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

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, DataCore, Library).
- Produce a consistent, submission-ready specification document in LaTeX.

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 DATACORE.
  - *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:

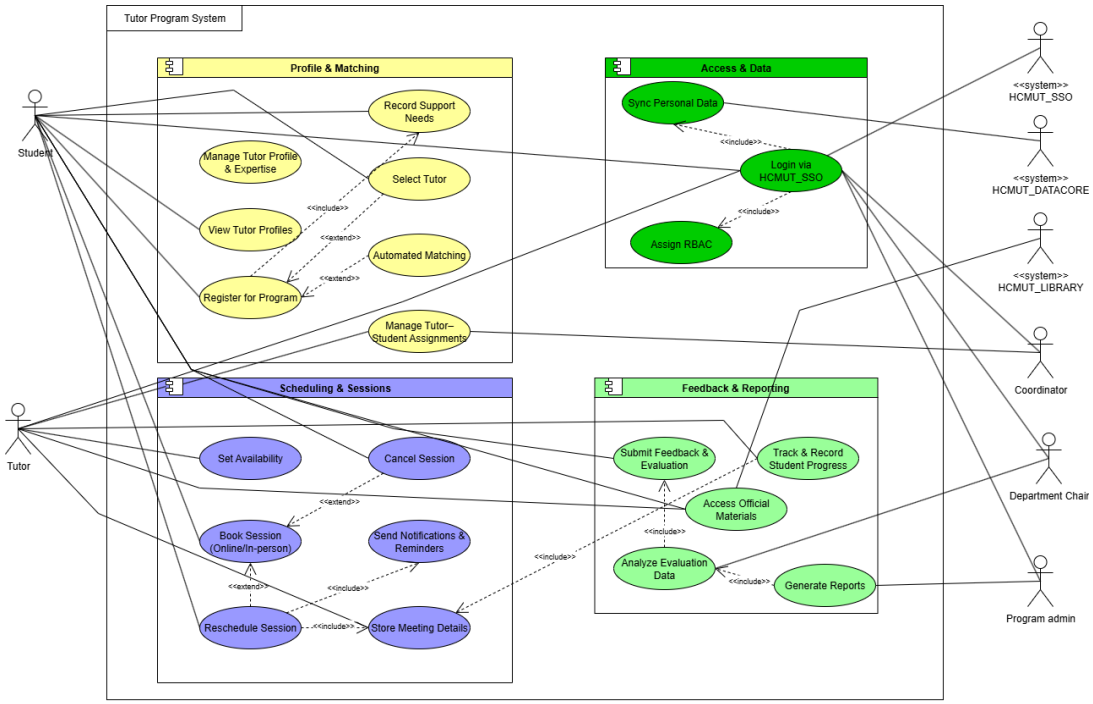

Figure 1: General Tutor program system

6.2 UC-01 Log in & Profile Management

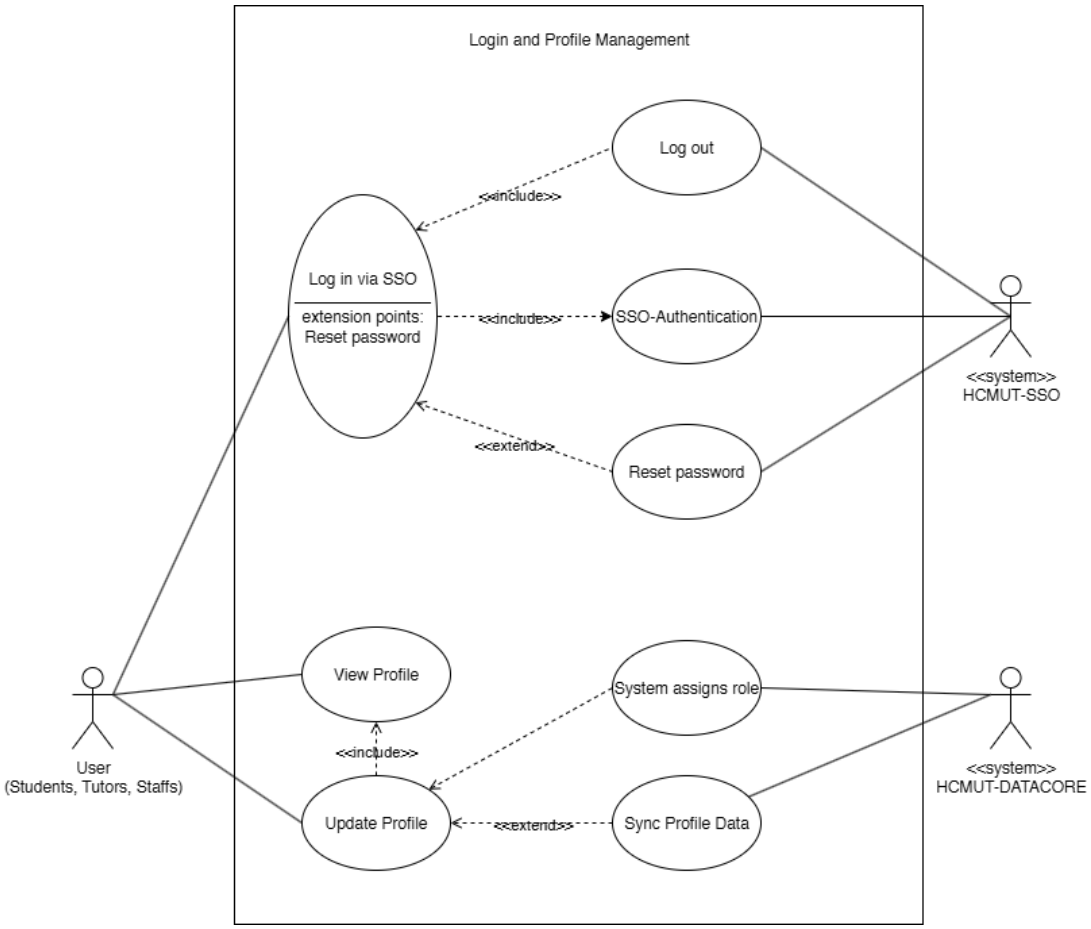

Figure 2: Log in and Manage Profile

Use-case ID	UC-01a
Use-case name	Login via SSO
Use-case overview	To allow students, tutors, and staff to log in securely via HCMUT_SSO with role assignment handled by the system.
Actors	User (Students, Tutors, Staff), HCMUT_SSO, System
Preconditions	1. User has valid SSO credentials. 2. HCMUT_SSO service is available.
Trigger	User selects the "Login via SSO" option.
Steps	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.
Postconditions	User is authenticated and session established; role assignment is ready for system use.
Alternative Flows	A1: Invalid login → Access denied with error message. A2: Role update in DATACORE synced during login.
Exception Flow	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

Use-case ID	UC-01b
Use-case name	Profile Management
Use-case overview	To allow students and tutors to view and update their profiles, with core data synchronized from HCMUT_DATACORE.
Actors	Student, Tutor, HCMUT_DATACORE, System
Preconditions	1. User is authenticated via SSO. 2. Profile data exists in DATACORE.
Trigger	User selects the "View/Update Profile" option.
Steps	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.
Postconditions	User profile is updated and synchronized with DATACORE; changes are timestamped and logged.
Alternative Flows	A1: DATACORE unavailable → Updates stored locally until sync resumes.
Exception Flow	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

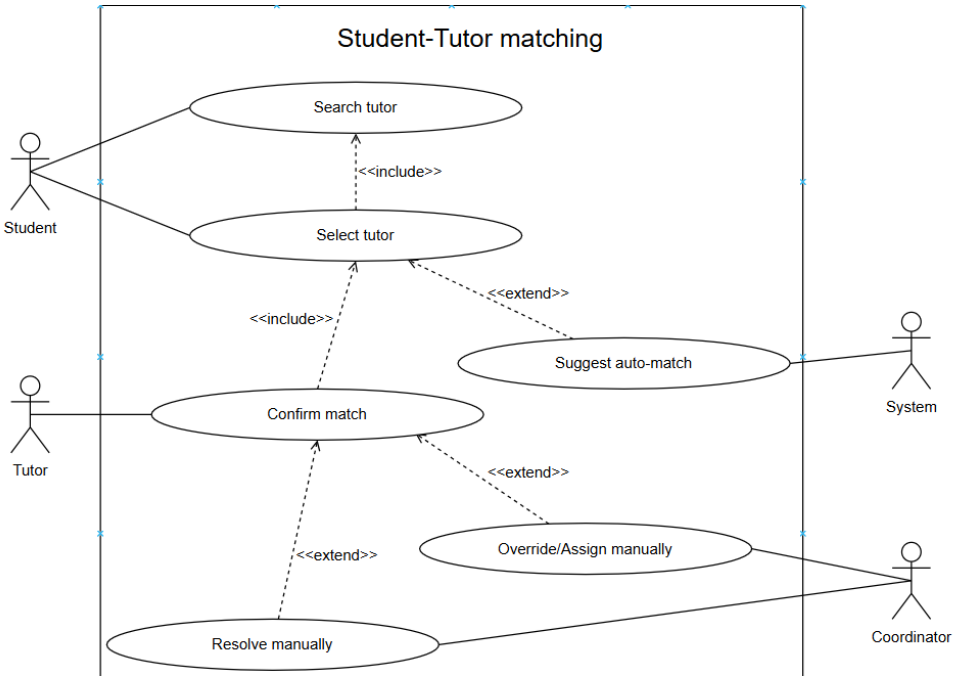

Figure 3: Tutor–Student Matching

Use-case ID	UC-02
Use-case name	Tutor–Student Matching
Use-case overview	To allow students to search and select tutors manually or request an automated match, with tutor confirmation and coordinator intervention when necessary.
Actors	Student (primary), Tutor, Coordinator, System
Preconditions	1. Student and tutor profiles exist in the system. 2. The system is operational and accessible. 3. Student is authenticated in the system.
Trigger	Student initiates a search for tutors or requests an auto-match.
Steps	1. Student searches for tutors by subject, availability, or preferences. 2. Student selects a tutor; a pending match is created. 3. System may suggest an auto-match (ranked list) based on the student's criteria. 4. Tutor reviews the pending match and confirms the match. 5. If confirmed, the system finalizes and logs the pairing.
Postconditions	A tutor–student pairing is established and logged in the system.
Alternative Flows	1. Auto-match rejected by student or tutor → Coordinator resolves manually. 2. No tutors found → System suggests broadening search criteria. 3. Tutor does not respond within time limit → Coordinator is notified to assign a tutor.
Exception Flow	1. Network failure prevents search or confirmation (system prompts user to retry). 2. Tutor or student profile missing/corrupted → System logs an error and notifies Coordinator.

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

6.4 UC-03 Session Scheduling Management

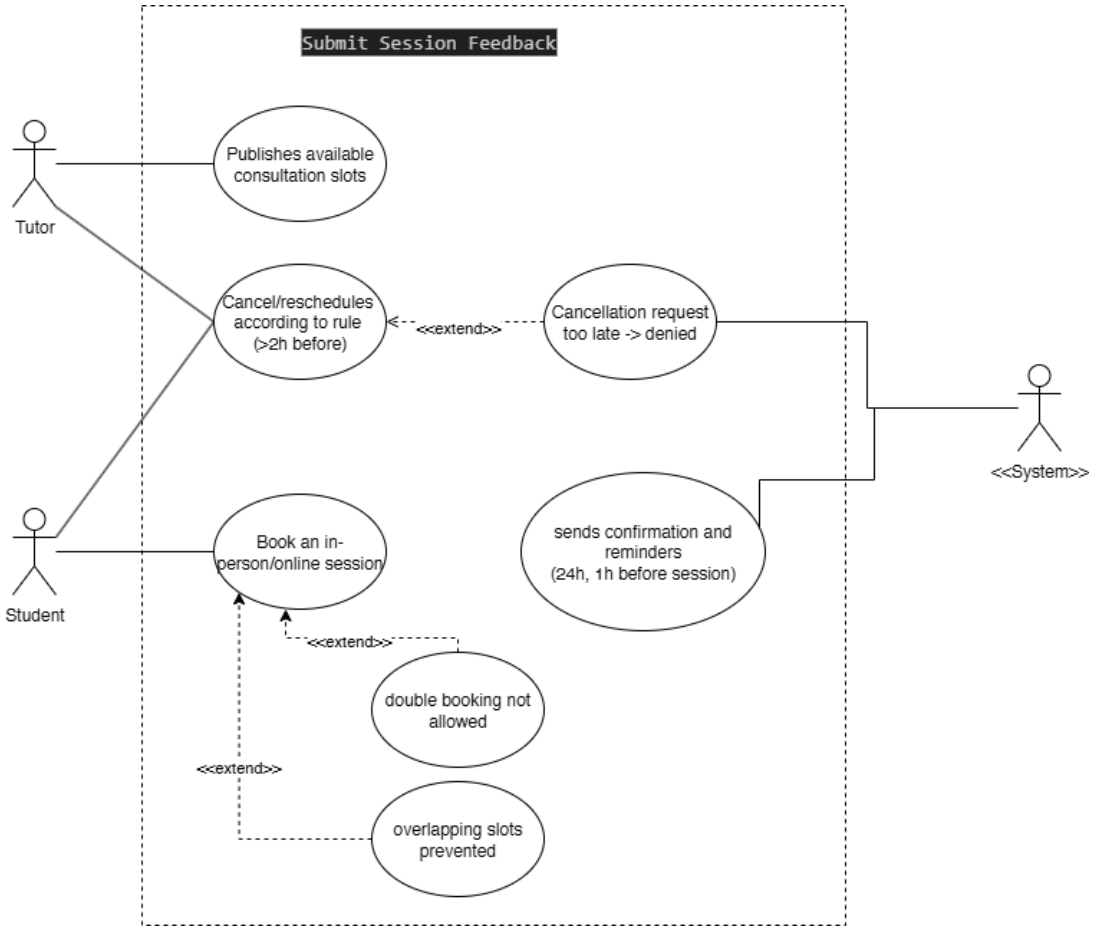

Figure 4: Session Scheduling Management

Use-case ID	UC-03a
Use-case name	Publish and Manage Tutor Availability
Overview	To allow a tutor to create, edit, and manage available consultation slots, while the system prevents overlapping sessions and validates scheduling rules.
Actors	Tutor, System
Preconditions	• Tutor is authenticated in the system. • The scheduling database is accessible.
Trigger	Tutor selects "Set Availability" or modifies existing slots.
Main Flow (Steps)	1. Tutor publishes available consultation slots. 2. System checks for overlapping slots. 3. If no conflicts are found, slots are confirmed and saved. 4. Tutor may later edit or delete slots before they are booked.
Postconditions	• Valid time slots are stored in the system and visible to students.
Alternative Flows	AF1: Overlapping slot detected → system rejects slot creation and displays conflict message.
Exception Flows	• Database or connection error → slot creation aborted, system logs error.

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

Use-case ID	UC-03b
Use-case name	Session Booking, Reminders, and Cancellation Rules
Overview	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.
Actors	Student, Tutor, System
Preconditions	• Tutor has published available slots. • Student is authenticated and matched with a tutor.
Trigger	Student initiates a booking or a cancellation/reschedule request.
Main Flow (Steps)	1. Student views tutor's available consultation slots. 2. Student selects a preferred slot and submits booking. 3. System checks for double booking or slot conflicts. 4. Booking is confirmed, and notifications are sent to student and tutor. 5. System automatically sends reminders (24h and 1h before the session). 6. If cancellation or reschedule is requested $\geq 2h$ before session it is approved.
Postconditions	• Session status updated (booked, rescheduled, or cancelled). • All notifications are logged and delivered.
Alternative Flows	AF1: Cancellation request too late (<2h before) → denied with reason. AF2: Tutor unavailable → student notified and slot reopened.
Exception Flows	• Network or database failure → transaction aborted, retry suggested.

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

6.5 UC-04 Feedback & Progress Tracking

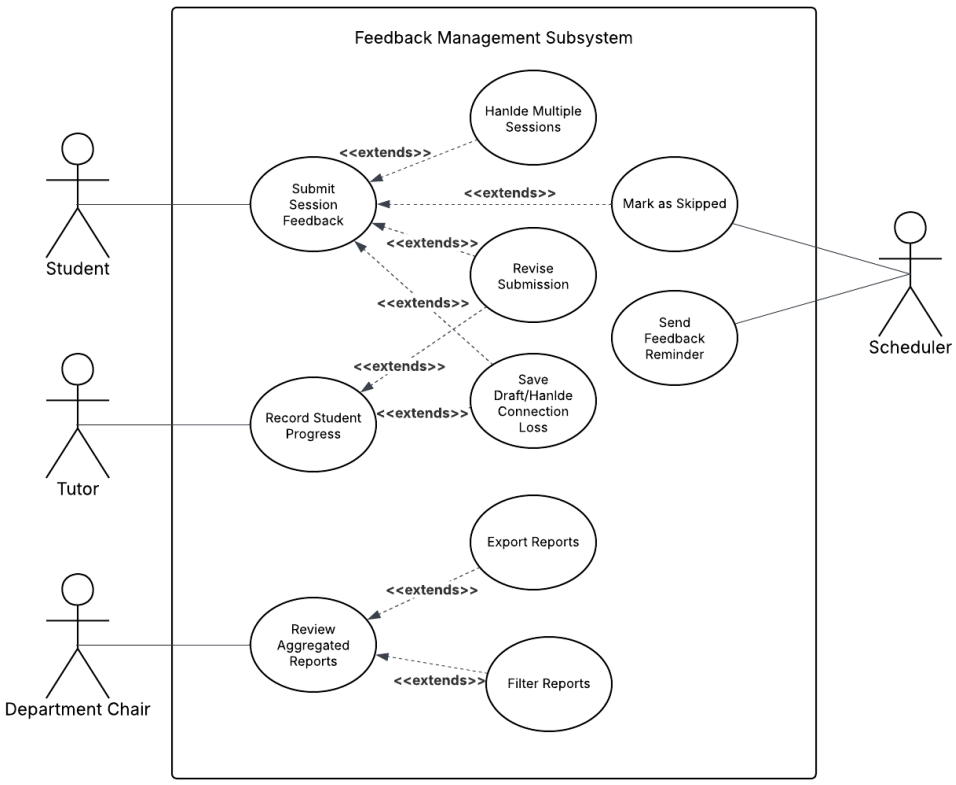

Figure 5: Submit Session Feedback

Use-case ID	UC-04a
Use-case name	Submit Session Feedback
Use-case overview	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.
Actors	• Student (primary) • Scheduler (secondary, for reminders and marking 'Skipped')
Preconditions	1. A tutoring session has been completed. 2. The system is running and accessible. 3. Student is authenticated in the system.
Trigger	The student clicks the "Submit Feedback" option after the session is completed.

Main Flow (Steps)	1. System displays a feedback form linked to the completed session. 2. Student fills in and submits the structured feedback. 3. System validates the input and saves the feedback in the database. 4. System links the feedback to the corresponding session.
Postconditions	• Feedback is stored in the database and linked to the correct session. • If skipped, the system records a "Feedback Skipped" status for that session. • Data is available for tutors and department chairs in aggregated reports.
Alternative Flows	1. **Multiple Session Feedback:** If multiple sessions are pending, the system displays a list and allows the student to submit feedback sequentially. 2. **Draft Save/Connection Loss:** The student may save incomplete feedback as a draft and return later. System automatically saves the draft if connection is lost before final submission. 3. **Feedback Revision:** Within a defined grace period (e.g., 24h), the student may edit and resubmit feedback; the updated version replaces the original. 4. **Skipped Feedback:** If no feedback is submitted within the allowed time, the Scheduler triggers a process to mark the session's feedback status as "Skipped."
Exception Flows	• **Database Error:** 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. • **Missing Session Record:** If the session record is missing or corrupted, the system logs an error and notifies the coordinator.

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

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

Preconditions	1. A tutoring session has been completed. 2. The system is running and accessible. 3. Tutor is authenticated in the system.
Trigger	Tutor accesses the session record and selects "Record Progress."
Steps	1. Tutor opens the specific session record. 2. Tutor enters progress notes and, optionally, a session summary. 3. Tutor submits the record. 4. System validates the input and saves the record, linking it to the session.
Postconditions	1. Student progress and optional session summary are saved. 2. Data is linked to the session and available for departmental review.
Alternative Flows	A1: Optional Summary → Tutor skips writing a session summary (only progress notes are saved). A2: Draft Save / Connection Loss → Tutor may save incomplete notes as a draft; the system auto-saves the draft if the connection is lost. A3: Note Revision → Within a 24-hour grace period, tutor may edit and resubmit notes; the new version replaces the previous record.
Exception Flow	1. Database error → System logs the issue and notifies the tutor to retry. 2. Missing or locked session → System notifies the tutor and logs the error.
Priority	Should

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

Use-case ID	UC-04c
Use-case name	Review Aggregated Reports (Feedback & Progress)
Overview	To allow the department chair to review aggregated feedback and progress data for monitoring tutor performance and program effectiveness.
Actors	Department Chair
Preconditions	1. Feedback and progress records exist in the database. 2. The system is running and accessible. 3. Department Chair is authenticated in the system.
Trigger	Department Chair selects "Aggregated Reports Overview."

Main Flow (Steps)	1. Department Chair selects report criteria (e.g., date range, tutor, course). 2. System retrieves all relevant feedback and progress data. 3. System processes and displays aggregated statistics, charts, or reports.
Postconditions	• Aggregated data is available for decision-making. • Reports may be exported or stored for institutional use.
Alternative Flows	AF1 **Filtering:** Department Chair filters or drills down data by tutor, course, or specific time period. AF2 **Export Report:** Department Chair exports the report in a chosen format (e.g., PDF, CSV) for external analysis.
Exception Flows	• **Database/Retrieval Failure:** If database retrieval fails, the system shows a technical error message and logs the issue for system administrators. • **No Data:** If no data matches the selected criteria, the system displays a clear message: "No data available for this selection."

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

6.6 UC-05 Reporting & Analytics

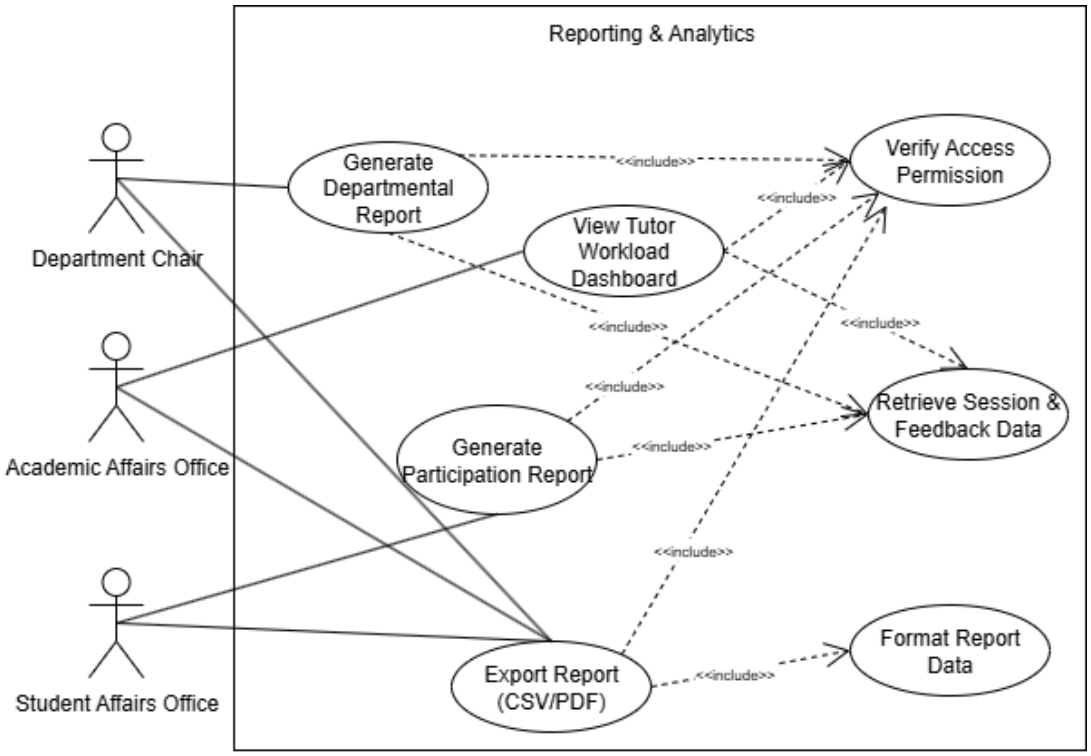

Figure 6: Reporting and Analytics

Use-case ID	UC-05a
Use-case name	Generate Departmental Report
Use-case overview	Department Chair generates a department-level report (attendance, performance, session counts) for academic monitoring and decisions.
Actors	Department Chair
Preconditions	1. Authenticated and authorized to view departmental reports. 2. Sessions, feedback, and (if applicable) progress data exist. 3. System is operational.
Trigger	Chair opens "Reporting" and selects "Departmental Report".
Steps	1. Set filters (term/date, program/department, tutor/cohort). 2. System retrieves and analyzes metrics (attendance, performance, session counts). 3. System displays tables/charts with data-source notes. 4. Optional: Export CSV/PDF with metadata (generation time, filters, version); log action.
Postconditions	1. Departmental report displayed; optional file generated. 2. Action recorded in the audit log.
Alternative Flows	1. Change filters/scope → system refreshes results.
Exception Flow	1. No data for selected criteria → show "No data available for this selection." 2. Export error (I/O or size) → suggest narrowing scope or retrying. 3. Data source unavailable → show service notice; allow retry when available.

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

Use-case ID	UC-05b
Use-case name	View Tutor Workload Dashboard
Use-case overview	Academic Affairs views tutor workload and student demand to allocate resources effectively.
Actors	Academic Affairs Office
Preconditions	1. Authenticated and authorized for workload dashboards. 2. Sessions/booking and feedback data exist. 3. System is operational.
Trigger	Office opens "Workload & Demand" dashboard.
Steps	1. Select filters (term/date, department/program). 2. System aggregates tutor load and demand indicators. 3. System displays dashboard (tables/charts, drill-down). 4. Optional: Export CSV/PDF with metadata; log action.
Postconditions	1. Workload dashboard visible; optional export ready. 2. Access recorded in the audit log.
Alternative Flows	1. Drill-down by tutor/program → refresh view.
Exception Flow	1. No data for selected criteria → show "No data available for this selection." 2. Export error (I/O or size) → suggest narrowing scope or retrying. 3. Data source unavailable → show service notice; allow retry when available.

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

Use-case ID	UC-05c
Use-case name	Generate Participation Report
Use-case overview	Student Affairs generates participation reports for credit/scholarship consideration.
Actors	Student Affairs Office
Preconditions	1. Authenticated and authorized for participation reports. 2. Sessions and attendance/feedback records exist. 3. System is operational.
Trigger	Office selects "Participation Report".
Steps	1. Choose filters (term/date, cohort, program, thresholds). 2. System compiles participation metrics and eligibility indicators. 3. System displays results; Optional: export CSV/PDF with metadata; log action.
Postconditions	1. Participation report displayed; optional file generated. 2. Action logged.
Alternative Flows	1. Adjust filters/criteria → refresh results.
Exception Flow	1. No data for selected criteria → show "No data available for this selection." 2. Export error (I/O or size) → suggest narrowing scope or retrying. 3. Data source unavailable → show service notice; allow retry when available.

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

6.7 UC-06 Integration with HCMUT Infrastructure

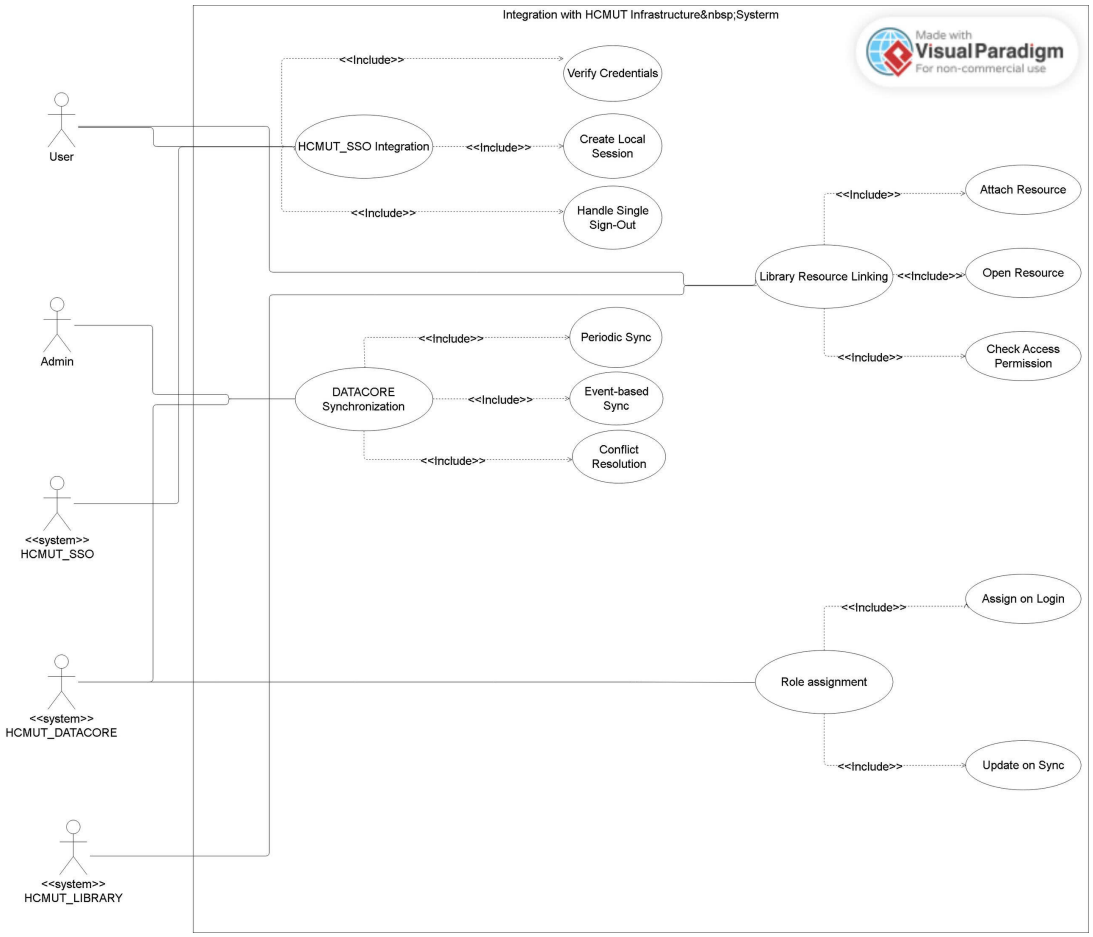

Figure 7: Integration with HCMUT Infrastructure

Use-case ID	UC-06a
Use-case name	HCMUT_SSO Integration
Use-case overview	The system integrates with HCMUT_SSO for unified authentication, allowing users to log in using university credentials and automatically manage single sign-out.
Actors	User, System, HCMUT_SSO
Preconditions	HCMUT_SSO service is operational and reachable.
Trigger	User initiates login via HCMUT_SSO.
Steps	1. User selects "Login with HCMUT_SSO". 2. System redirects to the authentication portal. 3. HCMUT_SSO validates credentials and returns a token. 4. System verifies the token and creates a session. 5. Upon sign-out at SSO, the system terminates the local session.
Postconditions	1. User is successfully authenticated. 2. Single sign-out ensures session consistency.
Exception Flow	1. Invalid or expired token → login attempt rejected. 2. SSO service unavailable → system displays maintenance message.
Priority	Must

Table 12: Use Case UC-06a: HCMUT_SSO Integration

Use-case ID	UC-06b
Use-case name	DATACORE Synchronization
Use-case overview	The system synchronizes personal and academic data from HCMUT_DATACORE periodically or in near real-time to ensure data consistency and reduce manual entry.
Actors	System, HCMUT_DATACORE, Administrator
Preconditions	DATACORE APIs are online and accessible.
Trigger	Scheduled synchronization or data-change event detected.
Steps	1. System triggers synchronization with DATACORE. 2. Retrieve updated profiles and academic data. 3. Validate and compare data with local records. 4. Update local data using DATACORE as source of truth. 5. Log synchronization status and timestamp.
Postconditions	1. Local data mirrors DATACORE. 2. Synchronization events logged for auditing.
Exception Flow	1. Connection timeout or API error → retry with exponential backoff. 2. Invalid data format → skip record and log validation error.
Priority	Must

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

Use-case ID	UC-06c
Use-case name	Role Assignment
Use-case overview	The system automatically assigns user roles (student, tutor, coordinator, department chair, or administrator) based on centralized role data from DATACORE and SSO.
Actors	User, System, HCMUT_DATACORE
Preconditions	User is authenticated via SSO and DATACORE role data is available.
Trigger	User login or scheduled role update.
Steps	1. Retrieve role mapping from DATACORE. 2. Match user ID with centralized role data. 3. Assign permissions based on role. 4. Apply changes in access-control policies. 5. Log the role-assignment event.
Postconditions	1. User roles align with centralized data. 2. Updated permissions take effect immediately.
Exception Flow	1. Role data missing → assign default "student" role and notify admin. 2. Role conflict → logged and flagged for manual verification.
Priority	Must

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

Use-case ID	UC-06d
Use-case name	Library Resource Linking
Use-case overview	The system connects with HCMUT_LIBRARY to let tutors and students attach or access library materials securely within tutoring sessions or summaries.
Actors	Student, Tutor, System, HCMUT_LIBRARY
Preconditions	Library API and authentication services are available.
Trigger	User attaches or opens a library resource.
Steps	1. User searches or selects a library resource. 2. System requests metadata and access permissions. 3. Verify user eligibility via role-based access. 4. Attach or open the resource in session view. 5. Log the access event.
Postconditions	1. Resource successfully linked to the session or summary. 2. Access rights enforced per library policy.
Exception Flow	1. Access denied → system displays permission error. 2. Library service unavailable → prompt user to retry or queue attachment.
Priority	Could

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

6.8 UC-07 Advanced / Optional Features

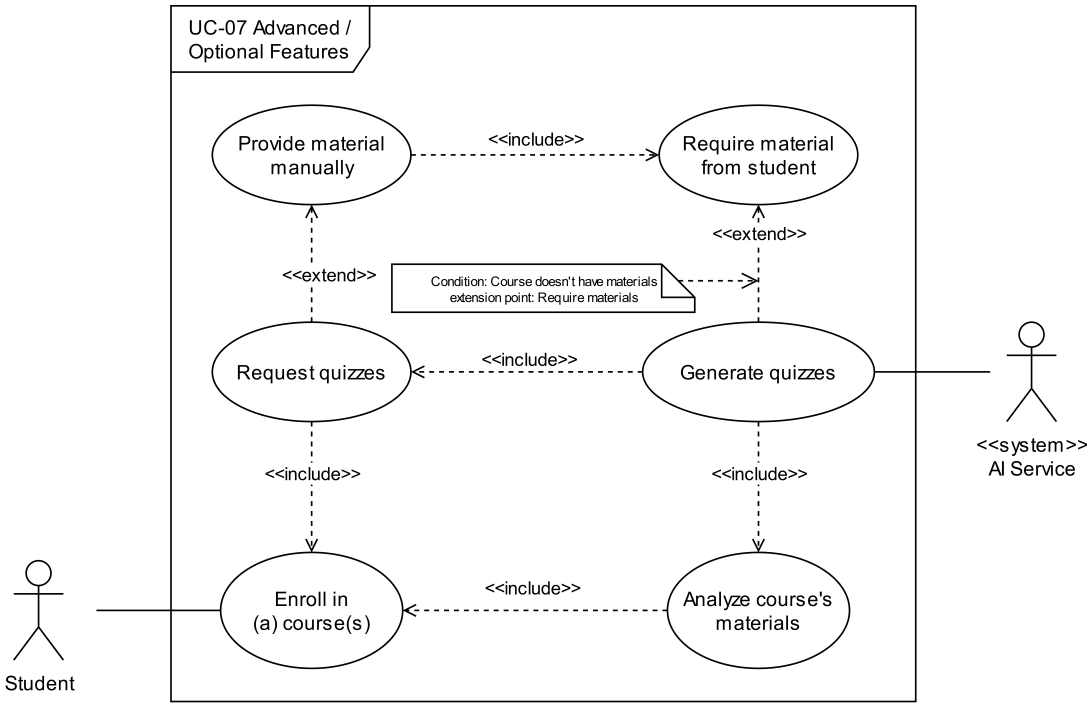

Figure 8: AI-generated review quizzes

Use-case ID	UC-07
Use-case name	AI-generated review quizzes
Use-case overview	Provide students with quizzes related to the courses they are enrolling.
Actors	Students, AI service
Preconditions	1. The system is running. 2. Internet connection is available. 3. AI service must be available. 4. The courses must have learning materials uploaded by tutors.
Trigger	Students use the AI-based quiz generation function.
Steps	1. Retrieve uploaded course materials of the course. 2. Process and analyze the content with AI. 3. Search the internet for related academic resources and quizzes. 4. Generate quiz questions covering the key topics. 5. Display the quiz to the requested students.
Postconditions	1. The quizzes are displayed on the screen of the requested students, and they can download the quizzes as a document file. 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
Alternative Flows	- 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.
Exception Flow	1. If AI service isn't available, display an error.

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

7 Sequence Diagram

7.1 SQ-01 Log in, Log out & Profile Management

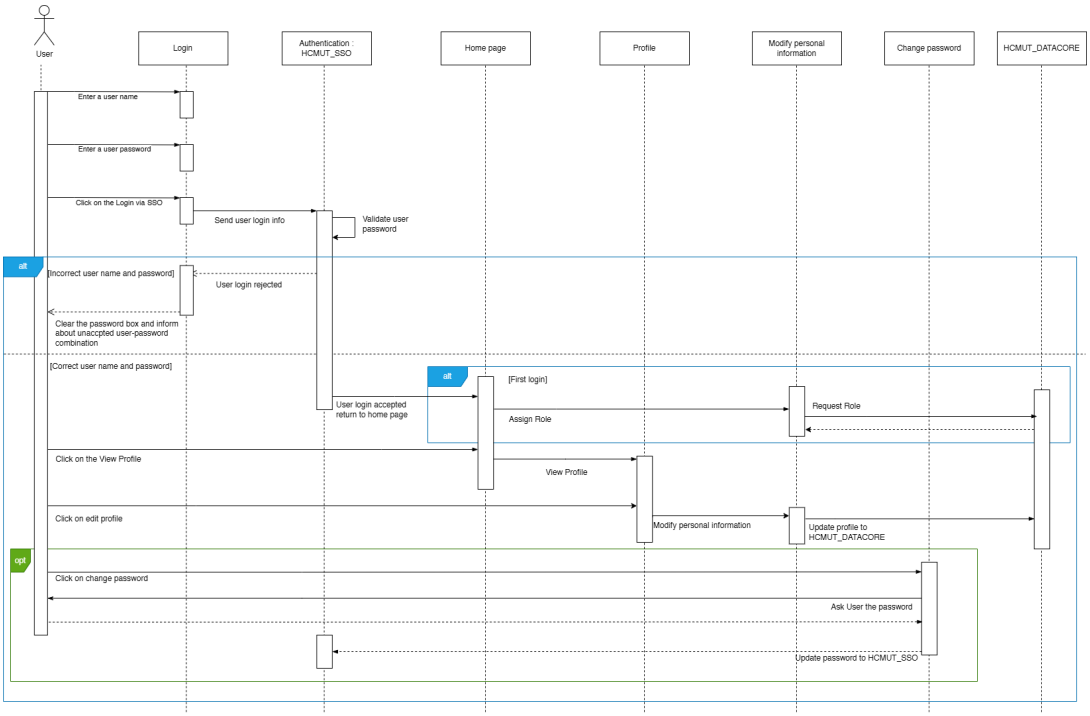

Figure 9: Log in, Log out & Profile Management

7.2 SQ-02 Tutor-Student Matching

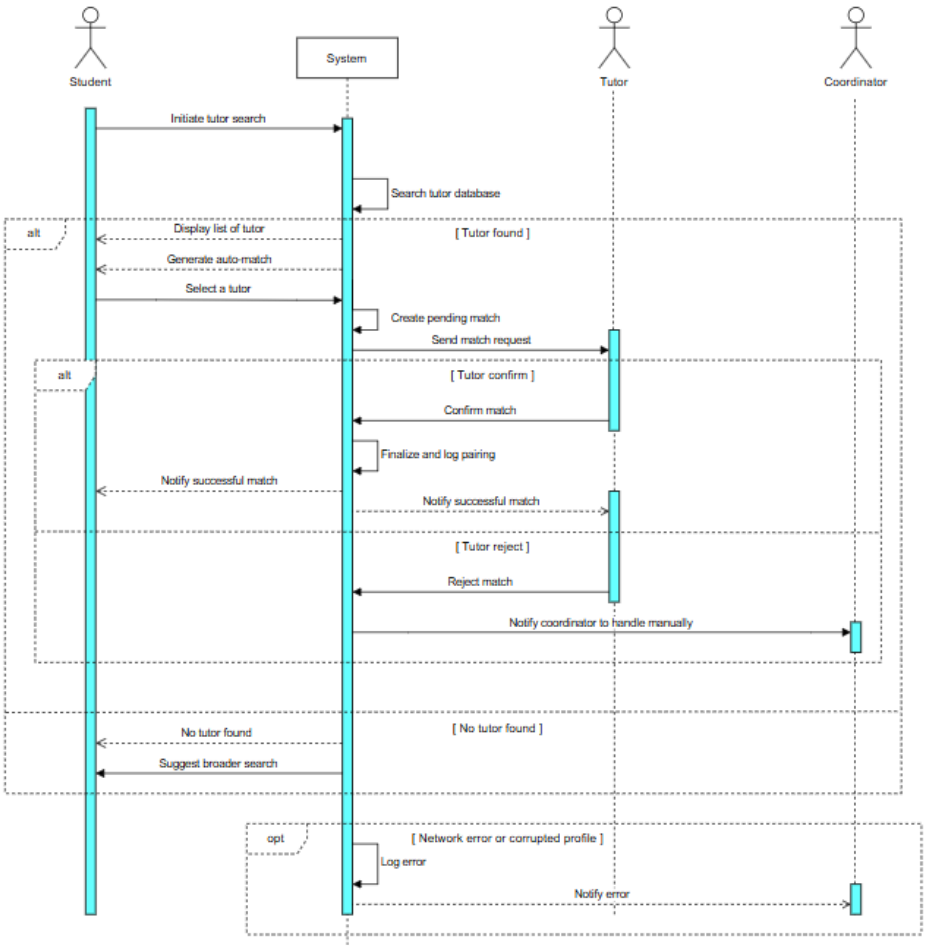

Figure 10: Tutor-Student Matching

7.3 SQ-03 Session Scheduling Management

Session Scheduling Management

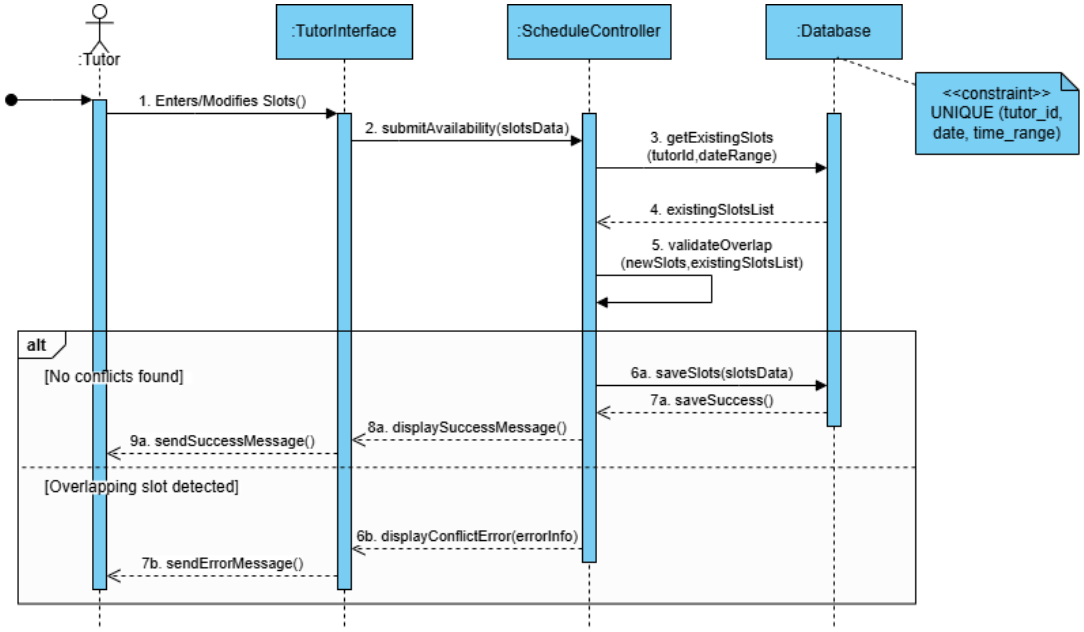

Figure 11: Tutor published available time slots

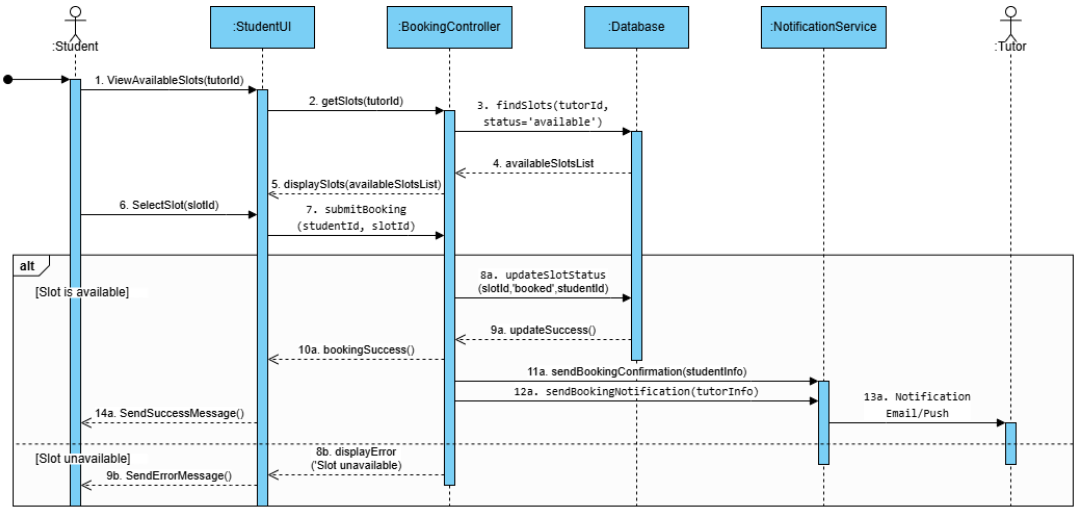

Figure 12: Student view and book session

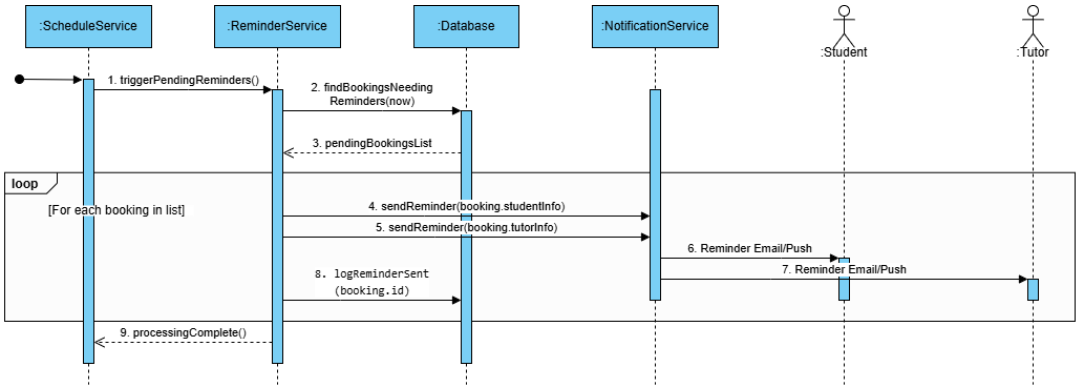

Figure 13: System sends notification reminders

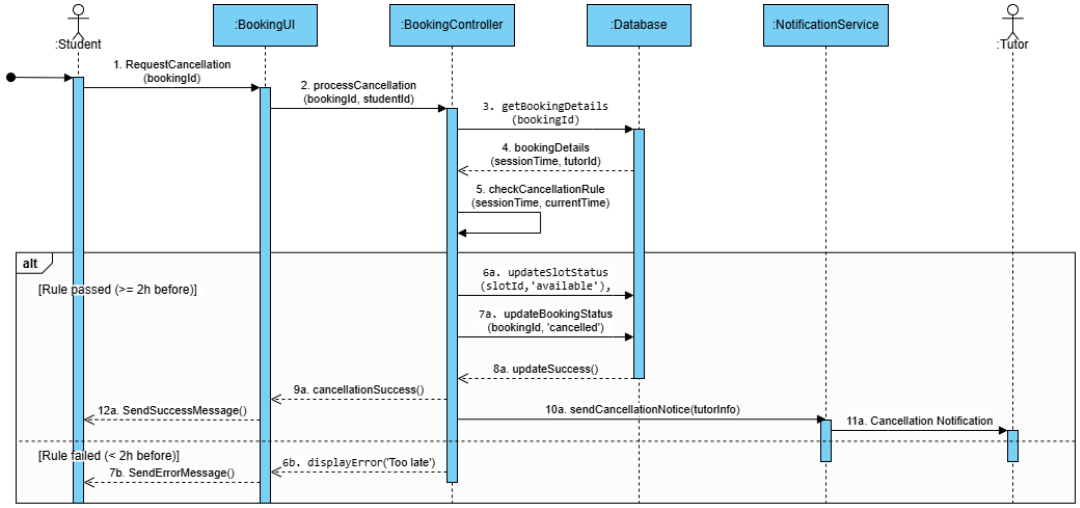

Figure 14: Cancellation and rescheduling

7.4 SQ-04 Feedback & Progress Tracking

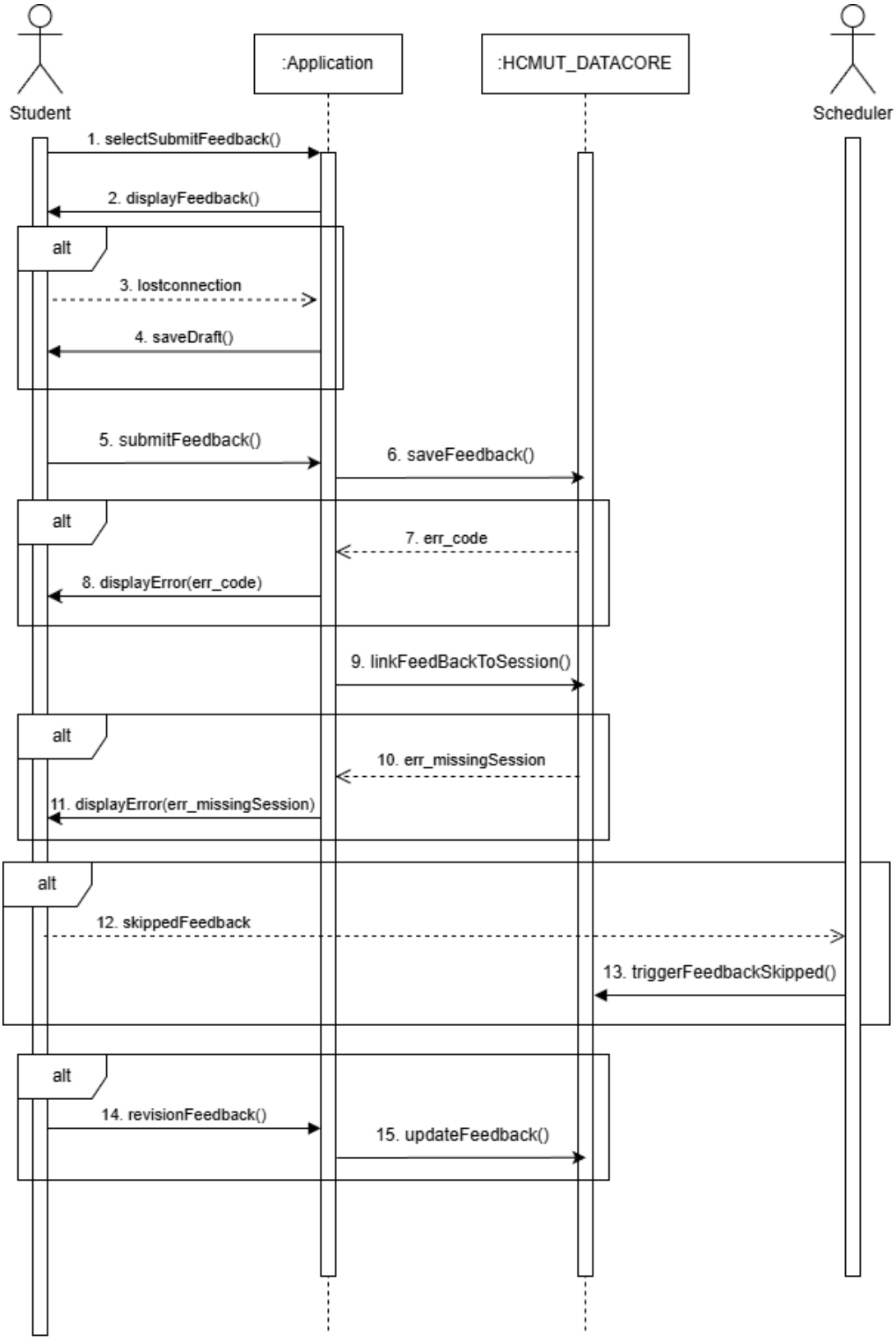

Figure 15: Feedback Submission

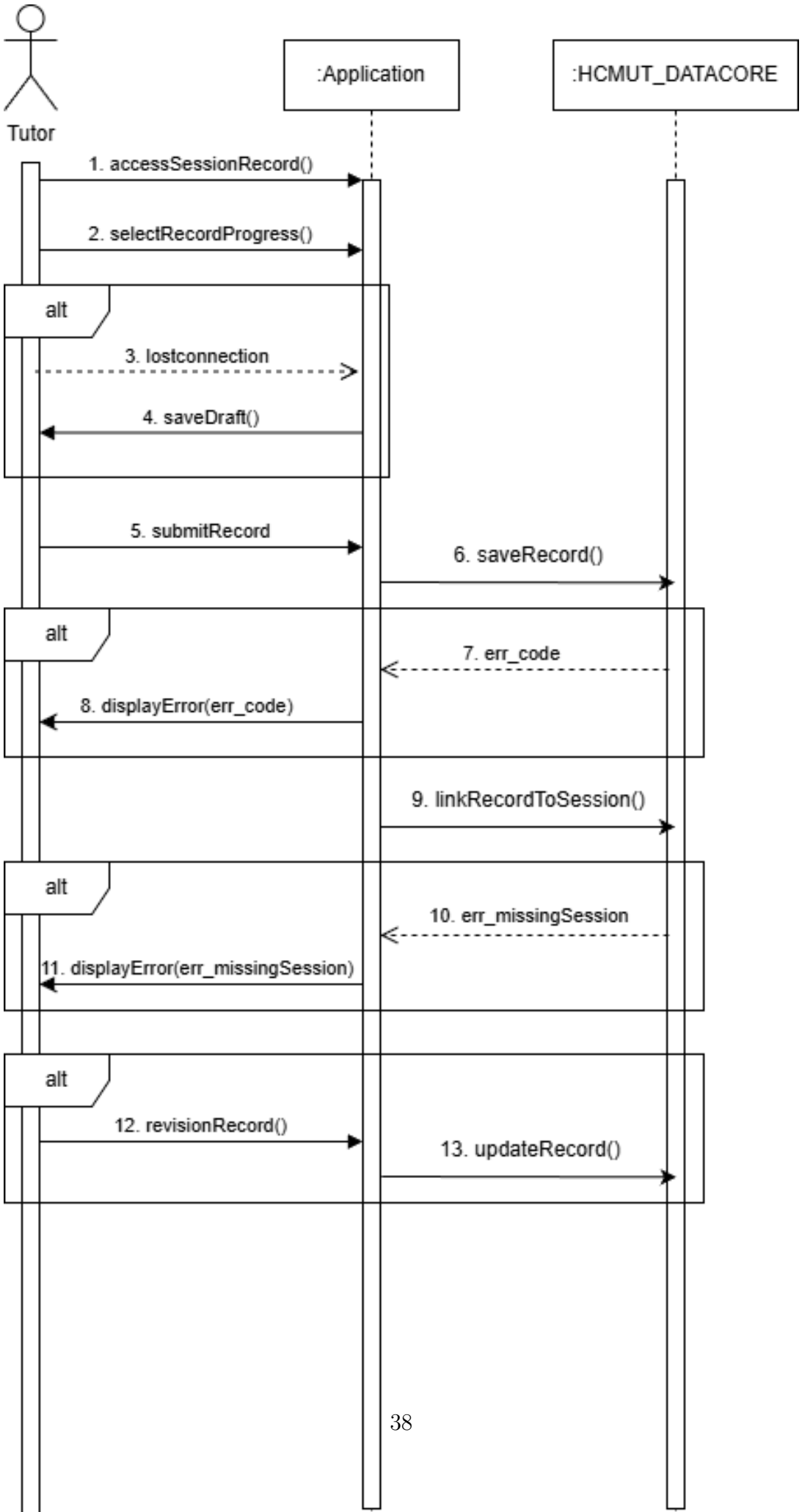
Tutor
:Application
:HCMUT_DATACORE
1. accessSessionRecord()
2. selectRecordProgress()
alt
3. lostconnection
4. saveDraft()
5. submitRecord
6. saveRecord()
alt
7. err_code
8. displayError(err_code)
9. linkRecordToSession()
alt
10. err_missingSession
11. displayError(err_missingSession)
alt
12. revisionRecord()
13. updateRecord()

Figure 16: Session Recording Access

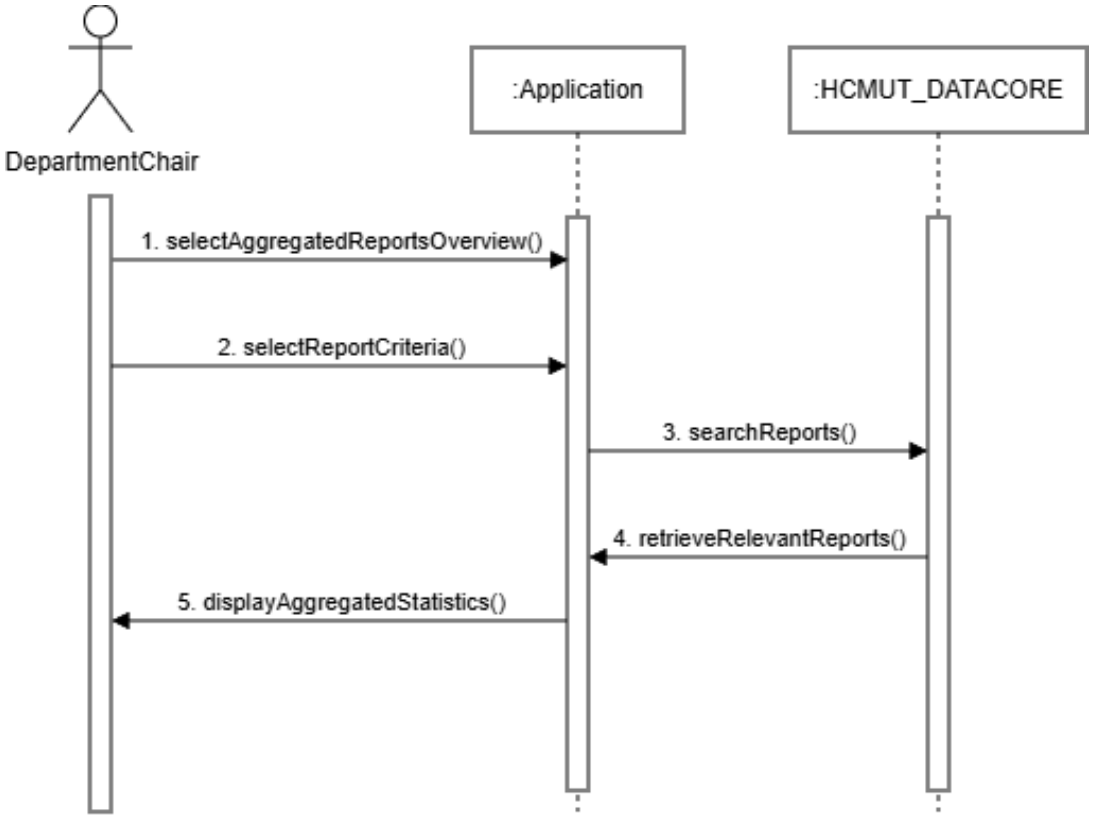

Figure 17: Report Tracking

7.5 SQ-05 Reporting & Analytics

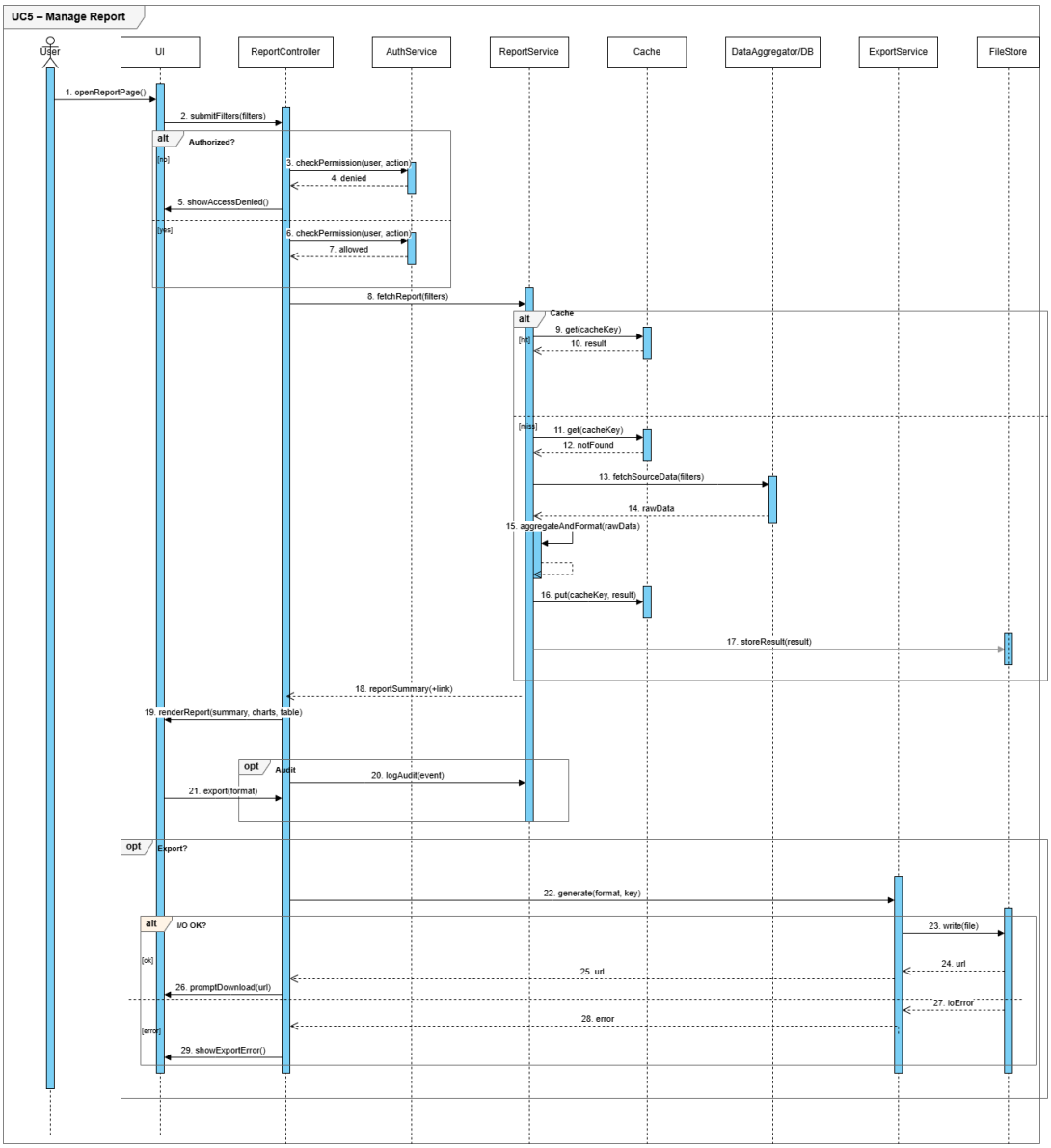

Figure 18: Reporting & Analytics

8 Activity Diagram

8.1 AC-01 Log in, Log out & Profile Management

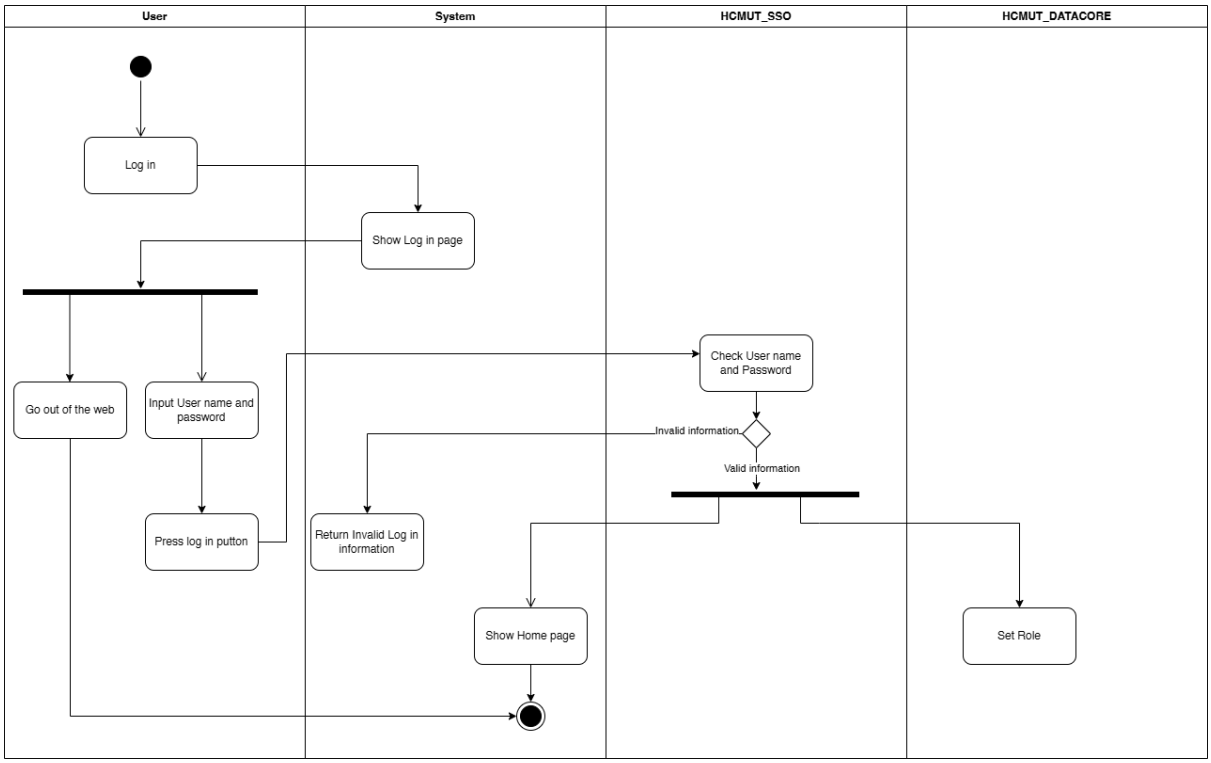

Figure 19: Log in

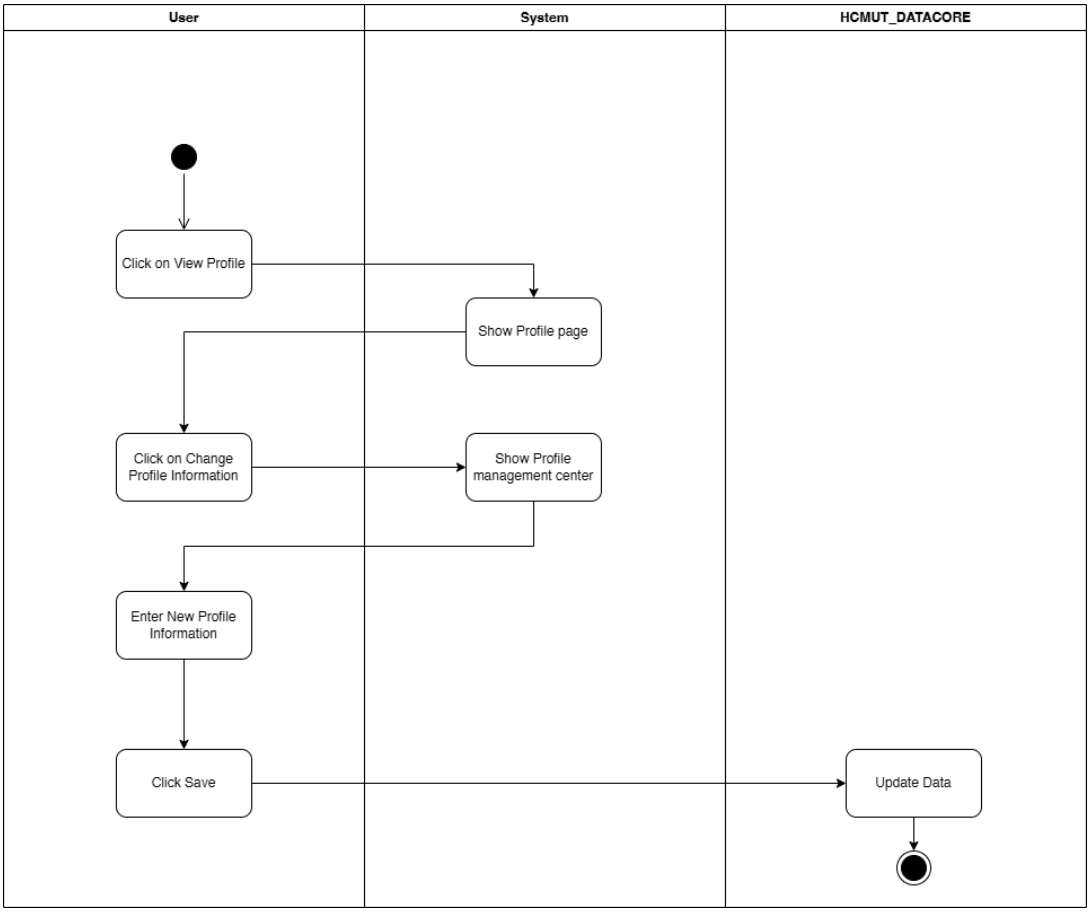

Figure 20: View Profile & Edit Profile

8.2 AC-02 Tutor-Student Matching

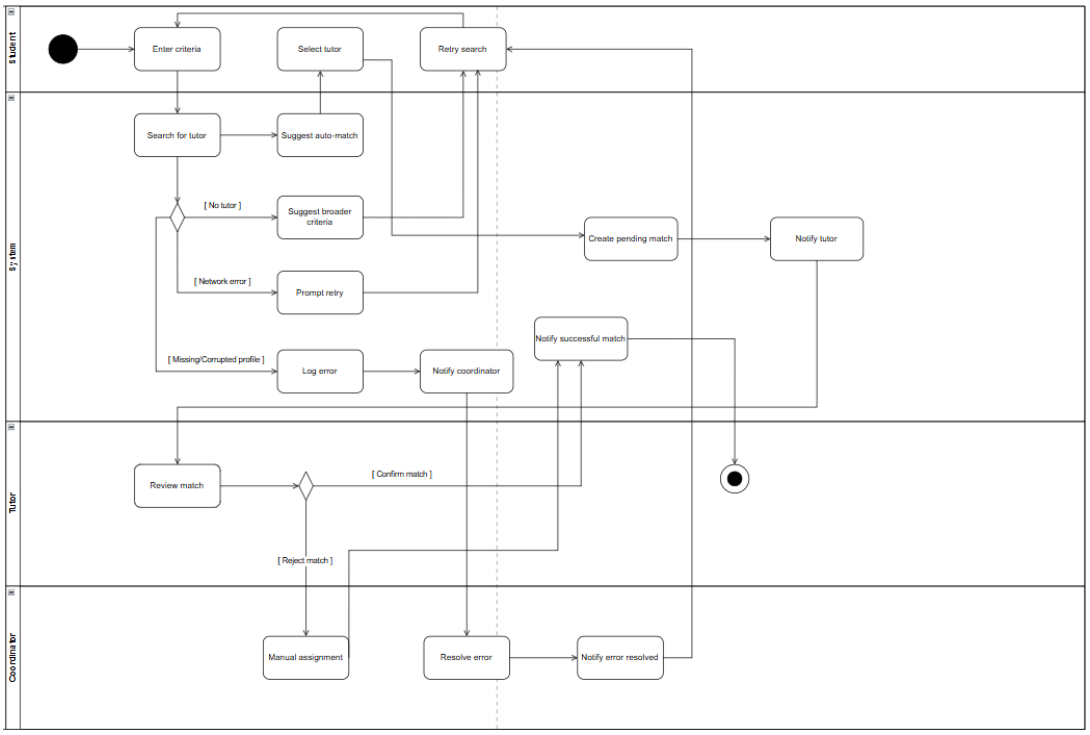

Figure 21: Tutor-Student Matching

8.3 AC-03 Session Scheduling Management

Session Scheduling Management

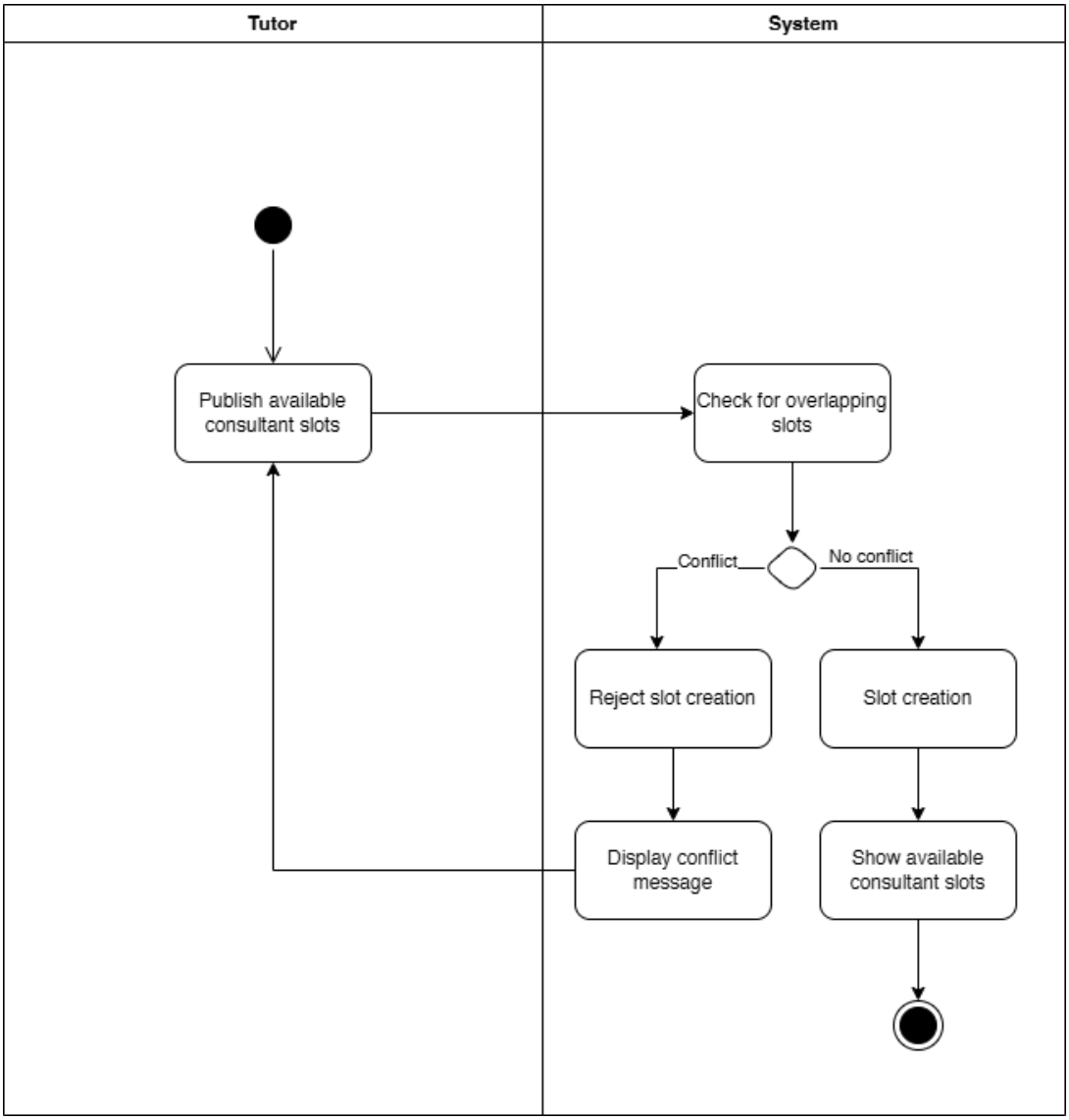

Figure 22: Tutor published available time slots

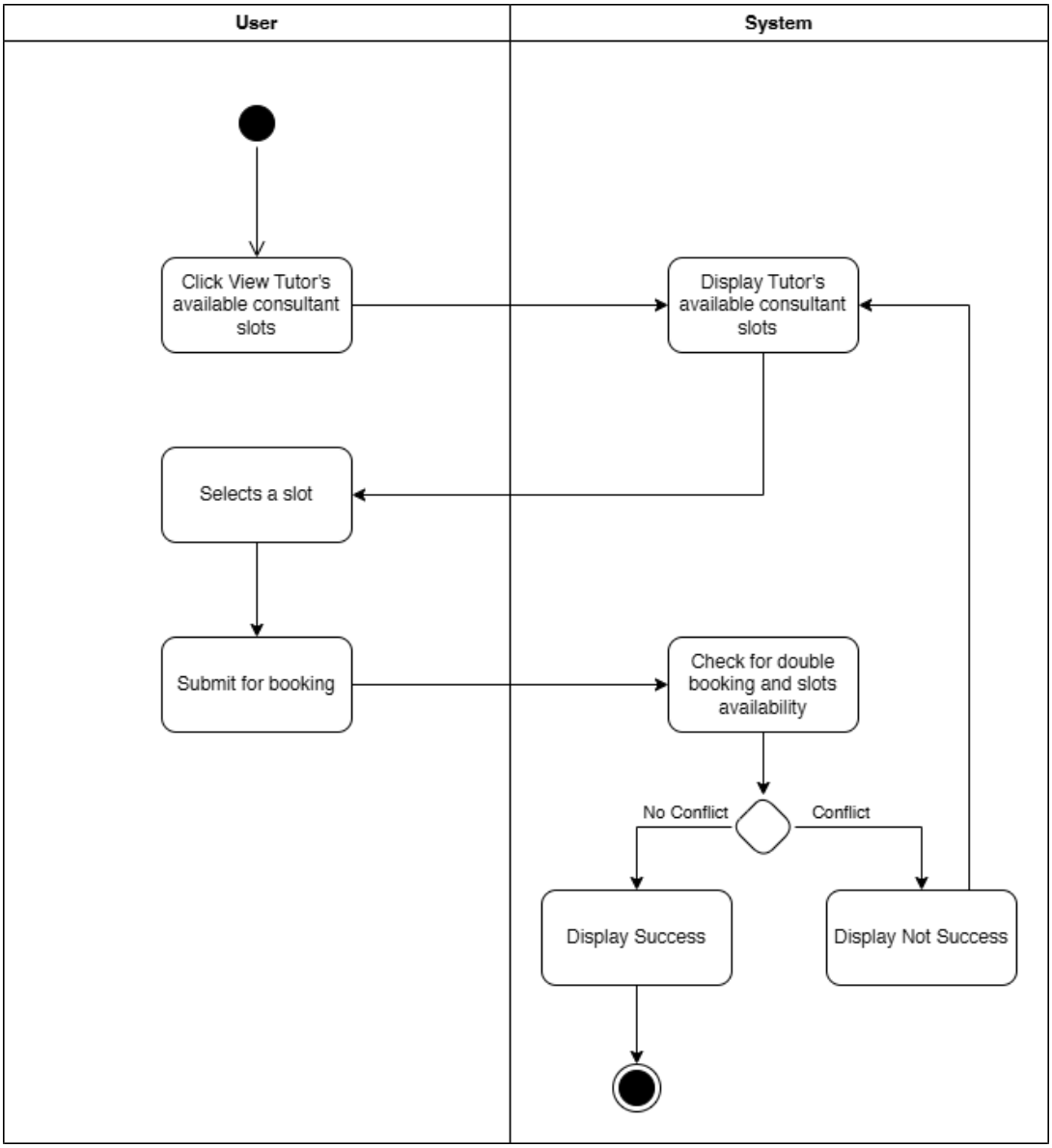

Figure 23: Student view and book session

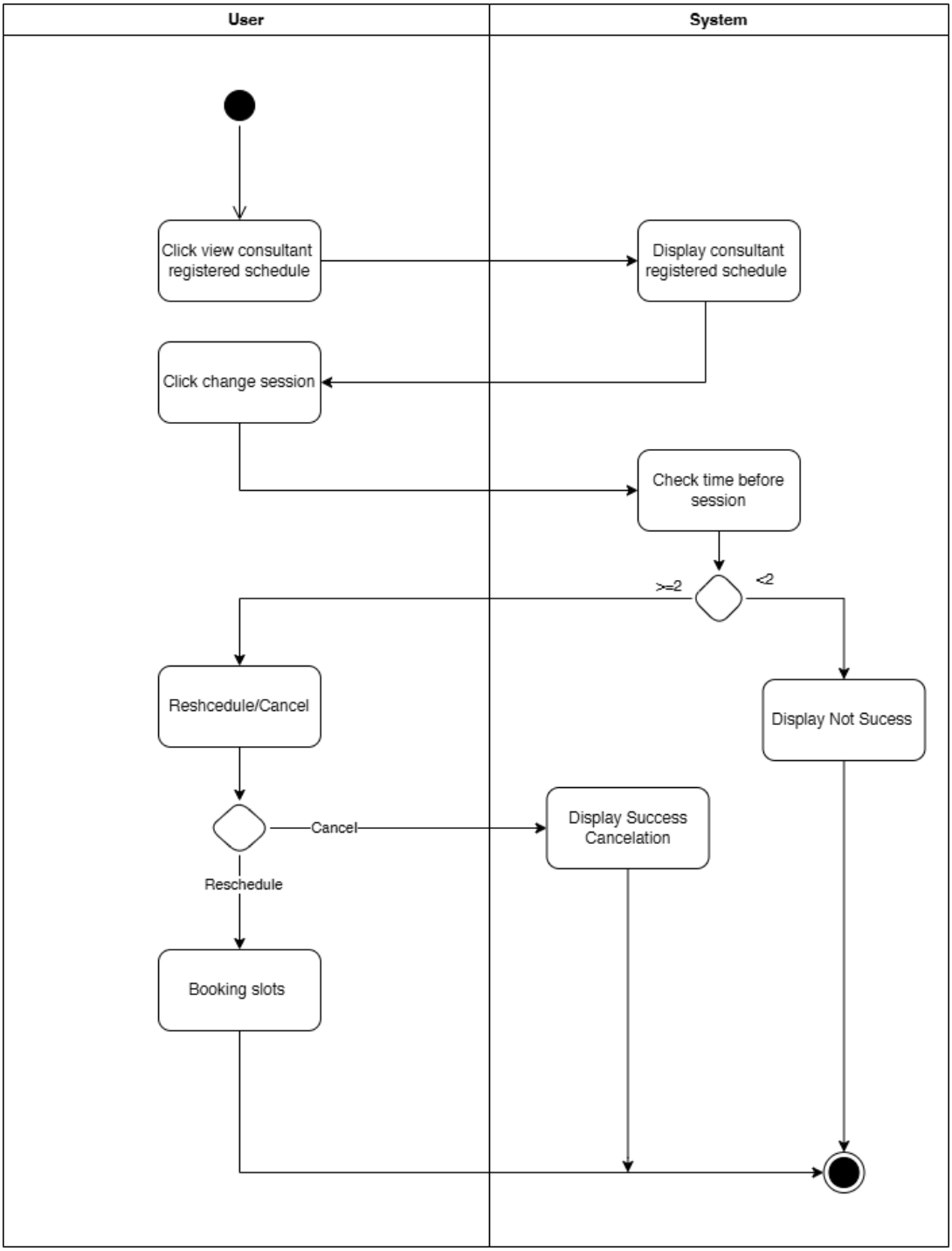

Figure 24: Cancellation and rescheduling

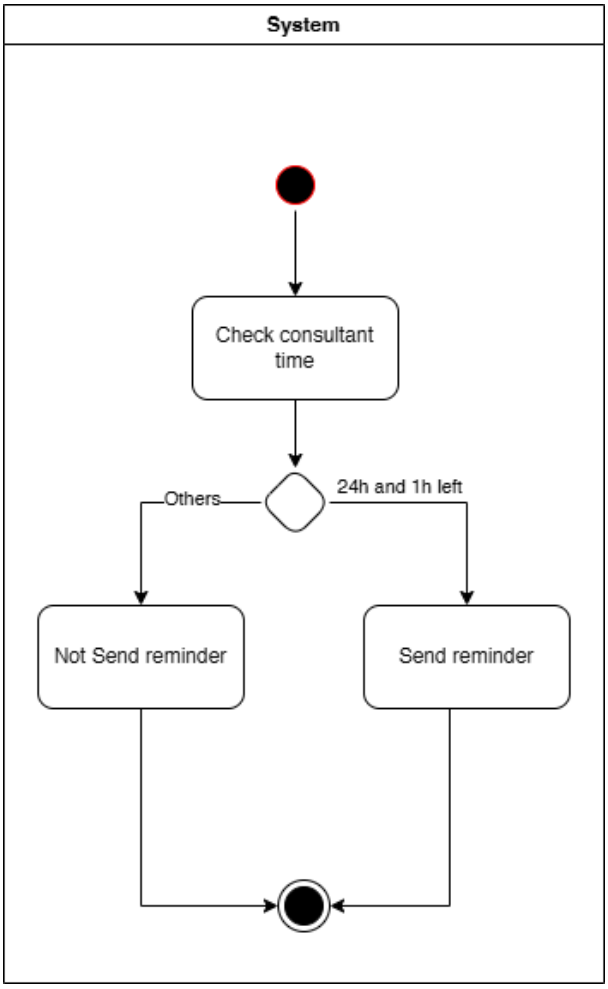

Figure 25: System sends notification reminders

8.4 AC-04 Feedback & Progress Tracking

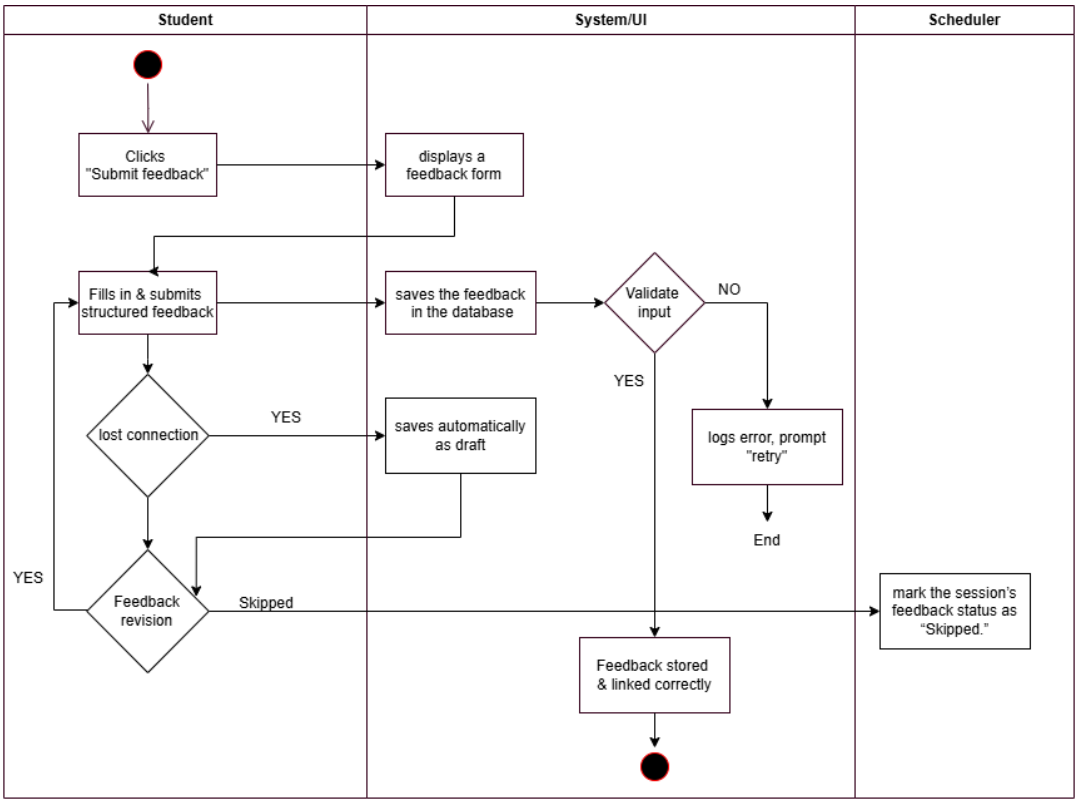

Figure 26: Feedback Submission

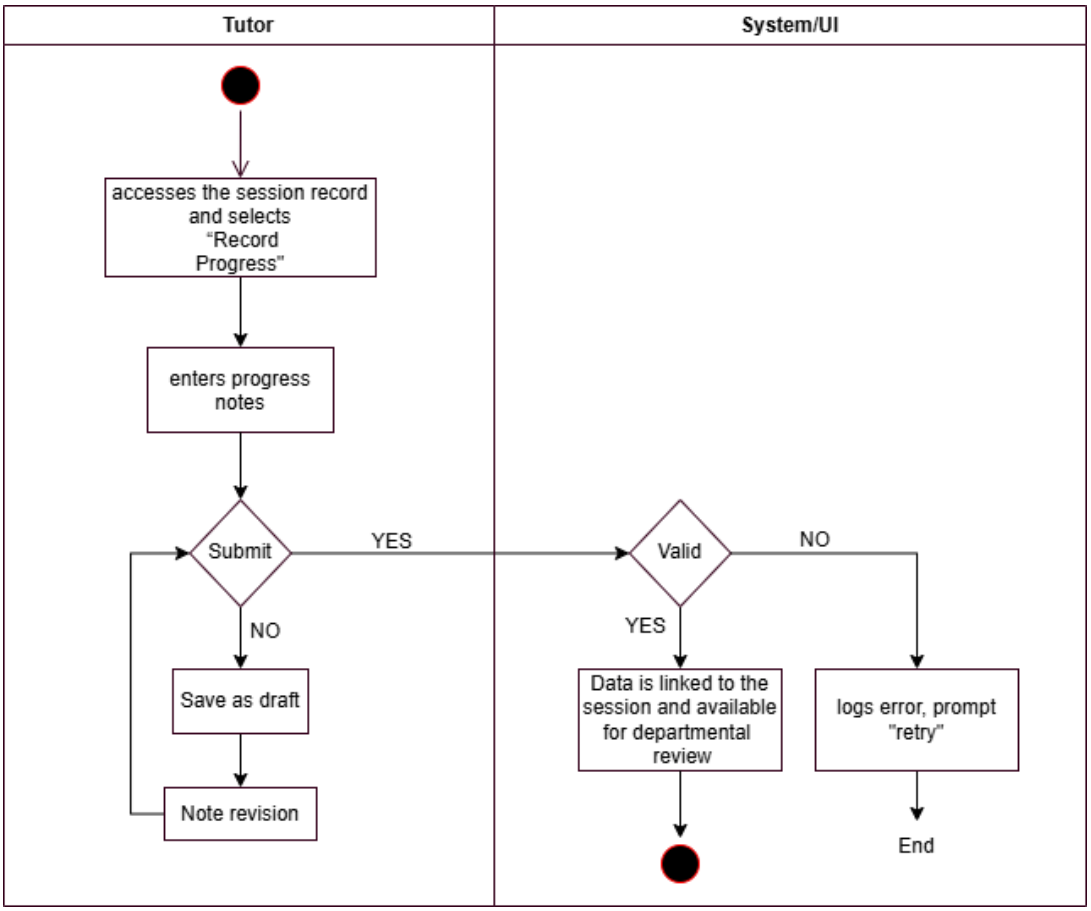

Figure 27: Session Recording Access

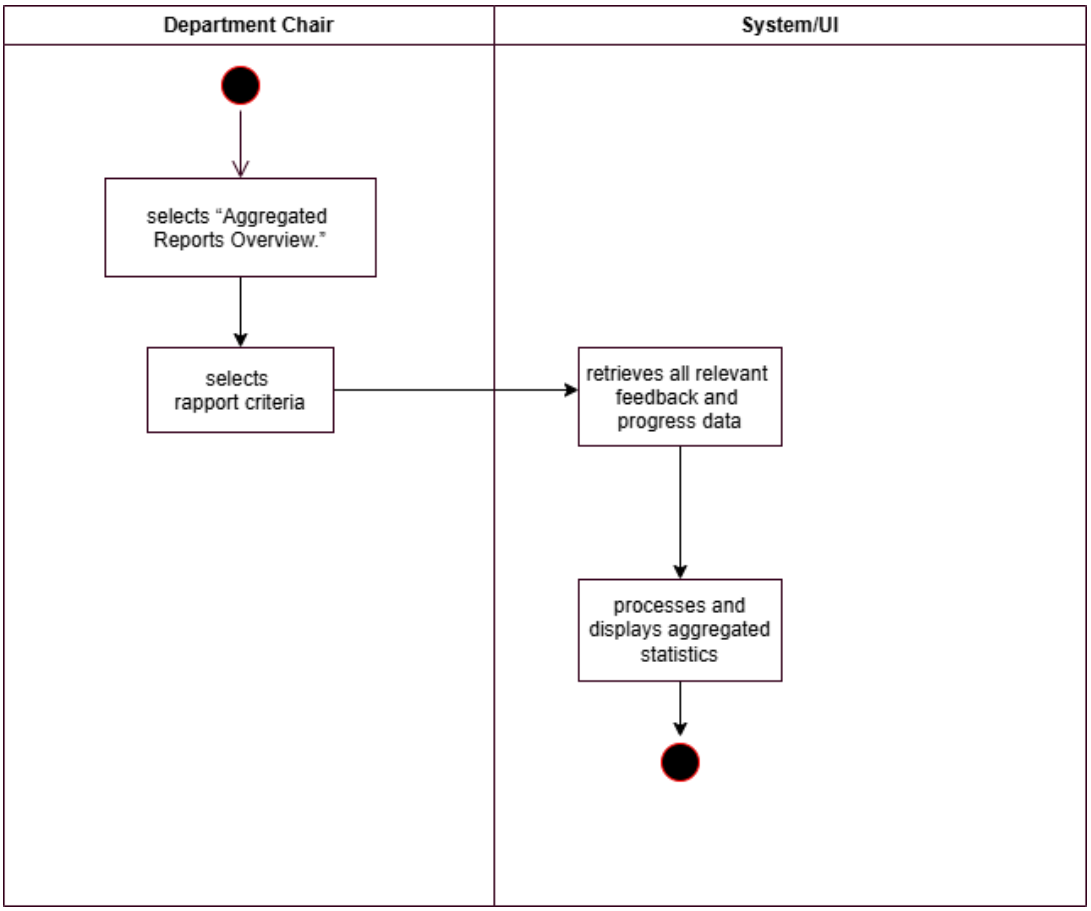

Figure 28: Report Tracking

8.5 AC-05 Reporting & Analytics

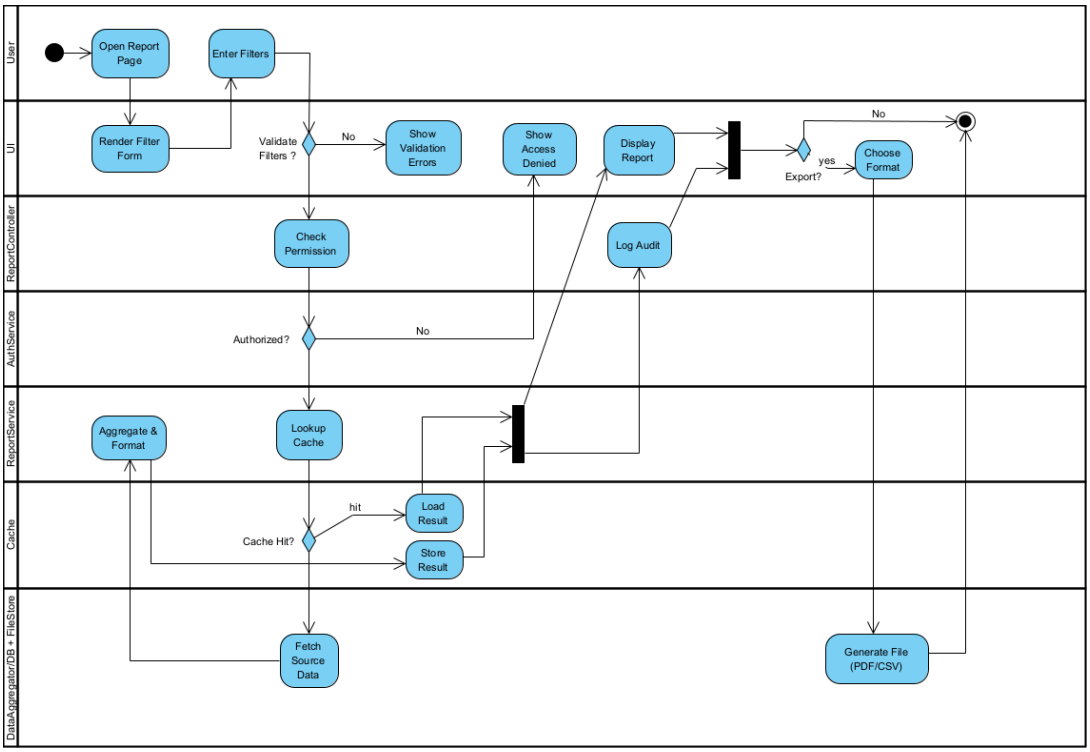

Figure 29: Reporting & Analytics

9 State Diagram

9.1 SD-01 Log in, Log out & Profile Management

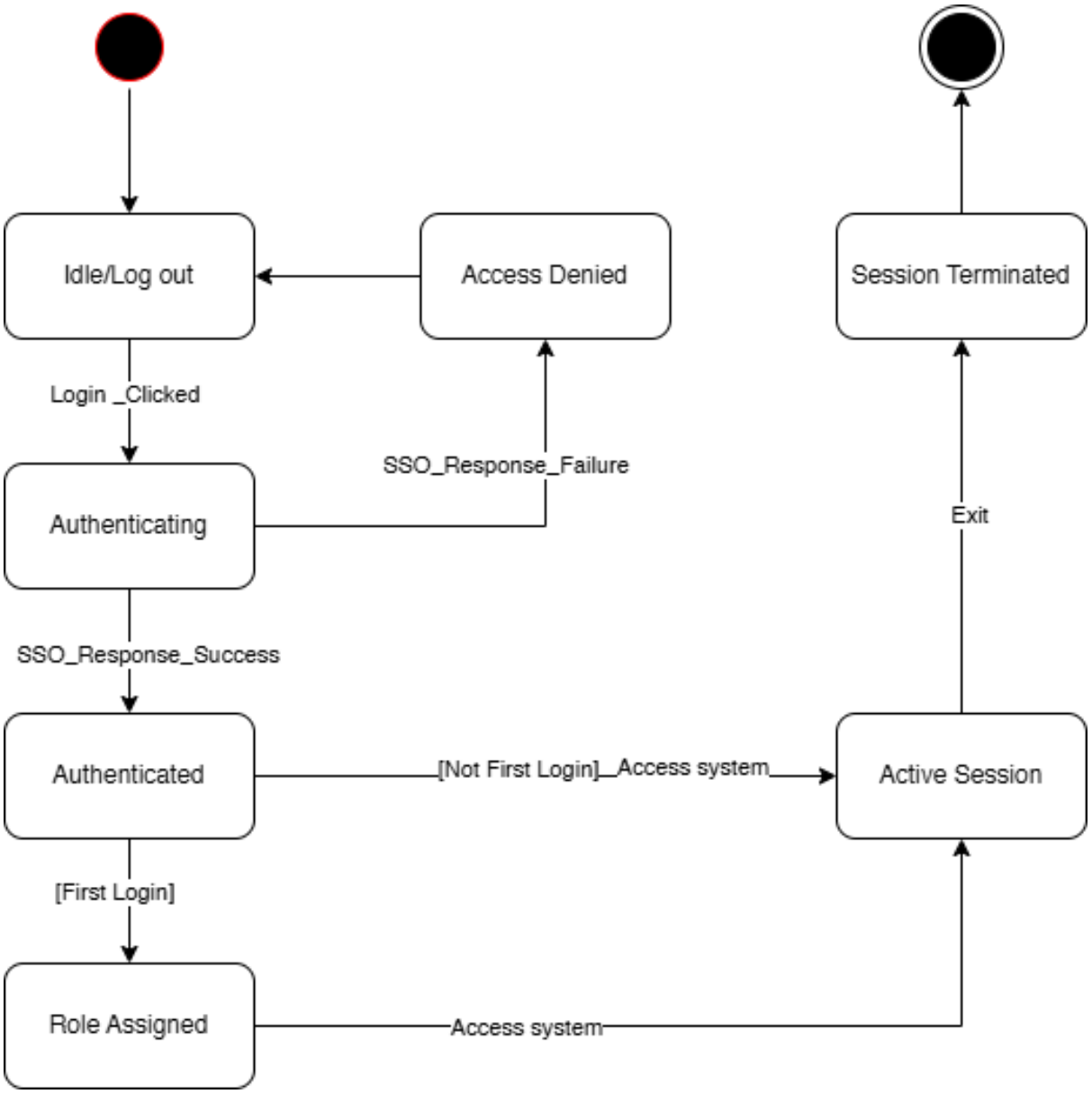

Figure 9: Log in

9.2 SD-02 Tutor-Student Matching

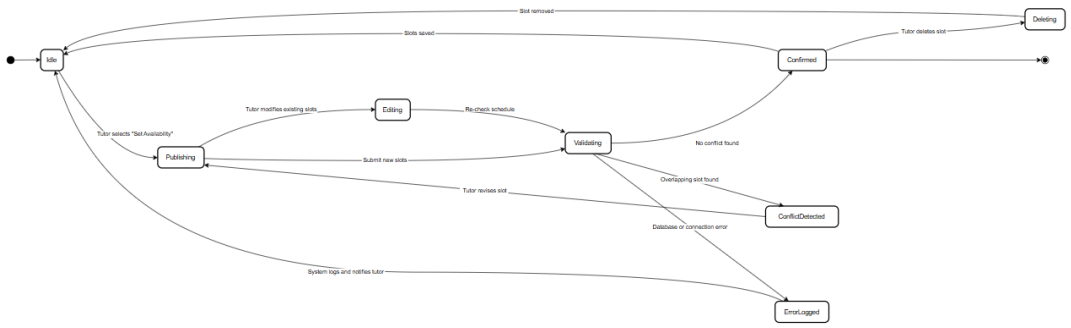

Figure 10: Tutor-Student Matching

9.3 SD-03 Session Scheduling Management

Session Scheduling Management

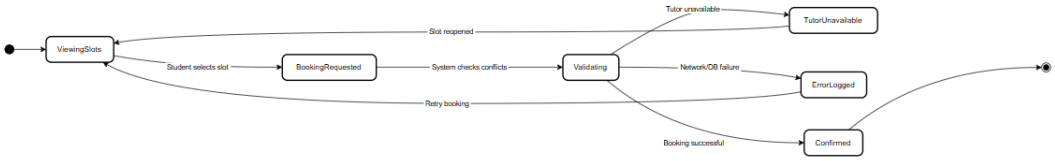

Figure 11: Booking a session

Figure 12: System Reminders

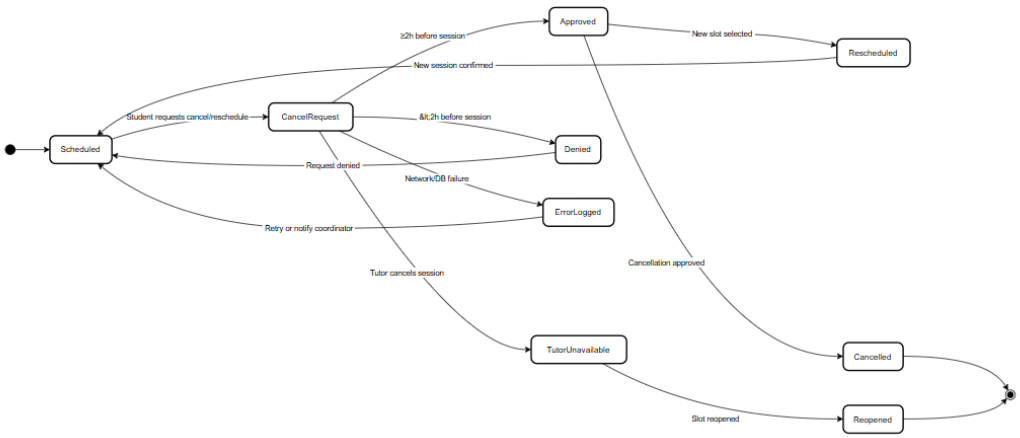

Figure 13: Cancellation and rescheduling

9.4 SD-04 Reporting & Analytics

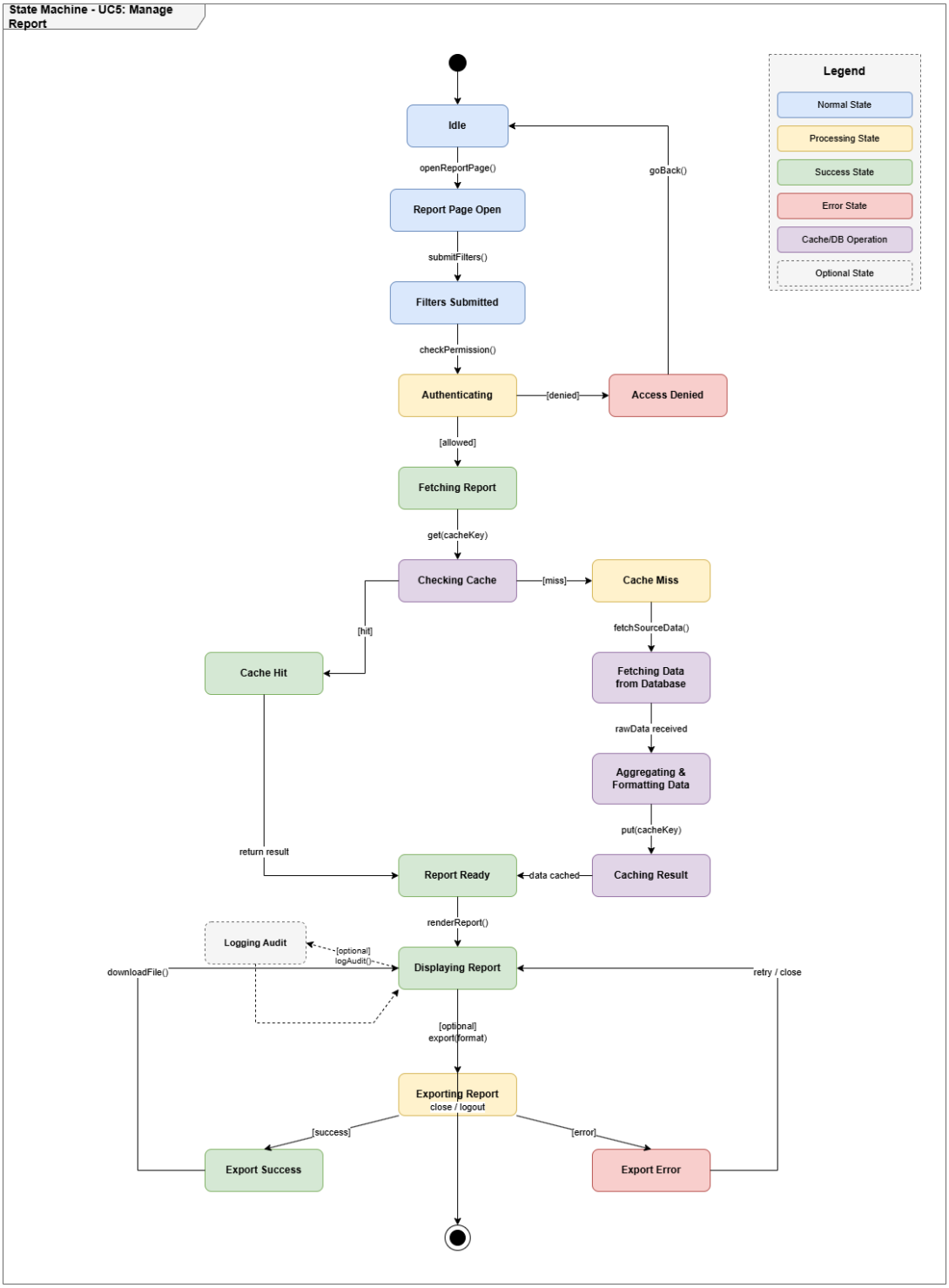

Figure 18: Manage Reports

10 Mockup

10.1 MU-01 Log in, Log out & Profile Management

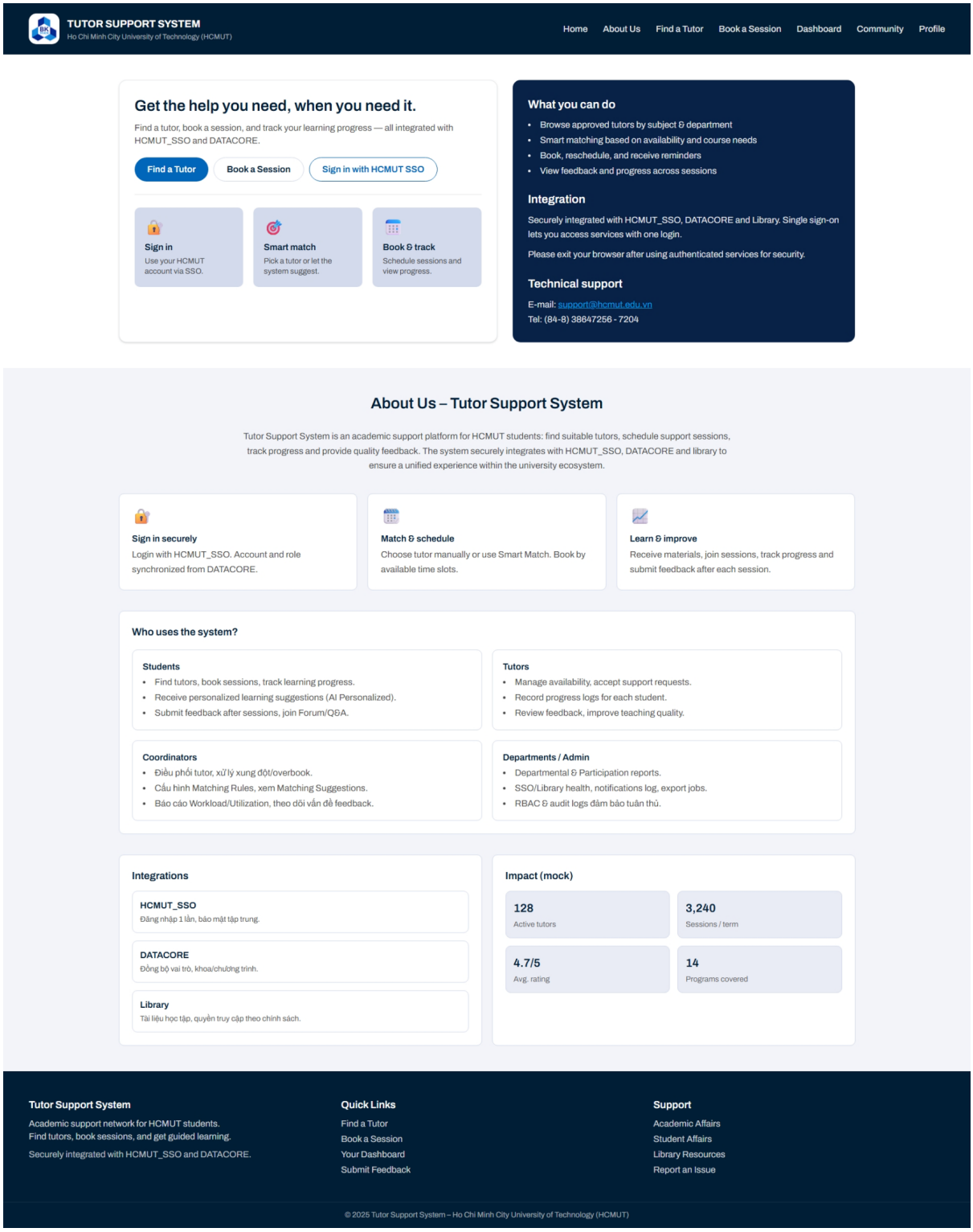

Figure 9: Landing page

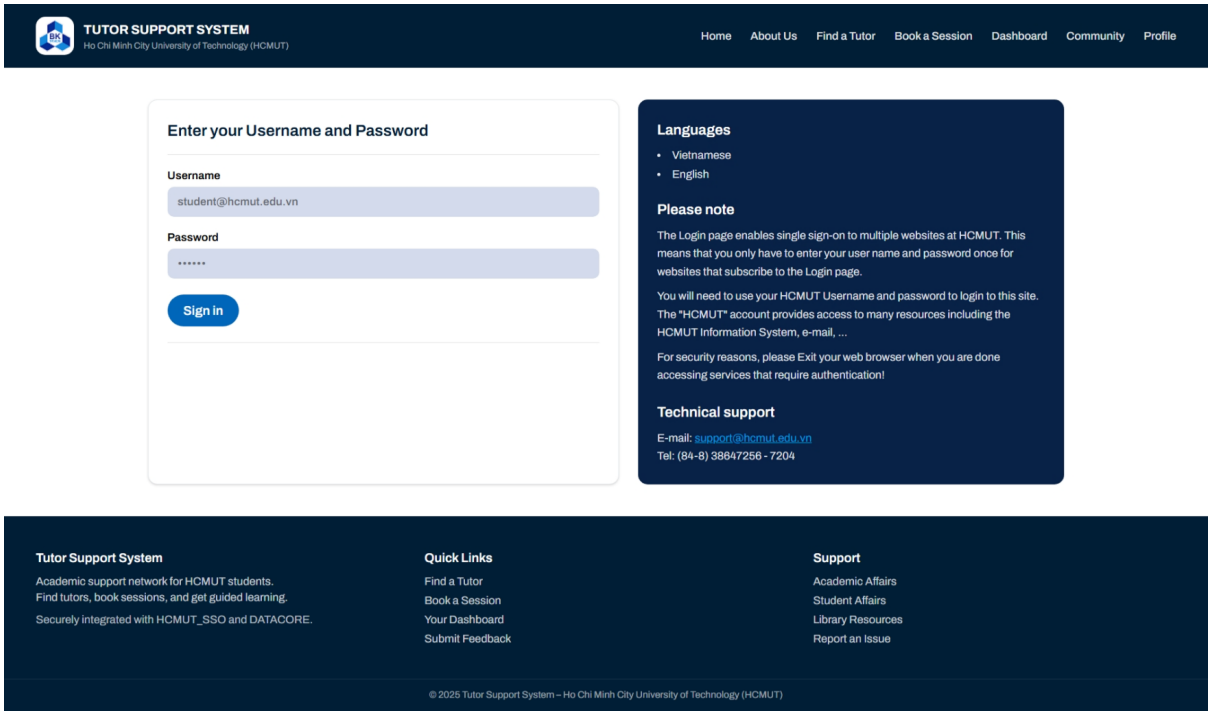

Figure 9: Log in page

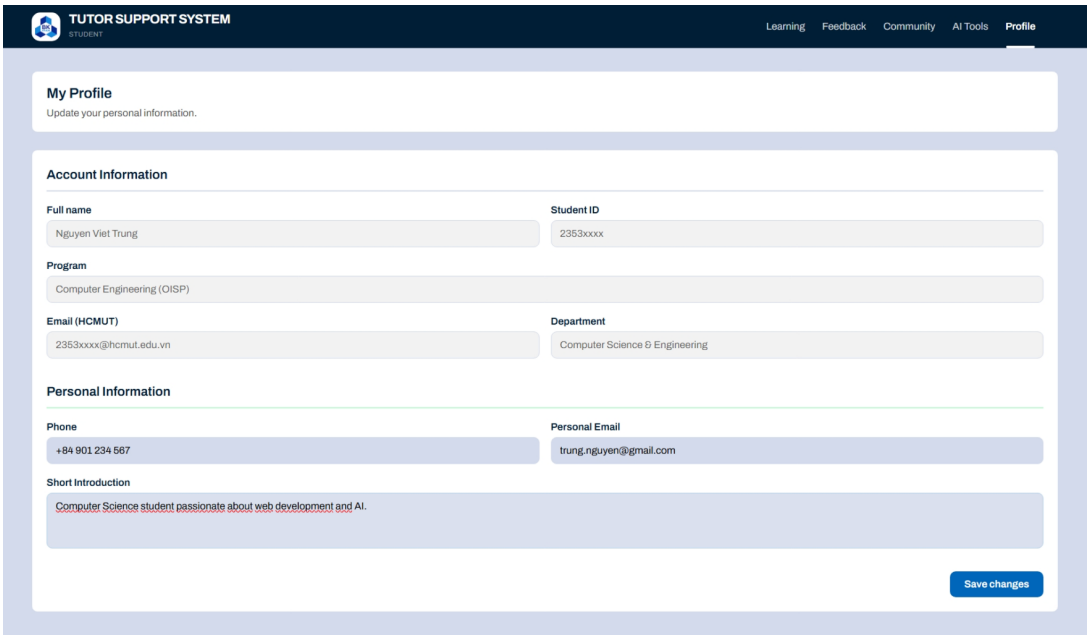

Figure 9: View profile & Edit profile page

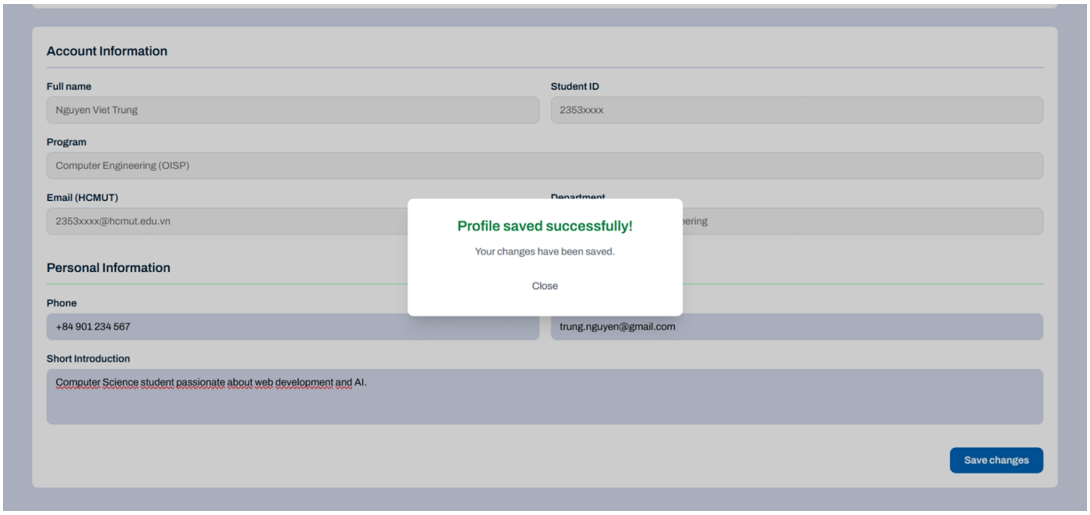

Figure 9: Save change Profile page

10.2 MU-02 Tutor-Student Matching

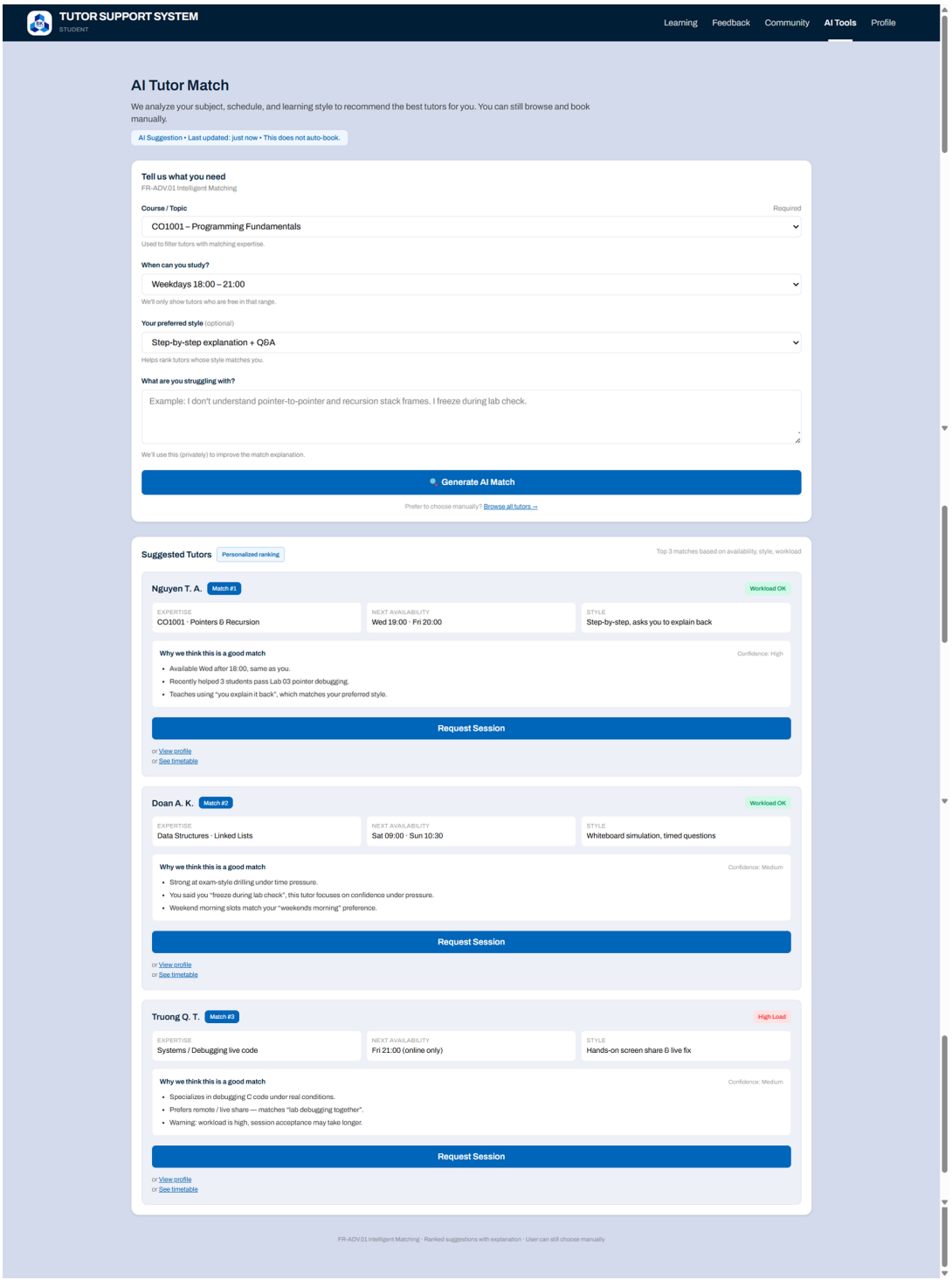

Figure 10: AI Tutor-Student Matching page

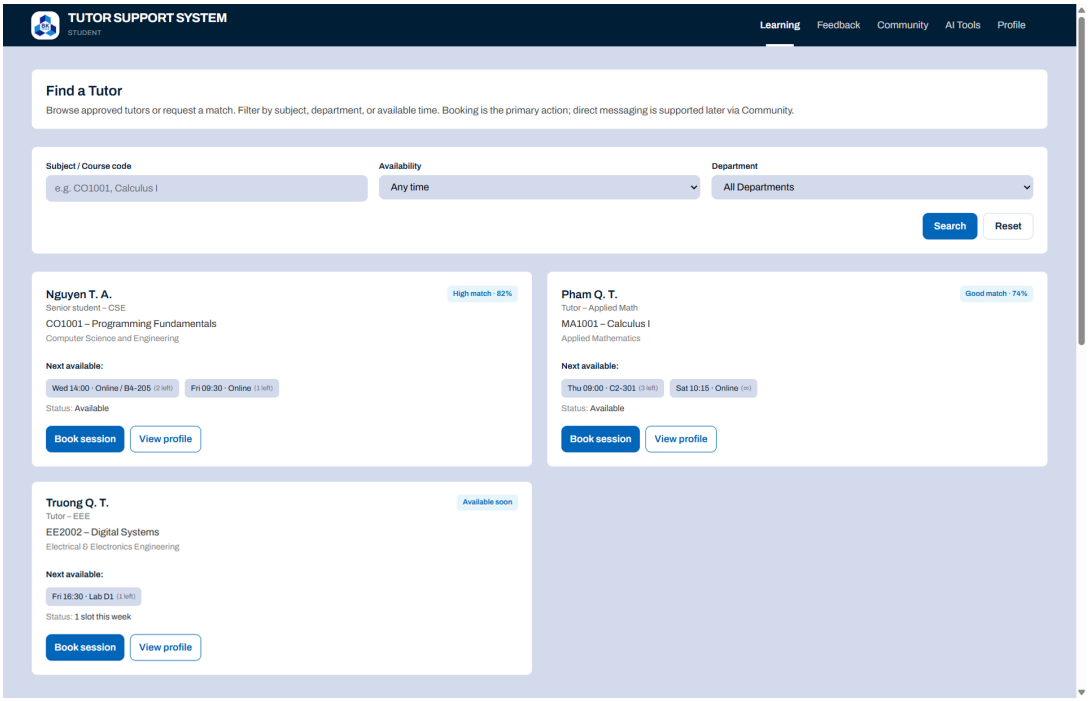

Figure 10: Find a Tutor page

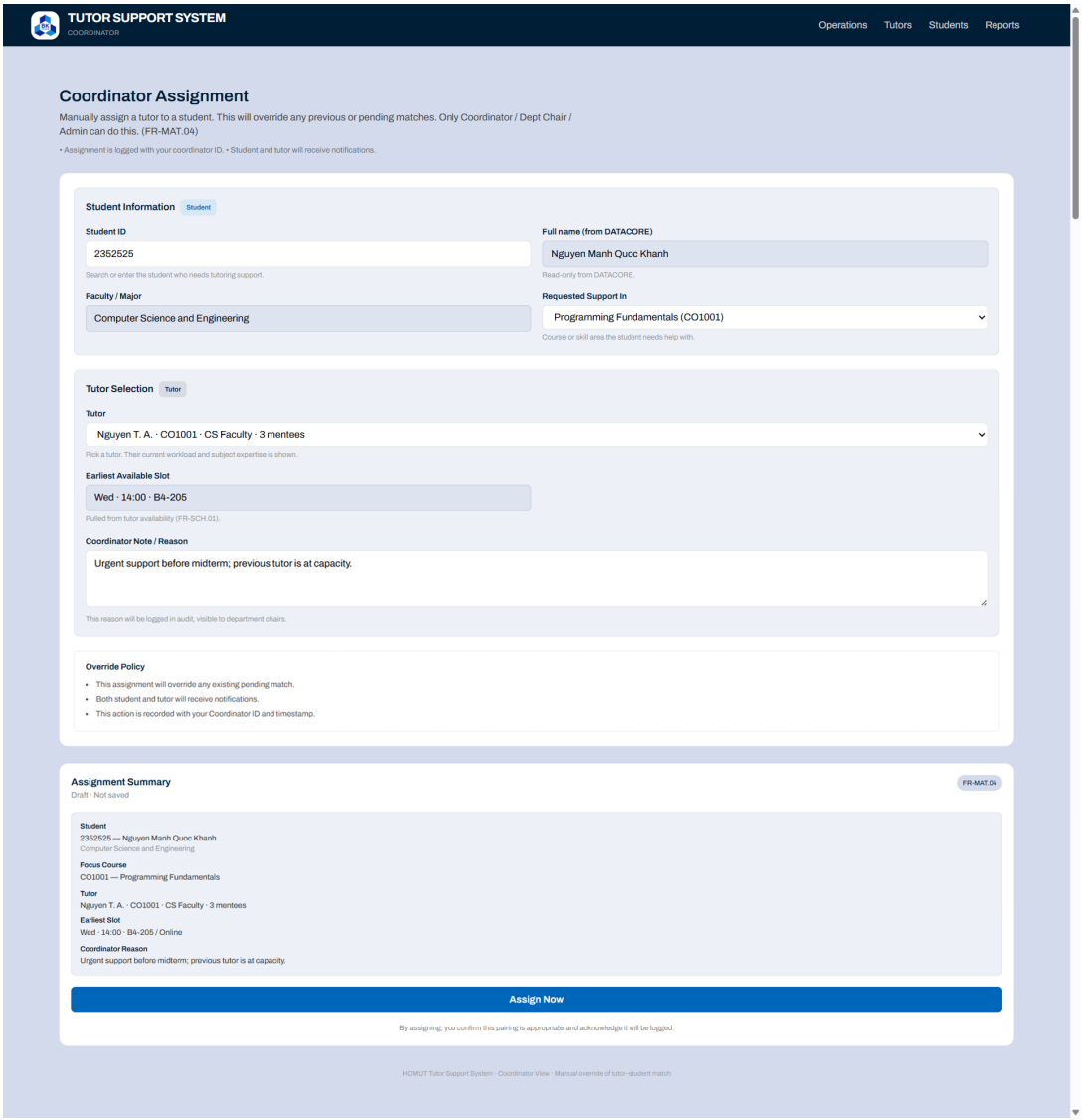

Figure 10: Coordinator assignment page

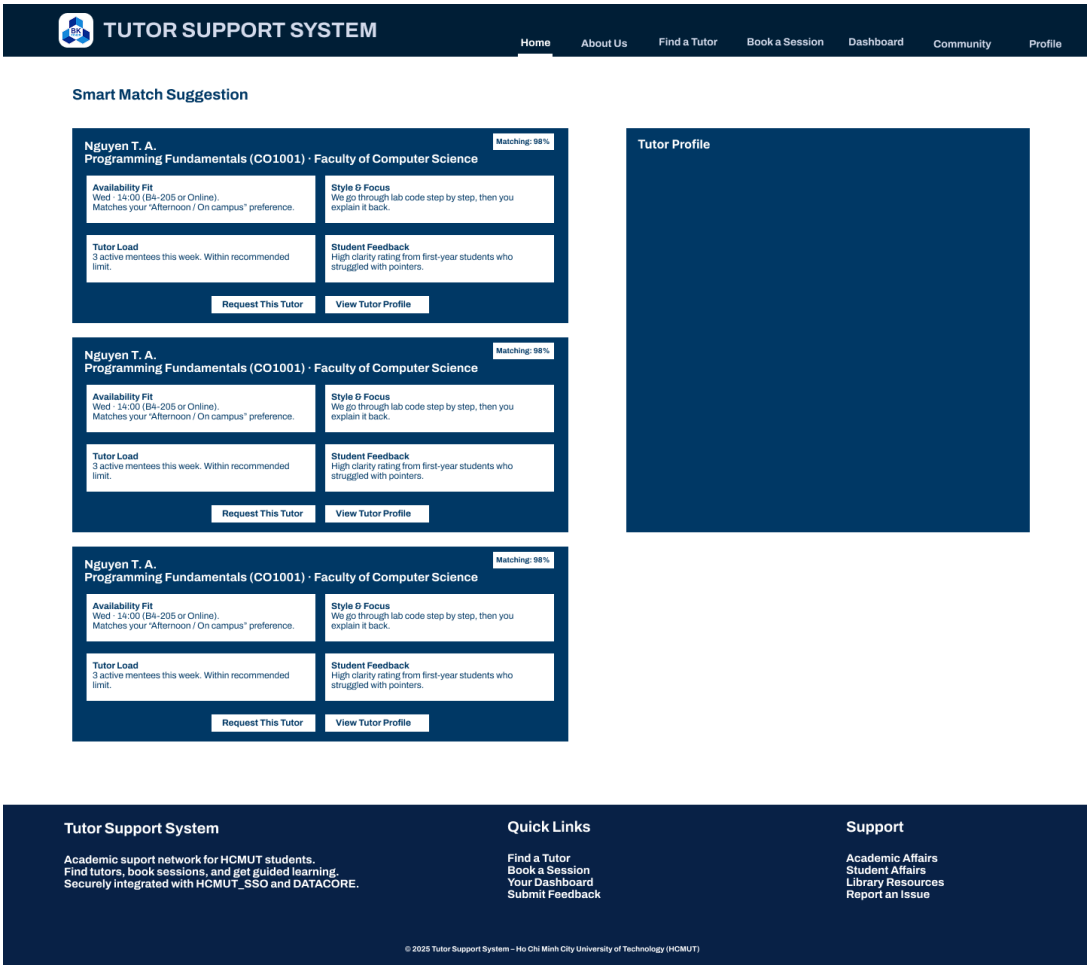

Figure 10: Smart match suggestion page

10.3 MU-03 Session Scheduling Management

Session Scheduling Management

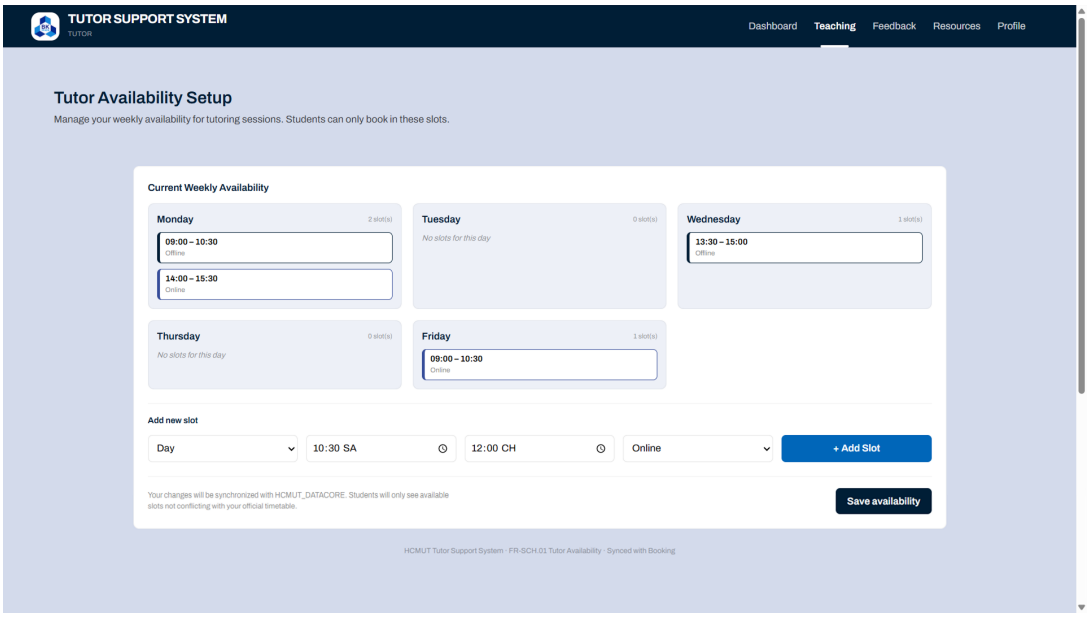

Figure 11: Tutor published available time slots

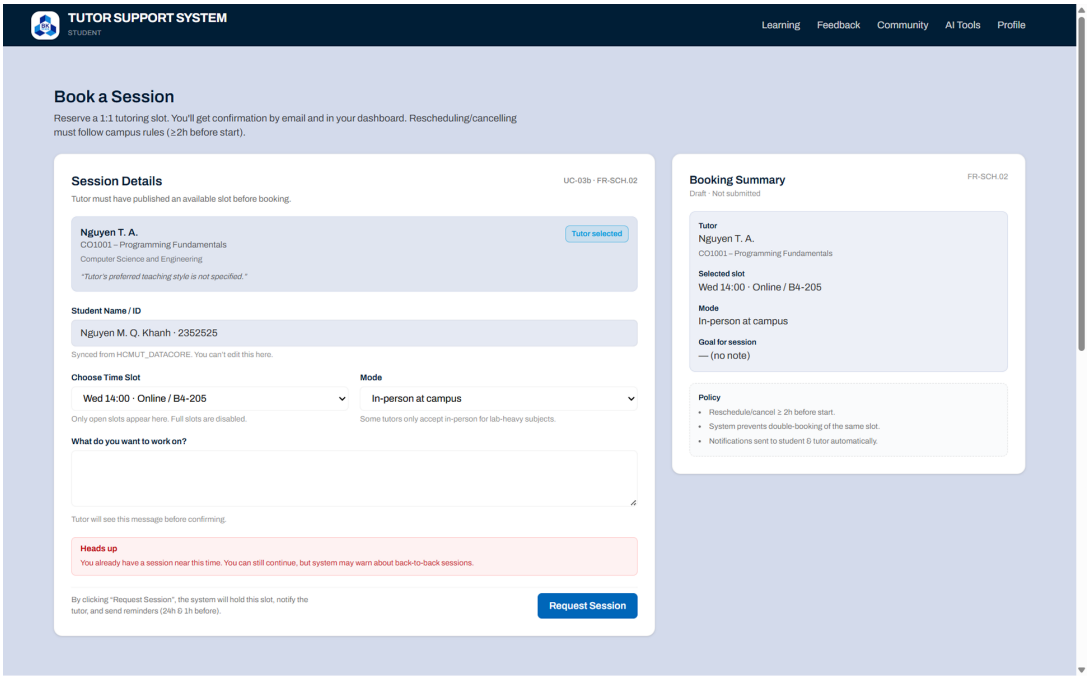

Figure 12: Student book session page

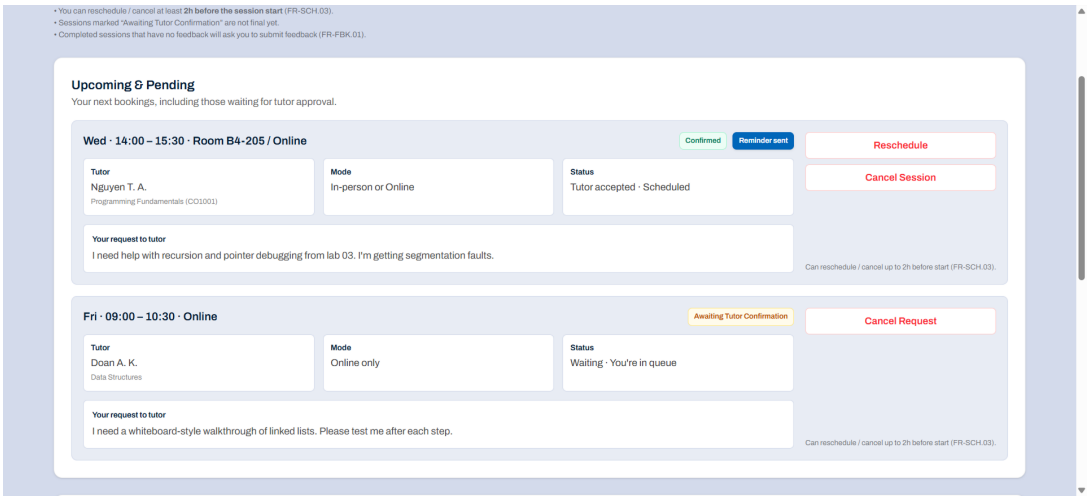

Figure 13: Cancellation and rescheduling page

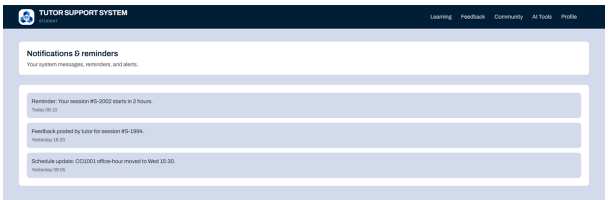

Figure 14: System sends notification reminders

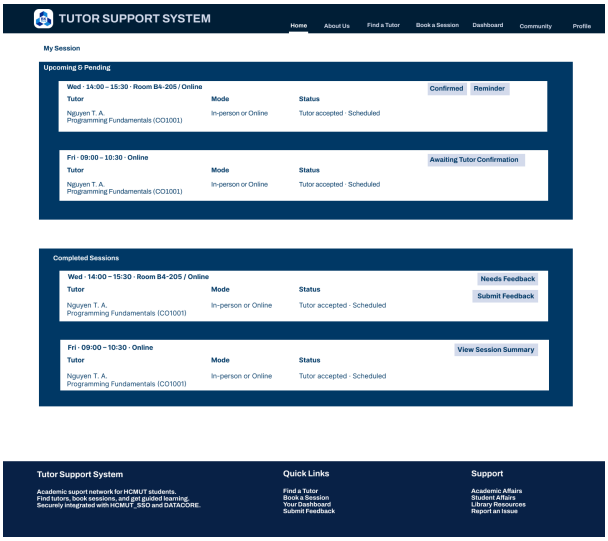

Figure 14: My registered sessions page

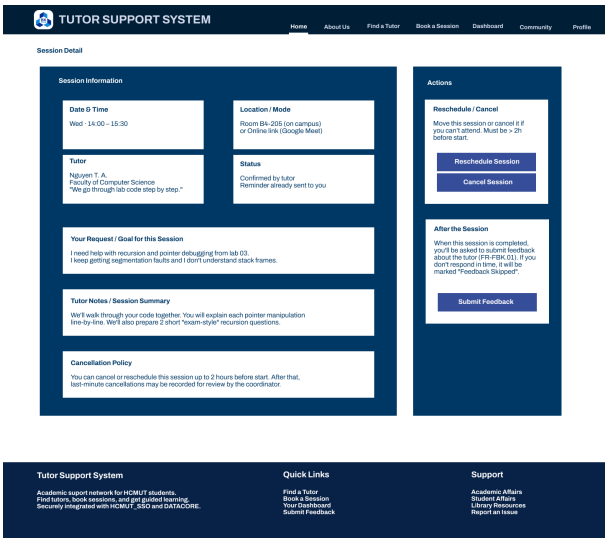

Figure 14: Session details page

10.4 MU-04 Feedback & Progress Tracking

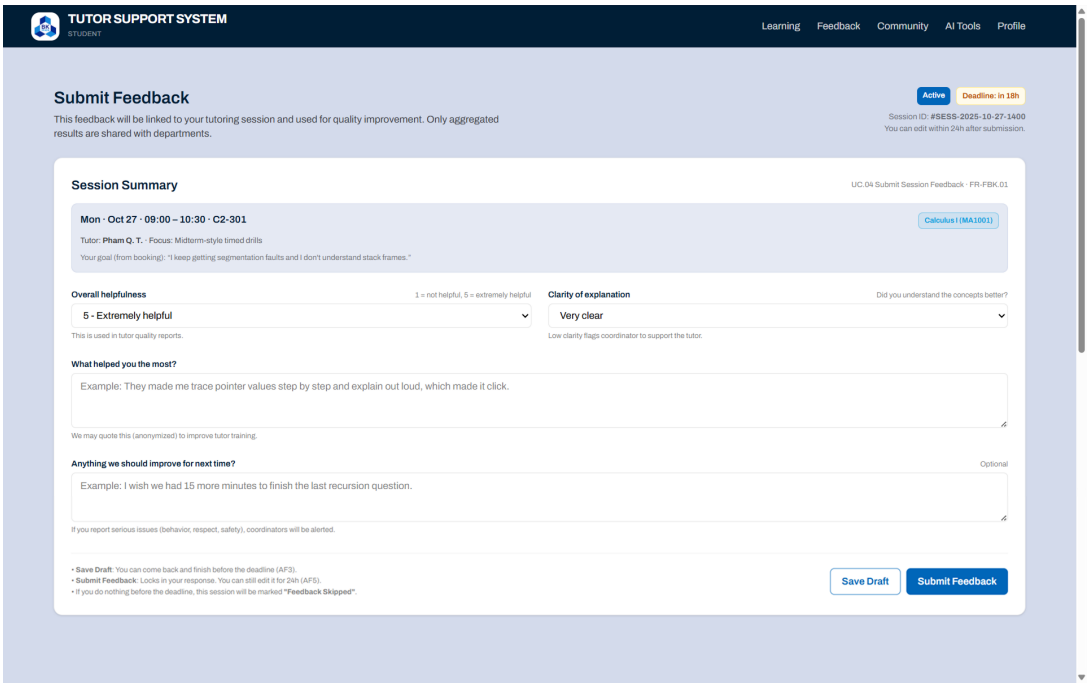

Figure 15: Submit feedback page

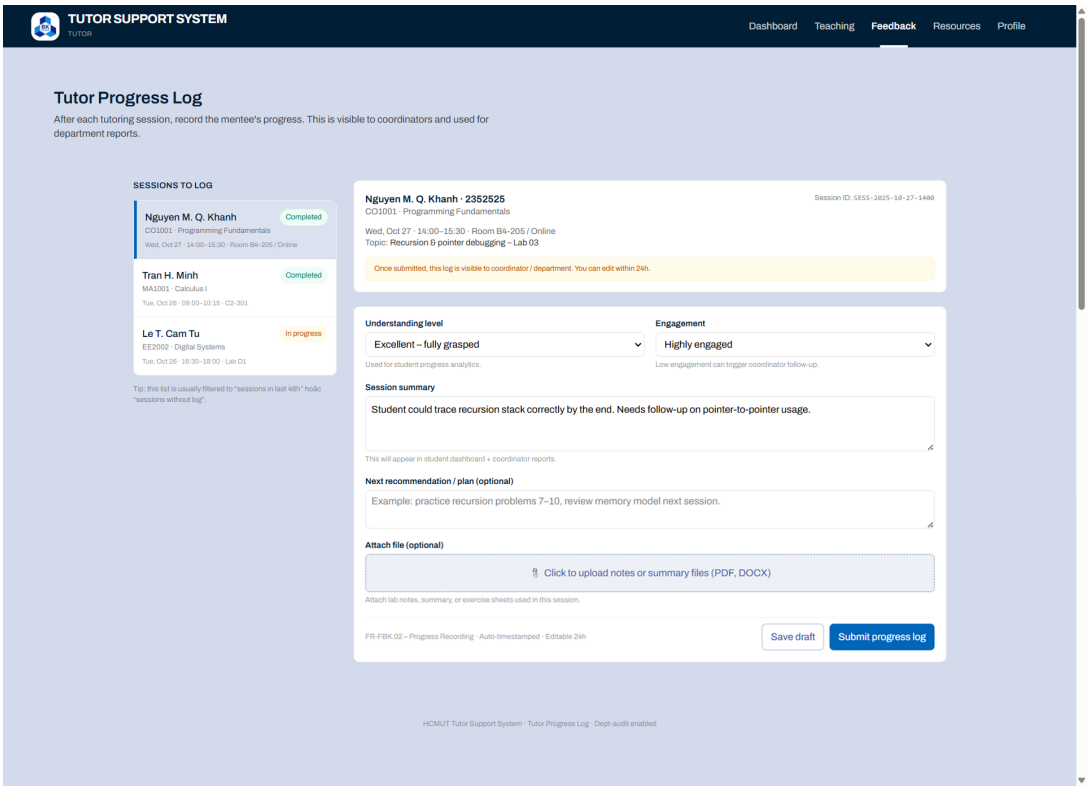

Figure 16: Tutor progress log page

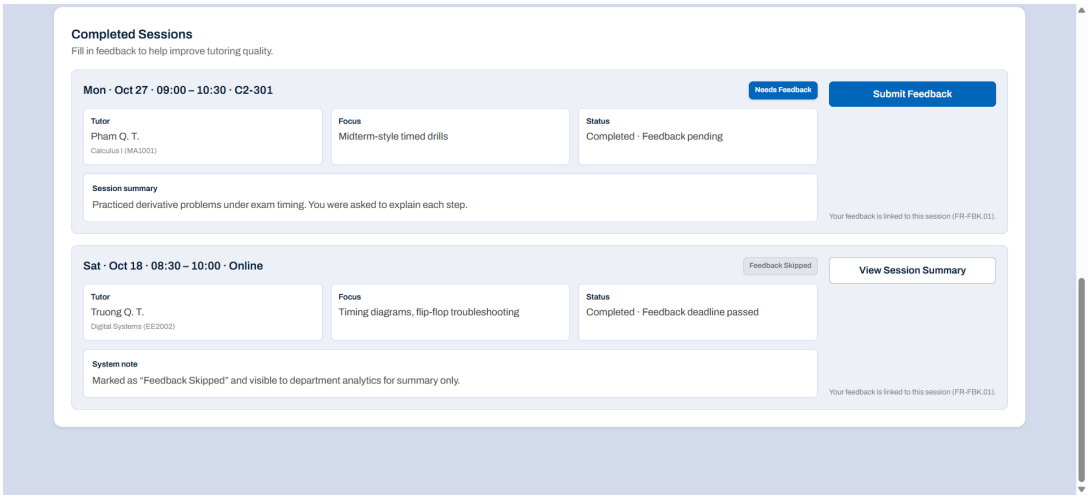

Figure 17: Completed session page

10.5 MU-05 Reporting & Analytics

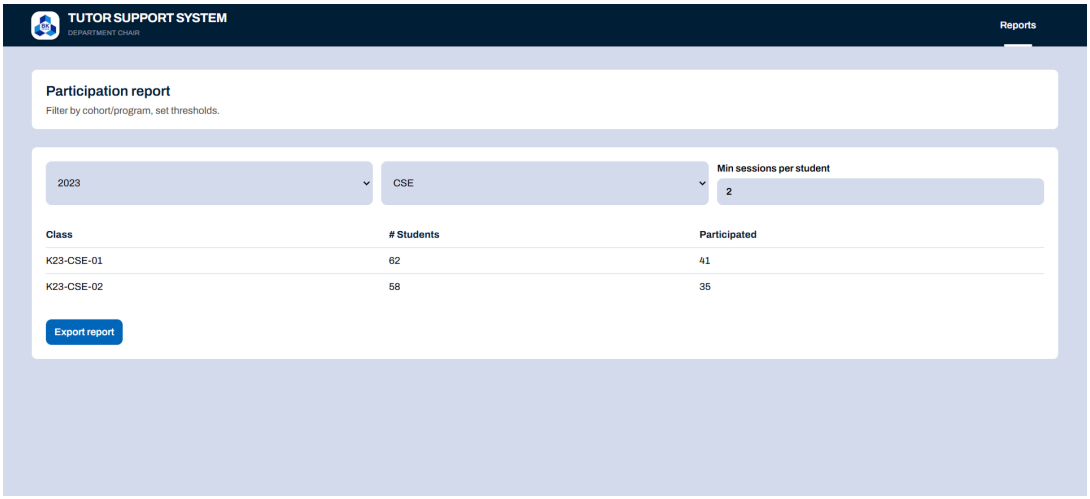

Figure 18: Participation report page

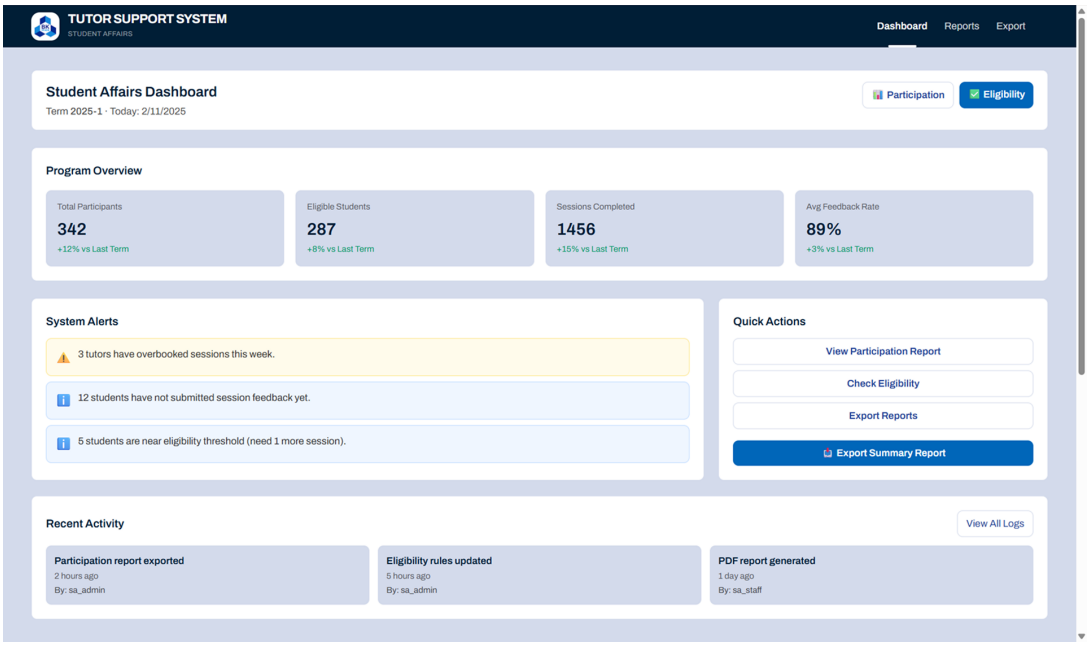

Figure 18: Student Affairs dashboard page

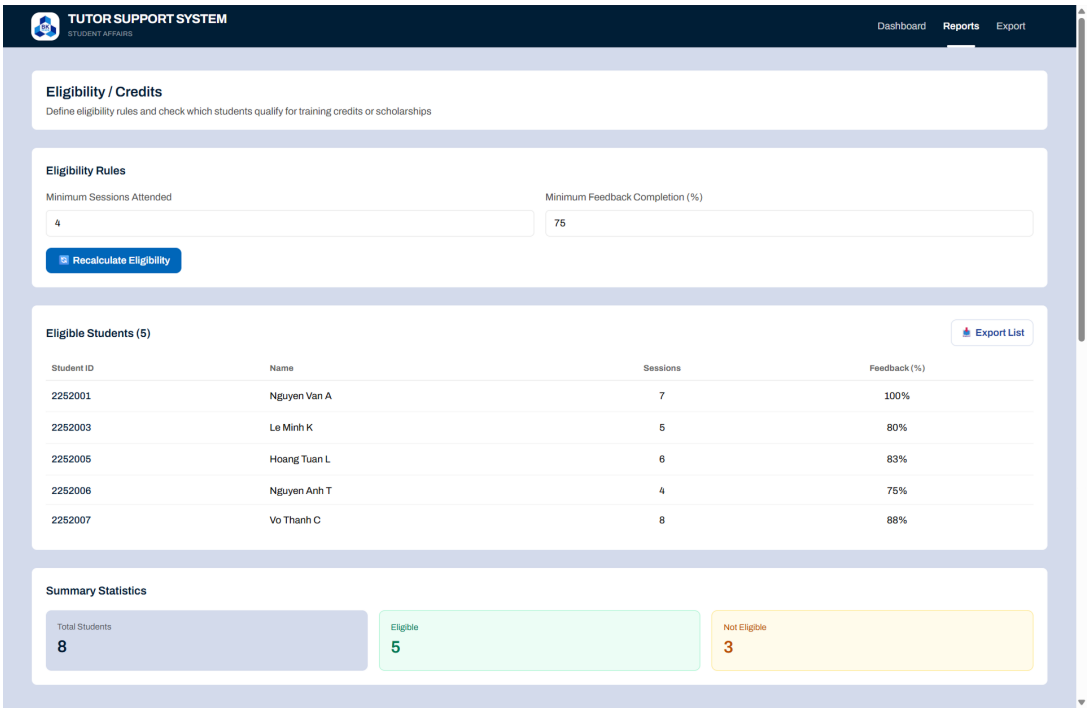

Figure 18: Eligibility & Credits page

10.6 MU-06 Integration with HCMUT Infrastructure

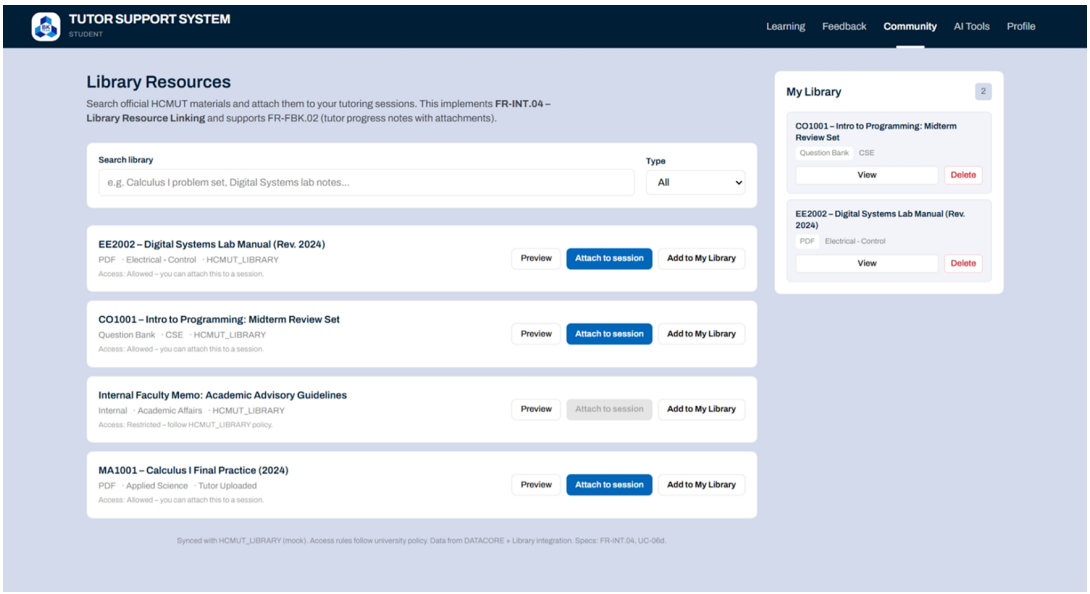

Figure 18: Library resource search page

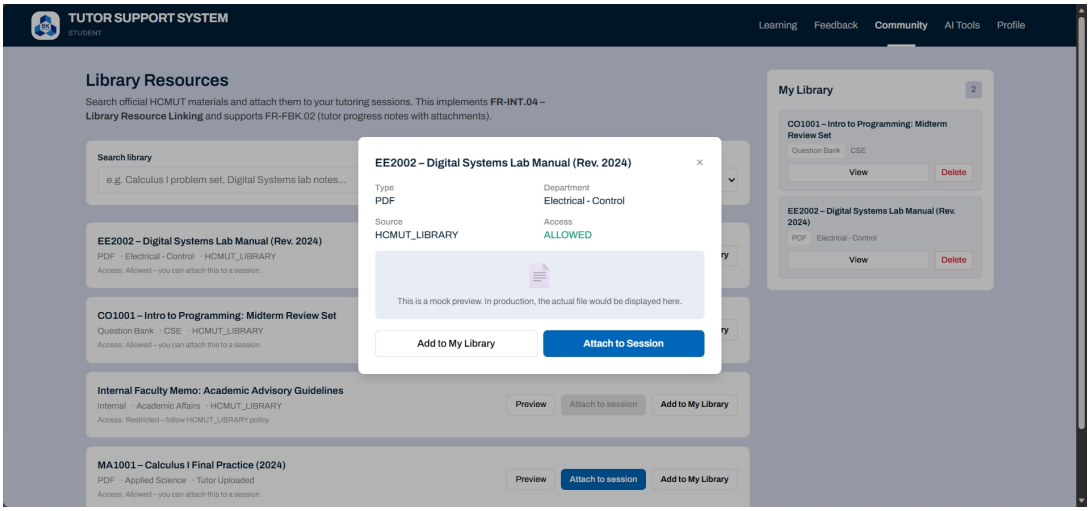

Figure 18: Attachment to session page

10.7 MU-07 Study & Advanced Features

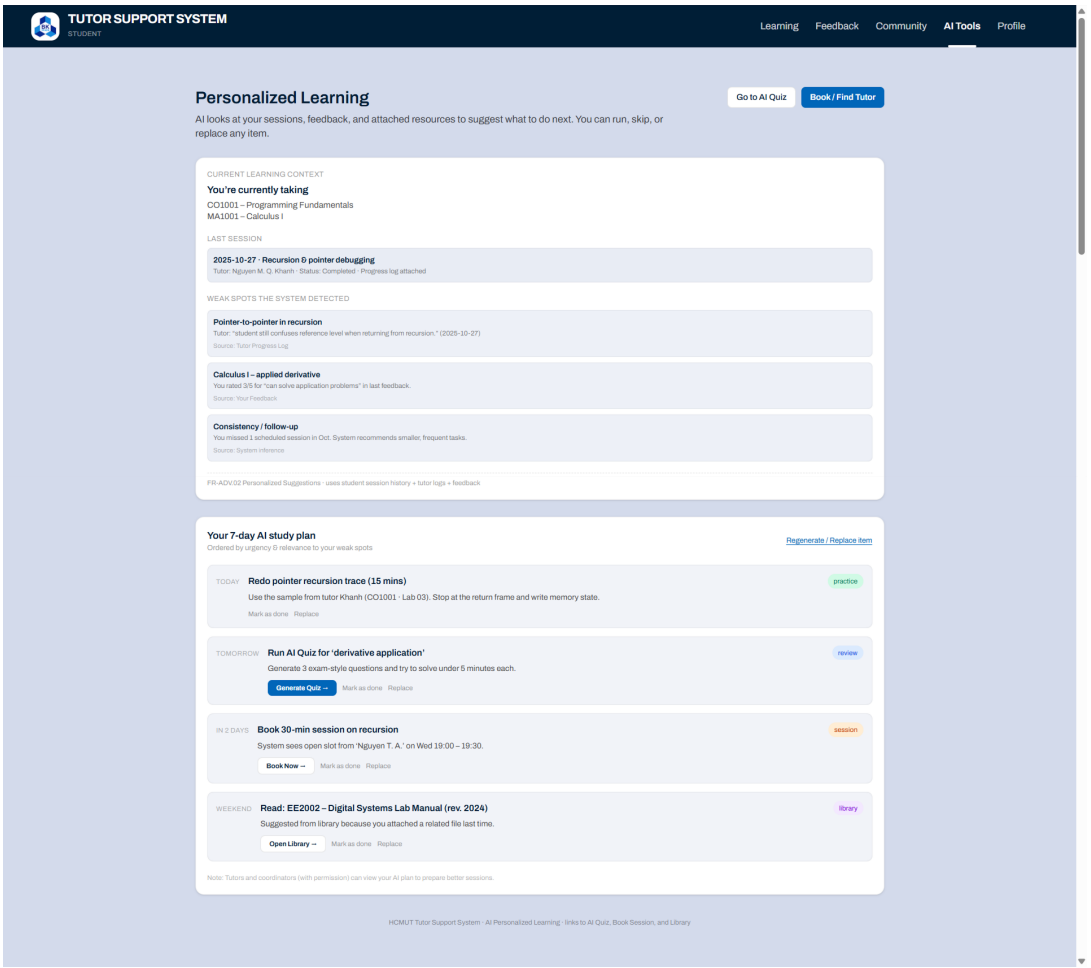

Figure 18: Personalized study plan page

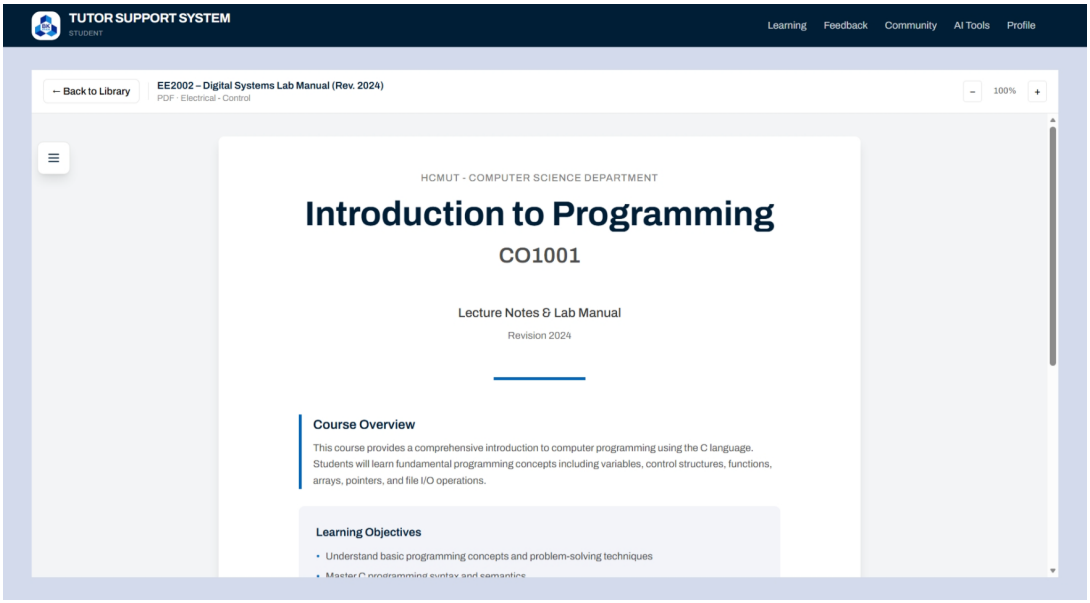

Figure 18: Document view page

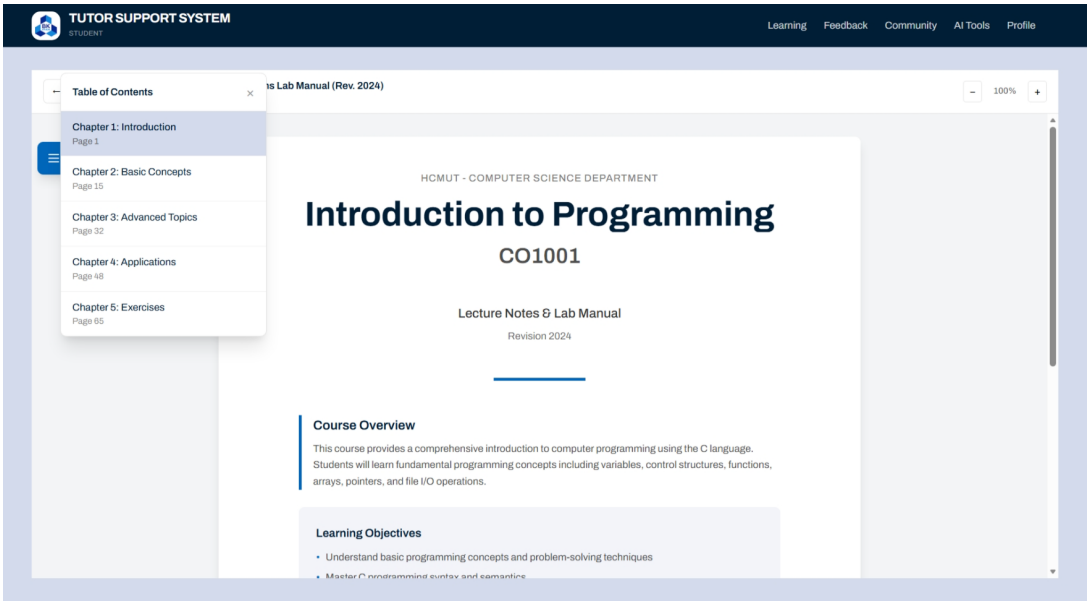

Figure 18: Table of contents page

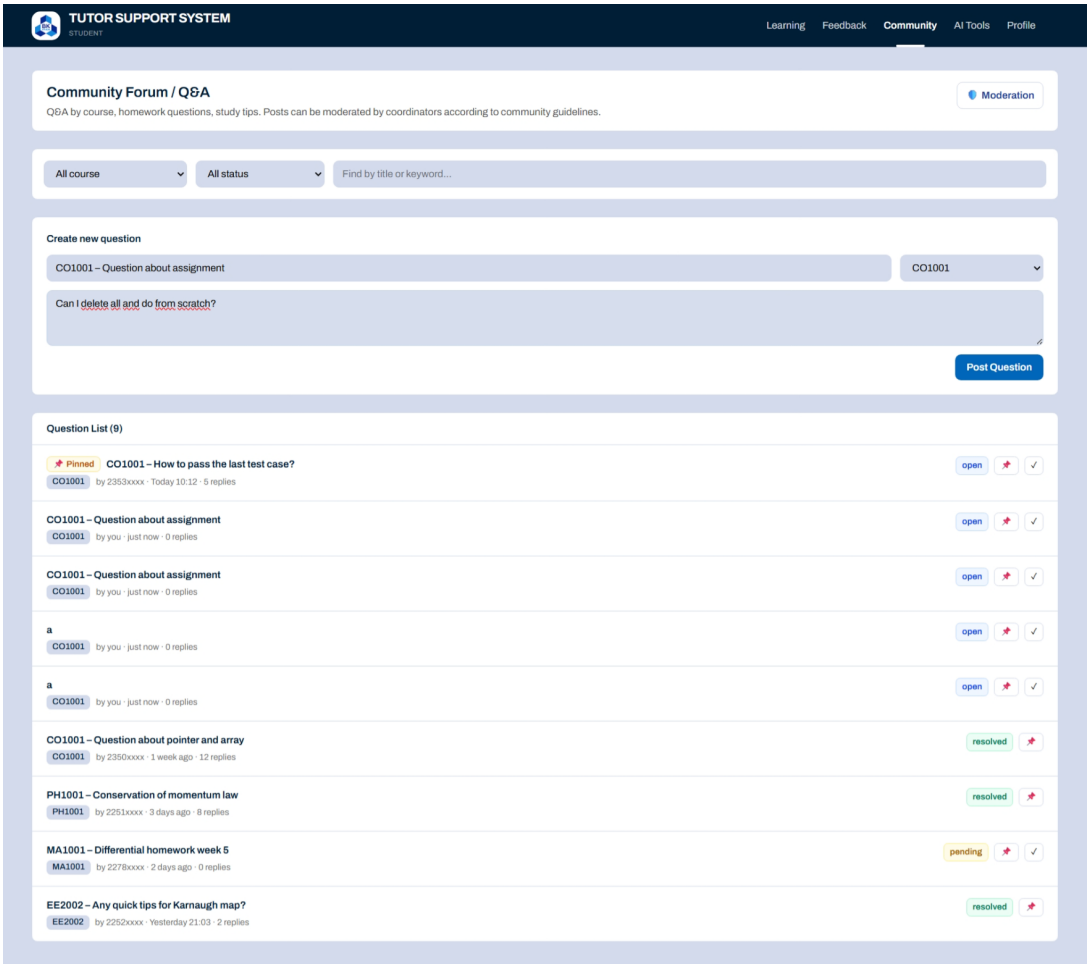

Figure 18: Comunity forum page

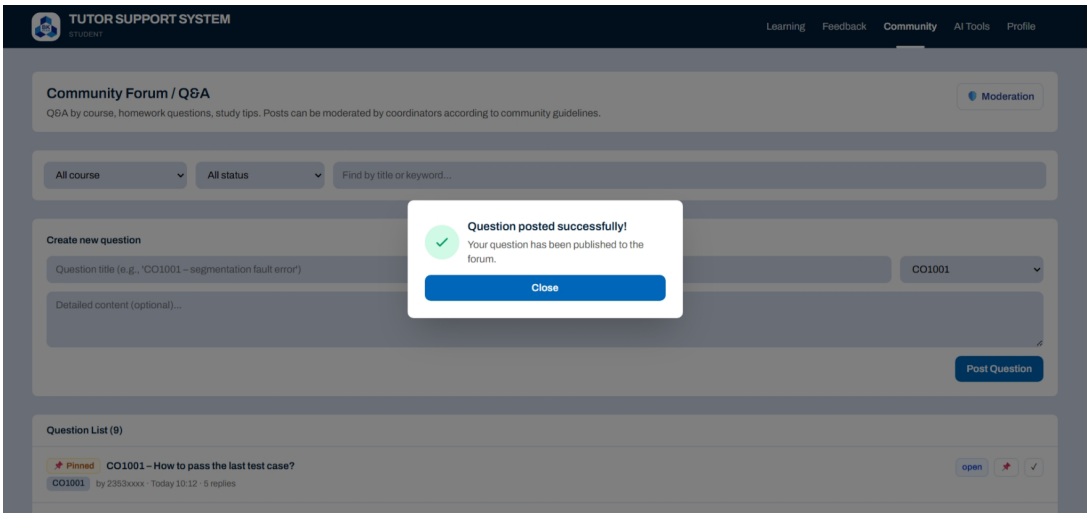

Figure 18: Successful question post page

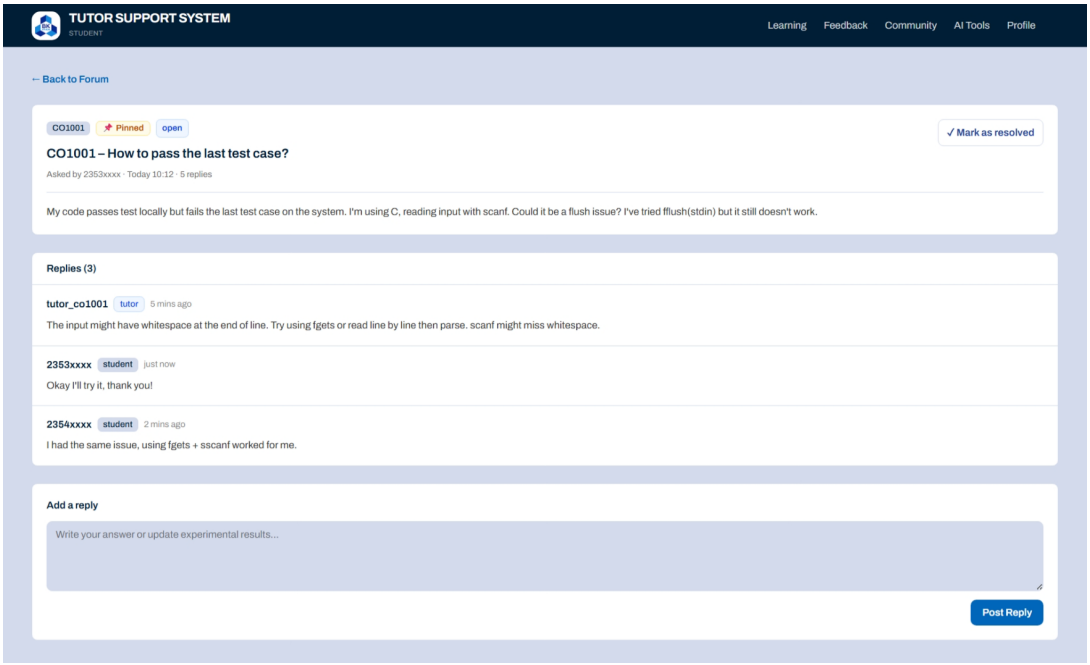

Figure 18: Reply forum's question page

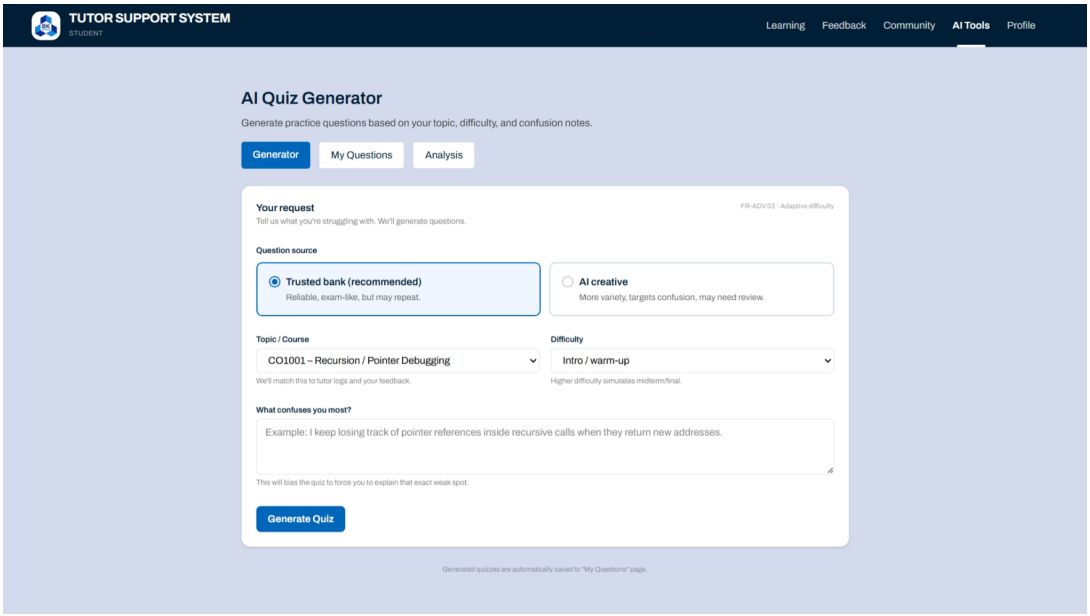

Figure 18: AI Quiz generator (1) page

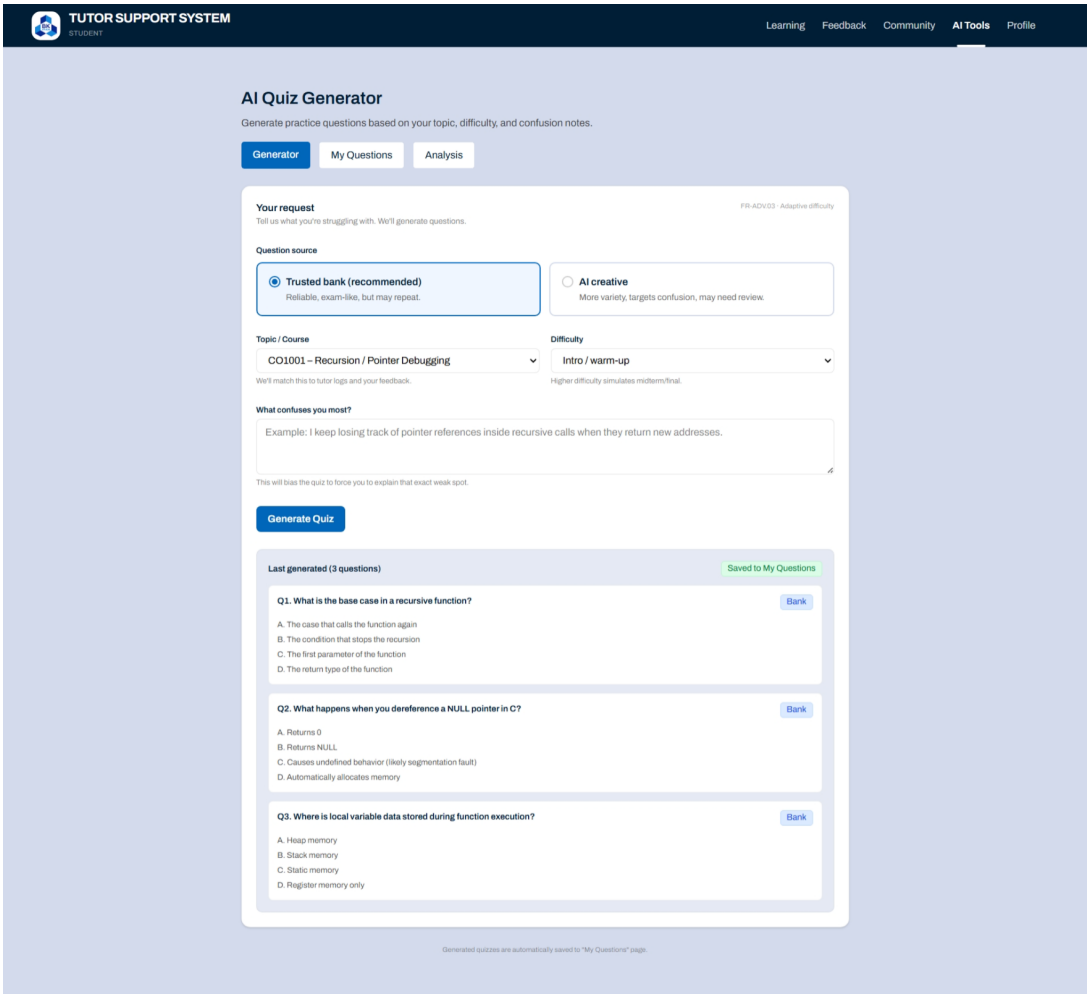

Figure 18: AI Quiz generator (2) page

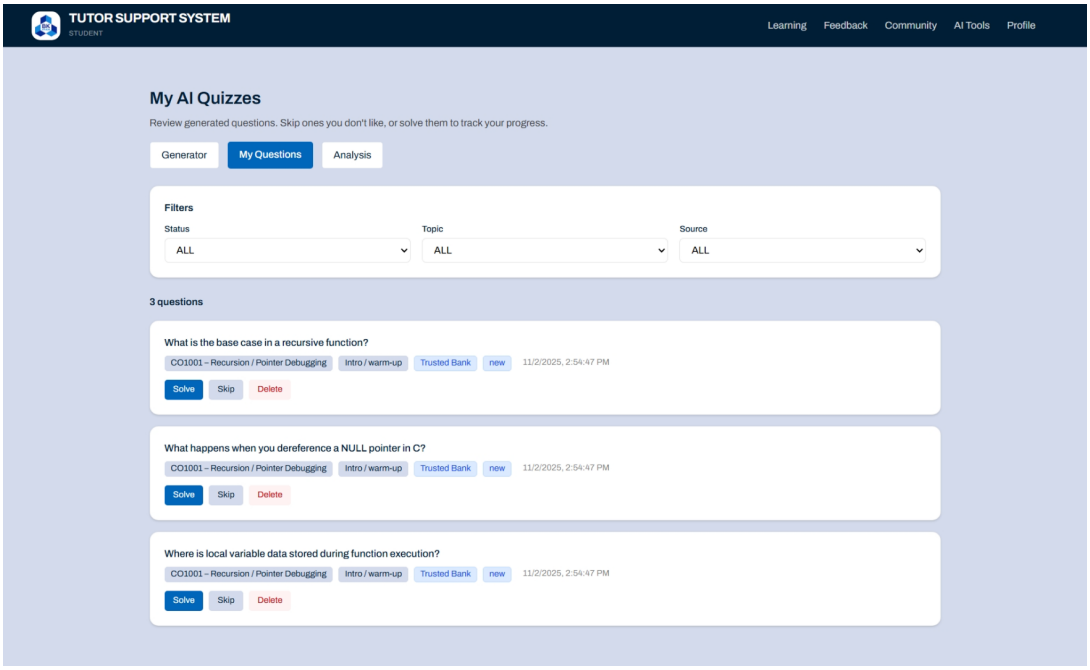

Figure 18: My AI Quizzes page

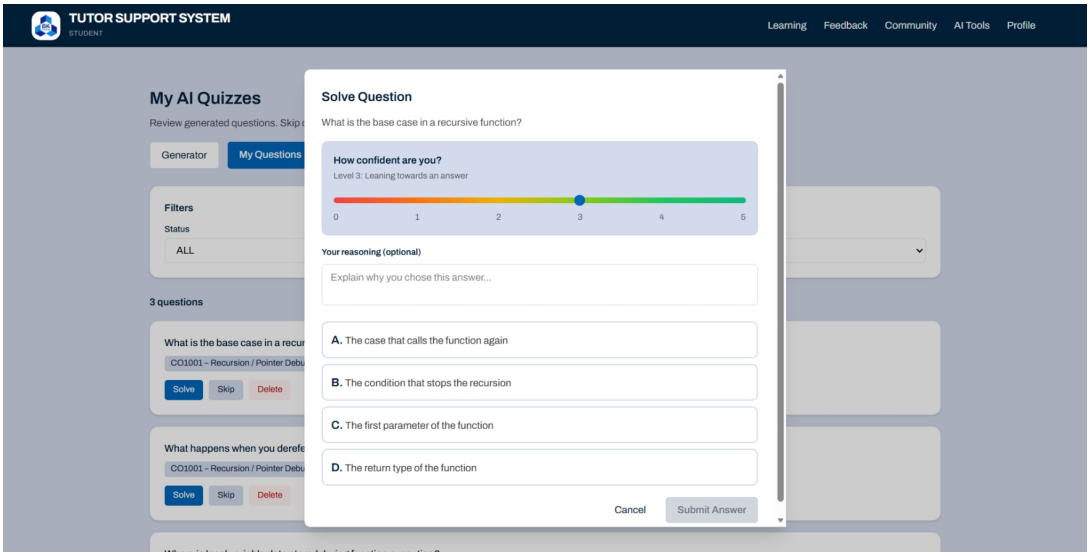

Figure 18: Solve question page (1)

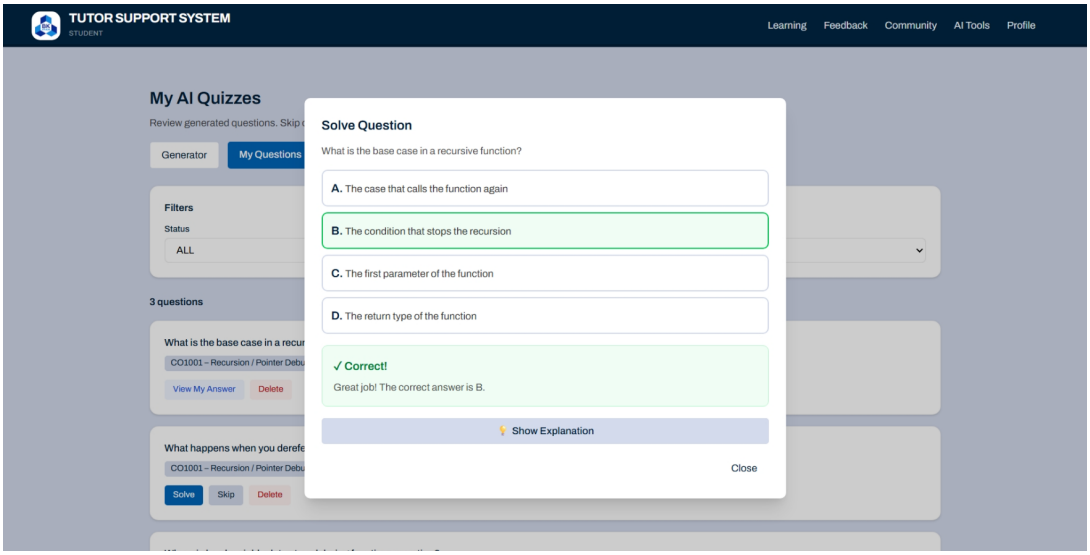

Figure 18: Solve question page (2)-answer

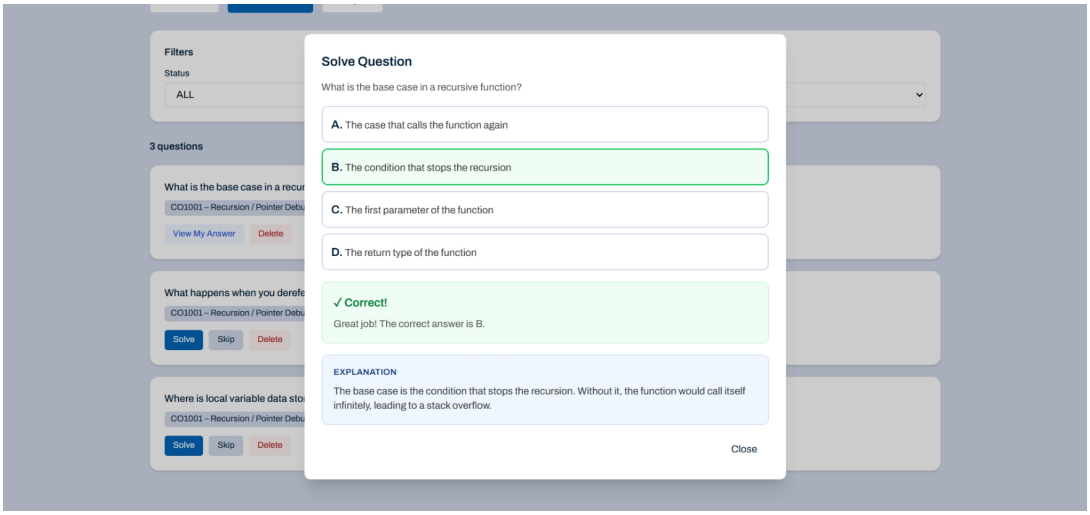

Figure 18: Solve question page (3)-explanation

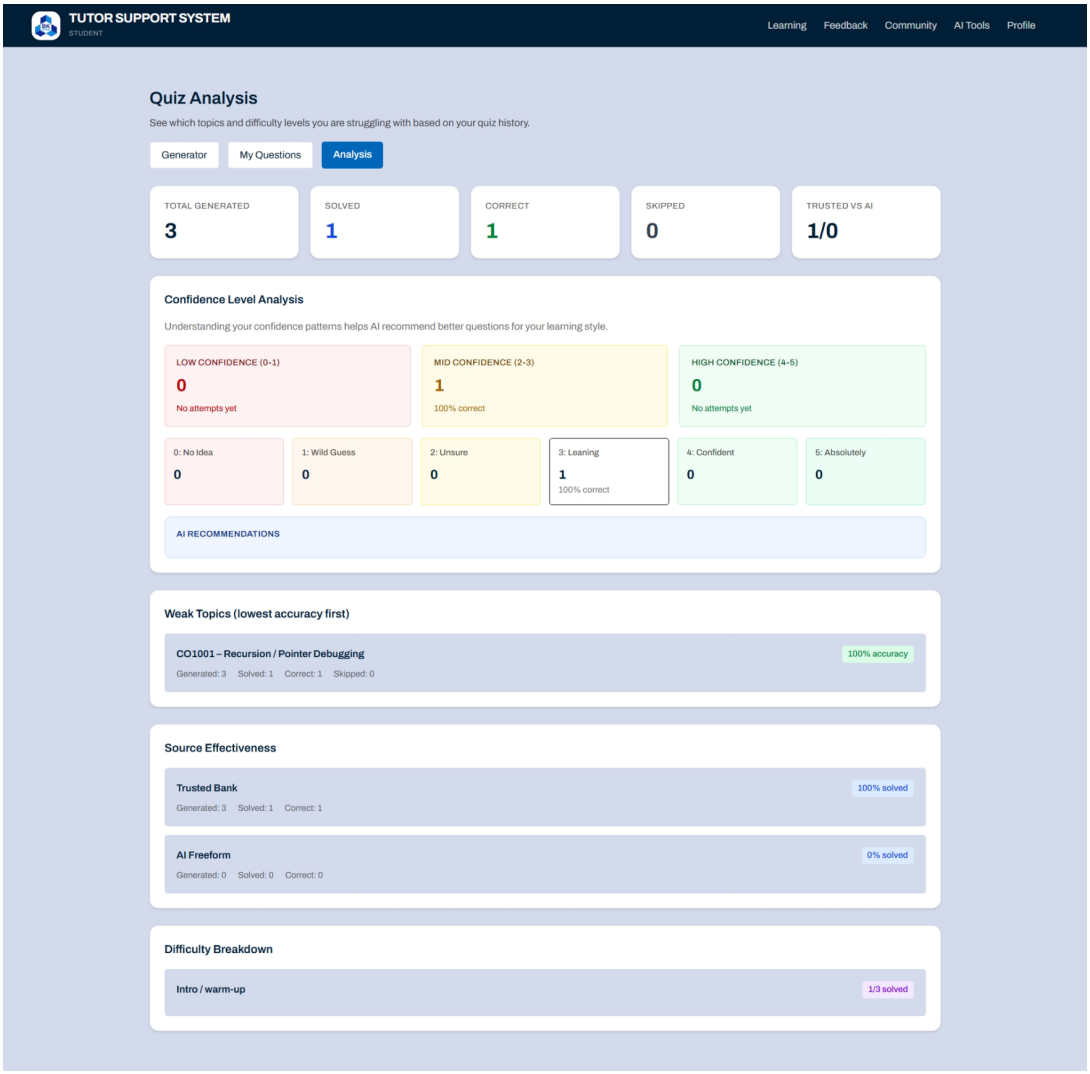

Figure 18: Quiz analysis page

10.8 MU-08 Admin Tools

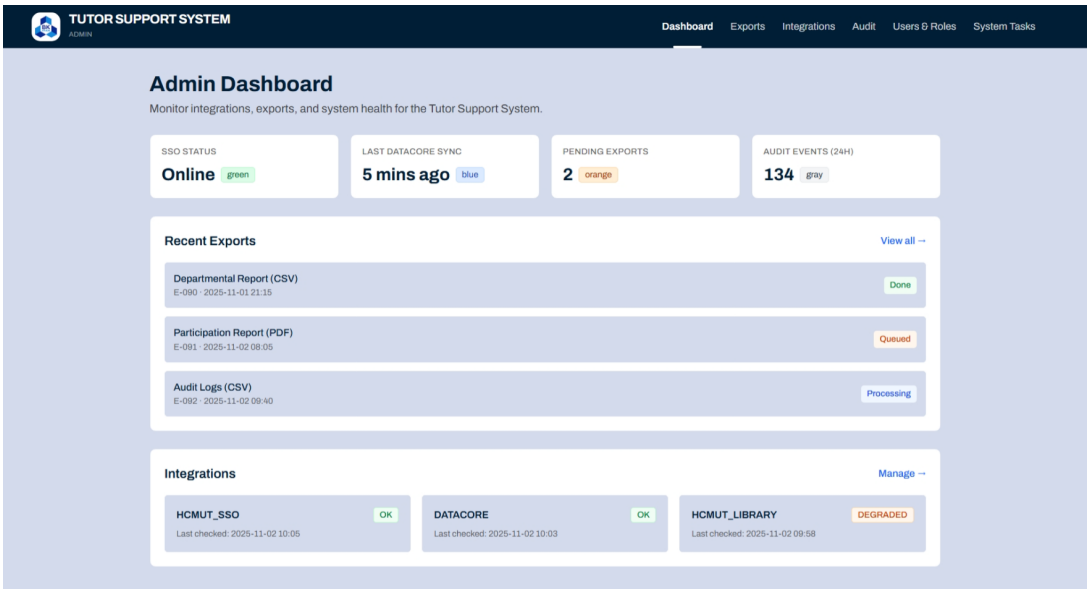

Figure 18: Admin dashboard page

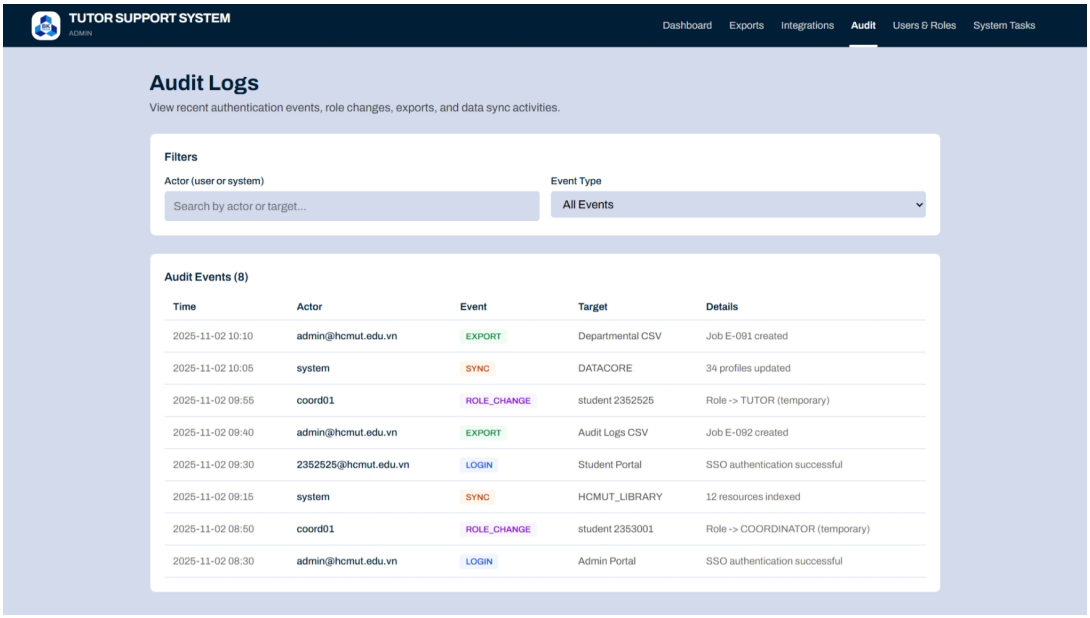

Figure 18: Audit logs page

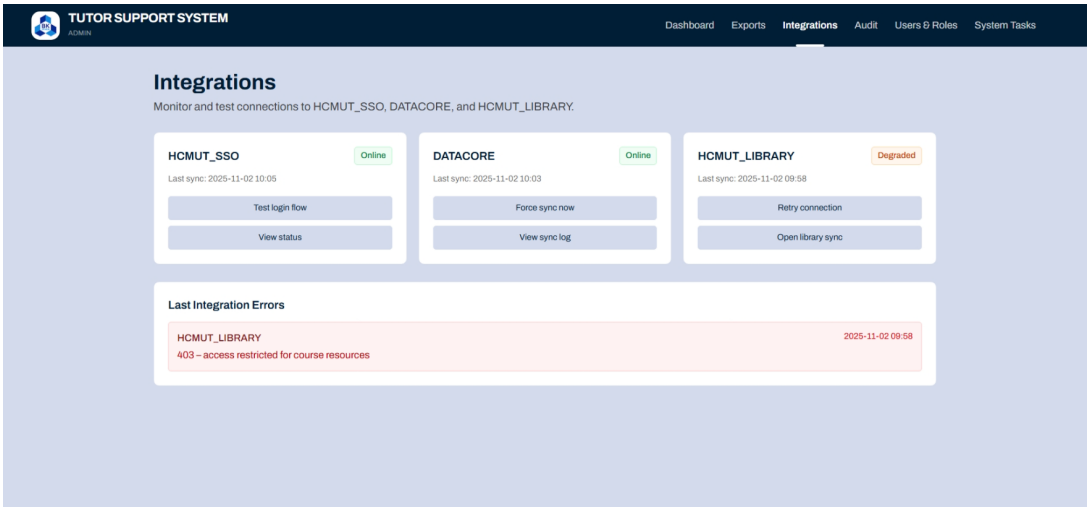

Figure 18: Integrations page

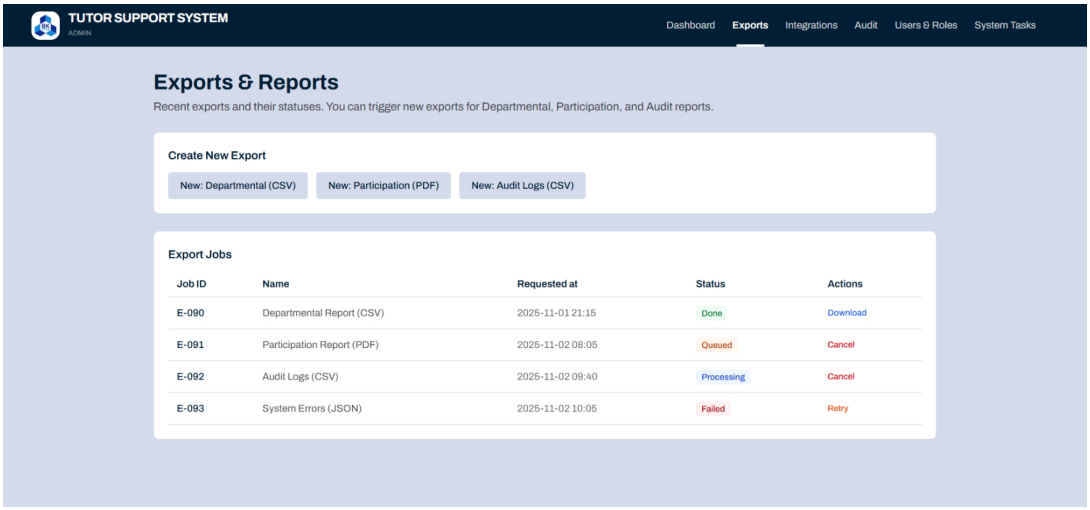

Figure 18: Export & Report page

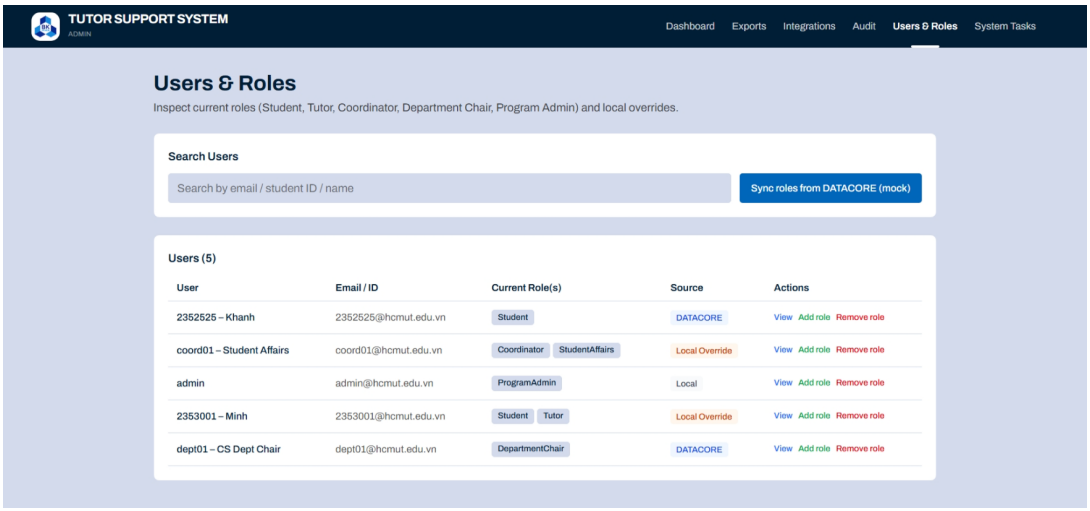

Figure 18: Users & Roles page

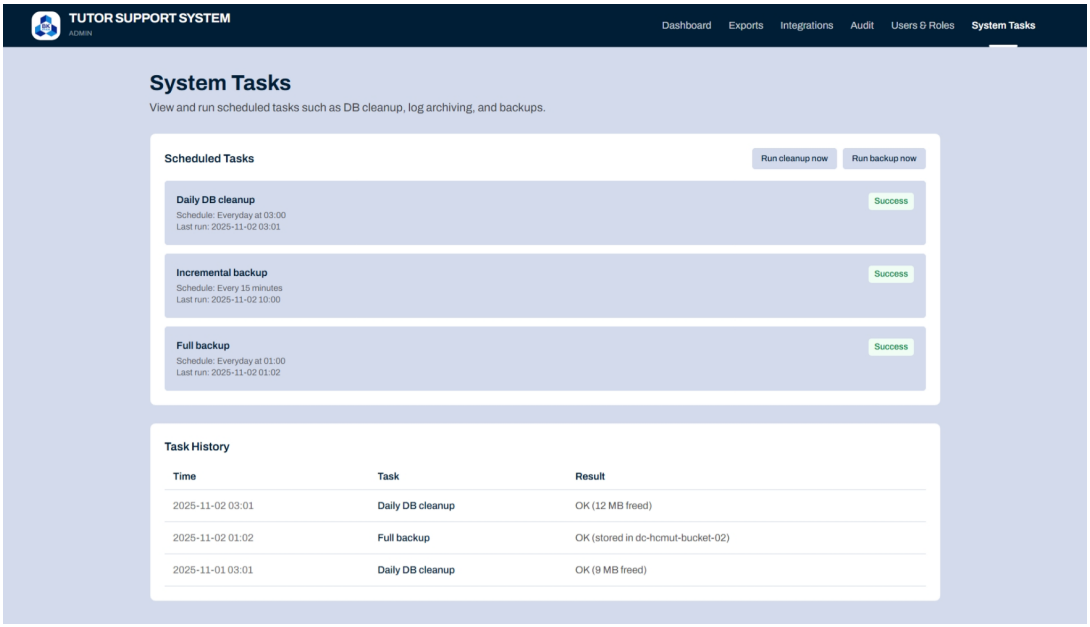

Figure 18: System tasks page

11 Deployment Diagram

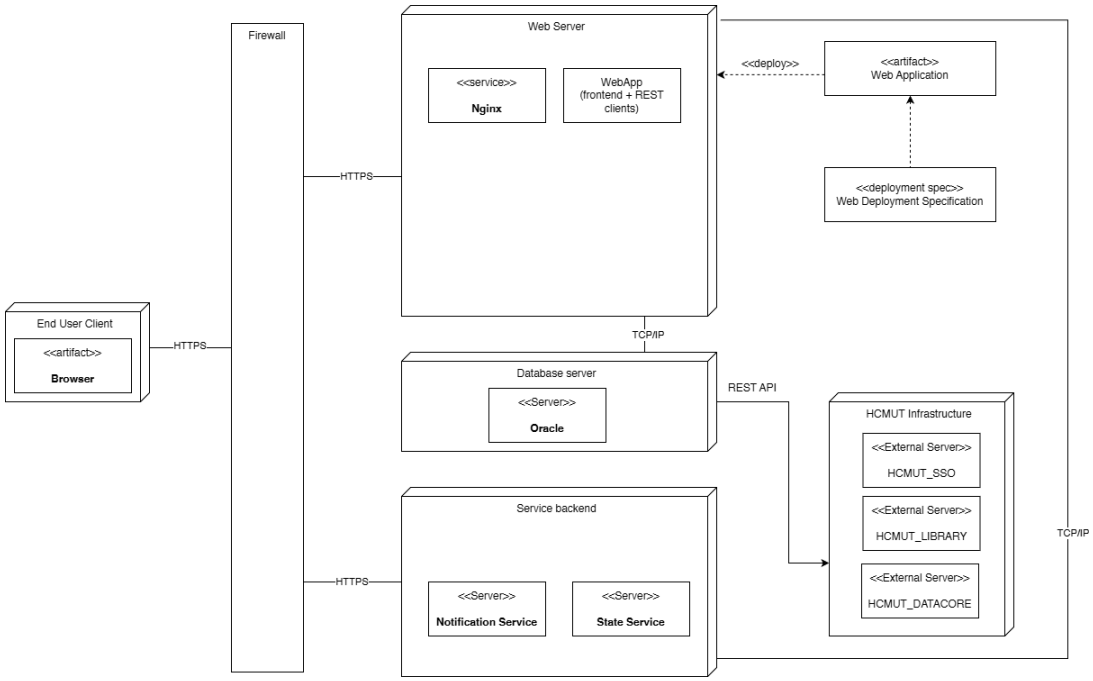

Figure 12: Deployment Diagram

11.1 Deployment View Description

The deployment diagram in Figure **??** illustrates the physical runtime environment of the Tutor Support System, showing how software components are deployed across servers and how they communicate through secure protocols.

End User Client Layer

End users interact with the system through a web browser, represented as an `«artifact»`. All communication between the client and the system is performed using HTTPS, ensuring encrypted and secure transmission of requests.

Firewall Layer

All incoming and outgoing traffic is filtered through a Firewall. This firewall protects the internal infrastructure by allowing only authorized protocols (e.g., HTTPS on port 443) and restricting unauthorized access from external networks.

Web Server Layer

The Web Server hosts two primary components:

- **Nginx**: Operates as a reverse proxy and static file server, routing incoming HTTPS requests to appropriate backend services.
- **WebApp (frontend + REST clients)**: Contains the user interface and the REST clients responsible for communicating with backend services and external systems.

The Web Application artifact is deployed onto this node according to the *Web Deployment Specification*, which defines the packaging and deployment procedures. Communication from the Web Server to backend services uses secure protocols depending on the target module.

Database Server Layer

The Database Server hosts the Oracle database, which stores persistent application data such as user profiles, tutor availability, booking information, session records, and system logs. Communication between the Web Server and the Oracle database is performed over TCP/IP to ensure consistent and reliable data transactions.

Service Backend Layer

This layer hosts two supporting infrastructure services:

- **Mail Server**: Responsible for sending notification emails, including booking confirmations and schedule updates.
- **State Service**: Provides in-memory storage for transient system state such as session data, caching, and booking locks.

Communication between the Web Server and this backend layer is secured through HTTPS.

HCMUT Infrastructure Layer

The system integrates with a set of university-provided external services:

- **HCMUT_SSO**: Provides authentication services through SAML/OAuth-based single sign-on.
- **HCMUT_LIBRARY**: Provides access to academic resources and related materials.
- **HCMUT_DATACORE**: Supplies institutional user data for identity synchronization and academic information retrieval.

These services are accessed from the Web Server using REST APIs over HTTPS, ensuring secure and authenticated data exchange.

Overall System Communication Flow

When a user interacts with the system, requests are sent from the Browser through the Firewall to the Web Server. Nginx forwards these requests to the WebApp component, which communicates with internal backend services (database, mail, state). For authentication or user data synchronization, the WebApp interacts with the HCMUT_SSO and HCMUT_DATACORE services. When academic materials are needed, the system retrieves them from the HCMUT_LIBRARY service. Processed responses are returned through the Firewall back to the user's browser over HTTPS.

This deployment architecture ensures security, scalability, maintainability, and seamless integration with university services.

12 Development Diagram

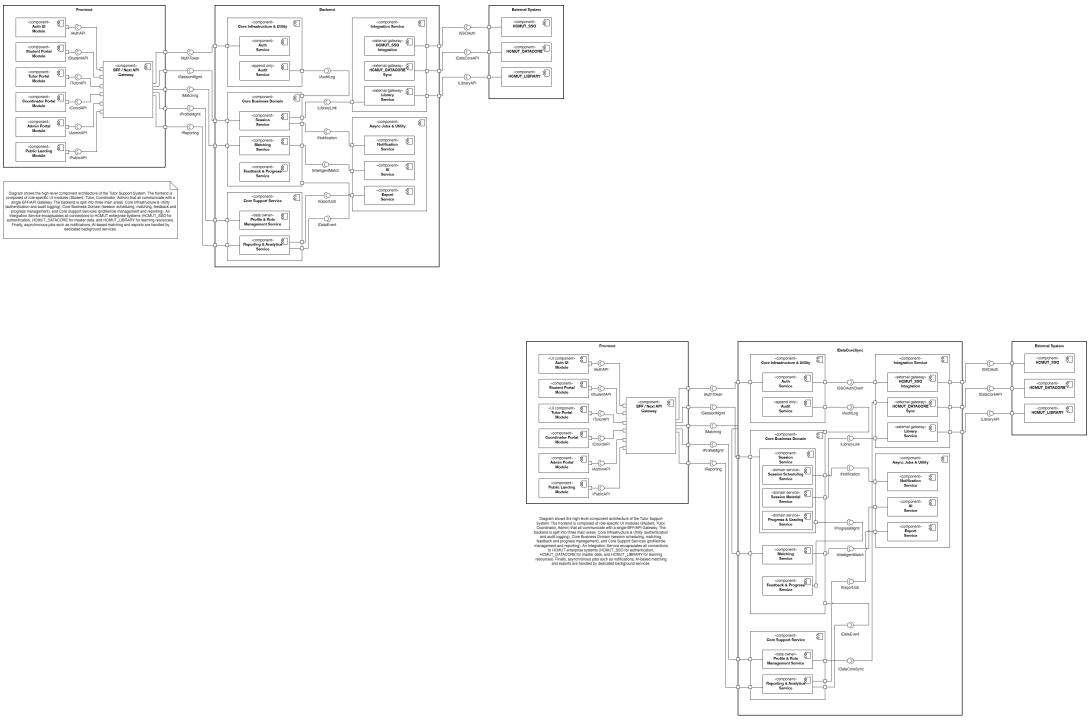

Figure 13: Component Diagram

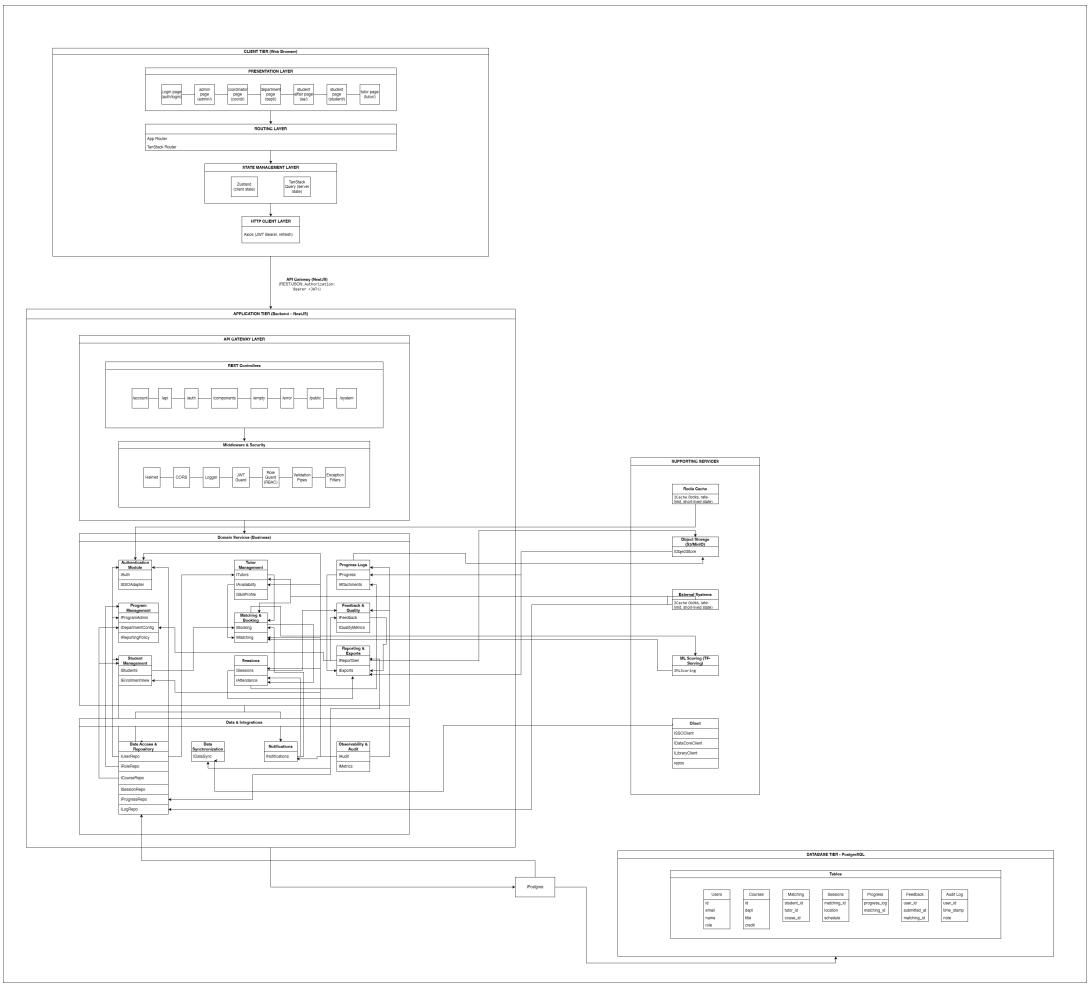

Figure 14: Package Diagram

13 Class Diagram

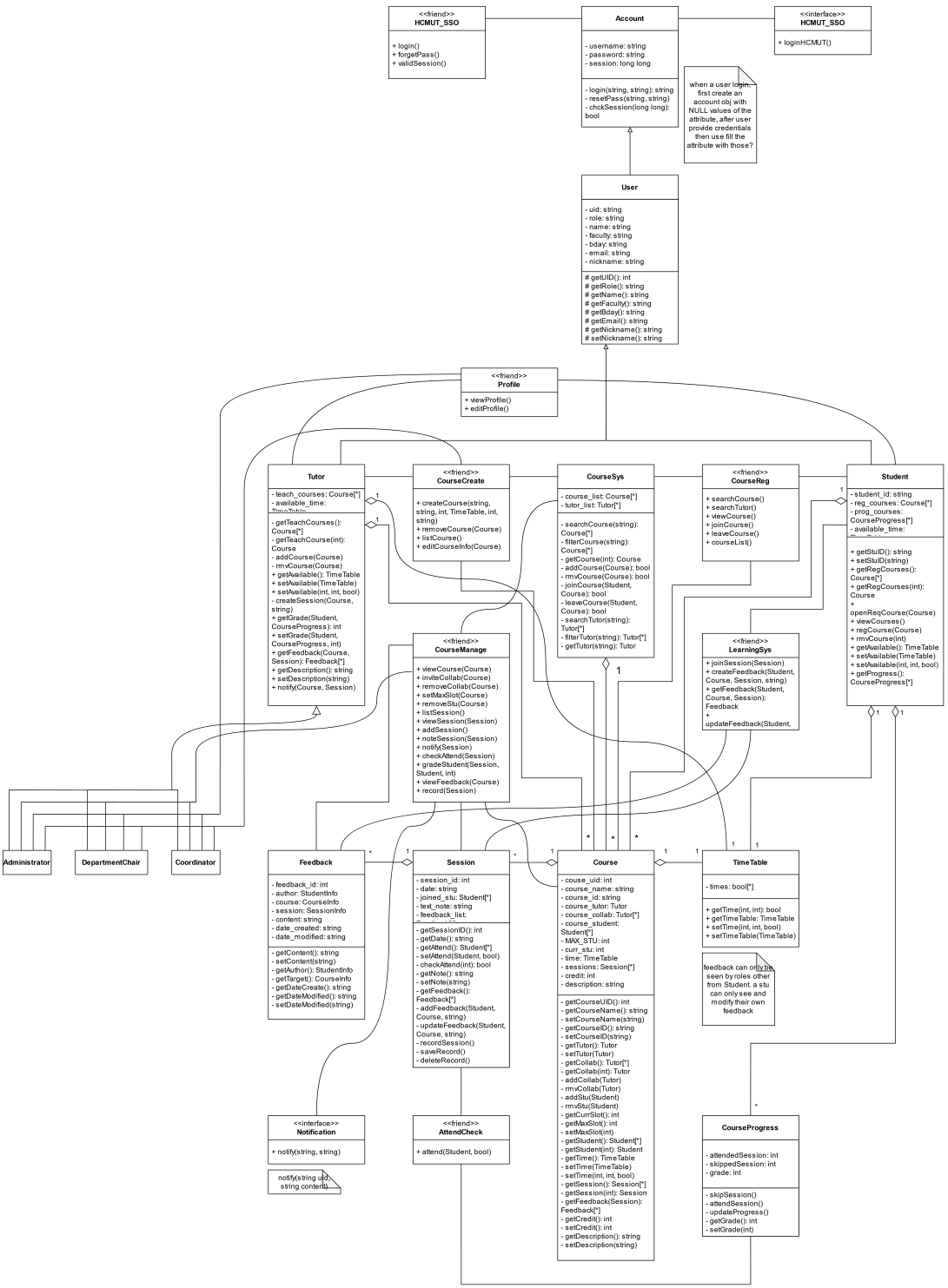

Figure 15: Class Diagram

14 Test case

14.1 Testcase Use-case 1

Test case	UC01a-01: Successful Login via SSO
Test description	User successfully logs in with valid SSO credentials and is redirected to the appropriate dashboard.
Related screens	Login Page → HCMUT_SSO Portal → Home Dashboard
Pre-conditions	1. User has valid HCMUT SSO credentials. 2. HCMUT_SSO service is operational. 3. User role exists in DATACORE.
Actions	1. Navigate to application login page. 2. Click "Login via SSO" button. 3. Enter valid SSO username and password. 4. Click "Submit". 5. System validates credentials via HCMUT_SSO. 6. System retrieves role from DATACORE. 7. User is redirected to role-appropriate dashboard.
Inputs	Any valid HCMUT SSO username and password combination that exists in HCMUT_SSO database (student, tutor, coordinator, department chair, or admin accounts).
Expected Outputs	1. User is authenticated successfully. 2. Role is assigned automatically based on DATACORE. 3. Session is established and logged. 4. User sees dashboard appropriate to their role (Student/Tutor/Coordinator/etc.).
Testing environment	Web application on Windows 10, Chrome

Figure 15: Testcase Successful Login via SSO

Test case	UC01a-02: Failed Login with Invalid SSO Credentials
Test description	User attempts to log in with invalid SSO credentials and access is denied.
Related screens	Login Page → HCMUT_SSO Portal
Pre-conditions	HCMUT_SSO service is operational.
Actions	1. Navigate to login page. 2. Click "Login via SSO". 3. Enter invalid username or password. 4. Click "Submit".
Inputs	Any invalid username/password combination including: non-existent usernames, incorrect passwords for valid usernames, expired credentials, deactivated accounts, or malformed email addresses.
Expected Outputs	1. Access is denied. 2. Error message displayed: "Invalid credentials. Please try again." 3. User remains on login page. 4. No session is created. 5. Failed login attempt is logged.
Testing environment	Web application on Windows 10

Figure 16: Testcase Failed Login with Invalid SSO Credentials

Test case	UC01a-03: Login with Empty/Null Credentials
Test description	System validates that both username and password fields are required.
Related screens	Login Page
Pre-conditions	HCMUT_SSO service is operational.
Actions	1. Navigate to login page. 2. Click "Login via SSO" button. 3. Leave one or both fields empty. 4. Click "Submit".
Inputs	Empty username, empty password, or both fields empty (null values).
Expected Outputs	1. Validation error displayed. 2. Error message: "Please enter both username and password." 3. Required fields highlighted. 4. No authentication request sent to SSO. 5. User remains on login page.
Testing environment	Web application on Windows 10

Figure 17: Testcase Login with Empty/Null Credentials

Test case	UC01a-04: Login When SSO Service is Unavailable
Test description	System displays appropriate error message when SSO service is down or unreachable.
Related screens	Login Page
Pre-conditions	HCMUT_SSO service is down or unreachable (mocked).
Actions	1. Navigate to login page. 2. Click "Login via SSO" button. 3. System attempts to connect to SSO service.
Inputs	Any input (service unavailability tested regardless of input).
Expected Outputs	1. Error message: "SSO service is currently unavailable. Please try again later." 2. User cannot proceed with login. 3. Error is logged in system logs. 4. No session is created.
Testing environment	Test environment with mocked SSO service outage

Figure 18: Testcase Login When SSO Service is Unavailable

Test case	UC01a-05: Role Assignment for All User Types
Test description	System correctly assigns appropriate role based on DATACORE data for all user types.
Related screens	Login Page → HCMUT_SSO → Role-Specific Dashboard
Pre-conditions	1. User has valid SSO credentials. 2. User role data exists in DATACORE. 3. HCMUT_SSO service is available.
Actions	1. Navigate to login page. 2. Complete SSO authentication. 3. System fetches role from DATACORE. 4. System assigns appropriate role. 5. User is redirected to role-specific dashboard.
Inputs	Valid credentials for any user type: Student, Tutor, Coordinator, Department Chair, or Administrator accounts.
Expected Outputs	1. User authenticated successfully. 2. Correct role assigned based on DATACORE (Student/Tutor/Coordinator/Chair/Admin). 3. User sees role-appropriate dashboard and features. 4. User menu shows role-specific options. 5. Role assignment is logged.
Testing environment	Web application on Windows 10, Chrome

Figure 19: Testcase Role Assignment for All User Types

Test case	UC01b-01: View Profile Information
Test description	Authenticated user can view their complete profile information synchronized from DATACORE.
Related screens	Profile Page
Pre-conditions	1. User is authenticated via SSO. 2. User profile exists in DATACORE. 3. System is operational.
Actions	1. Log in with valid credentials. 2. Navigate to "Profile" or "My Profile" section. 3. View profile information.
Inputs	Any authenticated user (all role types).
Expected Outputs	1. Profile page displays all fields: Student/Staff ID, Full Name, Email, Faculty/Department, Role. 2. Data matches DATACORE information. 3. Profile view is logged.
Testing environment	Web application on Windows 10

Figure 20: Testcase View Profile Information

Test case	UC01b-02: Update Non-Core Profile Details
Test description	User can successfully update editable profile fields with various valid inputs.
Related screens	Profile Page → Edit Profile
Pre-conditions	1. User is authenticated via SSO. 2. User profile exists in DATACORE.
Actions	1. Navigate to profile page. 2. Click "Edit Profile". 3. Update non-core fields (phone, bio, preferences). 4. Click "Save" or "Update".
Inputs	Any valid data for editable fields: phone numbers (various formats), bio text (up to character limit), preferences (all available options), profile picture, language preference, notification settings.
Expected Outputs	1. Profile updated successfully. 2. Success message: "Profile updated successfully". 3. Changes saved to database with timestamp. 4. Updated information visible on profile page. 5. Update event is logged.
Testing environment	Web application on Windows 10, Firefox

Figure 21: Testcase Update Non-Core Profile Details

Test case	UC01b-03: Update Profile with Invalid Data
Test description	System validates and rejects invalid profile update inputs.
Related screens	Profile Page → Edit Profile
Pre-conditions	User is authenticated and has a profile.
Actions	1. Navigate to profile page. 2. Click "Edit Profile". 3. Enter invalid data in editable fields. 4. Click "Save".
Inputs	Various invalid inputs: malformed phone numbers, text exceeding maximum length limits, special characters in restricted fields, SQL injection attempts, XSS scripts, invalid file types for profile picture, empty required fields.
Expected Outputs	1. System validates input and rejects invalid data. 2. Appropriate error messages displayed for each validation failure. 3. No changes saved to database. 4. User remains on edit page to correct errors. 5. Invalid fields highlighted.
Testing environment	Web application on Windows 10

Figure 22: Testcase Update Profile with Invalid Data

Test case	UC01b-04: Attempt to Update Core Profile Fields
Test description	Core profile fields synchronized from DATACORE cannot be edited by any user.
Related screens	Profile Page → Edit Profile
Pre-conditions	User is authenticated and has a profile synchronized from DATACORE.
Actions	1. Navigate to profile page. 2. Click "Edit Profile". 3. Attempt to modify core fields (ID, Name, Email, Faculty, Role).
Inputs	Any attempted modification to core fields: Student/Staff ID, Full Name, Email address, Faculty/Department, Role.
Expected Outputs	1. Core fields are read-only (disabled/greyed out). 2. Error message if modification attempted: "Core profile fields are managed by HCMUT DATACORE and cannot be modified". 3. No changes are saved to core fields regardless of method used.
Testing environment	Web application on Windows 10

Figure 23: Testcase Attempt to Update Core Profile Fields

Test case	UC01b-05: Profile Sync with DATACORE
Test description	Profile updates are synchronized with DATACORE when service is available.
Related screens	Profile Page
Pre-conditions	1. User is authenticated. 2. DATACORE is available. 3. User has updated non-core profile fields.
Actions	1. Update non-core profile fields. 2. Save changes. 3. System initiates sync with DATACORE.
Inputs	Any valid updates to non-core fields.
Expected Outputs	1. Profile update saved locally. 2. System transmits data to DATACORE. 3. Sync status logged with timestamp. 4. Confirmation of successful sync displayed.
Testing environment	Web application with DATACORE integration

Figure 24: Testcase Profile Sync with DATACORE

14.2 Testcase Use-case 2

Test case	UC02-01: Successful Manual Tutor Search by Subject
Test description	Student searches for tutors by subject and receives a list of matching tutors.
Related screens	Find a Tutor → Search Results
Pre-conditions	1. Student is authenticated. 2. At least one tutor profile with subject expertise exists. 3. System is operational.
Actions	1. Log in as student. 2. Navigate to "Find a Tutor" page. 3. Enter subject name "Calculus" in search field. 4. Click "Search" button. 5. View search results.
Inputs	Any subject name that exists in the tutor database (e.g., Calculus, Physics, Programming, Mathematics, etc.)
Expected Outputs	1. System displays list of tutors teaching Calculus. 2. Each tutor shows: name, subject(s), availability, rating. 3. Results sorted by relevance or rating. 4. "Select" or "Request Match" button available for each tutor. 5. Search query logged.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Successful Manual Tutor Search by Subject

Test case	UC02-02: Search Tutor by Availability Filter
Test description	Student filters tutor search results by availability time slots.
Related screens	Tutor Search Page → Filtered Results
Pre-conditions	1. Student is authenticated. 2. Multiple tutors exist with different availability schedules.
Actions	1. Navigate to tutor search page. 2. Apply availability filter: "Monday 9:00-12:00". 3. Click "Search" or "Apply Filter". 4. View filtered results.
Inputs	Any valid time slot filter (day of week and time range)
Expected Outputs	1. Only tutors available during selected time displayed. 2. Each result shows availability slots. 3. No tutors without matching availability shown. 4. Filtered search logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Search Tutor by Availability Filter

Test case	UC02-03: No Tutors Found - Broaden Search Suggestion
Test description	When no tutors match search criteria, system suggests broadening the search.
Related screens	Tutor Search Page
Pre-conditions	Search criteria is very specific with no matching tutors.
Actions	1. Log in as student. 2. Navigate to tutor search. 3. Enter very specific criteria (rare subject + specific time). 4. Click "Search".
Inputs	Very specific/rare combinations of subject and availability that have no matching tutors
Expected Outputs	1. Empty result set returned. 2. Message: "No tutors found matching your criteria. Try broadening your search." 3. Suggestions: Remove filters, try different times, search related subjects. 4. Option to request coordinator assistance. 5. Zero-result search logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase No Tutors Found – Broaden Search Suggestion

Test case	UC02-04: Successful Manual Tutor Selection
Test description	Student successfully selects a tutor and sends a match request.
Related screens	Search Results → Match Request Confirmation
Pre-conditions	1. Student has searched and found suitable tutors. 2. Tutor is available.
Actions	1. Search for tutors. 2. Review search results. 3. Click "Select" or "Request Match" for a tutor. 4. Confirm selection.
Inputs	Any available tutor from search results
Expected Outputs	1. Pending match request created. 2. Confirmation: "Match request sent to [Tutor Name]. Awaiting tutor confirmation." 3. Tutor receives notification of match request. 4. Match status = "Awaiting Tutor Confirmation". 5. Match request logged.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Successful Manual Tutor Selection

Test case	UC02-05: Tutor Confirms Match Request
Test description	Tutor reviews and confirms a pending match request from a student.
Related screens	Pending Match Requests → Confirmation Page
Pre-conditions	1. Pending match request exists from student. 2. Tutor is authenticated. 3. Tutor has capacity to accept more students.
Actions	1. Log in as tutor. 2. Navigate to "Pending Match Requests". 3. View match request details. 4. Click "Accept" or "Confirm Match". 5. Confirm action.
Inputs	Any pending match request
Expected Outputs	1. Match confirmed and finalized. 2. Both student and tutor receive confirmation notification. 3. Student sees matched tutor in "My Tutor" section. 4. Tutor sees student in "My Students" section. 5. Match status = "Active", logged with timestamp.
Testing environment	Web application on Windows 10, Firefox

Figure XX: Testcase Tutor Confirms Match Request

Test case	UC02-06: Tutor Rejects Match Request
Test description	Tutor declines a match request and student is notified to search for another tutor.
Related screens	Pending Match Requests
Pre-conditions	Pending match request exists, tutor is authenticated.
Actions	1. Log in as tutor. 2. Navigate to pending match requests. 3. View match request. 4. Click "Reject" or "Decline". 5. Optionally provide reason: "Schedule conflict". 6. Confirm rejection.
Inputs	Any pending match request with optional rejection reason
Expected Outputs	1. Match request rejected. 2. Student notification: "Your match request was declined. You can search for another tutor or request coordinator assistance." 3. Match status = "Rejected". 4. Student can initiate new search. 5. Rejection logged with reason.
Testing environment	Web application on Windows 10

Figure XX: Testcase Tutor Rejects Match Request

Test case	UC02-07: Request Auto-Match (Automated Matching)
Test description	Student requests automated tutor matching and receives ranked recommendations.
Related screens	Find a Tutor → Auto-Match → Recommendations
Pre-conditions	1. Student is authenticated. 2. Multiple tutors available. 3. Auto-match algorithm is operational.
Actions	1. Navigate to "Find a Tutor". 2. Click "Auto-Match" or "Smart Match". 3. Enter preferences: subject, availability, mode. 4. Click "Find Best Match". 5. System runs matching algorithm.
Inputs	Subject name, availability preference (morning/afternoon/evening/weekend), and session mode (online/in-person)
Expected Outputs	1. System returns ranked list of recommended tutors. 2. Recommendations based on: expertise, availability, ratings, preferences. 3. Top 3-5 tutors displayed with match score/percentage. 4. Student can select from recommendations. 5. Auto-match request logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Request Auto-Match (Automated Matching)

Test case	UC02-08: Tutor Does Not Respond - Coordinator Notification
Test description	When tutor doesn't respond within time limit, coordinator is notified to handle the match.
Related screens	Match Request Status → Coordinator Dashboard
Pre-conditions	1. Pending match request exists. 2. Tutor has not responded within 48 hours. 3. System timeout mechanism configured.
Actions	1. Student sends match request. 2. Wait for timeout period (48 hours). 3. Tutor does not respond. 4. System triggers timeout action.
Inputs	System timeout period (configured value) and any pending match request
Expected Outputs	1. System detects timeout. 2. Coordinator receives notification: "Match request MAT-003 has not been responded to. Please assign a tutor." 3. Student notification: "Your match request is being handled by a coordinator." 4. Match status = "Coordinator Review". 5. Timeout event logged.
Testing environment	Test environment with timeout simulation

Figure XX: Testcase Tutor Does Not Respond – Coordinator Notification

Test case	UC02-09: Coordinator Manual Assignment
Test description	Coordinator manually assigns a tutor to a student when automated matching fails.
Related screens	Coordinator Dashboard → Pending Assignments
Pre-conditions	1. Coordinator is authenticated. 2. Match request is in "Coordinator Review" status. 3. Available tutors exist.
Actions	1. Log in as coordinator. 2. Navigate to "Pending Assignments". 3. View student's match request details. 4. Search for suitable tutor. 5. Select appropriate tutor. 6. Click "Assign Tutor" and confirm.
Inputs	Student needing assignment, subject required, and available suitable tutor
Expected Outputs	1. Tutor assigned to student. 2. Both parties receive notification. 3. Match status = "Active (Coordinator Assigned)". 4. Student can schedule sessions with assigned tutor. 5. Assignment logged with coordinator ID.
Testing environment	Web application on Windows 10

Figure XX: Testcase Coordinator Manual Assignment

14.3 Testcase Use-case 3

Test case	UC03a-01: Successfully Create Availability Slot
Test description	Tutor creates a new availability slot for students to book sessions.
Related screens	Set Availability → Create Slot
Pre-conditions	1. Tutor is authenticated. 2. Scheduling database is accessible. 3. No conflicting slots exist.
Actions	1. Log in as tutor. 2. Navigate to "Set Availability" or "Manage Schedule". 3. Click "Add New Slot" or "Create Availability". 4. Enter slot details: date, start time, end time, mode, location. 5. Click "Save" or "Publish".
Inputs	Future date, valid start time, valid end time (end > start), session mode (online/in-person), optional location
Expected Outputs	1. Availability slot created successfully. 2. Confirmation: "Availability slot created successfully". 3. Slot appears in tutor's calendar. 4. Slot visible to students for booking. 5. Creation logged with timestamp.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Successfully Create Availability Slot

Test case	UC03a-02: Attempt to Create Overlapping Slot
Test description	System prevents creation of overlapping availability slots.
Related screens	Set Availability
Pre-conditions	Existing slot: Monday 14:00-16:00.
Actions	1. Log in as tutor. 2. Attempt to create new overlapping slot. 3. Date: Same Monday, Start: 15:00, End: 17:00. 4. Click "Save".
Inputs	Any new time slot that overlaps with existing slot (same day, overlapping time range)
Expected Outputs	1. System detects overlap. 2. Slot creation rejected. 3. Error: "This slot overlaps with an existing availability slot (Monday 14:00-16:00). Please choose a different time." 4. New slot NOT created. 5. Conflict logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Attempt to Create Overlapping Slot

Test case	UC03a-03: Delete Unbooked Availability Slot
Test description	Tutor successfully deletes an availability slot that has not been booked.
Related screens	Availability Calendar
Pre-conditions	Tutor is authenticated, unbooked slot exists.
Actions	1. Navigate to availability calendar. 2. Select unbooked slot. 3. Click "Delete" or "Remove". 4. Confirm deletion.
Inputs	Any unbooked availability slot
Expected Outputs	1. Confirmation prompt: "Are you sure you want to delete this slot?" 2. Slot deleted upon confirmation. 3. Success message: "Slot deleted successfully". 4. Slot removed from calendar. 5. No longer visible to students. 6. Deletion logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Delete Unbooked Availability Slot

Test case	UC03a-04: Attempt to Delete Booked Slot
Test description	System prevents deletion of a slot that has been booked by a student.
Related screens	Availability Calendar
Pre-conditions	Booked slot exists.
Actions	1. Navigate to availability calendar. 2. Select booked slot. 3. Attempt to delete.
Inputs	Any booked availability slot
Expected Outputs	1. "Delete" button disabled or not visible. 2. Error if attempted: "Cannot delete a booked slot. Please cancel the session first." 3. Slot not deleted. 4. Attempt logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Attempt to Delete Booked Slot

Test case	UC03b-06: Cancellation Denied (Less than 2 Hours Before)
Test description	Cancellation is denied when requested less than 2 hours before session start.
Related screens	My Sessions
Pre-conditions	1. Session is booked. 2. Current time is less than 2 hours before session.
Actions	1. Session scheduled for 14:00. 2. Current time: 12:30 (1.5 hours before). 3. Navigate to session. 4. Click "Cancel Session".
Inputs	Session and cancellation request time with gap < 2 hours
Expected Outputs	1. Cancellation denied. 2. Error: "Cancellation denied. Sessions must be cancelled at least 2 hours in advance. Please contact your tutor directly." 3. Session remains active. 4. Denial logged with timestamp. 5. Alternative action suggested: "Contact tutor".
Testing environment	Web application on Windows 10

Figure XX: Testcase Cancellation Denied (Less than 2 Hours Before)

Test case	UC03b-01: Successful Session Booking
Test description	Student successfully books an available consultation slot with matched tutor.
Related screens	My Tutor → Book Session → Confirmation
Pre-conditions	1. Student is authenticated and matched with tutor. 2. Tutor has published available slots. 3. Slot is not booked.
Actions	1. Navigate to "My Tutor" or "Book Session". 2. View tutor's available slots. 3. Select preferred slot. 4. Choose session mode if applicable. 5. Click "Book Session" and confirm.
Inputs	Any available slot from matched tutor, with session mode selection
Expected Outputs	1. Booking confirmed. 2. Confirmation: "Session booked successfully". 3. Both parties receive notification immediately. 4. Session appears in both calendars. 5. Slot status = "Booked", no longer available to others. 6. Booking logged with timestamp.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Successful Session Booking

Test case	UC03b-02: Prevent Double Booking of Same Slot
Test description	System prevents two students from booking the same slot simultaneously.
Related screens	Book Session
Pre-conditions	Student A and B both viewing same available slot.
Actions	1. Student B books slot Mon 14:00-16:00. 2. Student A (who loaded page earlier) attempts to book same slot. 3. Student A clicks "Book Session".
Inputs	Same available slot accessed by multiple students simultaneously
Expected Outputs	1. System detects slot already booked. 2. Booking rejected. 3. Error: "This slot has already been booked. Please select another time." 4. Student A's booking NOT created. 5. Available slots refreshed. 6. Conflict logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Prevent Double Booking of Same Slot

Test case	UC03b-03: Booking Confirmation Notification
Test description	Both student and tutor receive immediate notification upon successful booking.
Related screens	Notification System
Pre-conditions	Student successfully books a session, notification system operational.
Actions	1. Student books session. 2. Check student's notifications/email. 3. Check tutor's notifications/email.
Inputs	Any booked session
Expected Outputs	1. Student receives notification within 2 seconds: "Your session with [Tutor Name] on [Date] at [Time] is confirmed". 2. Tutor receives notification within 2 seconds: "[Student Name] has booked a session with you on [Date] at [Time]". 3. Session details included in both notifications. 4. Notifications logged.
Testing environment	Web application with notification system

Figure XX: Testcase Booking Confirmation Notification

Test case	UC03b-04: 24-Hour Reminder Notification
Test description	Automatic reminder sent 24 hours before scheduled session to both parties.
Related screens	Notification System
Pre-conditions	1. Session is booked. 2. Current time is exactly 24 hours before session. 3. Automated reminder system running.
Actions	1. Book session for tomorrow at 14:00. 2. Wait for 24-hour reminder trigger (or simulate time). 3. Check notifications.
Inputs	Session scheduled 24 hours in the future
Expected Outputs	1. Reminder sent at T-24h to both student and tutor. 2. Student: "Reminder: Your session with [Tutor] is in 24 hours ([Date] at [Time])". 3. Tutor: "Reminder: Your session with [Student] is in 24 hours ([Date] at [Time])". 4. Delivery within 2 seconds. 5. Reminder logged.
Testing environment	Test environment with time simulation

Figure XX: Testcase 24-Hour Reminder Notification

Test case	UC03b-05: Successful Cancellation (More than 2 Hours Before)
Test description	Session can be cancelled when requested at least 2 hours before start time.
Related screens	My Sessions → Cancel Session
Pre-conditions	1. Session is booked. 2. Current time is more than 2 hours before session. 3. Student is authenticated.
Actions	1. Navigate to "My Sessions" or "Upcoming Sessions". 2. Select booked session. 3. Click "Cancel Session". 4. Optionally provide reason: "Schedule conflict". 5. Confirm cancellation.
Inputs	Session and cancellation request time with gap >= 2 hours
Expected Outputs	1. Cancellation approved. 2. Confirmation: "Session cancelled successfully". 3. Tutor receives cancellation notification. 4. Slot becomes available again. 5. Session status = "Cancelled by Student". 6. Cancellation logged with reason and timestamp.
Testing environment	Web application on Windows 10

Figure XX: Testcase Successful Cancellation (More than 2 Hours Before)

14.4 Testcase Use-case 4

Test case	UC04a-01: Successfully Submit Complete Feedback
Test description	Student successfully submits complete feedback after a tutoring session.
Related screens	Submit Feedback Form
Pre-conditions	1. Tutoring session has been completed. 2. Student is authenticated. 3. Feedback has not been submitted yet for this session.
Actions	1. Log in as student. 2. Navigate to "Submit Feedback" or view completed sessions. 3. Click "Submit Feedback" for completed session. 4. Fill all required fields (rating, comments). 5. Click "Submit".
Inputs	Completed session ID, rating value (1-5 scale), and text comments/feedback
Expected Outputs	1. Feedback validated and accepted. 2. Success message: "Thank you! Your feedback has been submitted successfully". 3. Feedback stored and linked to session. 4. Submission timestamp recorded. 5. Feedback status = "Submitted". 6. Data available for aggregated reports.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Successfully Submit Complete Feedback

Test case	UC04a-02: Attempt to Submit Without Required Fields
Test description	Feedback submission fails when required fields are missing.
Related screens	Submit Feedback Form
Pre-conditions	Session completed, student is authenticated.
Actions	1. Open feedback form. 2. Leave required field(s) empty (e.g., rating). 3. Fill optional fields. 4. Click "Submit".
Inputs	Missing required fields (e.g., rating), with or without optional fields filled
Expected Outputs	1. Validation error displayed. 2. Error message: "Please fill in all required fields". 3. Required field(s) highlighted in red. 4. Feedback NOT submitted. 5. User remains on feedback form. 6. Can correct and resubmit.
Testing environment	Web application on Windows 10

Figure XX: Testcase Attempt to Submit Without Required Fields

Test case	UC04a-03: Save Feedback as Draft
Test description	Student can save incomplete feedback as draft for later completion.
Related screens	Submit Feedback Form
Pre-conditions	Session completed, draft functionality available.
Actions	1. Open feedback form. 2. Fill some fields (partially complete). 3. Click "Save as Draft" or "Save for Later".
Inputs	Partially completed feedback data
Expected Outputs	1. Draft saved successfully. 2. Message: "Draft saved. You can complete it later." 3. Draft stored with session link. 4. Student can return to complete later. 5. Draft timestamp recorded. 6. Status = "Draft".
Testing environment	Web application on Windows 10, Firefox

Figure XX: Testcase Save Feedback as Draft

Test case	UC04a-04: Revise Feedback Within Grace Period (24 hours)
Test description	Student can edit and resubmit feedback within 24-hour grace period.
Related screens	Submitted Feedback → Edit
Pre-conditions	1. Feedback was submitted. 2. Less than 24 hours have passed since submission.
Actions	1. Student submits feedback at T=0. 2. At T=10 hours, student logs in. 3. Navigate to submitted feedback. 4. Click "Edit" or "Revise Feedback". 5. Modify rating or comments. 6. Click "Resubmit".
Inputs	Any submitted feedback within 24-hour grace period and modified field values
Expected Outputs	1. Edit option available (within 24h window). 2. Previous feedback loads for editing. 3. Changes can be made. 4. Resubmission successful. 5. Updated version replaces original. 6. Revision timestamp recorded. 7. Message: "Feedback updated successfully".
Testing environment	Web application on Windows 10

Figure XX: Testcase Revise Feedback Within 24-Hour Grace Period

Test case	UC04b-01: Successfully Record Student Progress
Test description	Tutor successfully records student progress after a tutoring session.
Related screens	Record Progress Form
Pre-conditions	1. Tutoring session completed. 2. Tutor is authenticated.
Actions	1. Log in as tutor. 2. Navigate to completed sessions. 3. Select specific session. 4. Click "Record Progress". 5. Enter progress notes (required). 6. Optionally enter session summary. 7. Click "Submit".
Inputs	Completed session ID and progress notes text
Expected Outputs	1. Progress record validated and saved. 2. Success message: "Progress recorded successfully". 3. Data stored in database. 4. Record linked to session and student. 5. Timestamp recorded. 6. Available for departmental review. 7. Submission logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Successfully Record Student Progress

Test case	UC04b-02: Attempt to Submit Without Required Progress Notes
Test description	Progress cannot be submitted without required notes field.
Related screens	Record Progress Form
Pre-conditions	Session completed, tutor is authenticated.
Actions	1. Open progress recording form. 2. Leave required progress notes empty. 3. Fill optional summary. 4. Click "Submit".
Inputs	Missing required progress notes field
Expected Outputs	1. Validation error. 2. Error: "Progress notes are required". 3. Progress not submitted. 4. Form remains editable. 5. Required field highlighted.
Testing environment	Web application on Windows 10

Figure XX: Testcase Attempt to Submit Without Required Progress Notes

Test case	UC04c-01: Successfully Generate Aggregated Report
Test description	Department chair generates aggregated feedback and progress reports.
Related screens	Reports → Aggregated Reports
Pre-conditions	1. Feedback and progress records exist in database. 2. Department Chair is authenticated.
Actions	1. Log in as Department Chair. 2. Navigate to "Reports" or "Aggregated Reports". 3. Select report type: "Feedback & Progress Overview". 4. Choose criteria (date range, tutor, course). 5. Click "Generate Report".
Inputs	Any valid date range and department/program filter
Expected Outputs	1. System retrieves relevant data. 2. Report generated successfully. 3. Displays: average ratings, session counts, feedback submission rate, progress indicators. 4. Charts and visualizations included. 5. Report displays within 3 seconds. 6. Generation logged.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Successfully Generate Aggregated Report

Test case	UC04c-02: Filter Report by Date Range
Test description	Reports can be filtered by specific date range for focused analysis.
Related screens	Aggregated Reports
Pre-conditions	Department Chair is authenticated, data exists across multiple time periods.
Actions	1. Navigate to reports. 2. Select date filter: "Last 3 months". 3. Generate report.
Inputs	Any relative or absolute date range filter
Expected Outputs	1. Only data from selected period included. 2. Report clearly shows date range. 3. Statistics accurate for period. 4. Filtering works correctly.
Testing environment	Web application on Windows 10

Figure XX: Testcase Filter Report by Date Range

Test case	UC04c-03: Export Report to PDF
Test description	Aggregated reports can be exported to PDF for distribution and archival.
Related screens	Report Display → Export
Pre-conditions	Report has been generated, export functionality available.
Actions	1. Generate aggregated report. 2. Click "Export to PDF". 3. Download file. 4. Open PDF.
Inputs	Generated report with any valid filter criteria
Expected Outputs	1. PDF generated successfully. 2. PDF includes: all report data, charts, visualizations, metadata. 3. PDF well-formatted with department branding. 4. Download completes successfully. 5. Export action logged.
Testing environment	Web application on Windows 10

Figure XX: Testcase Export Report to PDF

Test case	UC04c-04: No Data Available for Selected Criteria
Test description	System displays appropriate message when no data matches selected criteria.
Related screens	Aggregated Reports
Pre-conditions	Selected criteria has no matching data.
Actions	1. Log in as Department Chair. 2. Select very specific filters (e.g., future date range). 3. Click "Generate Report".
Inputs	Filter criteria that returns no matching records
Expected Outputs	1. System detects no data. 2. Message: "No data available for this selection". 3. Suggestions: "Try different date range" or "Broaden filter criteria". 4. Empty report not generated. 5. User can modify criteria.
Testing environment	Web application on Windows 10

Figure XX: Testcase No Data Available for Selected Criteria

14.5 Testcase Use-case 5

Test case	UC05a-01: Generate Departmental Report (Valid Filters)
Test description	The Department Chair generates a departmental report using valid filters and existing data.
Related screens	Reporting → Departmental Report Screen
Pre-conditions	1. User is authenticated as Department Chair. 2. Reporting module is available. 3. Attendance/performance/session data exists. 4. Reporting screen is open.
Actions	1. Open Departmental Report. 2. Select filters (term, program). 3. Click "Generate". 4. System retrieves and aggregates metrics. 5. System displays tables and charts.
Inputs	Term = Fall 2025, Program = Computer Science
Expected Outputs	1. Report is displayed with correct metrics (attendance, performance, session counts). 2. Charts and tables match underlying data.
Testing environment	Web application on Windows 10, Chrome

Figure XX: Testcase Generate Departmental Report (Valid Filters)

Test case	UC05a-02: Change Filters → Report Refresh
Test description	User changes the filters after the report is displayed, and the system refreshes results.
Related screens	Departmental Report Screen
Pre-conditions	A generated report is already visible.
Actions	1. User changes the filter (e.g., change term). 2. Click "Apply". 3. System reloads data and refreshes the report.
Inputs	Term changed from Fall 2025 → Spring 2026
Expected Outputs	Report updates according to the new filters.
Testing environment	Web application on Windows 10

Figure XX: Testcase Change Filters → Report Refresh

Test case	UC05a-03: No Data for Selected Criteria
Test description	The selected filters return no matching data, and the system displays an informative message.
Related screens	Departmental Report Screen
Pre-conditions	Database contains no matching records for selected filters.
Actions	1. Select unavailable filters. 2. Click "Generate".
Inputs	Term = 1999, Program = Computer Science
Expected Outputs	Display message: "No data available for this selection."
Testing environment	Web application on Windows 10

Figure XX: Testcase No Data for Selected Criteria

Test case	UC05a-04: Export Error (I/O or Size Limit)
Test description	Error occurs when exporting the report due to file size limit or I/O failure.
Related screens	Departmental Report → Export
Pre-conditions	1. Report is generated. 2. System simulates I/O or export size error.
Actions	1. Click "Export → PDF".
Inputs	Extremely large dataset (mocked)
Expected Outputs	Error message: "Export failed. Narrow scope or retry." No corrupted file produced; error logged.
Testing environment	Web application with mocked I/O failure

Figure XX: Testcase Export Error (I/O or Size Limit)

Test case	UC05a-05: Data Source Unavailable
Test description	Data service or database becomes unavailable while generating the report.
Related screens	Departmental Report Screen
Pre-conditions	Reporting service is offline/mocked down.
Actions	Click "Generate".
Inputs	Any valid filters (with service down)
Expected Outputs	Display message: "Data source unavailable. Please try again later."
Testing environment	Test environment with mocked service outage

Figure XX: Testcase Data Source Unavailable

Test case	UC05b-01: Load Tutor Workload Dashboard (Normal)
Test description	Academic Affairs loads the workload dashboard using valid filters.
Related screens	Workload & Demand Dashboard
Pre-conditions	1. User is authenticated as Academic Affairs. 2. Session/booking data exists. 3. Dashboard module is operational.
Actions	1. Open Dashboard. 2. Select term and department. 3. Click "Load". 4. System computes and displays workload charts.
Inputs	Term = Fall 2025, Department = Mathematics
Expected Outputs	Workload dashboard displays correct aggregates and charts.
Testing environment	Web app on Windows 10, Firefox

Figure XX: Testcase Load Tutor Workload Dashboard (Normal)

Test case	UC05b-02: Drill-down Tutor Details (Alternative)
Test description	User selects a tutor on the dashboard to view detailed workload information.
Related screens	Dashboard → Tutor Detail View
Pre-conditions	Dashboard is displayed with at least one tutor.
Actions	1. Click tutor (e.g., T01). 2. System shows session list, hours, and feedback.
Inputs	Tutor ID = T01
Expected Outputs	Detailed tutor workload is displayed correctly.
Testing environment	Web application on Windows 10

Figure XX: Testcase Drill-down Tutor Details (Alternative)

Test case	UC05b-03: No Data for Dashboard
Test description	No matching data exists, and the dashboard cannot be generated.
Related screens	Workload & Demand Dashboard
Pre-conditions	DB has zero matching session/booking records.
Actions	Select empty filters and click "Load".
Inputs	Term = 1998, Department = Physics
Expected Outputs	Display message: "No data available for this selection."
Testing environment	Web app on Windows 10

Figure XX: Testcase No Data for Dashboard

Test case	UC05b-04: Dashboard Export Error
Test description	Exporting dashboard fails due to size limit or I/O error.
Related screens	Workload Dashboard → Export
Pre-conditions	Mocked I/O or file write failure.
Actions	Click "Export (CSV)".
Inputs	Large dataset or mocked I/O failure
Expected Outputs	Error message displayed; system suggests narrowing scope or retrying.
Testing environment	Windows 10 + Mock I/O failure

Figure XX: Testcase Dashboard Export Error

Test case	UC05b-05: Data Source Unavailable (Dashboard)
Test description	Data service becomes unavailable when loading the dashboard.
Related screens	Workload Dashboard
Pre-conditions	Aggregation or database service is down.
Actions	Click "Load Dashboard".
Inputs	Any valid filters (service offline)
Expected Outputs	"Service unavailable. Try again later."
Testing environment	Test environment with mocked outage

Figure XX: Testcase Data Source Unavailable (Dashboard)

Test case	UC05c-01: Generate Participation Report (Normal)
Test description	Student Affairs generates a participation report with valid filters and thresholds.
Related screens	Participation Report Screen
Pre-conditions	1. User is Student Affairs. 2. Attendance/session data exists. 3. Reporting module functional.
Actions	1. Open Participation Report. 2. Select term, cohort, threshold. 3. Click "Generate". 4. System computes metrics and eligibility. 5. System displays results.
Inputs	Term = Fall 2025, Cohort = 2023, Threshold = 70%
Expected Outputs	Report displays correct participation metrics and eligibility classification.
Testing environment	Windows 10, Chrome

Figure XX: Testcase Generate Participation Report (Normal)

Test case	UC05c-02: Adjust Filters / Threshold (Alternative)
Test description	User adjusts threshold or cohort and the report refreshes.
Related screens	Participation Report Screen
Pre-conditions	Report is already displayed.
Actions	1. Change threshold (70% → 80%). 2. Click "Apply".
Inputs	Threshold = 80%
Expected Outputs	Updated eligibility results according to new threshold.
Testing environment	Windows 10

Figure XX: Testcase Adjust Filters / Threshold (Alternative)

Test case	UC05c-03: No Data for Participation Report
Test description	No participation or attendance data found for selected filters.
Related screens	Participation Report Screen
Pre-conditions	Database contains no matching attendance records.
Actions	Select filter with no data and click "Generate".
Inputs	Term = 1990, Cohort = 1970
Expected Outputs	"No data available for this selection."
Testing environment	Windows 10

Figure XX: Testcase No Data for Participation Report

Test case	UC05c-04: Export Error (Participation Report)
Test description	Export fails due to file size or I/O error.
Related screens	Participation Report → Export
Pre-conditions	Export engine mocked to fail.
Actions	Click "Export → PDF".
Inputs	Large dataset or simulated I/O failure
Expected Outputs	Error message; suggestion to narrow scope or retry; error logged.
Testing environment	Windows 10 + mocked I/O failure

Figure XX: Testcase Export Error (Participation Report)

Test case	UC05c-05: Data Source Unavailable (Participation Report)
Test description	Data service unavailable when generating participation report.
Related screens	Participation Report Screen
Pre-conditions	Database or aggregation service offline.
Actions	Click "Generate".
Inputs	Any valid filters (service down)
Expected Outputs	"Data source unavailable. Please try again later."
Testing environment	Test environment with mocked outage

Figure XX: Testcase Data Source Unavailable (Participation Report)

14.6 Testcase Use-case 6

Test case	UC06a-01: Successful SSO Login
Test description	User successfully logs in via HCMUT_SSO using valid credentials.
Related screens	Login Screen, HCMUT_SSO Portal
Pre-conditions	1. HCMUT_SSO service is online. 2. User has valid credentials.
Actions	1. Click "Login with HCMUT_SSO". 2. System redirects to SSO portal. 3. User enters valid credentials. 4. SSO validates credentials and returns token. 5. System creates local session and redirects user.
Inputs	Valid username and password
Expected Outputs	User is authenticated and dashboard is displayed.
Testing environment	Windows 10, Chrome, HCMUT SSO staging server

Figure XX: Testcase Successful SSO Login

Test case	UC06a-02: Automatic Single Sign-Out
Test description	Logging out from SSO logs the user out from the system automatically.
Related screens	Dashboard, Login Page
Pre-conditions	1. User authenticated via SSO. 2. Active SSO session exists.
Actions	1. User logs out at SSO portal. 2. System receives logout callback. 3. Local session terminates.
Inputs	Valid SSO session token
Expected Outputs	User is logged out and returned to login screen.
Testing environment	Browser + SSO session manager

Figure XX: Testcase Automatic Single Sign-Out

Test case	UC06a-03: Invalid or Expired Token
Test description	System must reject login when SSO returns invalid or expired token.
Related screens	Login Screen
Pre-conditions	Token validation enabled.
Actions	1. User initiates SSO login. 2. SSO returns invalid/expired token. 3. System rejects login.
Inputs	Invalid or expired token
Expected Outputs	"Invalid token. Please try again."
Testing environment	Mock invalid-token generator

Figure XX: Testcase Invalid or Expired Token

Test case	UC06a-04: SSO Service Unavailable
Test description	System handles scenario where SSO is offline or unreachable.
Related screens	Login Page
Pre-conditions	SSO endpoint offline.
Actions	1. User selects "Login with HCMUT_SSO". 2. System fails to connect to SSO.
Inputs	None
Expected Outputs	"HCMUT_SSO is unavailable. Please try again later."
Testing environment	Simulated SSO outage

Figure XX: Testcase SSO Service Unavailable

Test case	UC06a-05: Network Timeout During SSO Redirect
Test description	Network timeout occurs during redirect to SSO service.
Related screens	Login Screen
Pre-conditions	Slow or unstable network.
Actions	1. User selects SSO login. 2. Browser fails to reach SSO endpoint.
Inputs	None
Expected Outputs	"Unable to reach SSO service."
Testing environment	Network throttling simulator

Figure XX: Testcase Network Timeout During SSO Redirect

Test case	UC06b-01: Successful Synchronization
Test description	System completes synchronization with DATACORE successfully.
Related screens	Admin Sync Logs
Pre-conditions	1. DATACORE API is online. 2. System has scheduled or manual sync enabled.
Actions	1. Sync job starts. 2. System retrieves updated profiles and academic data. 3. Data is validated and compared with local records. 4. Local database updated.
Inputs	Valid DATACORE API response
Expected Outputs	Local data matches DATACORE; sync logged.
Testing environment	Controlled sync environment with DATACORE test API

Figure XX: Testcase Successful Synchronization

Test case	UC06b-02: Retry with Exponential Backoff
Test description	System retries synchronization when DATACORE API times out.
Related screens	Admin Sync Logs
Pre-conditions	DATACORE API unstable or response delayed.
Actions	1. Sync job begins. 2. DATACORE API times out. 3. System retries using exponential backoff (1s, 2s, 4s...). 4. Sync either succeeds or fails after final retry.
Inputs	Simulated timeout responses
Expected Outputs	Retry attempts logged; sync succeeds or fails gracefully.
Testing environment	DATACORE timeout simulation

Figure XX: Testcase Retry with Exponential Backoff

Test case	UC06b-03: Invalid Data Format
Test description	System handles malformed or incomplete data returned from DATACORE.
Related screens	Admin Sync Logs
Pre-conditions	Sync job running normally.
Actions	1. System retrieves malformed JSON. 2. Fails data validation. 3. Skips affected record. 4. Logs validation error.
Inputs	Malformed DATACORE API record
Expected Outputs	Invalid record skipped; error logged.
Testing environment	Mock DATACORE malformed data feed

Figure XX: Testcase Invalid Data Format

Test case	UC06b-04: Sync Failure After All Retries
Test description	System aborts synchronization after failing all retry attempts.
Related screens	Admin Sync Logs
Pre-conditions	DATACORE API fully offline.
Actions	1. Sync job starts. 2. All retry attempts fail. 3. System aborts sync. 4. Error recorded and admin notified.
Inputs	Continuous timeout/no response
Expected Outputs	Sync aborted; failure logged; alert sent.
Testing environment	DATACORE offline simulation

Figure XX: Testcase Sync Failure After All Retries

Test case	UC06c-02: Missing Role Data
Test description	System assigns default "student" role if DATACORE does not return any role for the user.
Related screens	Login Page, Dashboard
Pre-conditions	1. User logged in through SSO. 2. DATACORE returns empty or null role data.
Actions	1. System reads empty role. 2. Default "student" role is assigned. 3. System logs incident and notifies admin.
Inputs	Empty role record
Expected Outputs	1. User assigned "student" role. 2. Admin receives notification about missing role.
Testing environment	DATACORE mock API returning empty role list

Figure XX: Testcase Missing Role Data

Test case	UC06c-03: Role Conflict Detected
Test description	System detects conflicting roles from DATACORE and resolves safely.
Related screens	Admin Logs
Pre-conditions	1. User logs in. 2. DATACORE returns conflicting roles (e.g., Tutor + Admin).
Actions	1. System reads conflicting roles. 2. Conflict is logged for audit. 3. System assigns least-privileged role (student) to ensure safety.
Inputs	Conflicting role list
Expected Outputs	1. Conflict recorded in logs. 2. User assigned fallback (student) role.
Testing environment	Mock API returning two or more conflicting roles

Figure XX: Testcase Role Conflict Detected

Test case	UC06d-01: Successful Resource Linking
Test description	User attaches a library resource to a tutoring session or summary.
Related screens	Session View, Library Search Interface
Pre-conditions	1. Library API is online. 2. User has valid access rights.
Actions	1. User searches for a resource. 2. System requests metadata from Library API. 3. System verifies access rights. 4. Resource is attached or opened.
Inputs	Valid resource ID
Expected Outputs	1. Resource linked successfully. 2. Access logged.
Testing environment	Mock Library API

Figure XX: Testcase Successful Resource Linking

Test case	UC06d-02: Access Denied
Test description	System denies access when user lacks permission for selected resource.
Related screens	Library Access Popup, Session View
Pre-conditions	1. Resource requires higher privileges. 2. User's role does not meet requirements.
Actions	1. User selects restricted resource. 2. System checks permissions. 3. System denies access.
Inputs	Restricted resource ID
Expected Outputs	1. Error displayed. 2. Event logged.
Testing environment	Permission-controlled Library API

Figure XX: Testcase Access Denied

Test case	UC06d-03: Library Service Unavailable
Test description	Library API is offline or unreachable.
Related screens	Library Search, Session View
Pre-conditions	Library API unavailable.
Actions	1. User attempts to attach resource. 2. System sends API request. 3. API does not respond.
Inputs	Any resource ID
Expected Outputs	1. Message displayed: "Library service unavailable". 2. User may retry. 3. Event is logged.
Testing environment	Simulated Library API outage

Figure XX: Testcase Library Service Unavailable

14.7 Testcase Use-case 7

Test case	UC07-01: Successful Quiz Generation
Test description	System generates quizzes using course materials and online resources.
Related screens	Quiz Generation Screen
Pre-conditions	1. System running. 2. Internet connection active. 3. AI service online. 4. Course has uploaded learning materials.
Actions	1. Student selects "Generate AI Quiz". 2. System retrieves course materials. 3. AI processes and analyzes the content. 4. Internet search for related academic resources. 5. Quiz is generated and displayed.
Inputs	Course ID with materials
Expected Outputs	1. Quiz displayed on screen. 2. Student may download quiz as a file. 3. Quiz persists until logout or course completion.
Testing environment	Browser, AI API mock, stable internet

Figure XX: Testcase Successful Quiz Generation

Test case	UC07-02: No Learning Materials
Test description	Course has no materials → system requests manual topic entry.
Related screens	Quiz Generation Screen
Pre-conditions	Course contains zero uploaded learning materials.
Actions	1. Student requests quiz generation. 2. System detects missing materials. 3. System prompts student to manually type key topics.
Inputs	Empty material repository
Expected Outputs	Message: "No materials found. Please provide topics manually."
Testing environment	Browser + course with no materials

Figure XX: Testcase No Learning Materials

Test case	UC07-03: Invalid Login
Test description	User not logged in attempts to access AI quiz generation.
Related screens	Login Screen
Pre-conditions	Student session invalid or expired.
Actions	1. Student opens quiz generator. 2. System checks login token. 3. Access denied.
Inputs	Invalid/expired authentication token
Expected Outputs	Error displayed: "Invalid login. Access denied."
Testing environment	Browser, expired login session

Figure XX: Testcase Invalid Login

Test case	UC07-04: AI Service Unavailable
Test description	AI engine offline → quiz cannot be generated.
Related screens	Quiz Generation Screen
Pre-conditions	AI API offline or unreachable.
Actions	1. Student requests AI quiz. 2. System sends request to AI service. 3. AI endpoint does not respond.
Inputs	Any quiz request
Expected Outputs	Error message: "AI service unavailable. Please try again later."
Testing environment	AI API outage simulation

Figure XX: Testcase AI Service Unavailable

Test case	UC07-05: Internet Connection Lost
Test description	Internet disconnects during quiz generation.
Related screens	Quiz Generation Screen
Pre-conditions	Internet unstable or limited.
Actions	1. AI processes materials. 2. Internet interruption occurs. 3. System cannot fetch online resources.
Inputs	Any request during unstable internet
Expected Outputs	Message: "Internet connection lost. Retry once stable."
Testing environment	Network throttling/offline simulator

Figure XX: Testcase Internet Connection Lost

Test case	UC07-06: Corrupted Course Materials
Test description	Uploaded documents unreadable or corrupted.
Related screens	Course Materials Viewer
Pre-conditions	One or more files unreadable by system or AI.
Actions	1. System attempts to load course materials. 2. Corrupted file detected. 3. Affected files are skipped.
Inputs	Corrupted PDF/Word/text file
Expected Outputs	1. Notification: "Some materials cannot be processed." 2. Only readable content used for quiz.
Testing environment	Corrupted file samples

Figure XX: Testcase Corrupted Course Materials

Test case	UC07-07: No Online Academic Resources Found
Test description	AI search returns no usable related academic content.
Related screens	Quiz Generation Screen
Pre-conditions	AI search engine operational but zero results found.
Actions	1. AI analyzes course materials. 2. AI queries search engines. 3. Zero relevant results returned.
Inputs	Course with niche/uncommon topics
Expected Outputs	Message: "No external academic resources found. Using course materials only."
Testing environment	Mock search API returning empty results

Figure XX: Testcase No Online Academic Resources Found

Test case	UC07-08: AI Quiz Generation Timeout
Test description	Quiz generation exceeds allowed processing time.
Related screens	Quiz Loading Screen
Pre-conditions	AI service slow or overloaded.
Actions	1. Student starts quiz generation. 2. AI begins processing. 3. Processing exceeds timeout threshold.
Inputs	Any quiz request during overload
Expected Outputs	Message: "Quiz generation timed out. Please try again later."
Testing environment	AI API slowdown simulation

Figure XX: Testcase AI Quiz Generation Timeout

15 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.

16 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.)

17 Submission and Deliverables

Deliverables:

- PDF generated from this 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.

18 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.

19 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).
2.0	Complete State, Activity, Sequence and Mockup diagrams (01 November 2025).
3.0	Complete Development, Class, Test case, Deployment diagrams (17 November 2025).