assigned batches
- Cannot approve content or issue certificates

**c) Administrator**

Administrators have full system oversight and governance authority.

**Permitted Access**

- Manage all users (students and tutors): create, update, suspend, and deactivate
- Approve or reject tutor registrations
- Create, update, and archive courses, batches, and learning content
- Assign tutors to batches
- Monitor system activity and generate reports
- View and manage all payment transactions

- Issue and verify certificates
- Send system-wide or targeted notifications
- Moderate community content and user feedback

**Restricted Access**

- No functional restrictions within the system scope (highest-privilege role)

## Authentication and Authorization

- All users must authenticate using valid credentials (email or phone number and password).
- Passwords are securely stored using industry-standard hashing algorithms.
- After successful authentication, permissions are assigned based on the user's role.

**Authorization checks are enforced at:**

- Application level (API and UI access control)
- Database level (role-validated queries)

## Data Access Control

- Users can access only their own data unless explicitly authorized by their role.
- Tutors can access student data only within their assigned batches.
- Administrators have full read and write access to all system data.
- Sensitive data (e.g., payments, personal information, voice recordings) is protected and never exposed to unauthorized roles.

## Account Status Control

Each user account operates under one of the following statuses:

- **Active** – Full access according to assigned role
- **Suspended** – Login permitted, but access to learning and payment features is restricted
- **Deactivated** – No system access

Only administrators are authorized to change account status.

**Security Mechanisms**

The system implements multiple security measures, including:

- Limited login attempts to prevent brute-force attacks
- Temporary account locking after multiple failed login attempts
- Secure communication using HTTPS
- Role validation on every API request
- Audit logs for critical actions (e.g., payments, content updates, account status changes)
- Regular data backups with restricted restoration access (administrator-only)

# References


1. *FastAPI, "FastAPI documentation: Python web framework for building APIs," 2020. [Online]. Available: https://fastapi.tiangolo.com*
2. *React Native, "React Native: Create native apps for Android and iOS using React," 2020. [Online]. Available: https://reactnative.dev*
3. *LangChain, "LangChain documentation: Building applications with LLMs," 2023. [Online]. Available: https://python.langchain.com*
4. *Jitsi, "Jitsi Meet API documentation," 2020. [Online]. Available: https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-iframe*
5. *Chapa, "Chapa API documentation: Ethiopian payment gateway," 2020. [Online]. Available: https://developer.chapa.co*

# Appendix

**Age group:**

- Under 18
- 18–25
- 26–35
- 36+

**Current English proficiency level:**

- Beginner
- Intermediate
- Advanced

**What are the main difficulties you face when learning English online?**

- Lack of motivation
- Limited tutor access
- Payment issues

**Rank the following features by importance:**

- Live tutor sessions
- Community