

**Student/Instructor Portal for CMU High School**

**BIS-698: Information Systems Project**

**SUBMITTED BY: GROUP – 5**

**Mechineni, Chanakya**

**Muskula, Sreeja**

**Nelaturi, Ganesh**

**Nyalakonda, Karan Siddharth**

**Manigandla, Rudrasaiprasad**

A purple text on a black background

Description automatically generated

Contents

[**1. Introduction** 3](#_Toc160741941)

[**1.1. Purpose of Document** 3](#_Toc160741942)

[**1.2. Project Summary** 3](#_Toc160741943)

[**1.3. Background** 3](#_Toc160741944)

[**1.4. Project Objectives and Scope** 4](#_Toc160741945)

[**2. Functional Requirements** 4](#_Toc160741946)

[**3. Non-Functional Requirements** 5](#_Toc160741947)

[**4. Context Model:** 6](#_Toc160741948)

[**System Externals** 7](#_Toc160741949)

[**5. The Use Case Model** 9](#_Toc160741950)

[**5.1 System Use Case Diagram** 9](#_Toc160741951)

[**5.2 Use Case Descriptions** 10](#_Toc160741952)

[**6. Acceptance Criteria** 17](#_Toc160741953)

[**7. Constraints** 19](#_Toc160741954)

[**8. Assumptions** 19](#_Toc160741955)

[**9. Risks** 19](#_Toc160741956)

[**10. Testing** 20](#_Toc160741957)

[**11. Conclusion** 21](#_Toc160741958)

Student/Instructor Portal

System Requirements Document (SRD)

# **1. Introduction**

## **1.1. Purpose of Document**

The purpose of this document is to outline the system requirements for the development of the Student/Instructor Portal for CMU High School. This portal aims to provide a platform for student enrolment, course registration, instructor management, attendance tracking, gradebook management, and communication between students and instructors.

This document defines the system requirements and serves as a blueprint for the development, implementation, and testing phases, ensuring that the final product meets the needs of its users effectively and efficiently.

## **1.2. Project Summary**

Online education can occur in either a synchronous or asynchronous setting. In a synchronous setting, interactions between students and teachers happen in real time. This requires participants to be present at a specific time, which might be considered a drawback. Nonetheless, having a scheduled time can aid students in maintaining their Classes/courses effectively by managing their time. Student/Instructor portal is an example of a synchronous platform. Through Student-Instructor portal, student enrolment, course registration, instructor management, attendance tracking, gradebook management, and communication between students and instructors.

The Student Instructor Portal is a comprehensive platform designed to facilitate interaction between students and instructors. It aims to streamline various academic and administrative processes, including course registration, attendance tracking, grade management, and communication.

## **1.3. Background**

The student-Instructor Portal is a strategic response to the growing complexities in managing academic records and enhancing communication within the educational sector. Developed to counter the inefficiencies of traditional administrative methods, it serves as a central digital hub that simplifies and streamlines essential educational processes like enrolment, course registration, and grade management. This system not only facilitates smoother operations but also fosters a more integrated, interactive educational environment. By promoting direct communication and collaboration between students and instructors, it aims to make education more accessible, personalized, and engaging. The portal signifies a shift towards leveraging technology to better support the educational mission, highlighting a commitment to transforming the academic experience for both students and educators by addressing critical administrative and communicative tasks.

The increasing complexity of managing academic records and communication between students and instructors necessitates a robust system that can handle these tasks seamlessly. The portal is intended to address these needs, enhancing the educational experience for all parties involved.

## **1.4. Project Objectives and Scope**

Objectives: To develop a user-friendly, secure, and efficient platform that supports academic management tasks, improves communication, and enhances the overall educational process.

Scope: The project encompasses user authentication, student enrolment, course registration, instructor management, attendance tracking, gradebook management, and a notification system.

# **2. Functional Requirements**

**2.1. User Authentication**

* The portal shall allow users to create their account by sign-in into our portal.
* Secure login/logout processes for students and instructors through unique usernames and passwords.

**2.2. Student Enrolment and Course Registration**

* Interface for students to browse, enroll in the class, and register for courses.
* This portal shows 100% availability of all the courses which are presented in their school.
* Students can enroll into their interested courses by joining the waiting list which will be approved by the instructor.

**2.3. Course Management**

* Instructors can manage their class size by approving the desired number of students.
* Students can register or drop out of a course.
* Instructors can approve or reject the student's registration.

**2.4. Attendance Tracking**

* For each section allotted, Instructors can take the attendance and post it in the portal.
* Students can view/print their attendance of different courses from the portal.

**2.5. Gradebook Management**

* This portal shall allow instructors to post grades of their students according to their performance in the exams which are conducted Face-to-Face.
* Also, students can view their performance assessment through this portal.

**2.6. Notification System**

* This system shall notify students about their approval/rejection for a class.
* This system also notifies students about their grades once they are published.

**2.7 Technology used.**

* For portal development we used Python and Tkinter.
* For database we are using SQLlite3

**2.8 Admin Management**

* This portal should allow the admin to add or delete the courses.
* This portal should allow the admin to add or delete the sections.

# **3. Non-Functional Requirements**

**3.1. Performance**

* This portal doesn’t limit to a few users and shall take any number of users registrations.
* Fast response times and high availability during peak usage periods.

**3.2. Usability**

* Easy user interface design for ease of navigation and operation.
* User interface is easy where a 13–14-year kid can use this portal.
* Since it is customized for CMU high school, only students and instructors of this school have the authorization to use this portal.
* The user interface should be intuitive, with an average learning curve of 15 minutes for new users.

**3.3. Compatibility**

* Supports windows Operating systems only.

**3.4 Version Control Systems**

* The system shall utilize a version control system (e.g., Git) to manage source code, track changes, and facilitate collaboration among development team members.

**3.5 Supportability**

* This portal shall be open to additional changes as per the requirement of the Institution.

# **4. Context Model:**

**Context level DFD:**

A diagram of a student instruction

Description automatically generated

SR-Student registration\*

Figure 1: Context level DFD.

**Level – 0 DFD:**

A diagram of a computer

Description automatically generated

Figure 2: Level-0 DFD

**Level – 1 DFD:**

A diagram of a course

Description automatically generated

Figure 3: Level-1 DFD

**System Externals**

**Instructors:** Instructors play a central role in the project, beginning with user registration to establish their credentials within the portal. They are responsible for managing course requests, accepting, or rejecting student enrollment based on their assigned courses. Instructors record and update student attendance for specific classes, maintaining accurate records of student participation. Posting grades for assessments, assignments, or exams is a critical duty, and instructors manage the gradebook to provide students with transparent insights into their academic achievements. Instructors also utilize the portal to send notifications, ensuring effective communication and timely distribution of updates, announcements, and other pertinent information to the student body. Through these roles, instructors contribute to fostering a dynamic and efficient learning environment within the Student Instructor Portal.

**Students:** Students serve as active participants in the Student Instructor Portal, starting with the crucial step of user registration where they provide essential details for account creation. Students have the autonomy to explore available courses, register for classes, and manage their course enrollments within the portal. They can monitor their attendance, viewing and printing reports for their enrolled classes to track their participation and punctuality. Access to the gradebook allows students to check and print comprehensive reports of their grades for all registered courses, ensuring transparency in academic performance. Additionally, students interact with notifications, staying informed about updates, announcements, and relevant information communicated through the portal.

**ERD diagram**

The Entity-Relationship Diagram (ERD) for our Student/Instructor portal plays a major role in the design and development process, offering a visual representation of the data structure and relationships within the application.

**Entities:**

1. User
2. Course
3. Grade
4. Attendance
5. **Section**
6. **Notification**
7. **Course Registration**

**Relationships:**

1. A student can enroll in multiple sections, and a section can have multiple students enrolled in it.
2. An instructor can teach in multiple sections, but each section can have only one instructor.
3. A student can receive many grades, but each grade is associated with only one student.
4. A user can receive many notifications, but each notification is intended for only one receiving user.
5. A student can submit multiple course registration requests, but each request is linked to only one student.
6. Each section can receive multiple student course registration requests, but each request is assigned to only one section.
7. Each course can have multiple student grades, but each student grade is associated with only one student.
8. A course can have multiple sections, but each section belongs to only one course.
9. Each instructor can record attendance for multiple sessions.
10. Each student will have attendance recorded for multiple instances.

**ERD Representation:**

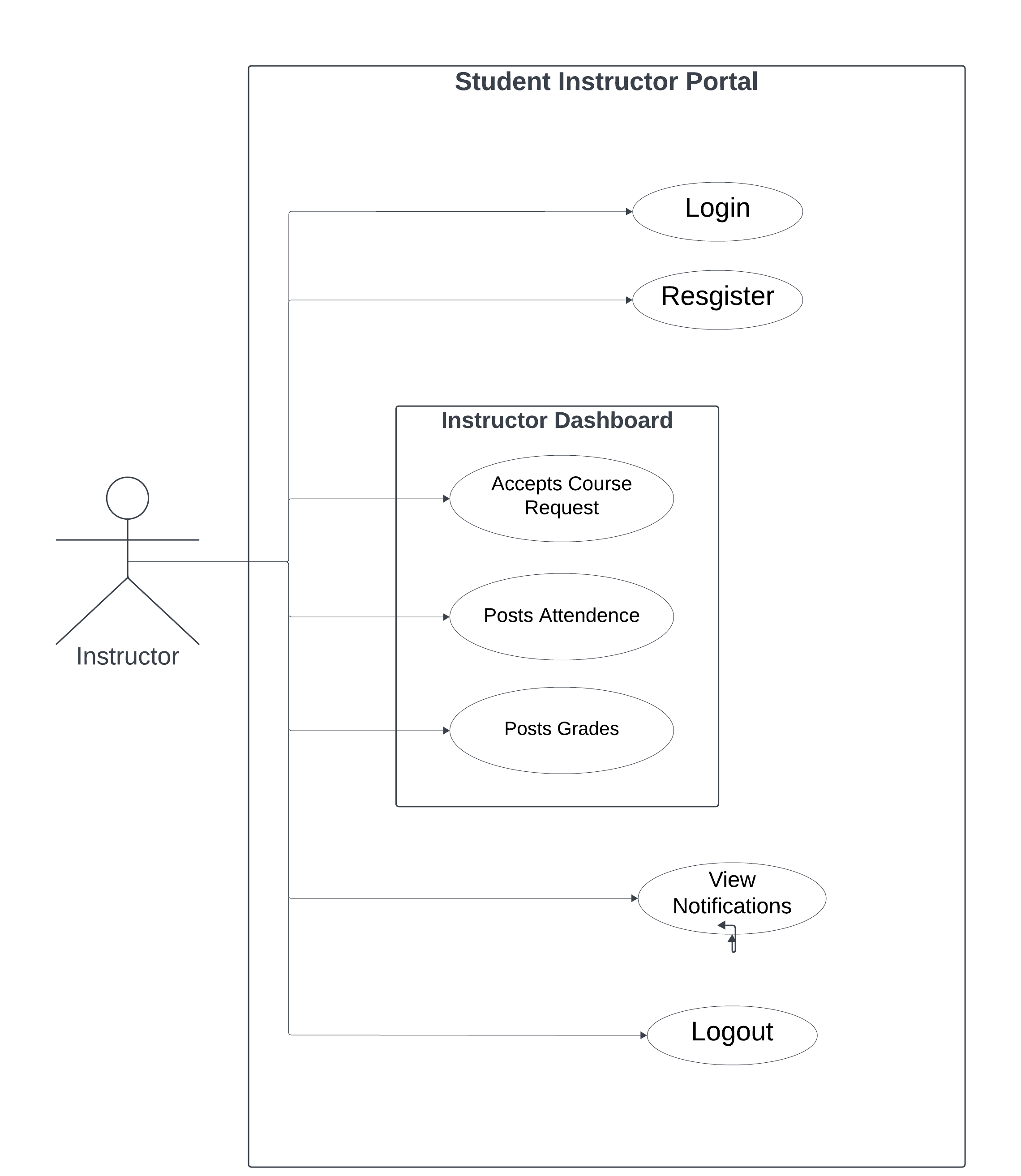
A diagram of a course

Description automatically generated

*Figure 4: Entity Relationship Diagram*

# **5. The Use Case Model**

## **5.1 System Use Case Diagram**





*Figure 5: Instructor Use Case*

A screenshot of a student instructor portal

Description automatically generated

*Figure 6: Admin Use Case*

A screen shot of a computer

Description automatically generated

*Figure 7: Student Use Case*

## **5.2 Use Case Descriptions**

In any given use case, the user can terminate/exit the process at any point requiring user input, thereby concluding the use case, and any information gathered during that specific use case will be discarded.

**Login User**

|  |  |
| --- | --- |
| Use Case Name: | Login User |
| Summary: | The user can login to the app to access the Student or Instructor Dashboard based on their role |
| Basic Flow: | 1. The use case starts when a user wants to login. 2. The system will request the username and password. 3. The user enters their username and password and clicks on login 4. The system will fetch the username and password against all registered user’s data from the database. 5. After successful authentication the system redirects the user to either Student or Instructor Dashboard based on their role |
| Alternative Flows: | Step 3:  If the username or password is invalid, the system will generate a message ‘Invalid username or password. Try again’ and use case goes back to step 2. |
| Extension Points: | none |
| Preconditions: | The user is registered. |
| Postconditions: | The user can now obtain data and perform functions according to his registered access level. |

**Register User**

|  |  |
| --- | --- |
| Use Case Name: | Register User |
| Summary: | This functionality allows new users, both students and instructors, to create accounts within the Student Instructor Portal |
| Basic Flow: | 1. User navigates to the registration page. 2. Enter personal details: First Name, Last Name, Email ID, Role, and Password. 3. The system checks and verifies the entered role. 4. If the role is identified as a student, the user provides additional information about their class. 5. User Clicks on Register button and the system automatically generates a username based on the entered details. 6. Displays a confirmation message and the username upon successful registration. |
| Alternative Flows: | * Step 5: If the password doesn’t meet specified strength criteria it displays a message with criteria to be met. And goes back to step-2 * Step 5: If User details already exists, will display a message ‘User already exists’ and redirects to login page. |
| Extension Points: | None |
| Preconditions: | User is not already registered |
| Postconditions: | The user can now login using username and password. |

**Course request Acceptance by Instructor**

|  |  |
| --- | --- |
| Use Case Name: | Course request Acceptance by Instructor |
| Summary: | This use case allows a registered instructor to accept or reject course request by students |
| Basic Flow: | 1. Instructor Navigates to the "Course Requests" section. 2. Reviews the list of pending course requests. 3. Selects a specific course request. 4. Chooses to accept or reject the course request. 5. If accepted, the system updates the course status and enrolls the student. 6. If rejected, the system notifies the student, and the course request is declined. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The instructor is registered and logged into the portal. * There are pending course requests to be reviewed. |
| Postconditions: | * The course request status is updated (accepted or rejected). * If accepted, the student is enrolled in the course. * If rejected, the student is notified, and the course request is marked as declined. * The course request log is updated. |

**Course registration by Student**

|  |  |
| --- | --- |
| Use Case Name: | Course registration by Student |
| Summary: | This use case allows a registered student to register or drop course by raising a request to instructor. |
| Basic Flow: | 1. Navigates to the "Course Registration" section. 2. View the list of available courses. 3. Selects a specific course for registration or dropping. 4. Raises a request to the instructor for course registration or dropping. 5. If approved, the student can see those below as registered courses. 6. If rejected, the system notifies the student of the decision. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The student is registered and logged into the portal. * Courses are available for registration or dropping. |
| Postconditions: | * If approved, the student's course status is updated (registered or dropped). * If rejected, the student is notified, and the course status remains unchanged. * The instructor reviews and updates the course request log. |

**Attendance (Instructor)**

|  |  |
| --- | --- |
| Use Case Name: | Attendance Posting by instructor |
| Summary: | This use case enables a registered instructor to record attendance for students in a specific section to which the instructor is assigned. |
| Basic Flow: | 1. Instructor Navigates to the "Attendance" section. 2. Selects the specific section or course for which attendance needs to be recorded. 3. View the list of enrolled students for the selected section. 4. Records attendance for each student, marking them as present, absent, or late. 5. The system updates the attendance records for the selected section. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The instructor is registered and logged into the portal. * The instructor is assigned to the specific section or course for which attendance needs to be recorded. |
| Postconditions: | Attendance records for the selected section are updated with the recorded attendance status for each student. |

**Attendance (Student)**

|  |  |
| --- | --- |
| Use Case Name: | View Attendance by Students |
| Summary: | This use case allows a registered student view attendance report of particular class |
| Basic Flow: | 1. Student Navigates to the "Attendance" section. 2. Selects the specific section for which they want to view attendance. 3. Views the attendance report, displaying details like dates, status (present, absent, late) 4. Chooses the option to print the attendance report. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The student is registered and logged into the portal. * The student is enrolled in the specific class for which they want to view attendance. |
| Postconditions: | The student successfully views and prints the attendance report for the selected class. |

**Gradebook (Instructor)**

|  |  |
| --- | --- |
| Use Case Name: | Post Grades by Instructor |
| Summary: | This use case enables a registered instructor to post grades for students in a specific section and course to which the instructor is assigned. |
| Basic Flow: | 1. Instructor Navigates to the "Gradebook" or "My Classes" section. 2. Selects the specific course and section for which grades need to be posted. 3. View the list of enrolled students in the selected course and section. 4. Enters grades for each student based on assessment results. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The instructor is registered and logged into the portal. * The instructor is assigned to the specific section and course for which grades need to be posted. |
| Postconditions: | Grades for the selected course and section are successfully posted in the gradebook. |

**Gradebook (Students)**

|  |  |
| --- | --- |
| Use Case Name: | View Grades by Instructor |
| Summary: | This use case enables a registered students to view and print grades for all the courses to which the student is registered. |
| Basic Flow: | 1. Student Navigates to the "Gradebook" or "My Classes" section. 2. Views a list of all the courses and grades to which the student is registered. 3. Choose the option to print the comprehensive grades report. 4. The system generates a printable version of the grades report for all registered courses. 5. The student prints the report for personal records. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The student is registered and logged into the portal. * The student is enrolled in courses. |
| Postconditions: | The student successfully views and prints a comprehensive grades report for all the courses to which they are registered. |

**Notifications (Instructor/Student)**

|  |  |
| --- | --- |
| Use Case Name: | View notifications |
| Summary: | Instructors or students can view the received notifications. |
| Basic Flow: | 1. Navigates to the "Notifications" section. 2. Views a list of notifications, including messages, updates, or alerts. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The user (Instructor/Student) is registered and logged into the portal. * Notifications are available or have been sent. |
| Postconditions: | The user successfully views and responds to notifications. |

**Add Courses by Admin**

|  |  |
| --- | --- |
| Use Case Name: | Add Courses |
| Summary: | Admin can add a new course by entering details such as course ID, course name, class, and credits. |
| Basic Flow: | 1. Admin navigates to the "Courses" or "Admin Dashboard" section. 2. Selects the option to "Add New Course." 3. Enter the necessary details for the new course, including Course ID, Course Name, Class, and Credits. 4. Submit the information to create the new course. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The admin is registered and logged into the administrative portal. |
| Postconditions: | * The new course is successfully added to the system. * Course details are stored in the database. * The course becomes available for instructors and students to enroll. |

**Delete Courses by Admin**

|  |  |
| --- | --- |
| Use Case Name: | Delete Courses |
| Summary: | Admin can delete an existing course from the system. |
| Basic Flow: | 1. Admin navigates to the "Courses" or "Admin Dashboard" section. 2. Selects the option to "Manage Courses" or "View Courses." 3. Locates the course to be deleted from the list. 4. Selects the option to "Delete" or "Remove" the chosen course. 5. Confirms the deletion when prompted. |
| Alternative Flows: | If the admin tries to delete a course with active enrollments, a warning message is displayed, and the admin must resolve the enrollments before proceeding with the deletion. |
| Extension Points: | None |
| Preconditions: | * The admin is registered and logged into the administrative portal. * There should courses to delete without any active enrollments. |
| Postconditions: | * The selected course is successfully removed from the system. |

**Add Section by Admin**

|  |  |
| --- | --- |
| Use Case Name: | Add Section |
| Summary: | Admin can add a new section within a course by providing necessary details. |
| Basic Flow: | 1. Admin navigates to the "Sections" or "Admin Dashboard" section. 2. Selects the option to "Add New Section" or "Manage Sections." 3. Chooses the course to which the new section will be added. 4. Enters details for the new section, including Section ID, Section Name, Schedule, and Room. 5. Submits the information to create the new section. |
| Alternative Flows: | None |
| Extension Points: | None |
| Preconditions: | * The admin is registered and logged into the administrative portal. * The course to which the section is added already exists. |
| Postconditions: | * The new section is successfully added to the selected course. * Section details are stored in the database. * The section becomes available for instructors to assign and for students to enroll. |

**Delete Section by Admin**

|  |  |
| --- | --- |
| Use Case Name: | Delete Section |
| Summary: | Admin can delete an existing section within a course from the system. |
| Basic Flow: | 1. Admin navigates to the "Sections" or "Admin Dashboard" section. 2. Selects the option to "Manage Sections" or "View Sections" for the respective course. 3. Locates the section to be deleted from the list. 4. Selects the option to "Delete" or "Remove" the chosen section. 5. Confirms the deletion when prompted. |
| Alternative Flows: | * If the admin attempts to delete a section with enrolled students, a warning message is displayed, and the admin must resolve the enrollments before proceeding with the deletion. |
| Extension Points: | None |
| Preconditions: | * The admin is registered and logged into the administrative portal. |
| Postconditions: | * The selected section is successfully removed from the system. |

# **6. Acceptance Criteria**

**Login User**:

* Users should be able to authenticate successfully based on a valid username and password.
* The system should display appropriate error messages for invalid login attempts.
* After successful authentication, users should be redirected to the Student or Instructor Dashboard based on their role.

**Register User:**

* Users should be able to navigate to the registration page.
* The system should verify user roles and prompt students for additional class information if applicable.
* Upon registration, users should see a confirmation message, including the automatically generated username.
* Password strength criteria should be enforced during registration.
* Users should be directed to log in if duplicate user details are detected.

**Course Request Acceptance by Instructor:**

* Instructors should be able to review pending course requests in the "Course Requests" section.
* Instructors should have the option to accept or reject course requests.
* Accepted requests should update course status and enroll students, while rejected requests should notify students.
* The course request log should be updated accordingly.

**Course Registration by Student:**

* Students should be able to view available courses and select specific courses for registration or dropping.
* Students should be able to raise requests to instructors for registration or dropping.
* Instructors should receive notifications and be able to approve or reject requests.
* Approved requests should update the student's course status, and students should receive notifications.
* Rejected requests should trigger notifications to students.

**Attendance Posting by Instructor:**

* Instructors should be able to record attendance for enrolled students in specific sections.
* The system should update attendance records, marking students as present, absent, or late.

**View Attendance by Students:**

* Students should be able to view attendance reports for specific classes.
* The option to print attendance reports should be available for personal records.

**Post Grades by Instructor:**

* Instructors should be able to post grades for students in specific sections and courses.
* The gradebook should be updated with assessment results.

**View and Print Grades by Students:**

* Students should be able to view grades for specific courses.
* The option to print comprehensive grades reports for all registered courses should be available.

**View Notifications:**

* Both instructors and students should be able to view notifications in a dedicated section.
* Notifications should include messages, updates, and alerts.
* Users should be able to acknowledge or take necessary actions based on the content of notifications.

**Manage Courses by Admin:**

* Admin can successfully add a new course by providing Course ID, Course Name, Class, and Credits.
* The new course is stored in the system and becomes available for enrollment.
* Admin can successfully delete an existing course.
* Deletion is confirmed with a prompt.

**Manage Sections by Admin:**

* Admin can successfully add a new section to an existing course.
* The new section becomes available for instructors and students.
* Admin can successfully delete an existing section.
* Deletion is confirmed with a prompt.

**General Success Criteria:**

* The system should be secure, protecting user data and ensuring data integrity.
* The user interface should be intuitive and user-friendly.
* System performance should meet acceptable response time standards.
* The project should be delivered on time and within the allocated budget.
* Stakeholders should express satisfaction with the functionality and usability of the Student Instructor Portal.Top of Form

# **7. Constraints**

* Project timeline and budget limitations.
* Tkinter provides basic theming and styling options, but it may not offer the same level of customization as other modern GUI frameworks. The appearance of widgets is somewhat limited compared to more feature-rich GUI libraries.
* SQLite does not provide built-in user authentication or access control mechanisms.
* SQLite databases are stored as files on disk. Can’t be connected to a cloud database.

# **8. Assumptions**

1. **Software Assumption**: The project assumes the user pc has required software (python, sqlite3), and libraries (tkinter, sqlite3, etc.) are already installed.
2. **Operational Assumption**: The project assumes that users will have basic computer literacy and familiarity with similar software applications.
3. **Technical Assumption**: The project assumes compatibility with the latest version of the operating system and requires a minimum hardware configuration of 4 GB RAM.
4. **Data Assumption:** The project assumes that database files are already present in the system.

# **9. Risks**

* Potential delays in development or integration phases.
* Insufficient communication and collaboration with stakeholders can result in misunderstandings and dissatisfaction with the final product.
* Continuous changes and additions to project requirements can result in scope creep, causing delays and increased project complexity.
* Issues within the development team, such as conflicts, lack of expertise, or turnover, can impact project progress and quality.
* Security vulnerabilities and data privacy concerns.

# **10. Testing**

| **Test Case ID** | **Test Description** | **Expected Result** | **Actual Result** | **Deviation** | **Remarks** |
| --- | --- | --- | --- | --- | --- |
| TC\_LU\_01 | User enters valid username and password and clicks login. | Redirected to the Student or Instructor Dashboard based on role. | Need to test | N/A | N/A |
| TC\_LU\_02 | User enters invalid username or password. | System displays 'Invalid username or password. Try again.' | Need to test | N/A | N/A |
| TC\_RU\_01 | User enters valid details and clicks register. | Confirmation message with the generated username is displayed. | Need to test | N/A | N/A |
| TC\_RU\_02 | User enters a weak password. | System displays a message with password strength criteria. | Need to test | N/A | N/A |
| TC\_RU\_03 | User details already exist in the system. | System displays 'User already exists' and redirects to login. | Need to test | N/A | N/A |
| TC\_CAR\_01 | Instructor accepts a pending course request. | Course status is updated to 'Accepted' and student is enrolled. | Need to test | N/A | N/A |
| TC\_CAR\_02 | Instructor rejects a pending course request. | System notifies the student, and course request is declined. | Need to test | N/A | N/A |
| TC\_CRS\_01 | Student successfully registers for a course. | Course status is updated to 'Registered' for the student. | Need to test | N/A | N/A |
| TC\_CRS\_02 | Student raises a request to drop a course. | If approved, course status is updated to 'Dropped.' | Need to test | N/A | N/A |
| TC\_API\_01 | Instructor records attendance for all students present. | Attendance records for the section are updated accordingly. | Need to test | N/A | N/A |
| TC\_API\_02 | Instructor records attendance for absent and late students. | Attendance records are updated with respective statuses. | Need to test | N/A | N/A |
| TC\_VAS\_01 | Student views attendance report for a specific class. | Details like dates, status (present, absent, late) are displayed. | Need to test | N/A | N/A |
| TC\_PGI\_01 | Instructor enters grades for all students in a course. | Grades are recorded and updated in the gradebook. | Need to test | N/A | N/A |
| TC\_PGI\_02 | Instructor updates grades for a specific assessment. | Updated grades are reflected in the gradebook. | Need to test | N/A | N/A |
| TC\_VGS\_01 | Student views comprehensive grades report for all courses. | All registered courses and respective grades are displayed. | Need to test | N/A | N/A |
| TC\_VNS\_01 | User navigates to the "Notifications" section. | A list of notifications, including messages, updates, or alerts is displayed. | Need to test | N/A | N/A |
| TC\_ACA\_01 | Admin adds a new course with valid details. | New course is successfully added to the system. | Need to test | N/A | N/A |
| TC\_ACA\_02 | Admin attempts to add a course with incomplete details. | System prompts to provide all necessary details. | Need to test | N/A | N/A |
| TC\_DCA\_01 | Admin deletes a course without active enrollments. | Selected course is successfully removed from the system. | Need to test | N/A | N/A |
| TC\_DCA\_02 | Admin attempts to delete a course with active enrollments. | Warning message is displayed, and admin must resolve enrollments before deletion. | Need to test | N/A | N/A |
| TC\_ASA\_01 | Admin adds a new section within a course with valid details. | New section is successfully added to the selected course. | Need to test | N/A | N/A |
| TC\_ASA\_02 | Admin attempts to add a section with incomplete details. | System prompts to provide all necessary details. | Need to test | N/A | N/A |
| TC\_DSA\_01 | Admin deletes an existing section within a course. | Selected section is successfully removed from the system. | Need to test | N/A | N/A |
| TC\_UX\_01 | Check overall usability and responsiveness of the portal interface. | Portal should have smooth navigation. | Need to test | N/A | N/A |

# **11. Conclusion**

In conclusion, the Software Requirements Document (SRD) provides a detailed and essential roadmap for the development team. It outlines user requirements, system functionalities, and constraints, serving as a crucial reference throughout the development lifecycle. The collaborative effort in creating this document ensures a shared vision and clear expectations, laying the foundation for a successful software development journey. Continuous communication and adherence to the SRD will be key in delivering a high-quality and aligned software solution.