Created by Yehia , Lokman, Abdelmoum Gp_1

Use case name: View Personal information

Actor: student

Goal: The student checking personal information(academic year,

assigned modules,timetable)

Scenario:the use case begins when student click the “information” button

Precondition:

1.Student Authentication: The student must be successfully logged into the system.

2.Valid Student Record: The student must have an active and valid record existing in the system.

Main flow

1. The student navigate to the main dashboard and clicks the “information button”.
2. The system displays a personal information overview screen showing key details (student name, ID, academic Year).

3. The system select category to view from a menu (“Assigned modules”).

4. The system retrieves and displays the list of modules the student is assigned for the current academic year.

5. The student select the timetable, schedule option.

6. The system generates and displays the student’s personal

7. The student reviews the timetable.

8. The use case ends when the studen closes the information window or navigates away.

Alternative Flow

A1. view timetable first

-The student immediately selects the "Timetable" option from the menu.
-The system generates and displays the student's personal timetable.
-The student reviews the timetable and the use case ends.

A.2 Return to Dashboard
- At any point after the information is displayed (after Step 2 of the Main Flow), the student can click a "Back to Dashboard" or "Close" button.

- The system closes the information view and returns the student to the main application dashboard. The use case ends.

Exception Flow:

E.1 Student Record Not Found

- The system aborts the use case and displays an error message: "Invalid student record. Please contact administration."

E.2 Timetable Not Yet Generated
The system displays an informative message: "Your timetable is not yet available. Please check back later."

Postcondition:

-The student's personal information, assigned modules, and timetable in the system database remain unmodified.

Constraint:
The student can only view their own personal information and must not be able to access the data of any other student.

Use case Mark Attendance (Student)

Actor: student.

Goal: the student mark their attendance to specific class.

Scenario Start:

the use case begins when student click “mark attendance” button after scanning the QR Code to enter the class.

Preconditions:

The student must be authenticated into the system.
The student must not be exclusion.
The class session is currently open for attendance marking.

Main Flow:

1. The system display the list of active sessions.
2. The student selects the desired sessions.
3. The system checks that the sessions is available.
4. The student clicks “Mark attendance”.
5. The system records the attendance and updates the student’s record.
6. A confirmation message appears.

Alternative Flow:

A1. student’s cancel the action
at any point before clicking “Mark attendance” between step 2 and 4 the student can click a cancel button.

Exception Flow:

E1. The system prevents the registration and displays an error message: "You cannot register, more than 30 minutes have passed since the session started."

E2. The QR code does not clear “scan again the QR code”.

E3. The system check that student already attendance “You already Attendance in this class.”.

E4. Network or database failure “check the internet the attendance cannot be saved”.

Postconditions:

The student’s attendance record is successfully updated in the database.

Constraints:
Attendance must be marked within the session’s active time window (within 30 minutes after class start).

Use Case Name: View Personal Attendance Information
Actor: Student

Goal: To allow the student to check their own attendance record for their enrolled modules or sessions.

Scenario: The use case begins when the student wants to review their attendance history and navigates to the attendance section from their dashboard.

Preconditions:

1.The student must be successfully logged into the system.

2.The student must be enrolled in at least one module for the current academic period.

Main Flow:
1.The student selects the "My Attendance" option from their dashboard.

2.The system retrieves and displays a summary of the student's attendance, typically showing modules and their overall percentages.

3.The student selects a specific module to view in detail.

4.The system displays the detailed attendance record for that module, listing dates, sessions, and statuses (Present, Absent, etc.).

5.The student finishes reviewing the record and the use case ends.

Alternative Flows:

1. From the dashboard, the student selects a "View Full Attendance Report" option.

2. The system immediately displays a detailed list of all attendance records across all modules, skipping the summary view.

Exception Flows :

1. No Attendance Records Found

2. Data Unavailable

Postconditions:

1.Information Displayed: The student has successfully viewed their attendance information, either as a summary, a detailed list, or both.

Data Freshness:

1. The displayed attendance information must be current as of the last synchronized data update from the official attendance system (it may not reflect a session marked just seconds ago).

Use Case Name: View Personal Information (Teacher)

Actor: Teacher

Goal: To allow the teacher to check their personal and professional information (schedule, assigned courses, contact details).

Scenario: The use case begins when the teacher navigates to their profile or information section from the main portal.

Main Flow:

1. The teacher logs in and selects the "My Profile" or "My Information" option from the dashboard.

2.The system retrieves and displays the teacher's basic personal information (Name, Employee ID, Email, Department).

3.The teacher selects the "My Schedule" tab.

4.The system retrieves and displays the teacher's weekly teaching timetable.

5.The teacher selects the "Assigned Courses" tab to view a list of courses they are teaching for the semester.

Preconditions:

1.The teacher must be successfully authenticated and logged into the system.

2.The teacher's profile and assignment data must exist and be accessible in the system.

Alternative Flows:

1. View Contact Details First: From the basic info screen (Step 2), the teacher directly selects a "Contact Details" section to view their office number and phone number before checking their schedule.

2. Print Schedule: While viewing the timetable (Step 4), the teacher clicks a "Print" button to generate a printer-friendly version of their schedule.

Exception Flows:

1. Schedule Not Published: The system cannot find a published schedule and displays: "Your schedule for the current semester is not yet available. Please contact the department."

2. Data Connection Lost: The system fails to load the information and displays a "Connection Error. Please check your network and try again." message with a retry button.

Postcondition:

The teacher has successfully viewed their non-sensitive personal and professional information. No data is modified.

Constraint:

The system must only display information related to the authenticated teacher's own profile and assignments.

Use Case Name: Validate Student Absences

Actor: Teacher

Goal: To allow the teacher to review and officially validate (approve or reject) student absence requests for their courses.

Scenario: The use case begins when the teacher navigates to the absence management section to check for pending requests.

Main Flow:

1.The teacher logs in and selects the **"Absence Requests"** or **"Attendance Approval"** option.
2.The system displays a list of pending absence requests from students in the teacher's courses, showing student name, course, date, and reason.
3.The teacher selects a specific absence request to view detailed information (supporting document, student's explanation).
4.The teacher validates the request by selecting an action: **"Approve"** or **"Reject"**.
5.The system records the decision, updates the student's attendance record, and notifies the student. A confirmation message is shown to the teacher.

Preconditions:

1.The teacher must be successfully authenticated and logged in.
2.The teacher must have at least one assigned course with pending student absence requests.

Alternative Flows:

1. Filter by Course: From the list (Step 2), the teacher filters the pending requests to show only those for a specific course.
2. Add a Comment: While approving or rejecting (Step 4), the teacher adds an optional comment (e.g., "Request approved, please check the recorded lecture") before submitting the decision.

Exception Flows:

1. Request Already Processed: When the teacher tries to validate a request, the system detects it was already approved/rejected by another teacher or administrator and displays: "This request has already been processed."

2. Invalid Data: The system fails to save the decision due to a data conflict and displays: "Action failed. The request may have been modified. Please refresh the page and try again."

Postcondition:

•The student's absence request has been updated with the teacher's decision (Approved/Rejected), and the official attendance record is modified accordingly.

Constraint:

1. teacher can only validate absence requests for students who are enrolled in their own assigned courses.

Use Case Name: Create Class Session

Actor: Teacher

Goal: To allow the teacher to schedule a new single class session for one of their assigned courses.

Scenario: The use case begins when the teacher needs to add a new class (a lecture, tutorial, or lab) to the timetable.

Main Flow:

1.The teacher navigates to the "Timetable" or "Schedule" section and clicks the "Create New Session" button.

2.The system displays a form, prompting for session details (Course, Date, Start/End Time, Location, Session Type).

3.The teacher fills in the required details and clicks "Submit".

4.The system saves the new session, adds it to the timetable, and displays a confirmation message: "Class session created successfully."

Preconditions:

1.The teacher must be successfully authenticated and logged into the system.

2.The teacher must have at least one assigned course for which they can create a session.

Alternative Flows:

1. Recurring Sessions: After Step 2, the teacher selects a "Recurring Session" option and defines a pattern (every Tuesday for 4 weeks), and the system creates all sessions at once.

Exception Flows:

1. Scheduling Conflict: The system detects a conflict (room double-booked, teacher already scheduled) and displays an error: "Scheduling conflict detected. Please choose a different time, date, or location."

2. Invalid Input: The teacher submits the form with missing or invalid data (end time before start time). The system rejects the submission and displays: "Please correct the errors in the form."

Postcondition:

A new class session is officially created and visible in the system's timetable for the specified course.

Constraint:

A session can only be created for a course that the authenticated teacher is officially assigned to teach.

Use Case Name: Assign Teacher to Modules

Actor: Administrator

Goal: To officially assign a teacher to teach one or more modules for an academic period.

Scenario: The use case begins when the administrator needs to link a teacher to a module in the system for scheduling and management.

Main Flow:

1.The administrator navigates to the "Faculty Management" section and selects "Assign Teacher to Module".

2.The system displays a form with dropdown lists for selecting an Academic Period, a Teacher, and available Modules.

3.The administrator selects the academic period, chooses a teacher from the list, and selects one or more modules to assign to them.

4.The administrator clicks the "Confirm Assignment" button.

5.The system saves the assignment, links the teacher to the selected modules, and displays a confirmation message: "Teacher has been successfully assigned to the module(s)."

Preconditions:

1.The administrator must be successfully logged in with appropriate privileges.

2.The teacher and the module(s) must already exist as active records in the system.

Alternative Flows:

1. Set Teaching Load/Role: During assignment (Step 3), the administrator can also specify a specific role ("Primary Lecturer," "Tutorial").

Exception Flows:

1. Duplicate Assignment: The system detects that the teacher is already assigned to one of the selected modules for the same period and displays: "This teacher is already assigned to [Module Code]. Please check the selection."

2. Teacher Not Available: The system checks the teacher's existing schedule and finds a potential overload or conflict and displays a warning: "Assignment may cause a teaching load conflict. Please verify before confirming."

Postcondition:

1.The teacher is officially assigned to the selected module(s) for the specified academic period, making them available for scheduling.

Constraint:

A teacher can only be assigned to modules that are active and offered in the selected academic period.

Use Case Name: Monitor Attendance Activity Across All Modules

Actor: Administrator

Goal: To provide an overview and detailed analysis of student attendance rates and patterns for all modules in the system.

Scenario: The use case begins when the administrator needs to review overall attendance compliance, identify trends, or generate reports.

Main Flow:

1.The administrator navigates to the "Attendance Dashboard" or "Reports" section.

2.The system updates the dashboard and displays a detailed list of modules with their respective attendance percentages and student counts.

3.The administrator selects a specific module from the list to view a detailed breakdown, including a list of students with high absence rates.

Preconditions:

1.The administrator must be successfully logged in with system-wide data access privileges.

2.Attendance data must exist in the system for the selected period and modules.

Alternative Flows:

1. Generate a Report: From the dashboard (Step 2), the administrator clicks a "Generate Report" button, selects a format (PDF), and the system creates and downloads a comprehensive attendance report.

2. From the detailed module view (Step 5), the administrator clicks on a specific student's name to view their complete individual attendance history across all their modules.

Exception:

1. No Data Available: The system finds no attendance records and displays: "No attendance data is available for the selected criteria."

2. Data Export Failure: During report generation, the system fails to create the file and displays: "Report generation failed. Please try again."

Postcondition:

1.The administrator has obtained a comprehensive view of attendance activity, which may be used for reporting or decision-making.

Constraint:

The administrator can only access and monitor attendance data for modules and students within their institutional purview, in compliance with data privacy policies.

Use Case Name: Manage Accounts (Student)

Actor: Administrator

Goal: To allow the administrator to perform CRUD (Create, Read, Update, Deactivate) operations on student accounts in the system.

Scenario: The use case begins when the administrator needs to handle student account lifecycle management, such as enrolling a new student or deactivating a graduate.

Main Flow:

1.The administrator navigates to the "User Management" section and selects "Manage Student Accounts".

2.The system displays a list of all student accounts with search and filter options.

3.The administrator selects a specific student account from the list.

4.The system displays the student's full account details and profile information.

5.The administrator updates the necessary information (academic status, email) and clicks "Save Changes". The system confirms the update was successful.

Preconditions:

1.The administrator must be successfully authenticated with high-level administrative privileges.

2.The administrator must have the necessary permissions to modify user accounts.

Alternative Flows:

1. Create New Student Account: From the list view (Step 2), the administrator clicks "Create New Account", fills in the required details (Student ID, Name, etc.), and the system generates the new account with default credentials.

2. Deactivate/Activate Account: From the student's detail view (Step 4), the administrator clicks a "Deactivate Account" button to disable login, or an "Activate" button to re-enable a deactivated account.

Exception Flows:

1. Duplicate Student ID: When creating a new account, the system detects that the provided Student ID already exists and displays an error: "A student with this ID already exists. Please use a unique Student ID."

2. Integrity Constraint Violation: When saving changes, the system fails due to invalid data (e.g., a non-existent academic program) and displays: "Update failed. Please check the entered data for errors."

Postcondition:

The student account is successfully created, updated, or have its status changed in the system as intended by the administrator.

Constraint:

The administrator cannot modify certain core system-generated data, such as the unique, internal student account ID, once it is created.

Use Case Name: Manage Accounts (Teacher)

Actor: Administrator

Goal: To allow the administrator to perform CRUD (Create, Read, Update, Deactivate) operations on teacher accounts in the system.

Scenario: The use case begins when the administrator needs to handle teacher account lifecycle management, such as onboarding a new teacher or deactivating an account for a departed staff member.

Main Flow:

1.The administrator navigates to the "User Management" section and selects "Manage Teacher Accounts".

2.The system displays a list of all teacher accounts with search and filter options.

3.The administrator selects a specific teacher account from the list.

4.The system displays the teacher's full account details, profile, and assigned modules.

5.The administrator updates the necessary information (e.g., assigned modules, department, status) and clicks "Save Changes". The system confirms the update was successful.

Preconditions:

1.The administrator must be successfully authenticated with high-level administrative privileges.

2.The administrator must have the necessary permissions to modify teacher accounts.

Alternative Flows:

1. Create New Teacher Account: From the list view (Step 2), the administrator clicks "Create New Account", fills in the required details (Employee ID, Name, Department, etc.), and the system generates the new account.

Exception Flows:

1. Duplicate Employee ID: When creating a new account, the system detects that the provided Employee ID already exists and displays an error: "An account with this Employee ID already exists."

2. Assignment Conflict: When assigning modules, the system detects a scheduling or authorization conflict and prevents the save,

displaying: "Cannot assign module. Conflict detected with [Module Code]."

Postcondition:

The teacher account is successfully created, updated, or has its status changed in the system as intended.

Constraint:

The administrator cannot modify a teacher's account in a way that violates institutional hierarchy (e.g., assigning modules outside the teacher's department without proper overrides).

Use Case Name: Manage Accounts (Module)

Actor: Administrator

Goal: To allow the administrator to perform CRUD (Create, Read, Update, Deactivate/Archive) operations on module/course accounts in the system.

Scenario: The use case begins when the administrator needs to add a new module for an upcoming semester, update details of an existing module, or archive an old one.

Main Flow:

1.The administrator navigates to the "Academic Management" section and selects "Manage Modules".

2.The system displays a list of all modules with names.

3.The administrator selects a specific module from the list.

4.The system displays the module's full details (name, description, credits, assigned department, assigned teachers).

The administrator updates the necessary information and clicks "Save Changes". The system confirms the update.

Preconditions:

1.The administrator must be successfully authenticated with academic management privileges.

2.The administrator's department must have permission to manage modules (if department-level control exists).

Alternative Flows:

1. Create New Module: From the list view (Step 2), the administrator clicks "Create New Module", fills in the required details (Module Code, Name, Credits, Department), and the system creates it.

Exception Flows:

1. Active Dependencies: When attempting to archive a module, the system checks for active dependencies (e.g., students enrolled, future classes scheduled) and blocks the action, displaying: "Cannot archive module. There are active dependencies. Please remove all enrollments and scheduled sessions first."

Postcondition:

The module is successfully created, updated, or archived in the system as intended, ensuring curriculum accuracy.

Constraint :

A module cannot be deleted from the system, only archived, to maintain historical data integrity for records and transcripts.