**4. System Requirements**

**4.1 Functional Requirements**

**1.User Management**

1. **User Roles & Permissions**

FR1.1 System shall support 4 user roles (Global Admin, Department Admin, Lecturer, Student).

FR1.2 Global Admin can add (UC-1 *Table 1*) or edit student accounts with the required details such as personal information, programme enrolment, and financial details.

FR1.3 Global Admin can add/edit department admin or lecturer accounts, assign them to specific departments.

FR1.4 Department Admin can manage student/lecturer but cannot add new student/lecturer within their department.

FR1.5 Department Admins can request Global Admins to create new Lecturer accounts.

FR1.6 All users can update their personal contact info (phone, email, address).

**b) Authentication**

FR1.7 Usernames auto generated (e.g., st123456 for students, sf6001for staff, ad9001 for admin).

FR1.8 Default passwords hashed using PHP’s password\_hash($password, PASSWORD\_DEFAULT).

FR1.9 Password reset requires email verification.

**2. Student Operations**

1. **Student Profile**

FR2.1: System shall store student attributes (name, DOB, address, education history etc.)

FR2.2: Dual-program students have records replicated in both department databases

1. **Enrolment Process include module selection**

FR2.3: The four-step enrolment workflow with progress tracking through a visual indicator (e.g., progress bar) that updates as students complete each step of the enrolment process. (UC-2 *Table 2*)

1. Accept rules/regulations → 2. Confirm Personal details

→ 3. Confirm Programme details & Module selection → 4. Fee payment

FR2.4: System shall show real-time module availability during selection

FR2.5: Prerequisite checks before module enrolment

FR2.6: Global/Department Admins can edit/delete enrolments.

**3. Academic Management**

1. **Programme Management**

FR3.1: Global Admin can add/edit/delete programmes across all departments

FR3.2: Department Admin can add/edit/delete their department's programmes only and assign module to the programme

FR3.3: Programme deletion blocked if active enrolments exist

1. **Module System**

FR3.4: Global Admin can add/edit/delete module across all departments

FR3.5: Department Admins can:

* add/edit/delete module in their department
* Assign modules as compulsory/optional in a programme
* Set capacity limits (e.g., max 50 students)
* Define prerequisites
* Assign lecturer to module

FR3.6: Module deletion blocked if active enrolments exist

FR3.7: Auto-update available slots when students enrol/drop

1. **Grading**

FR3.8: Department Admins and lecturer can add/edit grade of the student

FR3.9: System shall calculate final grades using the module’s weight:

$totalMarks = ($assignmentMarks \* ($assignmentWeight/100)) + ($examMarks \* ($examWeight/100));

FR3.10: Letter grades assigned via:

if ($totalMarks >= 90) return 'A+';

**d) Academic Tutor**

* **FR3.11:** Global/Department Admins can assign/unassign Academic Tutors (lecturer) to students.

**4. Financial System**

1. **Fee Management**

FR4.1: Auto-generate tuition fees based on programme taken

FR4.2: Support instalment payments (2 instalments per year)

FR4.3: Apply scholarship discounts automatically

1. **Payment Processing**

FR4.4: Students can view/download payment receipts

FR4.5: System shows payment statuses: Not Paid/Partially Paid/Fully Paid

1. **Scholarship Management**

FR4.6: Global admin can add/edit/delete scholarship and assign/unassign scholarship to student.

**5. Timetable & Scheduling**

1. **Scheduling**

FR5.1: Department Admins can add/edit/delete module timetables

FR5.2: Color-coded timetable display (green=lecture, blue=tutorial)

FR5.3: Conflict detection when scheduling classes

FR5.4: Lecturer & student can only view the timetable

**6. Assessment System**

1. **Assignments**
   * FR6.1: Department admin & Lecturers can add/edit/delete assignment files/descriptions
   * FR6.2: Students submit assignments with file attachments
   * FR6.3: Late submissions flagged automatically
2. **Examinations**
   * FR6.4: Department Admin can schedule exams within department
   * FR6.5: Lecturer & Students views exam timetable in list/calendar formats

**7. Support Services**

1. **Extenuating Circumstances (EC)**
   * FR7.1: Students can submit EC requests with documentation
   * FR7.2: Global Admins approve/reject requests
2. **Disability Accommodations**
   * FR7.3: Students can request special accommodations
   * FR7.4: Global Admins manage accommodation approvals/reject

**8. Reporting & Interfaces**

1. **Dashboards**
   * FR8.1: Role-specific dashboards:
     + Global Admins: Enrolment stats, financial reports
     + Department Admins: student
     + Students: Timetable, grades, fees
     + Lecturers: Teaching schedule, grade submission
2. **Data Export**
   * FR8.2: Student can generate finance report in PDF format

**9. System Integration**

1. **Database Operations**
   * FR9.1: Foreign Data Wrapper (FDW) connects central ↔ department databases
2. **API Endpoints**
   * FR9.3: REST APIs for all core functions (student CRUD, enrolment, etc.)
   * FR9.4: Mobile app compatibility via JSON responses

**10. Notifications**

1. **Alerts**
   * FR10.1: Email notifications for:
     + Enrolment completion
     + Payment due reminders
     + Grade publication
     + EC/disablity request updates

4.1.1USE CASE DIAGRAM – GLOBAL ADMIN



4.1.2 USE CASE DIAGRAM – DEPARTMENT ADMIN



4.1.3 USE CASE DIAGRAM – LECTURER



4.1.4 USE CASE DIAGRAM – STUDENT

A diagram of a student

AI-generated content may be incorrect.

4.1.5 USE CASE DESCRIPTION

|  |
| --- |
| **Use Case 1:** **Add Student** |
| **Use Case: UC-1** |
| **Actors:** Global Admin, Student |
| **Goal:** To add a new student to the system, ensuring all required details are entered, including personal, educational, and program-related information. |
| **Precondition:** The Global Admin is logged in with the necessary privileges to add students. |
| **Postcondition:** A new student is added to the system, with all relevant details captured and stored in the department database. The student's unique ID, username, and password are generated, and their enrolment details are linked to the appropriate departments and programmes. |
| **Flow of Events:**   1. The Global Admin logs into the system with valid credentials. 2. The Global Admin navigates to the "Student" page and there is a "+New" that allow to add new student. 3. The system displays the add student interface. 4. The Global Admin enters the following student details:   **Personal Details:** Name, Date of Birth (DOB), personal email, address, etc.  **Education Details:** Student type, education level, institutional name.  **Department & Program Details:** Department, Program, start date, current year, and if the student wishes to study a dual program, the system assigns them to both programmes (e.g., Computer Science and Business Management).   1. The Global Admin submit the student details. 2. The system validates the entered data (e.g., ensuring email format is correct, required fields are filled). 3. The system generates:    * **Student ID:** The system generates a unique student ID using the central database sequence (e.g., $student\_id = get\_nextval($share\_db\_conn, 'student\_id\_seq')).    * **Address ID:** The system generates an Address ID using the central database sequence (e.g., $address\_id = get\_formatted\_id($share\_db\_conn, 'address\_id\_seq', 'A', 8)).    * **Username:** The system generates the username in the format "st" followed by the last 6 digits of the student ID (e.g., st123456).    * **University Email:** The system generates the student’s email in the format: [username]@student.university.edu (e.g., [st123456@student.university.edu](mailto:st123456@student.university.edu)).    * **Password:** The system generates the Password for the student, which is set to last name + date of birth, and the password is stored in the database securely using password\_hash(). 4. The system sends a confirmation message to theGlobal Admin that the student was successfully added. 5. The system will send an email to the student as for login details, username and password. |
| **Alternative Flow:**  If any required field is missing, the system will prompt the admin to correct the errors and re-submit the form. |
| **Exceptions:**  - Invalid email address format or missing required fields.  - System unable to save the student data due to database issues. |

Table 1: Use Case Description for Add Student

|  |
| --- |
| **Use Case 2: Student Enrolment (Student Dashboard)** |
| **Use Case ID:** UC-2 |
| **Actor:** Student |
| **Goal:** To complete the student’s enrolment process, which involves confirming personal details, selecting modules, and paying fees. The process progresses through four steps: Rules and Regulations, Personal Details Confirmation, Programme and Module Selection, and Fees Payment. |
| **Precondition:** The student has been added to the system and is ready for enrolment. |
| **Postcondition:** The student's enrolment is complete, and the module selection is finalized. The student is enrolled in their chosen program(s), and the fees payment is recorded. |
| **Flow of Events (Student Enrolment Steps):**   1. The student logs into the system using their university username and password. 2. The system displays the Enrolment Dashboard, showing the four enrolment steps:   Step 1: Rules and Regulations  Step 2: Confirmation of Personal Details  Step 3: Confirmation of Programme Details and Module Selection  Step 4: Fees Payment   1. **Step 1: Rules and Regulations**    * The student reads and accepts the university's rules and regulations.    * The system records this as part of the student's progress (Progress Step: 1). 2. **Step 2: Confirmation of Personal Details**    * The system displays the student’s personal details.    * The student confirms that the details are correct and submits the confirmation.    * The system records this as part of the student's progress (Progress Step: 2). 3. **Step 3: Confirmation of Programme Details and Module Selection**    * The system displays the student’s current program(s) and available modules.    * The system shows compulsory modules and optional modules. The optional modules display the number of available slots (e.g., "50 students can be added").    * The student selects the optional modules they wish to enrol in.    * The system validates that the student has met the prerequisites for the selected optional modules and updates the available slots (e.g., if a module is selected, the available slots decrease by 1).    * The system also calculates and updates the student’s programme end date based on the duration of their program.    * The system records this as part of the student's progress (Progress Step: 3). 4. **Step 4: Fees Payment**    * The student is presented with the fees due for their program and any applicable scholarships.    * The system displays the scholarship details (if any) that have been assigned to the student.    * The student proceeds to pay the fees through the available payment methods.    * Once the payment is successful, the system creates a payment history record for the student.    * The system records this as part of the student's progress (Progress Step: 4, marking the enrolment as complete). 5. The system updates the student's enrolment status to “Enrolled” and confirms that all steps are completed. 6. The student’s module selection and payment details are saved in the system. |
| **Alternative Flow:**   * If the student does not meet prerequisites for a selected optional module, the system will prompt them to select another module or show which prerequisite modules need to be completed first. * If the student doesn’t have enough available slots in a module (due to the cap of 50 students), the system will display an error message and suggest alternative modules. |
| **Exceptions:**   * If there is a payment error during Step 4 (Fees Payment), the student will be notified, and they must retry payment. * If the student's program or modules are no longer available due to scheduling conflicts or other issues, the system will prompt them to select new options. |

Table 2: Use Case Description for Student Enrolment

**4.2 Non-Functional Requirements**

**Performance Requirements**

* **Response Time:**
  + The system should respond to any user interaction (e.g., clicking a button, loading a page) within 2 seconds under normal load conditions.
* **Throughput:**
  + The system should be able to handle 1000 concurrent users without performance degradation.
* **Scalability:**
  + The system must be able to scale horizontally, adding more servers to handle increased user load during peak times (e.g., during enrolment).

**Availability**

* **System Uptime:**
  + The system must ensure 99.9% uptime, with scheduled maintenance periods clearly communicated to users.
* **Backup and Recovery:**
  + The system should perform daily backups and have a recovery process in place that allows data restoration within 4 hours in case of a system failure.

**Security Requirements**

* **Data Encryption:**
  + All sensitive data, such as passwords and payment information, must be encrypted during transmission (using SSL/TLS) and at rest (using AES-256).
* **Authentication and Authorization:**
  + Role-Based Access Control (RBAC): The system will ensure that users only have access to the data and features relevant to their roles. This includes:
    - Global Admin: Full access to all departments, students, staff, financial information, modules, and academic records.
    - Department Admin: Limited to accessing and managing data within their specific department (e.g., only students, modules, and staff within their department).
    - Lecturer: Access to modules they teach and related student data.
    - Student: Access to their own personal information, enrolment details, module choices, grades, and financial records.
* **Password Management:**
  + Passwords should be stored using hashed and salted algorithms (e.g., bcrypt, argon2).
  + The system shall use PHP’s password\_hash() function, which applies the bcrypt algorithm by default for hashing passwords.
  + The password\_hash() function will generate a salt automatically, ensuring unique hash values even for identical passwords.
  + The system shall use password\_verify() to securely authenticate users without exposing their plain-text passwords.
* **Session Management:**
  + Sessions should expire after 30 minutes of inactivity, and users must be required to reauthenticate.

**Usability Requirements**

* **Ease of Use:**
  + The system should have an intuitive user interface that allows users (students and administrators) to navigate with minimal training.
* **Accessibility:**
  + The system should comply with WCAG 2.1 standards to ensure accessibility for users with disabilities.

**Compatibility Requirements**

* **Browser Compatibility:**
  + The system should work on the latest versions of popular web browsers (Chrome, Firefox, Safari, Edge).
* **Mobile Compatibility:**
  + The system must be responsive, meaning it should work seamlessly on mobile devices with screen sizes ranging from 5" to 12" in diagonal.

**Maintainability**

* **Code Quality:**
  + The system should follow best practices for software development, including modular code and proper documentation.
* **Error Logging and Debugging:**
  + The system must have error logging capabilities that allow system administrators to view and diagnose issues quickly. All errors must be logged in a structured format for debugging.

**4.2.1 SOFTWARE AND HARDWARE REQUIREMENTS**

**SOFTWARE REQUIREMENTS**

* PHP 5.0 and above
* APACHE HTTP Server
* Frontend: HTML, CSS, JavaScript
* Web designing language: PHP
* RDBMS: PostgreSQL 12 or later, with Foreign Data Wrapper (FDW) support for cross-database queries.
* Microsoft windows/Linux
* Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge.

**HARDWARE REQUIREMENTS**

According to [7] a web server that is capable of serving more than 1000 users should have the following specifitions.

* Intel Core i7 or h
* 512 MB Ram or Higher
* 20GB HDD or Higher
* Network Connectivity

But for development purposes, any desktop or laptop computer can be used.

5.

a) A diagram of a web browser

AI-generated content may be incorrect.

system architecture diagram

 **Presentation Layer (Frontend - UI)**

* This layer represents what the user sees and interacts with. It will run in a web browser, handling user requests and presenting data.
* The UI will communicate with the application layer to handle business logic and interact with the data layer (databases).
* The **Frontend** will include components like:
  + User Dashboard
  + Student Enrolment Process
  + Module Selection
  + Fee Payment
  + Notifications (e.g., payment due reminders)

 **Application Layer (Web Server - PHP + Application Logic)**

* This layer acts as an intermediary between the **Frontend** and the **Data Layer**. It processes requests, applies business rules, and interacts with databases.
* Components in this layer include:
  + **Web Server (PHP)**: Handles HTTP requests and responses.
  + **Authentication/Session Management**: Manages user logins and sessions.
  + **Business Logic**: Handles user-specific logic like program enrollment, module selection, fee processing.
  + **APIs (RESTful)**: Provides REST APIs for communication between the **Frontend** and the **Backend**.

 **Data Layer (Databases - Central Database, Department Databases)**

* This layer stores and manages the system's data. It consists of the central database and department-specific databases.
* The **Central Database** contains shared data, such as user accounts, financial records, and general system information.
* The **Department Databases (CS, BM)** store department-specific information, such as student enrollment in specific programs and modules.
* Components:
  + **Central Database**: Holds shared data.
  + **Department Databases (CS, BM)**: Holds department-specific student enrollment, courses, and programs.
  + **Email Service**: Handles email notifications (e.g., payment reminders, enrollment confirmation).
  + **Payment Gateway**: Handles payment transactions.

b) Database Design

A screenshot of a computer screen

AI-generated content may be incorrect.ER Diagram

A diagram of a flowchart

AI-generated content may be incorrect.

* **Data Flow Diagram (DFD) at Level 0**

A diagram of a company

AI-generated content may be incorrect.

**Data stores (cyan), processes (orange), and entities (yellow)** are visually distinct.

Green color in line is to differentiate the