

# Tutor Support System at HCMUT

## Assignment's Specification

Ho Chi Minh City University of Technology (HCMUT)

Faculty of Computer Science and Engineering

September 2025



*Version 1.0*

Instructor: Dr. Truong Thi Thai Minh

Student: Tran Huu Hoang Long 2352704

Doan Anh Khoi 2352601

Arthur Bardot 2460080

Nguyen Manh Quoc Khanh 2352525

Pham Quang Tuan 2353275

Phan Ngoc Lan Chi 2352137

Truong Quoc Thai 2353094

*This specification follows the structure and conventions used in prior course specifications, adapted to the Tutor Support System context.*

# Contents

<b>1</b>	<b>Assignment's outcome</b>	<b>3</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>Description</b>	<b>3</b>
3.1	Stakeholders, Roles, and Expectations . . . . .	3
3.1.1	Stakeholders . . . . .	3
3.1.2	Roles . . . . .	3
3.1.3	Expectations . . . . .	3
3.2	Objectives and Scope . . . . .	3
<b>4</b>	<b>Functional Requirements</b>	<b>4</b>
4.1	Functional Requirements List . . . . .	4
4.2	User & Information Management (FR-UM) . . . . .	4
4.3	Tutor-Student Matching (FR-MAT) . . . . .	5
4.4	Session & Scheduling Management (FR-SCH) . . . . .	6
4.5	Feedback & Progress Tracking (FR-FBK) . . . . .	7
4.6	Reporting & Analytics (FR-RPT) . . . . .	7
4.7	Integration with HCMUT Infrastructure (FR-INT) . . . . .	8
4.8	Advanced / Optional Features (FR-ADV) . . . . .	9
4.9	Non-interactive Functional Requirements (FR-NI) . . . . .	9
<b>5</b>	<b>Non-Functional Requirements</b>	<b>11</b>
5.1	Performance Requirements . . . . .	11
5.2	Security & Reliability Requirements . . . . .	11
5.3	Usability & Accessibility Requirements . . . . .	11
5.4	Software Quality Attributes . . . . .	11
5.5	Business Rules . . . . .	12
<b>6</b>	<b>Use-Case View</b>	<b>12</b>
6.1	General Use-Case Diagram . . . . .	12
6.2	UC-01 Log in & Profile Management . . . . .	13
6.3	UC-02 Tutor-Student Matching . . . . .	15
6.4	UC-03 Session Scheduling Management . . . . .	17
6.5	UC-04 Feedback & Progress Tracking . . . . .	20
6.6	UC-05 Reporting & Analytics . . . . .	24
6.7	UC-06 Integration with HCMUT Infrastructure . . . . .	28
6.8	UC-07 Advanced / Optional Features . . . . .	31
<b>7</b>	<b>System Integration</b>	<b>32</b>
<b>8</b>	<b>Coding Rules and Constraints</b>	<b>32</b>
<b>9</b>	<b>Submission and Deliverables</b>	<b>33</b>
<b>10</b>	<b>Other Regulations</b>	<b>33</b>



## 1 Assignment's outcome

Upon completion of this assignment, students will be able to:

- Identify stakeholders, roles, objectives, and scope for a university-scale information system.
- Specify functional and non-functional requirements clearly and traceably.
- Model core use-cases for tutoring workflows (registration, matching, booking, feedback).
- Define system integration touchpoints with HCMUT infrastructure (SSO, Data-Core, Library).
- Produce a consistent, submission-ready specification document in L<sup>A</sup>T<sub>E</sub>X.

## 2 Introduction

HCMUT requires a platform to support students in their academic and skills development journey. The **Tutor Support System (TSS)** will modernize the management of the Tutor/Mentor program, enabling scalable operations and data-driven improvement across departments.

## 3 Description

### 3.1 Stakeholders, Roles, and Expectations

#### 3.1.1 Stakeholders

Stakeholders are individuals or entities who can affect or are affected by the project outcomes.

**Primary:** HCMUT staff (customer), course instructors (project managers), development team, designers.

**Secondary:** Students (end-users), tutors, government, competitors.

Stakeholders may be internal (within HCMUT: Office of Academic Affairs, Office of Student Affairs, departments) or external (students, tutors).

#### 3.1.2 Roles

Roles are permissions and capabilities within the system: Student, Tutor, Coordinator, Department Chair, Program Administrator.

#### 3.1.3 Expectations

Each role expects secure access, clear workflows, and reliable performance. Students expect easy registration and booking; tutors expect manageable scheduling; administrators expect robust reporting.

### 3.2 Objectives and Scope

**Objectives** Design and develop an efficient, secure, and scalable software supporting the Tutor/Mentor program:

- Manage tutor/student information (profiles, expertise, support needs).
- Enable registration, selection or automated matching.
- Support scheduling, booking, cancellation, rescheduling (online or in-person).
- Provide progress tracking, feedback, evaluation.
- Generate reports for departments and offices to optimize resources and recognition.
- Integrate with HCMUT SSO/DataCore/Library for consistency and security.

**Scope** This specification covers:

- Core features: profile management, matching, scheduling, notifications, feedback, reporting.
- Integrations: HCMUT\_SSO (auth), HCMUT\_DATACORE (personal data), HCMUT\_LIBRARY (learning resources).
- Roles: Student, Tutor, Coordinator, Department Chair, Program Admin (RBAC).

**Out of Scope (MVP):** Full production DB, advanced AI features (smart matching, personalization), external integrations beyond HCMUT.

## 4 Functional Requirements

### 4.1 Functional Requirements List

The following table summarizes the functional requirements of the Tutor Support System. Requirements are grouped into thematic categories to ensure clarity and traceability.

**Prioritization Method:** In this project, we applied the MoSCoW prioritization technique to classify functional requirements. This method categorizes requirements into four levels:

- **Must:** Essential for the system to function; without them, the system fails to meet its objectives.
- **Should:** Important but not vital; the system can still operate without them in the first release.
- **Could:** Desirable enhancements that improve usability or efficiency if time/resources allow.
- **Won't (this time):** Explicitly excluded from the current scope, possibly considered for future releases.

This approach ensures clarity in requirement importance and helps manage project scope effectively.

### 4.2 User & Information Management (FR-UM)

- **FR-UM.01 – Profile**
  - *Description:* The system shall allow students and tutors to view and update their personal profiles, with core information (name, student ID, email, role, faculty/major) synchronized from the university's database.
  - *Acceptance Criteria:*

- \* Profiles automatically include core fields (name, student ID, email, role, faculty/major) synced from the university database; users are not required to manually enter these fields.
- \* Profile changes are timestamped and stored.
- *Priority*: Must
- **FR-UM.02 – Role-based Access Control**
  - *Description*: The system shall enforce role-based access control to regulate permissions for students, tutors, coordinators, department heads, and administrators.
  - *Acceptance Criteria*:
    - \* Each role has defined permissions.
    - \* Unauthorized access attempts are logged.
  - *Priority*: Must

### 4.3 Tutor–Student Matching (FR-MAT)

- **FR-MAT.01 – Manual Tutor Selection**
  - *Description*: The system shall allow students to register for the tutoring program.
  - *Acceptance Criteria*:
    - \* Students can successfully submit a registration request to join the tutoring program.
  - *Priority*: Must
- **FR-MAT.02 – Manual Tutor Selection**
  - *Description*: The system shall allow students to search for and manually select tutors based on expertise, availability, and preferences. Core tutor information (subject, department, schedule) is synchronized from the university database, while teaching preferences are provided by tutors.
  - *Acceptance Criteria*:
    - \* Students can filter tutors by at least three criteria (e.g., subject, availability, preferences).
    - \* Selection creates a pending match awaiting tutor confirmation.
  - *Priority*: Must
- **FR-MAT.03 – Smart Matching**
  - *Description*: The system shall provide automated tutor–student matching using predefined criteria such as subject, availability, and tutor workload. Matching relies on synchronized data from DATACORE combined with tutor-specified preferences.
  - *Acceptance Criteria*:
    - \* System generates a ranked list of tutors with explanation of matching factors.
    - \* Confirmation from both tutor and student finalizes the match.

- *Priority*: Should
- **FR-MAT.04 – Coordinator Assignment**
  - *Description*: The system shall allow coordinators, department chairs, or administrators to manually assign tutors to students when necessary, overriding automated or student-selected matches.
  - *Acceptance Criteria*:
    - \* Only authorized roles can assign tutors.
    - \* Manual assignment overrides previous matches.
    - \* Assignment details (who, when, reason) are logged and traceable.
  - *Priority*: Must

#### 4.4 Session & Scheduling Management (FR-SCH)

- **FR-SCH.01 – Tutor Availability**
  - *Description*: The system shall allow tutors to set and manage their availability for consultation sessions, synchronized with official university timetables where applicable.
  - *Acceptance Criteria*:
    - \* Only tutors can create, edit, and delete available slots.
    - \* The system prevents overlapping slots.
    - \* Slots cannot conflict with official class schedules imported from DATA-CORE.
  - *Priority*: Must
- **FR-SCH.02 – Session Booking**
  - *Description*: The system shall allow students to book in-person or online sessions with tutors based on available slots.
  - *Acceptance Criteria*:
    - \* Booking is allowed only within available tutor slots.
    - \* The system prevents double-booking of the same slot.
  - *Priority*: Must
- **FR-SCH.03 – Session Modification**
  - *Description*: The system shall allow students to cancel or reschedule booked sessions.
  - *Acceptance Criteria*:
    - \* Cancellation and rescheduling must follow configured rules (e.g., at least 2 hours before session start).
    - \* The system ensures new booking adheres to availability and no conflicts.
  - *Priority*: Must
- **FR-SCH.04 – Notifications & Reminders**
  - *Description*: The system shall automatically send notifications and reminders for upcoming sessions or schedule changes.

- *Acceptance Criteria:*
  - \* Notification sent immediately upon booking, cancellation, or reschedule.
  - \* Reminder sent at least 24h and 1h before session start.
- *Priority:* Must

## 4.5 Feedback & Progress Tracking (FR-FBK)

### • FR-FBK.01 – Session Feedback

- *Description:* The system shall enable students to provide structured feedback for each completed session.
- *Acceptance Criteria:*
  - \* Feedback form is available only after session completion.
  - \* Each student can submit one feedback entry per session.
  - \* Feedback is linked to session ID and timestamped.
- *Priority:* Must

### • FR-FBK.02 – Progress Recording

- *Description:* The system shall allow tutors to record mentee progress and generate optional summaries after sessions.
- *Acceptance Criteria:*
  - \* Only tutors can log progress, which is linked to session ID.
  - \* Summaries may include text notes and optional attachments.
  - \* All records are timestamped and stored for reporting.
- *Priority:* Should

## 4.6 Reporting & Analytics (FR-RPT)

### • FR-RPT.01 – Departmental Reports

- *Description:* The system shall generate reports for academic departments to monitor student learning performance.
- *Acceptance Criteria:*
  - \* Reports include attendance, performance indicators, and session counts.
  - \* Data exportable to CSV/PDF.
- *Priority:* Should

### • FR-RPT.02 – Academic Affairs Overview

- *Description:* The system shall provide overview reports for the Office of Academic Affairs to optimize resource allocation.
- *Acceptance Criteria:*
  - \* Reports show tutor workload distribution and student demand trends.
  - \* Dashboards update with latest synced data.
- *Priority:* Should

### • FR-RPT.03 – Student Affairs Outcomes



- *Description*: The system shall provide summarized participation data for the Office of Student Affairs to support training credits and scholarship considerations.
- *Acceptance Criteria*:
  - \* Reports list eligible students based on configured rules.
  - \* Calculation rules are transparent and logged.
- *Priority*: Should

## 4.7 Integration with HCMUT Infrastructure (FR-INT)

### • FR-INT.01 – HCMUT\_SSO Integration

- *Description*: The system shall integrate with HCMUT\_SSO for unified authentication.
- *Acceptance Criteria*:
  - \* Only valid SSO accounts can log in.
  - \* Single sign-out follows HCMUT SSO rules.
- *Priority*: Must

### • FR-INT.02 – DATACORE Synchronization

- *Description*: The system shall synchronize core personal and academic data from HCMUT\_DATACORE.
- *Acceptance Criteria*:
  - \* Sync occurs periodically or near real-time.
  - \* Conflicts resolved with DATACORE as source of truth.
- *Priority*: Must

### • FR-INT.03 – Role assignment

- *Description*: The system shall automatically assign roles (student, tutor, coordinator, department chair, administrator) based on centralized HCMUT role data.
- *Acceptance Criteria*:
  - \* System assigns roles upon login via SSO.
  - \* Role updates in DATACORE are reflected in the system within defined sync intervals.
- *Priority*: Must

### • FR-INT.04 – Library Resource Linking

- *Description*: The system shall connect with HCMUT\_LIBRARY to allow tutors and students to share relevant materials.
- *Acceptance Criteria*:
  - \* Users can attach library resources to sessions or summaries.
  - \* Access permissions follow HCMUT library rules.
- *Priority*: Could

## 4.8 Advanced / Optional Features (FR-ADV)

These features are not mandatory for the MVP but can enhance the Tutor Support System if resources permit:

- **FR-ADV.01 – Intelligent Matching (AI Integration)**
  - *Description:* The system may leverage AI techniques to optimize tutor–student pairing by analyzing multiple factors such as performance history, learning style, and tutor workload.
  - *Acceptance Criteria:*
    - \* AI suggestions ranked with justification.
    - \* Users can compare AI suggestion with manual choice.
  - *Priority:* Optional
- **FR-ADV.02 – Online Community Platform**
  - *Description:* The system may provide a forum or community space where tutors and students can exchange resources, discuss topics, and collaborate outside formal sessions.
  - *Acceptance Criteria:*
    - \* Users can create discussion threads and share files.
    - \* Moderation tools available for coordinators.
  - *Priority:* Optional
- **FR-ADV.03 – Personalized Learning Support**
  - *Description:* The system may use AI-driven recommendations to suggest learning materials, exercises, or tutoring approaches tailored to individual students.
  - *Acceptance Criteria:*
    - \* Recommendations adapt to student’s history and feedback.
    - \* Users can accept or reject suggestions.
  - *Priority:* Optional
- **FR-ADV.04 – Multi-Program Tutoring**
  - *Description:* The system may support both academic tutoring (courses, skills) and non-academic mentoring (career guidance, soft skills).
  - *Acceptance Criteria:*
    - \* System allows defining and tracking multiple tutoring program types.
    - \* Reports distinguish between academic and non-academic activities.
  - *Priority:* Optional

## 4.9 Non-interactive Functional Requirements (FR-NI)

- **FR-NI-01 – Automatic Notifications**
  - *Description:* The system shall automatically send confirmation, reminder, and cancellation/rescheduling notifications to students and tutors without manual intervention.
  - *Priority:* Must

- **FR-NI-02 – DataCore Sync**

- *Description*: The system shall periodically synchronize personal data (name, ID, faculty, email, status) from HCMUT\_DATACORE.
- *Trigger*: Hourly schedule + on change-webhook
- *Output*: Up-to-date user records with change log
- *Acceptance Criteria*:  $\geq 99\%$  updates reflected within 10 minutes of source change.
- *Priority*: Must

- **FR-NI-03 – Automatic Inactive Detection**

- *Description*: The system should detect logged-in accounts with no activity for a specific period of time and log them out to maintain stability and security.
- *Acceptance Criteria*:
  - \* The system saves the state of the inactive account before logging out.
  - \* The threshold for inactive time is dynamic, depending on the state of the system.
- *Priority*: Optional

- **FR-N-04 – Scheduled Database Cleanup**

- *Description*: The system shall automatically perform database cleanup on a scheduled basis, removing obsolete temporary data, expired logs, and error records to maintain storage efficiency and system performance.
- *Acceptance Criteria*:
  - \* Temporary data and logs older than 12 months are automatically deleted or archived.
  - \* Cleanup runs during off-peak hours to avoid disruption.
  - \* A cleanup summary report (records removed, storage freed) is logged.
  - \* Cleanup failures trigger an alert for administrators.
- *Priority*: Should

- **FR-NI-05 – Disaster Recovery & Backup**

- *Description*: The system shall maintain automated backup and disaster recovery mechanisms to ensure data resilience and continuity in case of failure or outage.
- *Acceptance Criteria*:
  - \* Full backups daily; incremental backups every 15 minutes.
  - \* Backups encrypted and stored in two geographically separate locations.
  - \*  $RPO \leq 15$  minutes,  $RTO \leq 4$  hours.
  - \* Backup integrity verified after each operation.
  - \* Failed backups trigger administrator alerts.
- *Priority*: Should

- **FR-NI-06 – Attendance Logging**

- *Description*: The system shall automatically mark attendance when a student

joins an online tutoring session via the platform.

- *Acceptance Criteria:*
  - \* Attendance log created within 1 minute of session start.
  - \* Logs include session ID, student ID, timestamp.
- *Priority:* Should

## 5 Non-Functional Requirements

### 5.1 Performance Requirements

- NFR1: Concurrent Users: The system shall handle at least **1000 concurrent users** without degradation, verified by load testing.
- NFR2: Response Time: 95% of key actions (login, enrollment, course access, page load) shall complete within **3 seconds** under normal load.
- NFR3: Real-Time Notifications: Notifications (reminders, announcements, deadlines) shall be delivered within **2 seconds** of the triggering event in 95% of cases.

### 5.2 Security & Reliability Requirements

- NFR4: Data Encryption: All personal and academic data shall be encrypted at rest with **AES-256** and in transit with **TLS 1.2 or higher**.
- NFR5: Access Control & Logging: 100% of authentication and access attempts shall be logged and retained for at least **90 days**.
- NFR6: Uptime: The system shall provide at least **99% uptime** during the academic year (excluding scheduled maintenance), monitored monthly.
- NFR7: Data Integrity: Recovery testing shall confirm **zero data loss** during transient failures; operations shall be idempotent.

### 5.3 Usability & Accessibility Requirements

- NFR8: User Interface: Usability testing (System Usability Scale) shall achieve a score of at least **80/100**.
- NFR9: Task Simplicity: Key workflows (course enrollment, submission, reschedule) shall require **no more than 3 steps**.
- NFR10: Multi-Platform Support: The system shall function correctly on at least **3 major browsers** (Chrome, Firefox, Edge) and on **desktop, tablet, and smartphone**.
- NFR11: Accessibility Standards: The system shall meet **WCAG 2.1 AA** accessibility compliance.

### 5.4 Software Quality Attributes

- NFR12: Scalability: The system shall support a **50% increase in users** beyond baseline load without major reconfiguration.

- NFR13: Maintainability: The codebase shall achieve at least **80% unit test coverage** and pass linting with no critical errors.
- NFR14: Extensibility: Adding a new feature (e.g., adaptive learning, AI tutor matching) shall require no more than **2 person-weeks** of integration effort.
- NFR15: Observability: The system shall provide logs covering **100% of authentication and enrollment events** and collect usage metrics (active users, completed courses) with **5-minute granularity**.

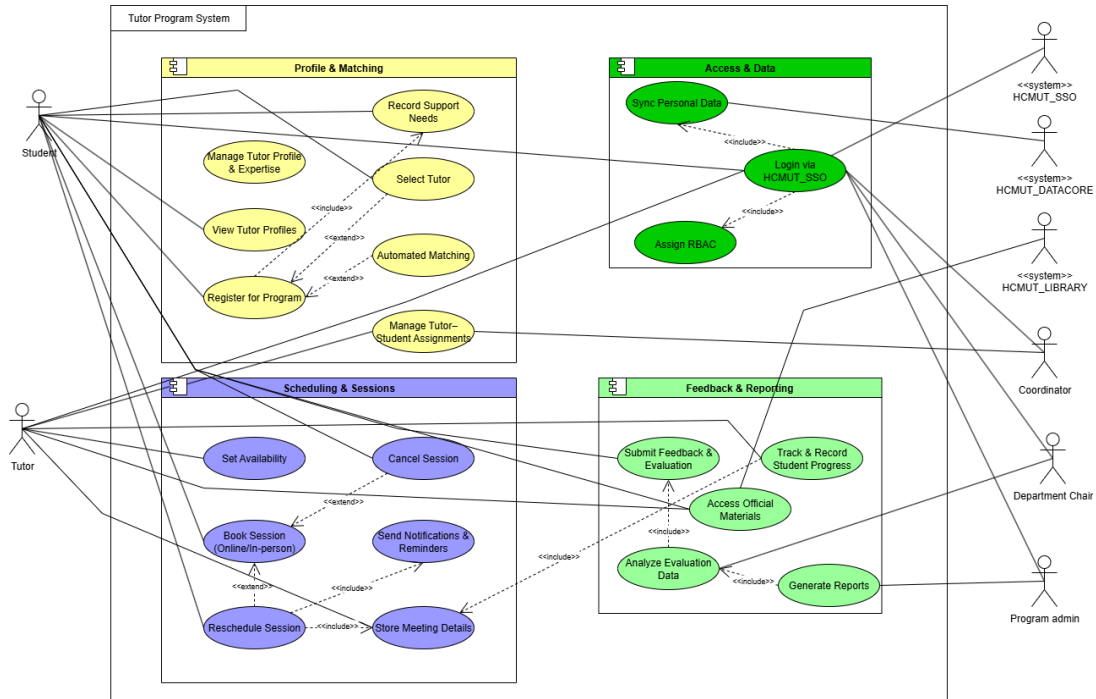
## 5.5 Business Rules

- NFR16: Legal Compliance: The system shall comply with Vietnamese data protection laws and pass an annual **compliance audit with zero critical findings**.

## 6 Use-Case View

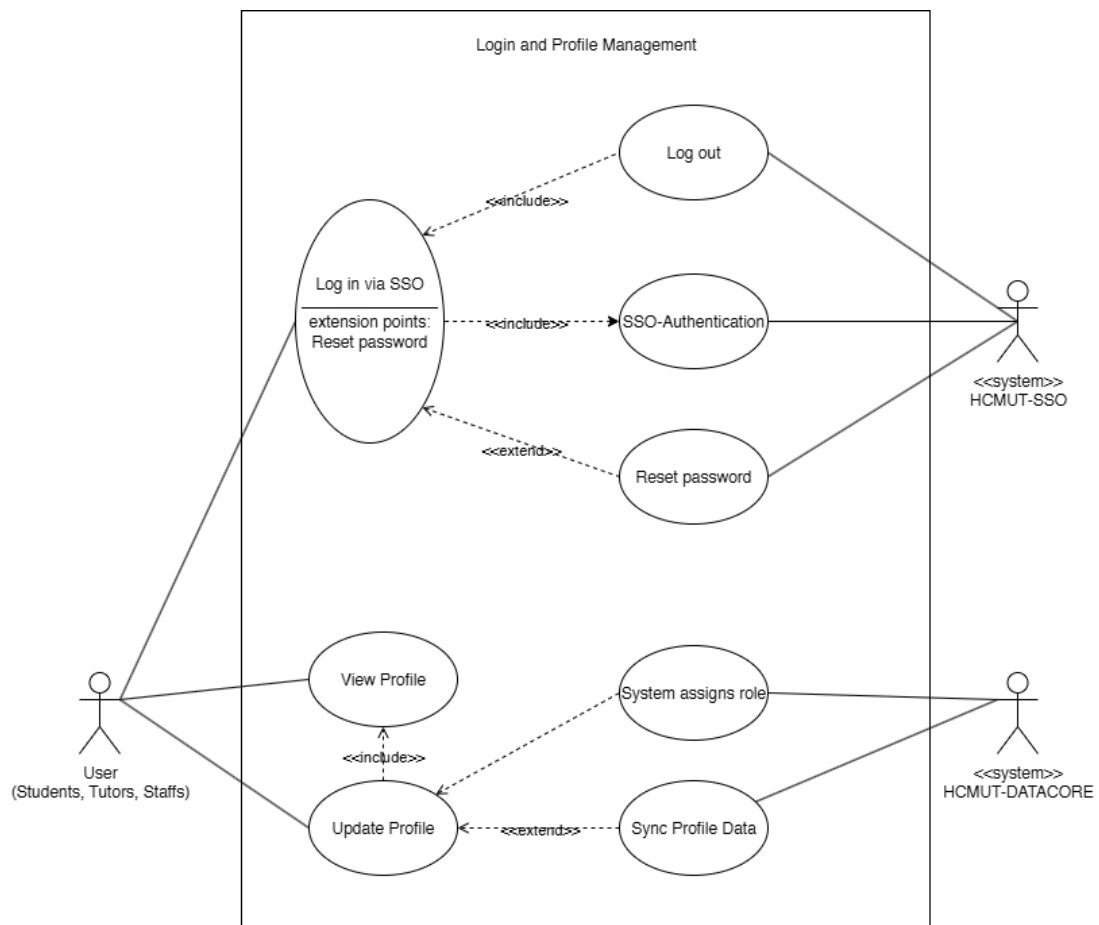
### 6.1 General Use-Case Diagram

Placeholder for diagram:



**Figure 1:** General Tutor program system

## 6.2 UC-01 Log in & Profile Management



**Figure 2:** Log in and Manage Profile

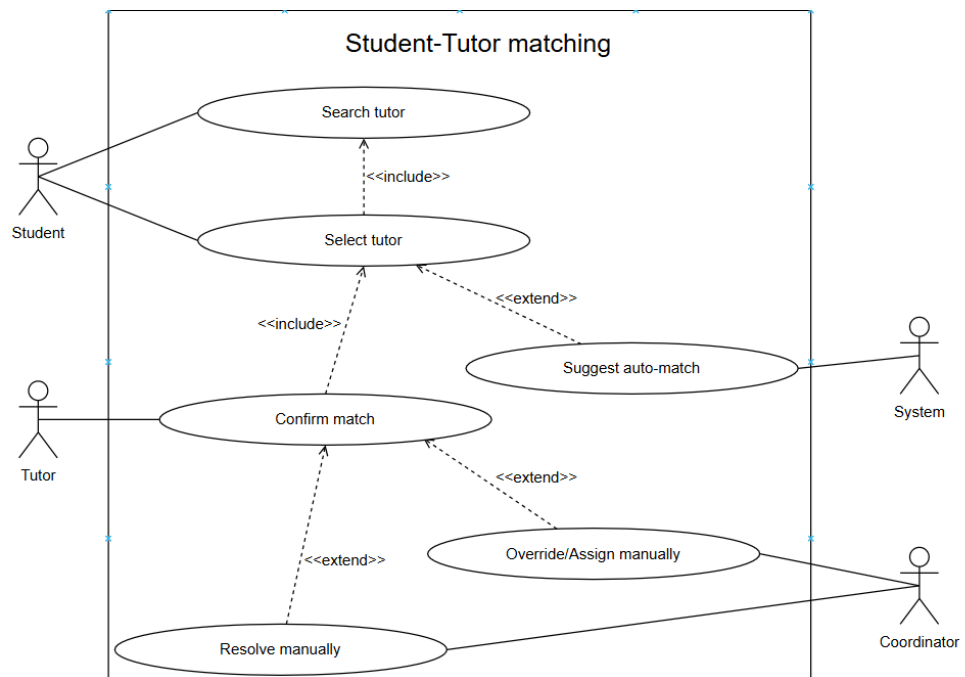
<b>Use-case ID</b>	UC-01a
<b>Use-case name</b>	Login via SSO
<b>Use-case overview</b>	To allow students, tutors, and staff to log in securely via HCMUT_SSO with role assignment handled by the system.
<b>Actors</b>	User (Students, Tutors, Staff), HCMUT_SSO, System
<b>Preconditions</b>	1. User has valid SSO credentials. 2. HCMUT_SSO service is available.
<b>Trigger</b>	User selects the “Login via SSO” option.
<b>Steps</b>	1. User initiates login via HCMUT_SSO. 2. System sends authentication request to SSO service. 3. HCMUT_SSO validates credentials. 4. On success, the system fetches user data and assigns role. 5. On failure, the system denies access.
<b>Postconditions</b>	User is authenticated and session established; role assignment is ready for system use.
<b>Alternative Flows</b>	A1: Invalid login → Access denied with error message. A2: Role update in DATACORE synced during login.
<b>Exception Flow</b>	1. SSO service unavailable → System shows maintenance/unavailable message. 2. Network error → User prompted to retry login.

Table 1: Use Case UC-01a: Login via SSO

<b>Use-case ID</b>	UC-01b
<b>Use-case name</b>	Profile Management
<b>Use-case overview</b>	To allow students and tutors to view and update their profiles, with core data synchronized from HCMUT_DATACORE.
<b>Actors</b>	Student, Tutor, HCMUT_DATACORE, System
<b>Preconditions</b>	1. User is authenticated via SSO. 2. Profile data exists in DATACORE.
<b>Trigger</b>	User selects the “View/Update Profile” option.
<b>Steps</b>	1. System retrieves profile information from DATACORE. 2. User views profile fields (ID, name, email, faculty, role). 3. User updates non-core profile details. 4. System validates and saves changes. 5. System syncs updated data with DATACORE.
<b>Postconditions</b>	User profile is updated and synchronized with DATACORE; changes are timestamped and logged.
<b>Alternative Flows</b>	A1: DATACORE unavailable → Updates stored locally until sync resumes.
<b>Exception Flow</b>	1. Invalid update request → System rejects and shows error. 2. Sync conflict with DATACORE → DATACORE treated as source of truth.

Table 2: Use Case UC-01b: Profile Management

### 6.3 UC-02 Tutor–Student Matching



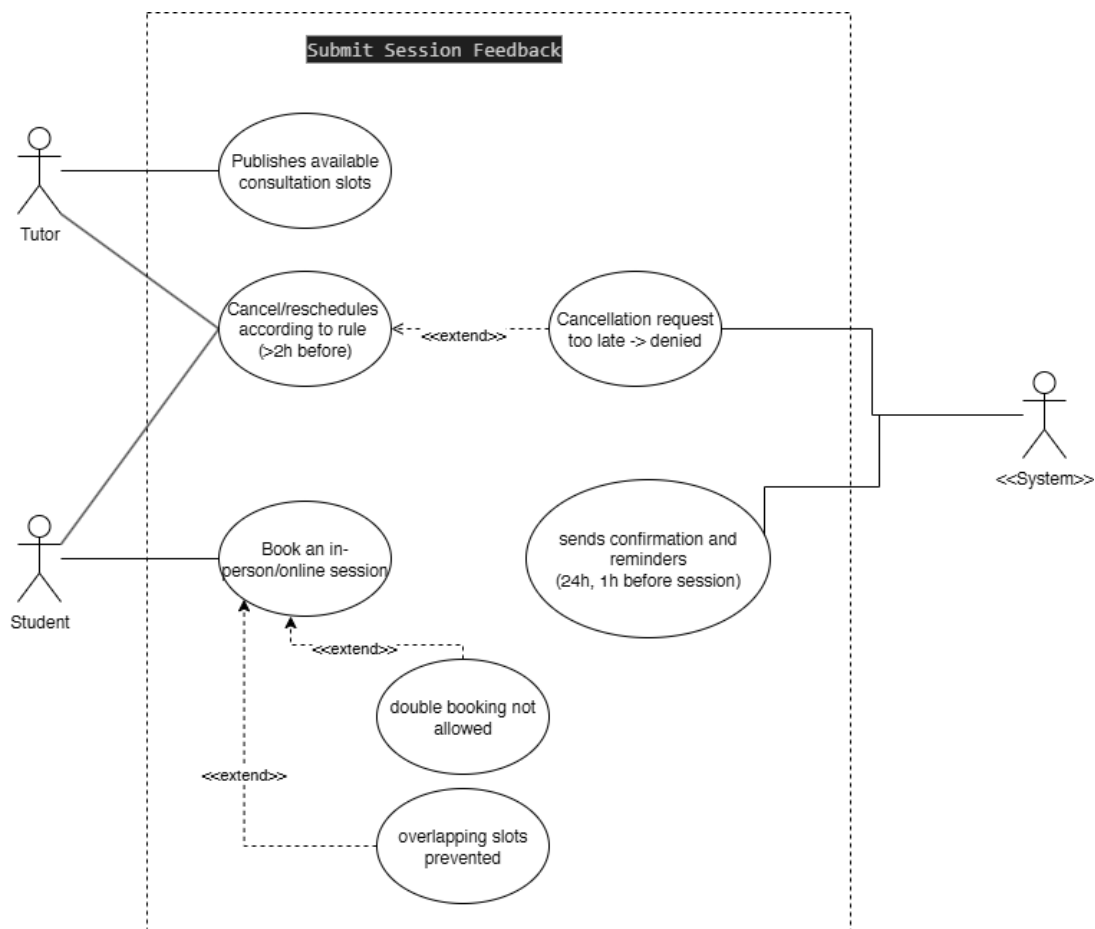
**Figure 3:** Tutor–Student Matching



<b>Use-case ID</b>	UC-02
<b>Use-case name</b>	Tutor–Student Matching
<b>Use-case overview</b>	To allow students to search and select tutors manually or request an automated match, with tutor confirmation and coordinator intervention when necessary.
<b>Actors</b>	Student (primary), Tutor, Coordinator, System
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Student and tutor profiles exist in the system.</li> <li>2. The system is operational and accessible.</li> <li>3. Student is authenticated in the system.</li> </ol>
<b>Trigger</b>	Student initiates a search for tutors or requests an auto-match.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Student searches for tutors by subject, availability, or preferences.</li> <li>2. Student selects a tutor; a pending match is created.</li> <li>3. System may suggest an auto-match (ranked list) based on the student’s criteria.</li> <li>4. Tutor reviews the pending match and confirms the match.</li> <li>5. If confirmed, the system finalizes and logs the pairing.</li> </ol>
<b>Postconditions</b>	A tutor–student pairing is established and logged in the system.
<b>Alternative Flows</b>	<ol style="list-style-type: none"> <li>1. Auto-match rejected by student or tutor → Coordinator resolves manually.</li> <li>2. No tutors found → System suggests broadening search criteria.</li> <li>3. Tutor does not respond within time limit → Coordinator is notified to assign a tutor.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. Network failure prevents search or confirmation (system prompts user to retry).</li> <li>2. Tutor or student profile missing/corrupted → System logs an error and notifies Coordinator.</li> </ol>

Table 3: Use Case UC-02: Tutor–Student Matching

## 6.4 UC-03 Session Scheduling Management



**Figure 4:** Session Scheduling Management

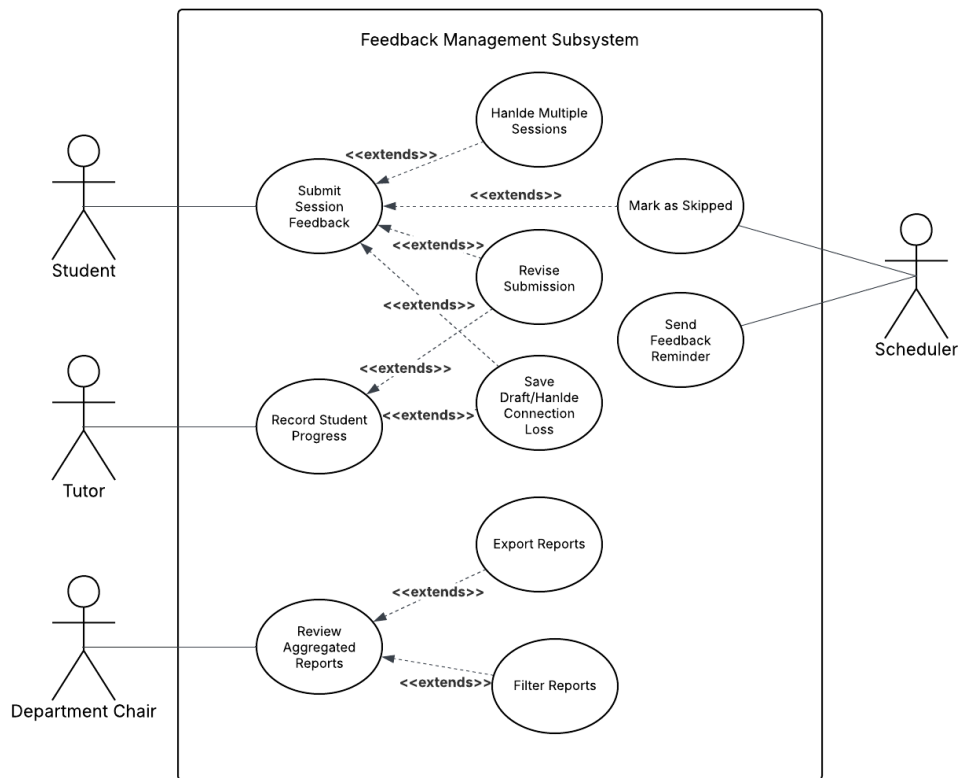
<b>Use-case ID</b>	UC-03a
<b>Use-case name</b>	Publish and Manage Tutor Availability
<b>Overview</b>	To allow a tutor to create, edit, and manage available consultation slots, while the system prevents overlapping sessions and validates scheduling rules.
<b>Actors</b>	Tutor, System
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Tutor is authenticated in the system.</li> <li>• The scheduling database is accessible.</li> </ul>
<b>Trigger</b>	Tutor selects “Set Availability” or modifies existing slots.
<b>Main Flow (Steps)</b>	<ol style="list-style-type: none"> <li>1. Tutor publishes available consultation slots.</li> <li>2. System checks for overlapping slots.</li> <li>3. If no conflicts are found, slots are confirmed and saved.</li> <li>4. Tutor may later edit or delete slots before they are booked.</li> </ol>
<b>Postconditions</b>	<ul style="list-style-type: none"> <li>• Valid time slots are stored in the system and visible to students.</li> </ul>
<b>Alternative Flows</b>	AF1: Overlapping slot detected → system rejects slot creation and displays conflict message.
<b>Exception Flows</b>	<ul style="list-style-type: none"> <li>• Database or connection error → slot creation aborted, system logs error.</li> </ul>

Table 4: Use Case UC-03a: Publish and Manage Tutor Availability

<b>Use-case ID</b>	UC-03b
<b>Use-case name</b>	Session Booking, Reminders, and Cancellation Rules
<b>Overview</b>	To allow a student to book an in-person or online session with a tutor, receive automatic confirmations and reminders, and manage cancellations or reschedules according to defined rules.
<b>Actors</b>	Student, Tutor, System
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Tutor has published available slots.</li> <li>• Student is authenticated and matched with a tutor.</li> </ul>
<b>Trigger</b>	Student initiates a booking or a cancellation/reschedule request.
<b>Main Flow (Steps)</b>	<ol style="list-style-type: none"> <li>1. Student views tutor's available consultation slots.</li> <li>2. Student selects a preferred slot and submits booking.</li> <li>3. System checks for double booking or slot conflicts.</li> <li>4. Booking is confirmed, and notifications are sent to student and tutor.</li> <li>5. System automatically sends reminders (24h and 1h before the session).</li> <li>6. If cancellation or reschedule is requested <math>\geq 2h</math> before session it is approved.</li> </ol>
<b>Postconditions</b>	<ul style="list-style-type: none"> <li>• Session status updated (booked, rescheduled, or cancelled).</li> <li>• All notifications are logged and delivered.</li> </ul>
<b>Alternative Flows</b>	<p>AF1: Cancellation request too late (<math>&lt;2h</math> before) <math>\rightarrow</math> denied with reason.</p> <p>AF2: Tutor unavailable <math>\rightarrow</math> student notified and slot reopened.</p>
<b>Exception Flows</b>	<ul style="list-style-type: none"> <li>• Network or database failure <math>\rightarrow</math> transaction aborted, retry suggested.</li> </ul>

Table 5: Use Case UC-03b: Session Booking, Reminders, and Cancellation Rules

## 6.5 UC-04 Feedback & Progress Tracking



**Figure 5:** Submit Session Feedback

<b>Use-case ID</b>	UC-04a
<b>Use-case name</b>	Submit Session Feedback
<b>Use-case overview</b>	To allow a student to submit structured feedback after a tutoring session. The feedback is linked to the session and stored for later evaluation and reporting.
<b>Actors</b>	<ul style="list-style-type: none"> <li>• Student (primary)</li> <li>• Scheduler (secondary, for reminders and marking 'Skipped')</li> </ul>
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. A tutoring session has been completed.</li> <li>2. The system is running and accessible.</li> <li>3. Student is authenticated in the system.</li> </ol>
<b>Trigger</b>	The student clicks the "Submit Feedback" option after the session is completed.

<b>Main Flow (Steps)</b>	<ol style="list-style-type: none"> <li>1. System displays a feedback form linked to the completed session.</li> <li>2. Student fills in and submits the structured feedback.</li> <li>3. System validates the input and saves the feedback in the database.</li> <li>4. System links the feedback to the corresponding session.</li> </ol>
<b>Postconditions</b>	<ul style="list-style-type: none"> <li>• Feedback is stored in the database and linked to the correct session.</li> <li>• If skipped, the system records a “Feedback Skipped” status for that session.</li> <li>• Data is available for tutors and department chairs in aggregated reports.</li> </ul>
<b>Alternative Flows</b>	<ol style="list-style-type: none"> <li>1. <b>Multiple Session Feedback:</b> If multiple sessions are pending, the system displays a list and allows the student to submit feedback sequentially.</li> <li>2. <b>Draft Save/Connection Loss:</b> The student may save incomplete feedback as a draft and return later. System automatically saves the draft if connection is lost before final submission.</li> <li>3. <b>Feedback Revision:</b> Within a defined grace period (e.g., 24h), the student may edit and resubmit feedback; the updated version replaces the original.</li> <li>4. <b>Skipped Feedback:</b> If no feedback is submitted within the allowed time, the Scheduler triggers a process to mark the session’s feedback status as “Skipped.”</li> </ol>
<b>Exception Flows</b>	<ul style="list-style-type: none"> <li>• <b>Database Error:</b> If the system fails to save the final submission due to a database error, it logs the error and prompts the student to retry later.</li> <li>• <b>Missing Session Record:</b> If the session record is missing or corrupted, the system logs an error and notifies the coordinator.</li> </ul>

Table 6: Use Case UC-04a: Submit Session Feedback

<b>Use-case ID</b>	UC-04b
<b>Use-case name</b>	Record Student Progress
<b>Use-case overview</b>	To allow tutors to record student progress and optionally add session summaries for reference and tracking purposes.
<b>Actors</b>	Tutor

<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. A tutoring session has been completed.</li> <li>2. The system is running and accessible.</li> <li>3. Tutor is authenticated in the system.</li> </ol>
<b>Trigger</b>	Tutor accesses the session record and selects “Record Progress.”
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Tutor opens the specific session record.</li> <li>2. Tutor enters progress notes and, optionally, a session summary.</li> <li>3. Tutor submits the record.</li> <li>4. System validates the input and saves the record, linking it to the session.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Student progress and optional session summary are saved.</li> <li>2. Data is linked to the session and available for departmental review.</li> </ol>
<b>Alternative Flows</b>	<p>A1: Optional Summary → Tutor skips writing a session summary (only progress notes are saved).</p> <p>A2: Draft Save / Connection Loss → Tutor may save incomplete notes as a draft; the system auto-saves the draft if the connection is lost.</p> <p>A3: Note Revision → Within a 24-hour grace period, tutor may edit and resubmit notes; the new version replaces the previous record.</p>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. Database error → System logs the issue and notifies the tutor to retry.</li> <li>2. Missing or locked session → System notifies the tutor and logs the error.</li> </ol>
<b>Priority</b>	Should

Table 7: Use Case UC-04b: Record Student Progress

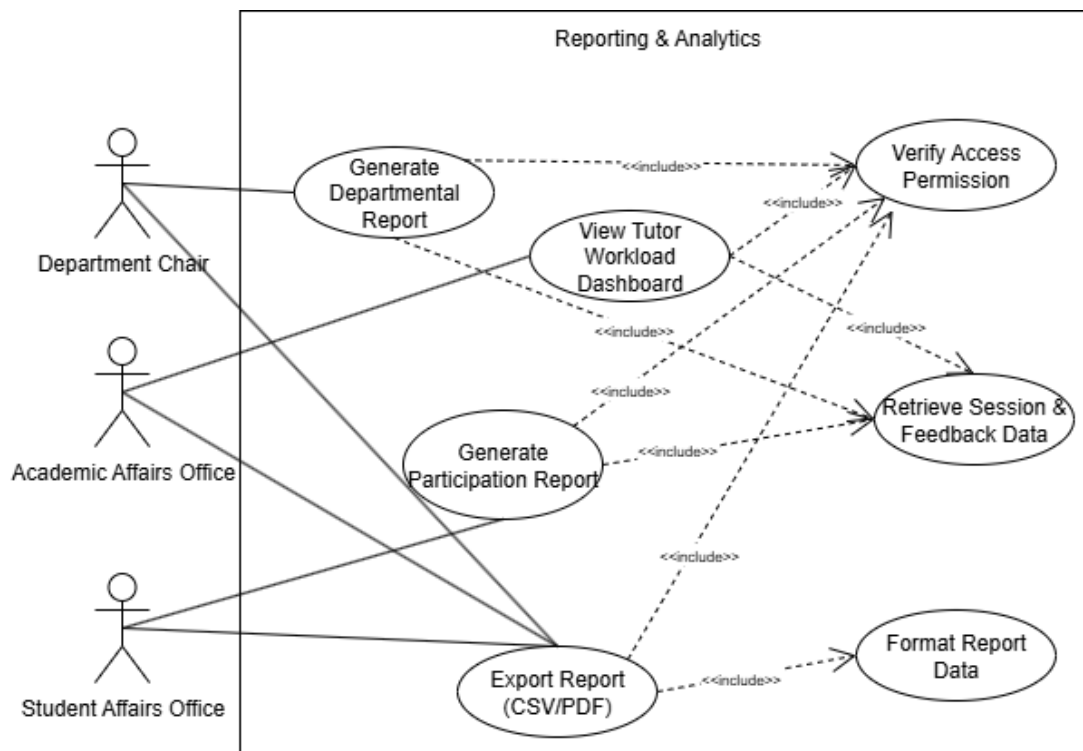
<b>Use-case ID</b>	UC-04c
<b>Use-case name</b>	Review Aggregated Reports (Feedback & Progress)
<b>Overview</b>	To allow the department chair to review aggregated feedback and progress data for monitoring tutor performance and program effectiveness.
<b>Actors</b>	Department Chair
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Feedback and progress records exist in the database.</li> <li>2. The system is running and accessible.</li> <li>3. Department Chair is authenticated in the system.</li> </ol>
<b>Trigger</b>	Department Chair selects “Aggregated Reports Overview.”

<b>Main Flow (Steps)</b>	<ol style="list-style-type: none"> <li>1. Department Chair selects report criteria (e.g., date range, tutor, course).</li> <li>2. System retrieves all relevant feedback and progress data.</li> <li>3. System processes and displays aggregated statistics, charts, or reports.</li> </ol>
<b>Postconditions</b>	<ul style="list-style-type: none"> <li>• Aggregated data is available for decision-making.</li> <li>• Reports may be exported or stored for institutional use.</li> </ul>
<b>Alternative Flows</b>	<p>AF1 <b>Filtering:</b> Department Chair filters or drills down data by tutor, course, or specific time period.</p> <p>AF2 <b>Export Report:</b> Department Chair exports the report in a chosen format (e.g., PDF, CSV) for external analysis.</p>
<b>Exception Flows</b>	<ul style="list-style-type: none"> <li>• <b>Database/Retrieval Failure:</b> If database retrieval fails, the system shows a technical error message and logs the issue for system administrators.</li> <li>• <b>No Data:</b> If no data matches the selected criteria, the system displays a clear message: “No data available for this selection.”</li> </ul>

Table 8: Use Case UC-04c: Review Aggregated Reports



## 6.6 UC-05 Reporting & Analytics



**Figure 6:** Reporting and Analytics

<b>Use-case ID</b>	UC-05a
<b>Use-case name</b>	Generate Departmental Report
<b>Use-case overview</b>	Department Chair generates a department-level report (attendance, performance, session counts) for academic monitoring and decisions.
<b>Actors</b>	Department Chair
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Authenticated and authorized to view departmental reports.</li> <li>2. Sessions, feedback, and (if applicable) progress data exist.</li> <li>3. System is operational.</li> </ol>
<b>Trigger</b>	Chair opens “Reporting” and selects “Departmental Report”.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Set filters (term/date, program/department, tutor/cohort).</li> <li>2. System retrieves and analyzes metrics (attendance, performance, session counts).</li> <li>3. System displays tables/charts with data-source notes.</li> <li>4. Optional: Export CSV/PDF with metadata (generation time, filters, version); log action.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Departmental report displayed; optional file generated.</li> <li>2. Action recorded in the audit log.</li> </ol>
<b>Alternative Flows</b>	<ol style="list-style-type: none"> <li>1. Change filters/scope → system refreshes results.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. No data for selected criteria → show “No data available for this selection.”</li> <li>2. Export error (I/O or size) → suggest narrowing scope or retrying.</li> <li>3. Data source unavailable → show service notice; allow retry when available.</li> </ol>

Table 9: Use Case UC-05a: Generate Departmental Report

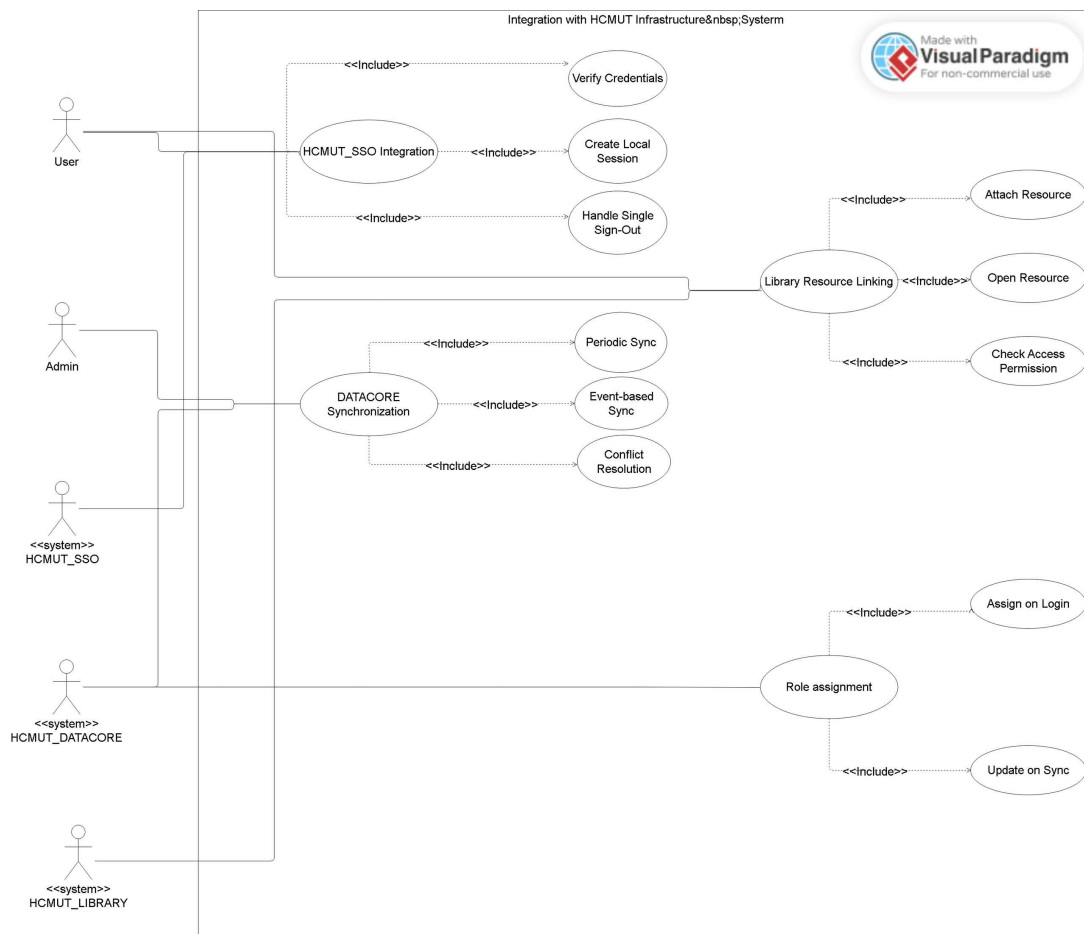
<b>Use-case ID</b>	UC-05b
<b>Use-case name</b>	View Tutor Workload Dashboard
<b>Use-case overview</b>	Academic Affairs views tutor workload and student demand to allocate resources effectively.
<b>Actors</b>	Academic Affairs Office
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Authenticated and authorized for workload dashboards.</li> <li>2. Sessions/booking and feedback data exist.</li> <li>3. System is operational.</li> </ol>
<b>Trigger</b>	Office opens “Workload & Demand” dashboard.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Select filters (term/date, department/program).</li> <li>2. System aggregates tutor load and demand indicators.</li> <li>3. System displays dashboard (tables/charts, drill-down).</li> <li>4. Optional: Export CSV/PDF with metadata; log action.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Workload dashboard visible; optional export ready.</li> <li>2. Access recorded in the audit log.</li> </ol>
<b>Alternative Flows</b>	<ol style="list-style-type: none"> <li>1. Drill-down by tutor/program → refresh view.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. No data for selected criteria → show “No data available for this selection.”</li> <li>2. Export error (I/O or size) → suggest narrowing scope or retrying.</li> <li>3. Data source unavailable → show service notice; allow retry when available.</li> </ol>

Table 10: Use Case UC-05b: View Tutor Workload Dashboard

<b>Use-case ID</b>	UC-05c
<b>Use-case name</b>	Generate Participation Report
<b>Use-case overview</b>	Student Affairs generates participation reports for credit/scholarship consideration.
<b>Actors</b>	Student Affairs Office
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Authenticated and authorized for participation reports.</li> <li>2. Sessions and attendance/feedback records exist.</li> <li>3. System is operational.</li> </ol>
<b>Trigger</b>	Office selects “Participation Report”.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Choose filters (term/date, cohort, program, thresholds).</li> <li>2. System compiles participation metrics and eligibility indicators.</li> <li>3. System displays results; Optional: export CSV/PDF with metadata; log action.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Participation report displayed; optional file generated.</li> <li>2. Action logged.</li> </ol>
<b>Alternative Flows</b>	<ol style="list-style-type: none"> <li>1. Adjust filters/criteria → refresh results.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. No data for selected criteria → show “No data available for this selection.”</li> <li>2. Export error (I/O or size) → suggest narrowing scope or retrying.</li> <li>3. Data source unavailable → show service notice; allow retry when available.</li> </ol>

Table 11: Use Case UC-05c: Generate Participation Report

## 6.7 UC-06 Integration with HCMUT Infrastructure



**Figure 7:** Integration with HCMUT Infrastructure

<b>Use-case ID</b>	UC-06a
<b>Use-case name</b>	HCMUT_SSO Integration
<b>Use-case overview</b>	The system integrates with HCMUT_SSO for unified authentication, allowing users to log in using university credentials and automatically manage single sign-out.
<b>Actors</b>	User, System, HCMUT_SSO
<b>Preconditions</b>	HCMUT_SSO service is operational and reachable.
<b>Trigger</b>	User initiates login via HCMUT_SSO.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. User selects “Login with HCMUT_SSO”.</li> <li>2. System redirects to the authentication portal.</li> <li>3. HCMUT_SSO validates credentials and returns a token.</li> <li>4. System verifies the token and creates a session.</li> <li>5. Upon sign-out at SSO, the system terminates the local session.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. User is successfully authenticated.</li> <li>2. Single sign-out ensures session consistency.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. Invalid or expired token → login attempt rejected.</li> <li>2. SSO service unavailable → system displays maintenance message.</li> </ol>
<b>Priority</b>	Must

Table 12: Use Case UC-06a: HCMUT\_SSO Integration

<b>Use-case ID</b>	UC-06b
<b>Use-case name</b>	DATAcore Synchronization
<b>Use-case overview</b>	The system synchronizes personal and academic data from HCMUT_DATAcore periodically or in near real-time to ensure data consistency and reduce manual entry.
<b>Actors</b>	System, HCMUT_DATAcore, Administrator
<b>Preconditions</b>	DATAcore APIs are online and accessible.
<b>Trigger</b>	Scheduled synchronization or data-change event detected.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. System triggers synchronization with DATAcore.</li> <li>2. Retrieve updated profiles and academic data.</li> <li>3. Validate and compare data with local records.</li> <li>4. Update local data using DATAcore as source of truth.</li> <li>5. Log synchronization status and timestamp.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Local data mirrors DATAcore.</li> <li>2. Synchronization events logged for auditing.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. Connection timeout or API error → retry with exponential backoff.</li> <li>2. Invalid data format → skip record and log validation error.</li> </ol>
<b>Priority</b>	Must

Table 13: Use Case UC-06b: DATAcore Synchronization

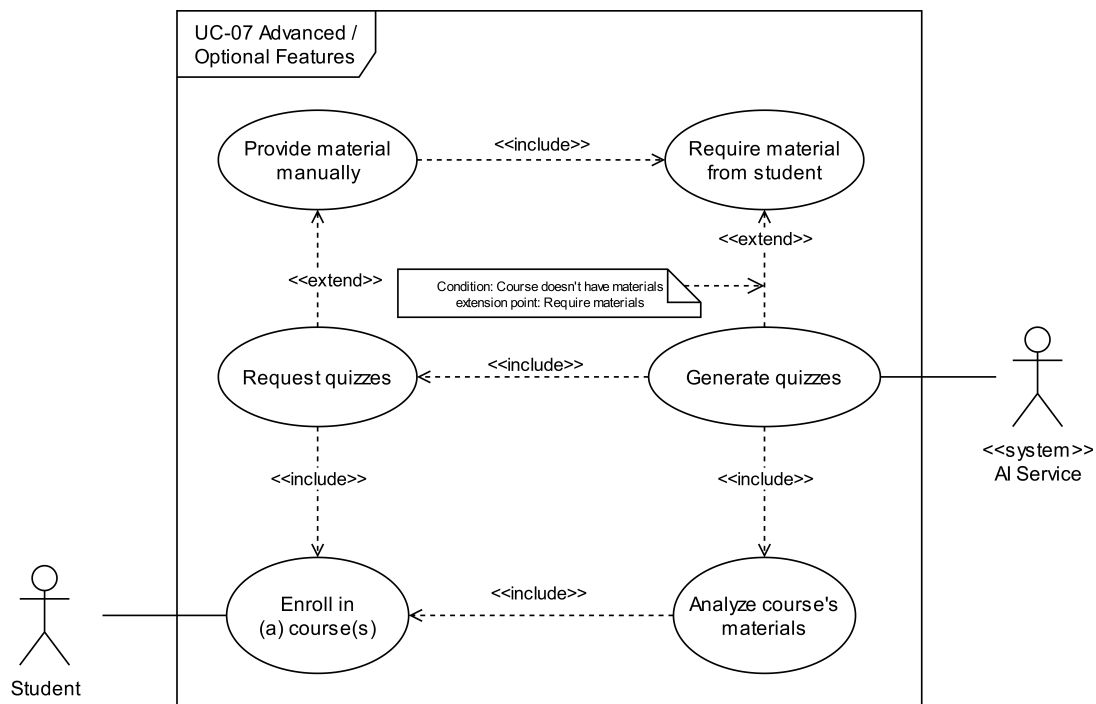
<b>Use-case ID</b>	UC-06c
<b>Use-case name</b>	Role Assignment
<b>Use-case overview</b>	The system automatically assigns user roles (student, tutor, coordinator, department chair, or administrator) based on centralized role data from DATACORE and SSO.
<b>Actors</b>	User, System, HCMUT_DATACORE
<b>Preconditions</b>	User is authenticated via SSO and DATACORE role data is available.
<b>Trigger</b>	User login or scheduled role update.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Retrieve role mapping from DATACORE.</li> <li>2. Match user ID with centralized role data.</li> <li>3. Assign permissions based on role.</li> <li>4. Apply changes in access-control policies.</li> <li>5. Log the role-assignment event.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. User roles align with centralized data.</li> <li>2. Updated permissions take effect immediately.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. Role data missing → assign default “student” role and notify admin.</li> <li>2. Role conflict → logged and flagged for manual verification.</li> </ol>
<b>Priority</b>	Must

Table 14: Use Case UC-06c: Role Assignment

<b>Use-case ID</b>	UC-06d
<b>Use-case name</b>	Library Resource Linking
<b>Use-case overview</b>	The system connects with HCMUT_LIBRARY to let tutors and students attach or access library materials securely within tutoring sessions or summaries.
<b>Actors</b>	Student, Tutor, System, HCMUT_LIBRARY
<b>Preconditions</b>	Library API and authentication services are available.
<b>Trigger</b>	User attaches or opens a library resource.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. User searches or selects a library resource.</li> <li>2. System requests metadata and access permissions.</li> <li>3. Verify user eligibility via role-based access.</li> <li>4. Attach or open the resource in session view.</li> <li>5. Log the access event.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Resource successfully linked to the session or summary.</li> <li>2. Access rights enforced per library policy.</li> </ol>
<b>Exception Flow</b>	<ol style="list-style-type: none"> <li>1. Access denied → system displays permission error.</li> <li>2. Library service unavailable → prompt user to retry or queue attachment.</li> </ol>
<b>Priority</b>	Could

Table 15: Use Case UC-06d: Library Resource Linking

## 6.8 UC-07 Advanced / Optional Features



**Figure 8:** AI-generated review quizzes



<b>Use-case ID</b>	UC-07
<b>Use-case name</b>	AI-generated review quizzes
<b>Use-case overview</b>	Provide students with quizzes related to the courses they are enrolling.
<b>Actors</b>	Students, AI service
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. The system is running.</li> <li>2. Internet connection is available.</li> <li>3. AI service must be available.</li> <li>4. The courses must have learning materials uploaded by tutors.</li> </ol>
<b>Trigger</b>	Students use the AI-based quiz generation function.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Retrieve uploaded course materials of the course.</li> <li>2. Process and analyze the content with AI.</li> <li>3. Search the internet for related academic resources and quizzes.</li> <li>4. Generate quiz questions covering the key topics.</li> <li>5. Display the quiz to the requested students.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. The quizzes are displayed on the screen of the requested students, and they can download the quizzes as a document file.</li> <li>2. The quizzes can be available until the end of the login session of the requested students, or until they finished the course if they chose to save the quizzes</li> </ol>
<b>Alternative Flows</b>	- If the course doesn't have any learning materials, notify the user and request them for manual typing in key topics needed for review. A1: Invalid login → Access denied with error message.
<b>Exception Flow</b>	1. If AI service isn't available, display an error.

Table 16: Use Case UC-07: AI-generated review quizzes

## 7 System Integration

- **HCMUT\_SSO**: OAuth2/OIDC-based single sign-on.
- **HCMUT\_DATACORE**: Read-only sync of personal data (name, ID, role, faculty, email).
- **HCMUT\_LIBRARY**: Link and share official materials within sessions.

## 8 Coding Rules and Constraints

To align with prior functional-programming constraints used in course assignments, if applicable:

- Only allowed imports per assignment rules.
- Prefer pure functions; avoid global state.
- Prefer higher-order functions and list comprehensions over loops.

- Single-assignment variables within functions to encourage immutability.
- (Adapt or remove this section if your instructor does not require functional-programming constraints.)

## 9 Submission and Deliverables

### Deliverables:

- PDF generated from this  $\text{\LaTeX}$  project.
- Any supporting diagrams (PNG/PDF) in the `images/` folder.

### Submission notes:

- Ensure the document compiles on Overleaf without custom fonts or non-standard packages.

## 10 Other Regulations

- Work must be original and comply with academic integrity policies.
- Instructor decisions are final.
- Post-grading test cases or rubrics may be summarized but not fully disclosed.

## 11 Changelog

Version	Notes
1.0	Initial draft tailored for Tutor Support System ( 26 September 2025).
1.1	Functional Requirement, Non-Functional Requirement and General Use-case diagram (24 September 2025).
1.2	Detail Use-case diagram and Table, Non-Interactive Functional Requirement and Fix 1.1 log submission(01 October 2025).