[2024-2025]

[T.A: Al-zhraa Ahmed]

Mohamed Hamdy - Ahmed Wael - Mohamed Hussein - Mohamed Ali

[Faculty management system]

Faculty Management System

# Project Overview

The Faculty Management System is a comprehensive digital solution designed to streamline and enhance academic institution operations. It serves students, faculty members, and administrators by offering an integrated platform for attendance tracking, grade management, academic content sharing, and communication. The system aims to improve efficiency, accessibility, and user experience while ensuring data security and scalability.

# Idea

The idea behind this project is to develop a centralized faculty management system that automates and digitizes critical academic processes. By leveraging modern technologies such as AI, mobile applications, and web platforms, the system provides a seamless and intuitive experience for all users, ensuring efficiency and ease of access to academic information.

# Problems Faced

Traditional faculty management systems often suffer from several challenges, including:

* Manual Attendance Tracking: Time-consuming and prone to errors.
* Inefficient Grade Management: Lack of a centralized, real-time grading system.
* Limited Communication: Ineffective methods for sending updates to students and faculty.
* Difficulty in Content Sharing: No streamlined way to distribute lecture materials.
* Lack of Data Insights: Difficulty in analyzing student performance trends.
* Guidance for New Students: Lack of structured information on faculty facilities, roadmaps, and academic guidelines.

# Solutions Provided by Our Idea

To address these challenges, the Faculty Management System introduces:

* Automated Attendance Tracking: Implementing QR code solution.
* Digital Grade Management: Enabling faculty to input and update grades seamlessly.
* Enhanced Communication: Introducing an in-app messaging system for real-time notifications.
* Academic Content Sharing: Providing a platform for uploading and accessing lecture materials.
* AI-Powered Chatbot: Assisting students with inquiries and academic guidance.
* Smart Analytics & Reporting: Generating insights into student performance and progress.
* Student Guide & Roadmap: Offering an interactive guide for new students, providing information about the faculty, academic roadmaps, facilities, and essential guidelines for a smoother transition into university life.

# Functional Requirements:

**Student-Facing Features:**

* **Student Dashboard**: Allow students to view their class schedules, attendance records, grades, and downloadable study materials in a personalized interface.
* **Attendance Tracking**: Enable students to check into classes using a QR code scanner via the mobile app, with real-time updates to their attendance record.
* **Grade Viewer**: Provide students with access to their current and past grades, including detailed breakdowns (e.g., assignments, exams).
* **Academic Content Access**: Allow students to download lecture notes, slides, and other materials uploaded by faculty.
* **Assignment Submission**: Enable students to upload assignments digitally with deadlines and submission status tracking.
* **Exam Schedule Viewer**: Display upcoming exam dates, times, and locations with reminders.
* **Student Guide & Roadmap**: Offer an interactive guide with faculty information, academic roadmaps, campus facilities, and onboarding tips for new students.
* **Event Notifications**: Notify students of campus events, deadlines, and announcements via push notifications or in-app alerts.
* **Feedback System**: Allow students to submit anonymous feedback on courses, faculty, or facilities.
* **Personalized Alerts**: Send reminders for assignment deadlines, class cancellations, or incomplete attendance.

**Faculty-Facing Features:**

* **Faculty Dashboard**: Provide faculty with tools to manage attendance, input grades, and upload academic content for their courses.
* **Attendance Management**: Enable faculty to verify and edit QR code-based attendance records manually if needed.
* **Grade Management**: Allow faculty to enter, edit, and publish grades for assignments, exams, and overall course performance.
* **Content Upload**: Permit faculty to upload lecture materials (e.g., PDFs, videos) and categorize them by course or topic.
* **Exam & Assignment Creation**: Facilitate the creation, scheduling, and distribution of exams and assignments to students.
* **Communication Tool**: Enable faculty to send announcements or direct messages to students within their courses.
* **Timetable Management**: Allow faculty to view and update their teaching schedules, with conflict detection.
* **Student Performance Analytics**: Provide faculty with reports on individual student progress, attendance trends, and grade distributions.

**Admin-Facing Features:**

* **Admin Dashboard**: Allow administrators to oversee student and faculty data, manage user accounts, and generate system-wide reports.
* **User Role Management**: Enable admins to assign and modify roles (e.g., student, faculty, admin) with specific permissions.
* **Report Generation**: Provide tools to create detailed reports on attendance, grades, and system usage for institutional analysis.
* **Data Backup & Recovery**: Ensure admins can schedule automatic backups and recover data in case of system failure.
* **System Configuration**: Allow admins to customize settings like academic calendars, grading scales, or notification preferences.
* **General System Features**
* **AI-Powered Chatbot**: Implement a chatbot to answer student inquiries, provide academic guidance, and assist with navigation of the system.
* **Multi-Language Support**: Offer the system interface in multiple languages to accommodate diverse users.
* **Real-Time Sync**: Ensure seamless synchronization of data (e.g., grades, attendance) across web and mobile platforms.
* **Security Features**: Implement role-based access control, data encryption, and secure login (e.g., two-factor authentication) to protect user information.
* **Scalability**: Design the system to support multiple faculties or institutions with cloud-based infrastructure and modular architecture.

# Non-functional Requirements:

* 1. **Security**
     + Implement role-based access control (RBAC) to restrict unauthorized access.
     + Encrypt all sensitive data at rest (AES-256) and in transit (TLS 1.3).
     + Enforce two-factor authentication (2FA) for admin and faculty logins.
     + Implement secure coding practices to prevent SQL injection, XSS, and CSRF attacks.

# Performance Requirements

* + - The system should support 500+ concurrent users without performance degradation.
    - Critical operations (e.g., login, grade viewing) should respond within ≤2 seconds under normal load.
    - Data synchronization between web and mobile versions should occur within ≤5 seconds.

# Availability Requirements

* + - Ensure 99.9% uptime, limiting downtime to ≤8.76 hours per year.
    - Use load balancing to distribute traffic and prevent overload.
    - Implement automatic failover to backup servers in case of hardware failure.
    - Notify users at least 24 hours in advance for scheduled maintenance.

# Scalability Requirements

* + - The system should handle up to 5,000 concurrent users without performance issues.
    - Use cloud-based and modular architecture to support multiple faculties/institutions.
    - Optimize the database for handling large datasets (student records for multiple years).

# Reliability Requirements

* + - Ensure continuous operation with self-healing mechanisms to restart failed services automatically.
    - Maintain data integrity with checksums and transaction logging to prevent corruption.
    - Implement automated alerts to notify administrators of system failures.
    - In case of a partial failure, graceful degradation should allow essential functions to remain operational.

# Interoperability Requirements

* + - Integrate with third-party tools like Moodle, Google Classroom, and Microsoft Teams.
    - Provide REST or GraphQL APIs for data exchange with university systems.
    - Support Single Sign-On (SSO) using university authentication methods (e.g., LDAP, OAuth 2.0).
    - Allow exporting data in CSV, PDF, and JSON formats for external use.

# Usability Requirements

* + - Ensure an intuitive UI where users can access key features within ≤3 clicks.
    - Follow WCAG 2.1 accessibility guidelines to accommodate users with disabilities.
    - Provide multi-language support (e.g., Arabic, English).
    - Implement a chatbot with ≥80% accuracy in responding to student inquiries.

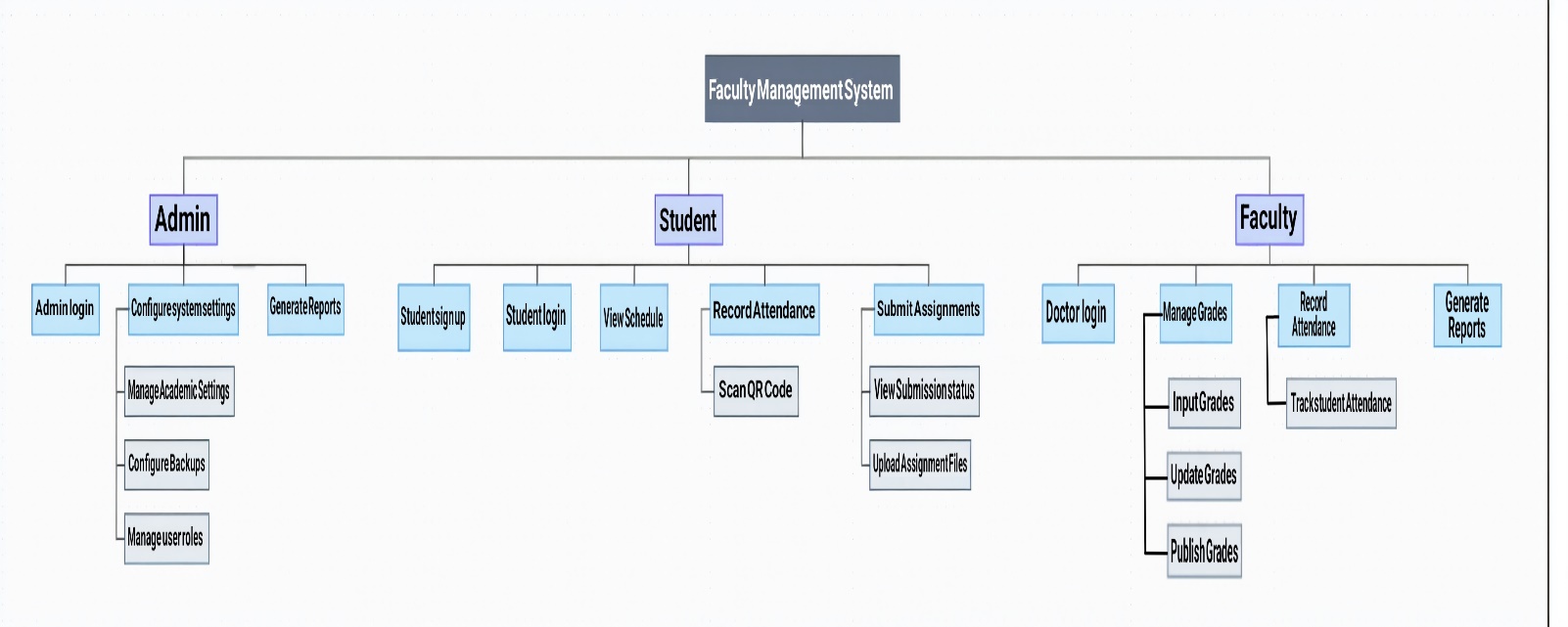
# Maintainability Requirements

* + - Use modular architecture to simplify updates and feature additions.
    - Maintain detailed system documentation for developers and administrators.
    - Design components to be independently upgradeable without affecting core functionality.

# Recovery Requirements

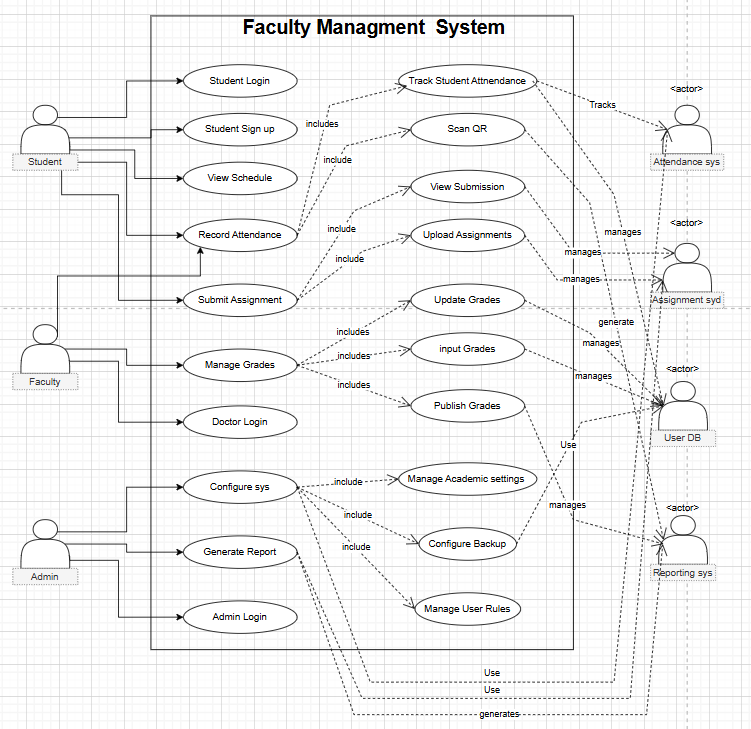
* + - Perform automated backups twice daily to ensure data safety.
    - Recovery Time Objective (RTO): Restore the system within ≤30 minutes after a failure.
    - Recovery Point Objective (RPO): Data loss should not exceed ≤5 minutes in case of failure.
    - Implement disaster recovery measures, including redundant database servers and cloud backups.
    - Notify users of system crashes with an estimated recovery time.

**(FDD) Functional Decomposition Diagram:**



1. **Use Case Model:**
   1. **Actors and Goals List**

|  |  |
| --- | --- |
| **Actor** | **Goals** |
| **Student** | * View attendance records and academic grades * Download course materials and resources * Submit assignments online * Receive notifications for events, exams, and updates * Interact with the AI Chatbot for assistance and academic inquiries |
| **Faculty** | * Manage and record student attendance * Enter and monitor student grades * Upload lectures, assignments, and educational content * Communicate with students via announcements and messages * Access course analytics and teaching schedules |
| **Administrator** | * Manage users, roles, and access permissions * Configure academic settings (e.g., grading scales, calendars) * Generate system and performance reports * Oversee system activity and logs * Handle data backup and disaster recovery |

* 1. **Use Case Diagram:  
     **
  2. **Use Case Formats**
     1. **Brief Use Case Formats**

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Primary Actors** | **Goal** |
| **Record Attendance** | Student, Faculty | Enable students to mark attendance via QR code and faculty to manage it. |
| **Manage Grades** | Faculty | Allow faculty to input, update, and publish student grades. |
| **Submit Assignments** | Student | Enable students to upload assignments and monitor submission status. |
| **Configure System Settings** | Admin | Allow administrators to manage roles, backups, and academic settings. |

* + 1. **Casual Use Case Formats**

**Use Case 1: Record Attendance Primary Actors**: Student, Faculty

1. **Main Success Scenario**:
   * Student logs into the system.
   * Navigates to the attendance page.
   * Scans the QR code displayed by the faculty.
   * System validates the QR code and student ID.
   * Attendance is marked and stored in the database.
   * Confirmation notification sent to student.
   * Faculty views updated attendance records.
2. **Alternative Scenarios**:
   * QR code not scannable → Manual check-in by faculty.
   * QR code expired → System prompts faculty to generate a new code.
   * Attendance missing → Student submits request to faculty.
   * Faculty corrects attendance manually if needed.
   * System downtime → Attendance stored locally and synced later.
   * Unauthorized scan detected → System denies and logs the attempt.
   * Faculty forgets to generate QR → System sends automated reminder.

**Use Case 2: Manage Grades Primary Actor**: Faculty

1. **Main Success Scenario**:
   * Faculty logs into the system.
   * Select a course from the dashboard.
   * Views enrolled students.
   * Input grades for each student.
   * System calculates final grade automatically.
   * Faculty reviews and confirms the entries.
   * Notifications sent to students.
2. **Alternative Scenarios**:
   * Wrong grade input → Faculty edits the entry.
   * System lag → Retry after page refresh.
   * Missing student → Admin adds student to course.
   * Calculation issues → Faculty manually adjusts.
   * Server crash during update → Auto rollback and notify.
   * Faculty forgets to publish → Reminder alert sent.
   * Data loss → Auto-save as draft enabled.

**Use Case 3: Submit Assignments Primary Actor**: Student

1. **Main Success Scenario**:
   * Student logs into the platform.
   * Navigates to the assignment section.
   * Selects the relevant course and assignment.
   * Uploads the assignment file.
   * System validates file format and size.
   * Successful upload confirmation displayed.
   * Submission confirmation sent via notification.
2. **Alternative Scenarios**:
   * File too large → Prompt to compress and retry.
   * Invalid format → System prompts for correct file type.
   * Upload fails → Retry or auto-reconnect options.
   * Missed deadline → Submission flagged as late.
   * Wrong file uploaded → Option to resubmit before deadline.
   * Server downtime → File saved locally and uploaded later.
   * Confirmation not received → Prompt to contact support.

**Use Case 4: Configure System Settings**

**Primary Actor:** Admin

1. **Main Success Scenario:**

* Admin logs into the system using admin credentials.
* Navigates to the System Configuration panel.
* The system displays configuration categories (User Roles, Backup Settings, Academic Settings).
* Admin selects the desired configuration category.
* The system displays current settings and configuration options for the selected category.
* Admin makes necessary changes:
  + For User Roles: Creates/modifies/deletes roles and adjusts permissions.
  + For Backup Settings: Sets backup frequency, storage location, and retention policy.
  + For Academic Settings:
  + Configure semesters, grading scales, and attendance policies.
  + Admin saves the configuration changes.
  + System validates the changes for consistency and security.
  + The system applies the changes and updates the configuration database.
  + System logs the configuration change with timestamp and admin ID.
  + Confirmation message displayed to admin.

1. **Alternative Scenarios**:
   * Invalid configuration → System displays validation errors and suggests corrections.
   * Conflicting settings detected → System displays conflict and provides resolution options.
   * Permission conflict → System warns about potential security issues and requests confirmation.
   * System critical settings change → System requires secondary verification (email code or supervisor approval).
   * Network interruption → System saves draft of changes and allows resuming configuration when connection restored.
   * Database error → System creates configuration backup before applying changes and provides rollback option.
   * Concurrent admin editing → System implements locking mechanisms and notifies about simultaneous edits
   * Academic year in progress → System warns about potential impact on current students and requires confirmation.
   * Backup storage full → System suggests cleanup options or alternative storage locations.
   * Security policy violation → System blocks change and notifies security administrator.
     1. **Fully Dressed Use Case Format**

|  |  |
| --- | --- |
| **Element** | **Description** |
| **1. Use Case Name** | Record Attendance |
| **2. Scope** | Faculty Management System |
| **3. Level** | User Goal |
| **4. Primary Actors** | Student, Faculty |
| **5. Stakeholders & Interests** | * **Student**: Wants easy and timely attendance registration to avoid penalties. * **Faculty**: Needs accurate records for evaluation. * **Admin**: Requires valid data for audits. * **University Management**: Monitors student participation. * **IT Support**: Ensures uptime and prevents errors. * **Parents** *(optional)*: May receive periodic attendance reports. |
| **6. Preconditions** | - Student has a valid login. - Faculty has generated a session-specific QR code. - System is live and functional. |
| **7. Success Guarantee (Postconditions)** | - Attendance is recorded for the right session and student. - Confirmation appears on student and faculty dashboards. - Data stored securely in the database. |
| **8. Main Success Scenario** | * Student logs into the mobile/web application. * Navigates to "Attendance". * Scans the QR code from the faculty. * System authenticates the session. * Attendance recorded. * Confirmation message sent. |

## Use Case 2: Manage Grades

|  |  |
| --- | --- |
| **Element** | **Description** |
| **1. Use Case Name** | Manage Grades |
| **2. Scope** | Faculty Management System |
| **3. Level** | User Goal |
| **4. Primary Actors** | Faculty |
| **5. Stakeholders & Interests** | - **Faculty**: Needs to efficiently record, update, and publish student grades.  **-Students**: Want timely and accurate grade information.  - **Department Head**: Requires grade statistics for program evaluation.  - **Academic Affairs**: Ensures compliance with grading policies.  - **IT Support**: Maintains system stability during grade processing periods.  - **Parents/Guardians**: May need access to grade reports. |
| **6. Preconditions** | - Faculty has valid system credentials.  - Course and student enrollments are configured in the system.  - Grading criteria and weightings are established.  - Faculty has appropriate permissions for the specific course. |
| **7. Success Guarantee (Postconditions)** | - Grades are accurately recorded and stored in the database.  - Grade calculations reflect the correct weighting.  - Students can view their grades when published.  - Grade reports can be generated for academic records.  - Grade history and changes are logged for audit purposes. |
| **8. Main Success Scenario** | * + Faculty logs into the system.   + Navigates to the "Grade Management" section.   + Select the appropriate course and assessment type.   + System displays the student roster with editable grade fields.   + Faculty enters grades for individual students or imports from spreadsheet.   + The system validates grade entries according to predefined criteria.   + Faculty reviews entered grades for accuracy.   + Faculty saves the grades to the system.   + Faculty chooses to publish grades to students.   + The system notifies students of grade availability. |
| **9. Extensions/Alternative Flows** | **8a. Invalid Grade Entry**:   * + System highlights invalid grade entries.   + Faculty corrects entries before proceeding.   **8b. Grade Calculation Error**:   * + The system detects mathematical inconsistency.   + Faculty reviews and adjusts calculation parameters.   **8c. Student Grade Appeal**:   * + Student submits grade appeal through the system.   + The faculty receives notification and reviews appeal.   + Faculty updates grade if appropriate.   **8d. Batch Grade Processing Failure**:   * + The system fails to process imported grades.   + Faculty is notified with specific error details.   + 3. Faculty corrects format and re-imports. |
| **10. Special Requirements** | - System must handle multiple grading scales (letter, percentage, GPA).  - Changes to grades must be logged with timestamp and user ID.  - Grade data must be encrypted at rest and in transit.  - Interface must support keyboard navigation for efficient data entry.  - System must allow grade entry for up to 500 students per course. |
| **11. Technology & Data Variations** | - Grade import supported from Excel, CSV, and LMS formats.  - Grade export available in PDF, Excel, and system-native formats.  - Curve-fitting algorithms available for statistical grade adjustment.  - Mobile interface supports limited grade entry functionality. |
| **12. Frequency of Occurrence** | - Heavy usage during mid-terms and finals (2-4 weeks per semester).  - Moderate usage for ongoing assignments throughout semester.  - Low usage for grade corrections outside active periods. |
| **13. Miscellaneous** | - System should maintain performance during peak grading periods.  - Automated backups should occur before and after major grading deadlines.  - System should provide analytics on grade distribution and student performance. |

## Use Case 3: Submit Assignments

|  |  |
| --- | --- |
| **Element** | **Description** |
| **1. Use Case Name** | Submit Assignments |
| **2. Scope** | Faculty Management System |
| **3. Level** | User Goal |
| **4. Primary Actors** | Student |
| **5. Stakeholders & Interests** | - **Student**: Wants reliable submission process with confirmation.  - **Faculty**: Needs organized access to submissions with timestamps.  - **Academic Integrity Office**: Requires plagiarism detection integration.  - **IT Support**: Ensures file storage and transfer reliability.  - **System Administrator**: Monitors storage capacity and file type security. |
| **6. Preconditions** | - Student has valid system credentials.  - Assignment has been created and published by faculty.  - Submission deadline is configured and active.  - Student is enrolled in the course. |
| **7. Success Guarantee (Postconditions)** | - Assignment files are uploaded and stored securely.  - Submission timestamp is recorded accurately.  - Faculty can access and review submitted files.  - Student receives confirmation of successful submission.  - Submission history is maintained for verification purposes. |
| **8. Main Success Scenario** | * + Student logs into the system.   + Navigates to the "Assignments" section.   + Selects the appropriate course and assignment.   + System displays assignment details and submission requirements.   + Student uploads required file(s) through the interface.   + System validates file types, sizes, and submission deadline.   + Student adds any required comments or submission notes.   + Student confirms submission.   + System processes upload and stores files.   + System generates submission receipt with timestamp.   + Students receive confirmation on screen and via email. |
| **9. Extensions/Alternative Flows** | **6a. Invalid File Type**:   * + The system rejects files and displays acceptable formats.   + Students convert files to acceptable formats and re-uploads.   **6b. File Size Exceeds Limit**:   * The system notifies students of size constraint. * Student compresses files or requests faculty assistance.   **6c. Late Submission**:   * System warns about deadline expiration. * Student confirms submission as late. * System marks submission as late for faculty review.   **9a. Upload Failure**:   * System detects network or storage issues. * Students are prompted to retry. * System provides troubleshooting options if persistent. |
| **10. Special Requirements** | - System must handle multiple file uploads simultaneously.  - Interface must show upload progress for large files.  - Storage system must maintain file integrity and prevent corruption.  - File previews should be available when possible.  - System must detect potentially malicious files. |
| **11. Technology & Data Variations** | - Uploads supported via web interface, mobile app, and API.  - File submissions supported from cloud storage integrations.  - Multiple compression formats supported for large submissions.  - Plagiarism detection integration for text submissions. |
| **12. Frequency of Occurrence** | - Moderate usage throughout semester.  - Heavy usage near assignment deadlines (typically evenings and weekends).  - Peak loads at end of semester for final projects. |
| **13. Miscellaneous** | - System should handle increased load during common submission periods.  - Recovery mechanisms should exist for interrupted uploads.  - Faculty should be able to download all submissions as a batch. |

## Use Case 4: Configure System Settings

|  |  |
| --- | --- |
| **Element** | **Description** |
| **1. Use Case Name** | Configure System Settings |
| **2. Scope** | Faculty Management System |
| **3. Level** | User Goal |
| **4. Primary Actors** | Admin |
| **5. Stakeholders & Interests** | - **Administrator**: Needs comprehensive control over system configuration.  - **Faculty**: Requires appropriate settings for teaching functions.  - **Students**: Need correctly configured academic parameters.  - **IT Department**: Ensures technical settings align with infrastructure.  - **University Management**: Requires compliance with institutional policies.  - **Security Team**: Needs proper access controls and backup protocols. |
| **6. Preconditions** | - Admin has authenticated with appropriate privileges.  - System is in maintenance mode or changes won't disrupt active users.  - Configuration change request has been approved if required.  - Previous configuration state is backed up. |
| **7. Success Guarantee (Postconditions)** | - System settings are updated according to specifications.  - Changes are logged for audit purposes.  - System maintains stability after configuration.  - Users experience the intended behavior based on new settings.  - Settings are properly stored and persisted across system restarts. |
| **8. Main Success Scenario** | * + Admin logs into the system with administrative credentials.   + Navigates to the “System Configuration" section.   + Selects configuration category (User Roles, Backups, Academic Settings).   + System displays current settings with modification options.   + Admin reviews and modifies necessary settings.   + System validates changes for consistency and security implications.   + Admin confirms changes and provides justification if required.   + The system applies changes and updates configuration database.   + System logs configuration change with timestamp and admin ID.   + The system displays confirmation of successful update. |
| **9. Extensions/Alternative Flows** | **6a. Validation Failure**:   * + System highlights invalid settings.   + Admin adjusts settings to valid values.   **8a. Critical System Setting Change**:   * The system requires secondary authorization. * Admin provides additional verification. * System applies changes with heightened logging.   **8b. Database Update Failure**:   * The system detects errors in configuration storage. * The system reverts to previous configuration. * Admin is notified of failure with details.   **8c. Configuration Conflict**:   * The system detects conflict between settings. * System suggests resolution options. * 3. Admin resolves conflicts before proceeding. |
| **10. Special Requirements** | - Configuration interface must be role-based with granular permissions.  - System must maintain configuration history with rollback capability.  - Critical settings require multi-factor authentication.  - Configuration changes must be schedulable for future activation.  - Settings must be exportable/importable for cross-environment deployment. |
| **11. Technology & Data Variations** | - Configuration via web interface and command-line tools.  - Batch configuration changes via structured files (JSON, YAML).  - API access for programmatic configuration management.  - Configuration templates for quick deployment of common settings. |
| **12. Frequency of Occurrence** | - Heavy usage during system setup and beginning of academic terms.  - Moderate usage for security and policy updates.  - Low usage for routine maintenance. |
| **13. Miscellaneous** | - System should provide simulation mode to test configuration changes.  - Automated testing of critical functions after configuration changes.  - Configuration documentation should be automatically generated. |

Class diagram:

