**SOFTWARE REQUIREMENTS SPECIFICATION**

***Student Course Enrollment System***

**Project Manager:**

**Design Lead:**

**Technical Lead:**

**1.0 Introduction**

The Student Course Enrollment System is a web-based application designed to simplify and streamline the course enrollment process for students and administrators in educational institutions. The system provides a centralized platform where students can view and enroll in available classes while administrators can manage class details and monitor student enrollments. The goal is to eliminate manual processes, reduce administrative workload, and provide students with a user-friendly interface to manage their course selections efficiently.

**1.1 Goals and Objectives**

The primary goals and objectives of the Student Course Enrollment System are:

* **Facilitate Enrollment:** To provide students with a streamlined process for viewing, enrolling, and tracking their classes.
* **Simplify Administration:** To enable administrators to create, update, and manage classes with predefined limits and schedules.
* **Ensure Compliance:** To ensure that students do not exceed the maximum limit of active classes, maintaining academic workload balance.
* **Centralized Management:** To provide a centralized platform where both students and administrators can view and manage enrollment statuses and class progress.
* **Improve Efficiency:** To reduce time and effort spent on manual enrollment processes and reduce human errors.

**1.2 Statement of Scope**

The scope of the Student Course Enrollment System includes functionalities for both students and administrators:

* **Student-Side Scope**:
  + Students can log in using university credentials and view a list of available classes.
  + Students can enroll in up to three active classes at a time.
  + Students can view class details, including seat availability, schedule, and description.
* **Admin-Side Scope**:
  + Administrators can log in and create new classes, setting limits for the number of students.
  + Administrators can manage class enrollments, update student progress, and mark classes as complete.

The system will not include functionalities like automated email reminders, bulk enrollment processing, or integrations with third-party systems in the initial release.

**1.3 Software Context**

The software is intended to be used by students and administrators within the institution's network and is not designed for public access. It will be hosted on a secure server, with access restricted to authorized users only. The application will use Azure for deployment and hosting, ensuring availability and scalability as needed.

The software will include the following modules:

* **User Authentication Module**: Handles login and access control for both students and administrators.
* **Student Enrollment Module:** Manages student access to view, enroll in, and track classes.
* **Admin Management Module:** Provides administrative functionalities to create, update, and manage classes and student enrollments.

**1.4 Major Constraints**

The development and implementation of the Student Course Enrollment System are subject to the following constraints:

1. Time Constraints:
   * The project must be completed within 12 weeks, with specific milestones defined for each phase (design, development, testing, deployment).
2. Resource Constraints:
   * The project is limited to a development team of 3 developers
3. Technical Constraints:
   * The system should handle a maximum of 30 concurrent users without performance degradation.
4. Regulatory Constraints:
   * The software must comply with university policies and regulations regarding data protection and user privacy.
   * The system must ensure that student data is encrypted and stored securely, in accordance with data protection laws (e.g., GDPR).
5. **Scope Constraints:**
   * The initial release will only include basic enrollment functionalities, with no advanced features such as email notifications or bulk enrollments.

**2.0 Usage Scenario**

The **Student Course Enrollment System** is designed to address the needs of both students and administrators within an educational institution, providing an efficient, user-friendly platform for managing class enrollments. The system aims to streamline the process of course registration, ensure compliance with academic rules, and reduce the administrative overhead typically associated with managing class rosters.

**Students** will use the system primarily to view available classes, enroll in their desired courses, and monitor their progress throughout the academic term.

**Administrators**, on the other hand, will utilize the platform to create and manage classes, handle student enrollments, and monitor overall class utilization and progress.

**2.1 User Profiles**

The system defines two primary user profiles—**Student** and **Admin**—each with distinct roles and responsibilities:

**2.1.1 Student**

* **Description**: A student is a user who is enrolled at the educational institution and has a unique student ID and university credentials. Students use the system to manage their class enrollments and view details about their selected courses.
* **Responsibilities**:
  + Log in using university credentials.
  + View available classes and their descriptions.
  + Enroll in up to three active classes at a time.
* **Access Rights**:
  + Can view classes, enroll in classes, and monitor personal progress.
  + Cannot create, delete, or modify class information.

**2.1.2 Admin**

* **Description**: An admin is a user responsible for managing class creation and student enrollments.
* **Responsibilities**:
  + Log in using admin credentials.
  + Create new classes, update existing classes, and set class limits.
  + Manage student enrollments and progress.
* **Access Rights**:
  + Full access to all system modules, including class management and enrollment tracking.
  + Can add, update, and delete classes, as well as generate administrative reports.

**2.2 Use Case Scenario:**

|  |  |
| --- | --- |
| **Use Case 1: Student Login** | |
|  |  |
| **Use Case ID** | **UC-001** |
| **Use Case Name** | Student Login |
| **Actor(s)** | Student |
| **Description** | This use case allows the student to log into the system using their university credentials. |
| **Preconditions** | - The student must be registered in the system. |
| - The student must have valid university credentials. |
| **Postconditions** | - The student is successfully logged into the system. |
| - The student is redirected to the home page to view available classes. |
| **Normal Flow** | 1. The student navigates to the login page. |
| 2. The student enters their university email and password. |
| 3. The system verifies the credentials. |
| 4. If valid, the student is redirected to the home page. |
| 5. The login time is recorded. |
| **Alternate Flows** | - If the credentials are invalid, the system displays an error message prompting the student to try again. |
| **Exceptions** | None |
|  |  |
|  |  |
| **Use Case 2: View Available Classes** | |
|  |  |
| **Use Case ID** | **UC-002** |
| **Use Case Name** | View Available Classes |
| **Actor(s)** | Student |
| **Description** | This use case allows students to view a list of available classes, including details such as class name, description, seat availability, and schedule. |
| **Preconditions** | - The student must be logged into the system. |
| **Postconditions** | - The student is able to see a list of available classes and their details. |
| **Normal Flow** | 1. The student navigates to the "Available Classes" page. |
| 2. The system displays a list of all available classes. |
| 3. The student can filter classes based on categories or search by keywords. |
| 4. The student clicks on a class to view more detailed information. |
| **Alternate Flows** | None |
| **Exceptions** | None |
|  |  |
|  |  |
| **Use Case 3: Enroll in a Class** | |
|  |  |
| **Use Case ID** | **UC-003** |
| **Use Case Name** | Enroll in a Class |
| **Actor(s)** | Student |
| **Description** | This use case allows a student to enroll in a class if there are seats available and they do not exceed the limit of three active classes. |
| **Preconditions** | - The student must be logged into the system. |
| - The class must have seats available. |
| - The student must not exceed the maximum limit of three active classes. |
| **Postconditions** | - The student is successfully enrolled in the selected class. |
| - The class seat availability is updated accordingly. |
| **Normal Flow** | 1. The student selects a class from the list of available classes. |
| 2. The system checks if the student is eligible to enroll. |
| 3. If eligible, the student clicks the "Enroll" button. |
| 4. The system updates the enrollment records and reduces the seat availability count. |
| 5. The system confirms the enrollment and displays a success message. |
| **Alternate Flows** | - If the class is full, the system displays a message indicating that there are no seats left. |
| - If the student is already enrolled in three active classes, the system displays a message indicating they cannot enroll in additional classes until one is completed. |
| **Exceptions** | None |
|  |  |
|  |  |
| **Use Case 4: Create a New Class** | |
|  |  |
| **Use Case ID** | **UC-004** |
| **Use Case Name** | Create a New Class |
| **Actor(s)** | Admin |
| **Description** | This use case allows an admin to create a new class with a specified name, description, and enrollment limit. |
| **Preconditions** | - The admin must be logged into the system. |
| **Postconditions** | - A new class is created. |
| - Students can now see it in the list of available classes. |
| **Normal Flow** | 1. The admin navigates to the "Create Class" page. |
| 2. The admin enters the class name, description, and maximum number of students. |
| 3. The admin sets the class schedule (e.g., start and end dates, times). |
| 4. The admin clicks the "Create" button. |
| 5. The system saves the class information and displays a success message. |
| **Alternate Flows** | None |
| **Exceptions** | - If the class already exists, the system displays an error message indicating duplication. |
|  |  |
|  |  |
| **Use Case 5: Manage Class Enrollment** | |
|  |  |
| **Use Case ID** | **UC-005** |
| **Use Case Name** | Manage Class Enrollment |
| **Actor(s)** | Admin |
| **Description** | This use case allows the admin to view and manage student enrollments for a specific class, including marking students’ progress as "complete". |
| **Preconditions** | - The admin must be logged into the system. |
| - The class must already exist in the system. |
| **Postconditions** | - The admin successfully manages student enrollments and progress for the selected class. |
| **Normal Flow** | 1. The admin navigates to the "Manage Enrollment" page for a specific class. |
| 2. The system displays a list of students currently enrolled in the class. |
| 3. The admin can update each student’s progress status. |
| 4. The admin marks a student’s progress as "complete" when they finish the class. |
| 5. The system updates the student’s status and releases the class slot for new enrollments. |
| **Alternate Flows** | None |
| **Exceptions** | None |

**2.3 Special Usage Considerations**

The following special usage considerations apply to the Student Course Enrollment System to ensure smooth operation and optimal user experience:

**2.3.1 Concurrent User Handling**

The system must be able to support concurrent access from 30 students and administrators, particularly during peak periods such as the beginning of a semester when class enrollments are at their highest. Special attention will be given to the optimization of database queries and load balancing to prevent performance bottlenecks.

**2.3.2 User Data Security**

Given that the system will handle sensitive user information, including student IDs, login credentials, and class details, data security is a top priority. All data must be encrypted both in transit .

**2.3.3 Scalability and Future Expansion**

The system should be designed with scalability in mind, allowing for future enhancements such as:

* Adding new user roles (e.g., teachers, supervisors) with limited access.
* Expanding the scope to include features like automated notifications, bulk enrollments, or integration with external learning management systems (LMS).

**2.3.4 Accessibility**

The system must be accessible to users with varying levels of technical proficiency. User interface design should follow best practices for usability, ensuring ease of navigation, readability, and accessibility.

**3.1 Data Description**

The data model of the system includes several core data objects that represent different entities within the course enrollment ecosystem. Each data object has a set of attributes, data types, and constraints that define its structure and purpose within the system.

**3.1.1 Data Objects**

1. **Student**
   * **Description**: Represents a student user in the system.
   * **Attributes**:
     + **StudentID** (Primary Key): Unique identifier for each student.
     + **FirstName**: The student's first name.
     + **LastName**: The student's last name.
     + **Email**: The student's university email address.
     + **Password**: Encrypted password for authentication.
     + **TotalActiveClasses**: The number of classes the student is currently enrolled in (max 3).
2. **Admin**
   * **Description**: Represents an administrative user in the system.
   * **Attributes**:
     + **AdminID** (Primary Key): Unique identifier for each admin.
     + **FirstName**: The admin’s first name.
     + **LastName**: The admin’s last name.
     + **Email**: The admin’s university email address.
     + **Role**: The role of the admin (e.g., Super Admin, Class Admin).
3. **Class**
   * **Description**: Represents a class offered by the institution.
   * **Attributes**:
     + **ClassID** (Primary Key): Unique identifier for each class.
     + **ClassName**: The name of the class.
     + **Description**: Detailed description of the class.
     + **MaxStudents**: Maximum number of students allowed to enroll in the class.
     + **SeatsAvailable**: The number of seats still available in the class.
     + **StartDate**: The date the class begins.
     + **EndDate**: The date the class ends.
     + **Status**: The status of the class (e.g., Open, Full, Completed).
4. **Enrollment**
   * **Description**: Represents the enrollment record for a student in a specific class.
   * **Attributes**:
     + **EnrollmentID** (Primary Key): Unique identifier for each enrollment record.
     + **StudentID** (Foreign Key): References the enrolled student.
     + **ClassID** (Foreign Key): References the class being enrolled in.
     + **EnrollmentDate**: The date the student enrolled in the class.
     + **CompletionStatus**: Status of the student’s progress in the class (e.g., In Progress, Completed).
     + **Grade**: The final grade of the student in the class (if applicable).

**3.1.2 Relationships**

The relationships between the data objects are as follows:

1. **Student – Enrollment Relationship**
   * **Type**: One-to-Many
   * **Description**: A student can have multiple enrollments, but each enrollment record links to a single student. This relationship is represented by the foreign key **StudentID** in the Enrollment table.
2. **Class – Enrollment Relationship**
   * **Type**: One-to-Many
   * **Description**: A class can have multiple students enrolled in it, but each enrollment record links to a single class. This relationship is represented by the foreign key **ClassID** in the Enrollment table.
3. **Admin – Class Relationship**
   * **Type**: One-to-Many
   * **Description**: An admin can create and manage multiple classes, but each class is managed by a single admin. This relationship is defined by the **AdminID** in the Class table.

**3.1.3 Entity Relationship Diagram (ERD) Overview**

The system’s Entity Relationship Diagram (ERD) visually represents the structure and relationships of the data objects in the Student Course Enrollment System:

1. **Student**: Connects to Enrollment through **StudentID.**
2. **Class**: Connects to Enrollment through **ClassID**.
3. **Enrollment**: Acts as a link table between Student and Class.
4. **Admin**: Connects to Class through **AdminID** for administrative oversight (if applicable).

**3.1.4 ER Diagram**:

A diagram of a diagram

Description automatically generated with medium confidence

FIG: 1 – ER Diagram

**4.0 Functional Model and Description**

The Student Course Enrollment System is designed to support key functionalities for both students and administrators, allowing them to efficiently manage and participate in the course enrollment process. Each function is broken down into its individual requirements, describing the specific features and expected behavior of the system.

**4.1 Description of Major Functions**

This section details the primary functionalities of the Student Course Enrollment System, focusing on the requirements for managing classes and enrollments. The system will provide the following major functions:

**4.1.1 Requirement 1:**

**Admin - Create and Manage Classes**

* **Description**: This function allows administrators to create new classes, update class details, and manage existing classes. Admins can specify class attributes such as class name, description, start and end dates, and maximum number of students.
* **Associated Use Cases:**
  + UC-004: Create a New Class
  + UC-005: Manage Class Enrollment
* **Functional Requirements:**
  + The admin should be able to create a new class by providing the class name, description, start date, end date, and maximum number of students.
  + The admin can edit or delete an existing class.
  + The admin should be able to view the list of all created classes and see details such as class capacity, seats filled, and the number of students enrolled.

**4.1.2 Requirement 2:**

**Admin - Manage Student Enrollments**

* **Description**: This function allows administrators to view and manage student enrollments for each class. Admins can monitor student progress, mark students as "complete," and manage enrollment limits.
* **Associated Use Cases:**
  + UC-005: Manage Class Enrollment
* **Functional Requirements:**
  + The admin should be able to view the list of students enrolled in a specific class.
  + The admin should be able to see if a student has exceeded the enrollment limit and take appropriate action if necessary.

**4.1.3 Requirement 3:**

**Student - View and Enroll in Classes**

* **Description**: This function allows students to view a list of available classes, including class details such as schedule, seat availability, and description. Students can enroll in classes and track their progress throughout the academic term.
* **Associated Use Cases**:
  + UC-002: View Available Classes
  + UC-003: Enroll in a Class
* **Functional Requirements:**
  + Students should be able to view the list of all available classes, including their descriptions, schedules, and seat availability.
  + Students should be able to search and filter classes based on categories or keywords.
  + Students can enroll in up to three active classes at any given time.
  + When a student enrolls in a class, the system should update the seat availability count and display a confirmation message.

**4.2 Software Interface Description**

**4.2.1 User Interface**

* **Login Screen:**
  + Allows both students and admins to log in using their credentials.
  + Displays error messages for invalid login.
* **Student Dashboard:**
  + Displays the list of available classes, current enrollments, and class progress.
  + Allows students to search for classes, view class details, and enroll in classes.
* **Admin Dashboard:**
  + Displays a summary of class statistics (e.g., total classes, total students enrolled).
  + Provides options to **create new classes, update existing classes, and manage student enrollments.**

**4.2.2 System Interface**

* **Data Access Layer:**
  + The system connects to database SQL SERVER for storing class and enrollment information.
  + APIs will be developed for data retrieval and modification, providing secure access to data for both frontend and backend components.

**5.0 Restrictions, Limitations, and Constraints**

The **Student Course Enrollment System** is designed to meet the specific needs of an educational institution for managing course enrollments and student participation. However, several restrictions, limitations, and constraints must be considered to ensure the system’s feasibility, maintainability, and compliance with institutional policies and technical requirements.

* 1. **Restrictions**

1. **User Access**:
   * Only authorized users (students and admins) with valid university credentials will have access to the system. No external users or guests will be permitted to access or use the platform.
2. **Data Visibility**:
   * Students can only view classes and enrollments related to their profiles and cannot see or modify information about other students or classes not related to them.
   * Admins have broader visibility but cannot access student personal data such as passwords or other confidential information beyond enrollment and class details.

**5.2 Limitations**

1. **Class Enrollment Limits**:
   * Students can only be enrolled in a maximum of three active classes at any given time. This limitation is imposed to maintain a balanced academic workload and cannot be overridden by the system.
2. **Class Capacity Management**:
   * The system does not support automated waitlisting or reservation mechanisms for classes. If a class is full, students must wait for an open seat or choose another class.
3. **Reporting Capability**:
   * The system does not currently support advanced data visualization or reporting formats (e.g., PDF, graphical charts) in the initial release.

**5.3 Constraints**

1. **Technical Constraints**:
   * The system will be hosted on Azure cloud services. Any changes to the hosting environment or cloud provider will require reconfiguration of deployment pipelines and service integrations.
2. **Compliance and Regulatory Constraints**:
   * The system must comply with university data protection policies and relevant regulations such as GDPR, ensuring that personal data is stored securely and not shared without authorization.
3. **Integration Constraints**:
   * Additional integrations with third-party services or platforms are not within the scope of the initial release and will require separate contracts or agreements.
4. **Security Constraints**:
   * All data exchanges must be secured using HTTPS. Direct database access by external entities or applications is strictly prohibited.
5. **Usability Constraints**:
   * The system is intended for users with basic to intermediate familiarity with web applications. Advanced technical skills are not required, and the user interface must be designed to accommodate users with varying levels of digital proficiency.
6. **Maintenance and Support Constraints**:
   * The initial release will not include ongoing maintenance or support services. Any post-deployment changes, bug fixes, or feature enhancements will require a separate maintenance agreement.