



**Department of Computer Science**

**Quaid-e-Azam University, Islamabad**

**Software**

**Requirements**

**Specifications**

**Student Course Registration System**

Prepared by:

**HURR MEHDI (Leader)**

**M. SEHAL BILAL**

**M. ASAD**

**ZOHAIB AHMED**

Submitted to:

**DR. ONAIZA MAQBOOL**

26 March 2025

According to ISO/IEC/IEEE 16326 Guidelines

**Course:** Software Construction

**Submitted by:** Group 8

**Team Lead:** Hurr Mehdi

**Team Members:** M.Sehal Bilal, Muhammad Asad, Zohaib Ahmed

**Submission Date:** 26/03/2025

**Audience:** Development Team, Client (Prof. Onaiza Maqbool) and stakeholders.

**Author:** Hurr Mehdi (Team Lead)

**Co-Authors:** M.Sehal Bilal, Muhammad Asad, Zohaib Ahmed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signature page**  **Team Member 1/Team Leader:** | |  | **Team Member 2** | |
| **By:** |  |  | **By:** |  |
|  | *(Signature)* |  |  | *(Signature)* |
| **Name:** | Hurr Mehdi |  | **Name:** | Muhammad Sehal Bilal |
|  |  |  |  |  |
| **Reg No. :** | 04072312048 |  | **Reg No. :** | 04072312015 |
| **Date:** | 26-03-2025 |  | **Date:** | 26-03-2025 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Team Member 3:** | |  | **Team Member 4** | |
| **By:** |  |  | **By:** |  |
|  | *(Signature)* |  |  | *(Signature)* |
| **Name:** | Muhammad Asad |  | **Name:** | Zohaib Ahmed |
|  |  |  |  |  |
| **Reg No. :** | 04072312023 |  | **Reg No. :** | 04072213001 |
| **Date:** | 26-03-2025 |  | **Date:** | 26-03-2025 |

|  |  |
| --- | --- |
| **Supervisor:** | |
| **By:** |  |
|  | *(Signature)* |
| **Name:** | Dr. Onaiza Maqbool |
|  |  |
| **Position:** | Associate Professor |
| **Date:** |  |

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Changed By | Description |
| **1.0** | 05 March 2025 | Hurr Mehdi | Project Plan |
| **2.0** | 26 March 2025 | Hurr Mehdi | SRS |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Remarks:**

|  |  |  |
| --- | --- | --- |
| **Approved By** |  | Date |

**Preface:**

*This document outlines the project management plan for developing the Student Course Registration System for the CS Office. The system automates course registration, enforces course prerequisites, and enables coordinators to track student progress. The plan adheres to ISO/IEC/IEEE 16326 guidelines.*

**Table of Contents:**

[1. Introduction 7](#_Toc193933041)

[1.1 Project Summary 7](#_Toc193933042)

[1.2 Purpose 8](#_Toc193933044)

[1.3 Scope 8](#_Toc193933049)

[1.4 Definitions, Acronyms, and Abbreviations 8](#_Toc193933053)

[1.5 References 9](#_Toc193933057)

[1.6 Overview 9](#_Toc193933058)

[2. Overall Description 9](#_Toc193933059)

[2.1 Product Perspective 9](#_Toc193933060)

[2.2 Product Functions 9](#_Toc193933061)

[2.3 User Characteristics 10](#_Toc193933062)

[2.4 Constraints 10](#_Toc193933063)

[2.5 Assumptions and Dependencies 10](#_Toc193933064)

[2.6 Project Deliverables 10](#_Toc193933065)

[2.7 Schedule Summary 11](#_Toc193933066)

[3. PROJECT CONTEXT 11](#_Toc193933067)

[3.1 Process Model 11](#_Toc193933068)

[3.2 Methods, Tools, and Techniques 11](#_Toc193933069)

[3.3 Product Acceptance Plan 12](#_Toc193933070)

[3.4 Project Work plans 12](#_Toc193933071)

[3.5 Work Activities 12](#_Toc193933072)

[3.6 Schedule Allocation 13](#_Toc193933073)

[3.7 Resource Allocation 13](#_Toc193933074)

[3.8 Risk Management 14](#_Toc193933075)

[4. Use Case Diagram 15](#_Toc193933076)

[4.1 Actors and Use Cases 16](#_Toc193933077)

[4.2 Specific Requirements 17](#_Toc193933078)

[5. Domain Model 32](#_Toc193933092)

[6. Data Dictionary 33](#_Toc193933093)

1. Introduction
   1. **Project Summary**

|  |  |
| --- | --- |
|  | *The Student Course Registration System is a platform designed to assist academic administration for the CS Department. The system will automate course enrollment processes, enforce prerequisite validations, and provide real-time tracking of student academic progress. The platform will replace manual record keeping with a digital solution. Key functionalities include dynamic course scheduling, prerequisite rule enforcement, and automated reporting for academic coordinators, ensuring alignment with institutional policies and reducing administrative overload.* |

* 1. **Purpose**

**Purpose:**

*This document outlines the requirements for the Student Course Registration System, a web-based platform designed to automate course registration, enforce prerequisites, and track academic progress for the Computer Science department. It adheres to ISO/IEC/IEEE 16326 guidelines and serves as a contract between stakeholders and the development team.*

**Audience:**

* ***Development Team:*** *To implement functionalities per requirements.*
* ***Course Coordinators:*** *To validate academic rule enforcement.*
* ***Students:*** *To understand system workflows.*
* ***University Administration:*** *To ensure compliance with academic policies.* 
  1. **Scope**

**System Name:**

***Student Course Registration System***

**Main Functions:**

* ***Course Registration****: Students register for courses with automated prerequisite checks.*
* ***Academic Progress Tracking:*** *View passed/failed courses and pending prerequisites.*
* ***Study Scheme Management:*** *Upload and manage curriculum structures for student batches.*
* ***Timetable Coordination:*** *Detect and resolve scheduling conflicts.*
* ***Reporting:*** *Generate academic reports for coordinators.*

**Objectives:**

* *Eliminate manual errors in course registration.*
* *Simplify timetable planning for coordinators.*
* *Ensure 100% adherence to prerequisite rules during registration.*
* *Provide coordinators with instant access to student academic histories.*
* *Automate deletion of outdated course entries after semester deadlines.*
* *Simplify course registration for students with intuitive interfaces.*

*Eliminate manual data entry for coordinators through batch uploads and system-generated reports.*

* 1. **Definitions, Acronyms, and Abbreviations**
* ***Prerequisite:*** *A course that must be completed before enrolling in another course.*
* ***Study Scheme:*** *Curriculum structure defining courses and prerequisites for a student batch.*
* ***SRS:*** *Software Requirements Specification.*
* ***Course Coordinator:*** *Manages course offerings, prerequisites, and academic policies.*
* ***Timetable Coordinator:*** *Resolves scheduling conflicts and plans class timetables.*
* ***Passed Courses:*** *Courses successfully completed by a student.*
* ***Skipped Courses:*** *Courses a student has not taken.*
  1. **References**
* ISO/IEC/IEEE 16326:2019
* Case Study 3 – Problem Description
* University's Course Registration Guidelines
* Relevant Software Engineering Textbooks/Papers
  1. **Overview**

*This document provides a comprehensive blueprint for the Student Course Registration System, a web-based platform designed to automate and streamline academic processes for the Computer Science department. This SRS details functional and non-functional requirements, use cases, system functions, and performance criteria. Appendices include a data dictionary, domain model, use case diagram and system sequence diagram with description.*

1. Overall Description
   1. **Product Perspective**

*The system is a standalone web application built using Java/Spring Boot for the backend, MySQL for the database, and HTML/CSS/JS for the frontend. It integrates with university systems for data consistency.*

* 1. **Product Functions**

1. ***User Authentication:*** *Login for students, course coordinators, and timetable coordinators.*
2. ***Course Registration:*** *Enforce prerequisites during enrollment.*
3. ***Academic Dashboard:*** *View passed/failed courses and pending requirements.*
4. ***Study Scheme Upload:*** *Define curriculum structures for batches.*
5. ***Timetable Conflict Detection:*** *Identify and resolve scheduling overlaps.*
6. ***Report Generation:*** *Export academic progress reports (PDF/Excel).* 
   1. **User Characteristics**

* ***Students:*** *Basic computer literacy to register for courses and view progress.*
* ***Course Coordinators:*** *Manage courses, prerequisites, and study schemes.*
* ***Timetable Coordinators:*** *Resolve conflicts and plan schedules.* 
  1. **Constraints**
* *Must use Java for backend development.*
* *Project completion within 16 weeks.*
* *Limited to CS department use (no university-wide integration).* 
  1. **Assumptions and Dependencies**
* *Students and coordinators have internet access.*
* *Prerequisite data is accurate and preloaded.*
* *Mock data is used for testing due to privacy restrictions.*

## Project Deliverables

|  |  |
| --- | --- |
| Deliverable | Description |
| **Project Plan** | *A formal document outlining the project’s scope, timelines, resource allocation, and risk management strategies.* |
| **SRS Documentation** | *A Software Requirements Specification detailing functional/non-functional requirements, use cases, and system constraints.* |
| **System Prototype** | *A mock-up UI design demonstrating core workflows (e.g., course registration, prerequisite checks).* |
| **Functional Application** | *A working system with all core features: prerequisite validation, pass/fail tracking, coordinator dashboards, and report generation.* |
| **Source Code** | *Well-commented Java code, configuration files, and dependencies for reproducibility.* |
| **User Manual** | *A brief guide explaining how students/coordinators can use the system (e.g., registration steps, report generation).* |
| **Test Plan and Reports** | *Documentation of test cases (e.g., prerequisite validation scenarios) and results to verify system accuracy.* |
| **Final Presentation** | *A PowerPoint and live demo summarizing the project’s phases, challenges, and outcomes for academic evaluation.* |

## Schedule Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Start Date | End Date | Deliverables |
| **Planning** | 19/02/2025 | 05/03/2025 | Project Plan |
| **Analysis** | 06/03/2025 | 26/03/2025 | SRS Documentation |
| **Design** | 27/03/2025 | 23/04/2025 | Prototype |
| **Development** | 24/04/2025 | 21/05/2025 | Functional System/User Manual/Source Code |
| **Testing** | 22/05/2025 | 28/05/2025 | Test Reports |
| **Presentation** | 29/05/2025 | 11/06/2025 | Presentation |

1. PROJECT CONTEXT
   1. **Process Model**

*For the Student Course Registration and Status Tracking System, the V-model is adopted to ensure rigorous testing at each stage aligning with ISO/IEC/IEEE 16326 standards. Each development phase is paired with a corresponding testing phase to verify that all requirements from course registration rules to prerequisite enforcement are met.*

*Key Phases:*

* *Requirements Analysis: Define all functional and non-functional requirements.*
* *System Design: Outline a high-level architecture including registration, course management, and reporting modules.*
* *Detailed Design: Develop detailed specifications and prepare unit test cases.*
* *Implementation: Code the system components according to the design.*
* *Unit & Integration Testing: Test individual modules and their interactions.*
* *System & Acceptance Testing: Validate the complete system in a simulated real-world environment.*
* *Deployment & Maintenance: Deploy the system and provide ongoing support.*
  1. **Methods, Tools, and Techniques**

***Frontend****: HTML, CSS, JavaScript*

***IDE****: Eclipse*

***Backend:*** *Java*

***Database:*** *MySQL*

***Framework:*** *Spring Boot / JSF*

***Version Management:*** *Github*

***Project Management:*** *Project Libre, Canva, PowerPoint*

* 1. **Product Acceptance Plan**
* *Acceptance Criteria: All features meet SRS requirements.*
* *User Acceptance Testing (UAT): Performed by students and coordinators*
* *Pre-requisite Checking Validation: Ensures only eligible courses can be registered.*
* *Performance Testing: Evaluates system efficiency under high load* 
  1. **Project Work plans**

*The project work plan will be divided into discrete phases (Planning, Analysis, Design, Development, Testing, and Presentation) with clearly defined tasks and milestones. Each phase will conclude with a review meeting to assess progress and prepare for the next phase.*

* 1. **Work Activities**

***Planning:***

* *Analyzing Case Study*
* *Developing Plan*
* *Create Work Plan*
* *Create Project Document*

***Analysis:***

* *Figure out Use Cases*
* *Develop Analysis Model*
* *Create SRS Model*

***Design:***

* *Database Schema Design (ERD)*
* *UI/UX Mockups (Prototype)*
* *Detailed System Design*

***Development:***

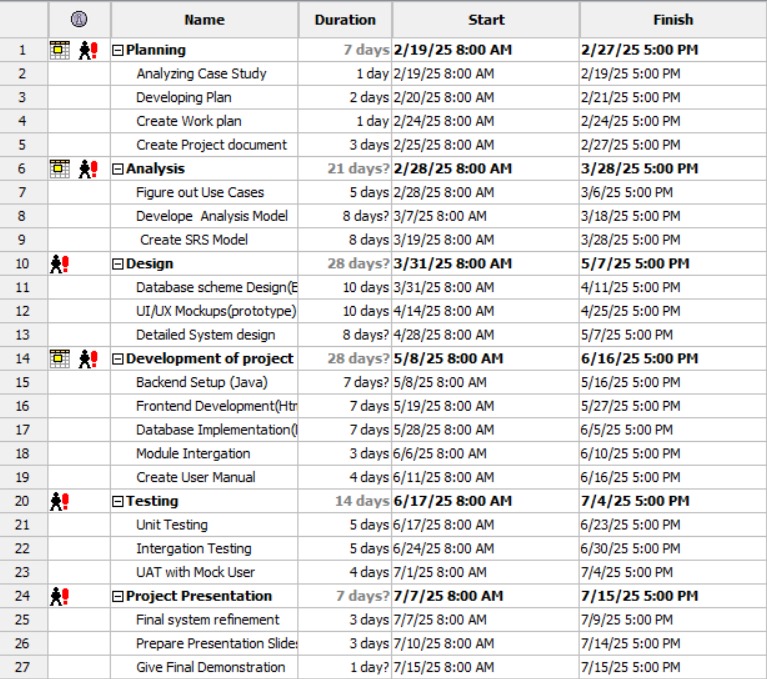
* *Backend Setup (Java)*
* *Frontend Development (HTML/CSS/JS)*
* *Database Implementation (MySQL)*
* *Module Integration*
* *Create User Manual*

***Testing:***

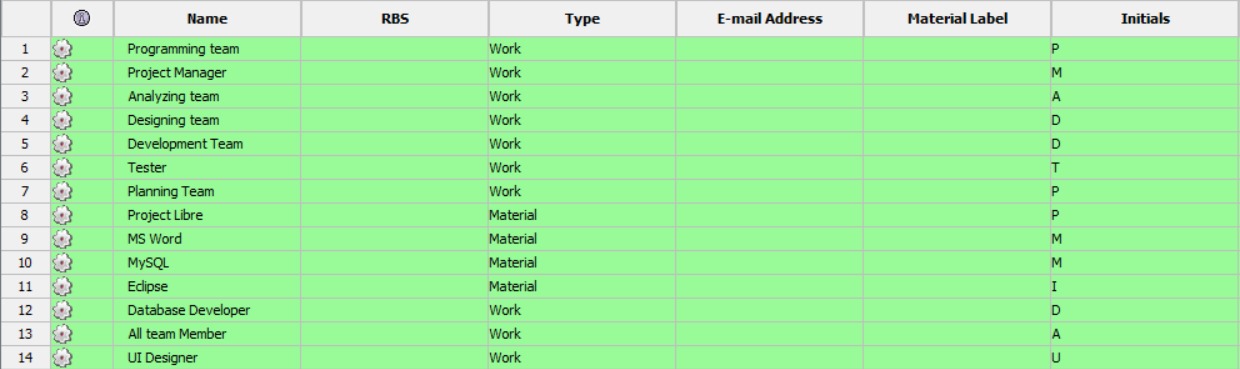
* *Unit Testing*
* *Integration Testing*
* *UAT with Mock Users*

***Presentation:***

* *Final System Refinement*
* *Prepare Presentation Slides*
* *Give Final Demonstration*
  1. **Schedule Allocation**



* 1. **Resource Allocation**



**3.8 Risk Management**

***Technical Risks:***

* *Inadequate Java Proficiency*

*Impact: Delays in backend development and potential code errors.*

*Mitigation: Conduct weekly peer code reviews and allocate time for team members to upskill via online Java tutorials.*

* *Module Integration Failures:*

*Impact: Incompatibility between registration, database, and UI modules.*

*Mitigation: Use modular design principles and test integrations incrementally during development.*

* *Time-Consuming Tool Learning Curve*

*Impact: Delays due to unfamiliarity with tools like Spring Boot, MySQL, or Eclipse.*

*Mitigation: Dedicate the first week of each phase to tool training, use guided tutorials, and leverage pair programming for knowledge sharing.*

***Operational Risks:***

* *Team Member Overload*

*Impact: Delays due to conflicting academic commitments.*

*Mitigation: Assign tasks based on availability and use a shared Gantt chart for progress tracking.*

* *Time Constraints*

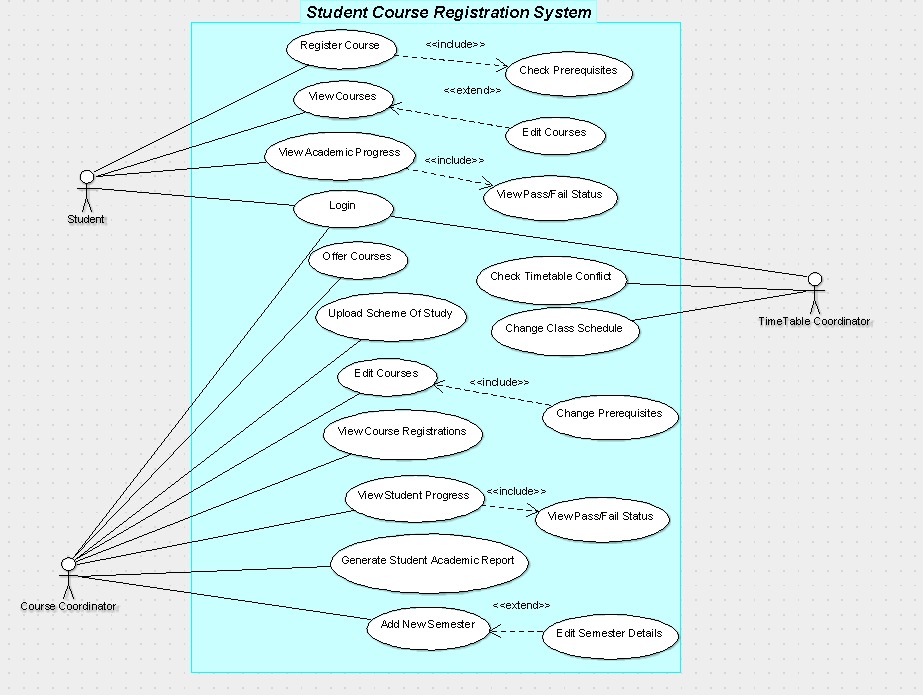
*Impact: Incomplete features by submission deadlines.*

*Mitigation: Prioritize core functionalities (prerequisite checks, progress tracking) over nice-to-haves.*

* *Team Member Unavailability (Sickness/Personal Issues)*

*Impact: Critical tasks stalled due to absenteeism.*

*Mitigation: Cross-train team members on all modules (e.g., frontend, backend), Maintain detailed documentation to enable easy handovers, Build buffer time into the schedule for unexpected absences.*

1. Use Case Diagram

|  |  |
| --- | --- |
|  | *The purpose of this use case diagram is to provide a visual representation of the functional requirements and interactions between users (actors) and the Student Course Registration System, as outlined in Case Study #3. It identifies key actors, their primary use cases, and the relationships between them.This use case diagram was designed using Argo UML, a UML modeling tool, to adhere to ensure clarity in system design. It serves as a foundational blueprint for upcoming development phases.* |

**4.1 Actors and Use Cases**

1. **Student**

* *Login (Authenticates Student access to the system.)*
* *Register Courses (Sign up for the courses being offered)*
* *Includes: Check Prerequisites (automated validation to ensure eligibility) (Mandatory step).*
* *View Courses (View the courses registered by the student)*
* *Extends: Edit Courses (viewing courses lead to editing registrations, e.g., dropping/adding) (Optional).*
* *View Academic Progress (Tracks passed/failed courses and remaining requirements.)*
* *Includes: View Pass/Fail status of courses.*

1. **Time Table Coordinator**

* *Login (Authenticates Time Table Coordinator access to the system).*
* *Check Timetable Conflict (Identifies scheduling overlaps).*
* *View Course Registrations (Monitors enrollments for timetable planning).*
* *Change Class Schedule (Adjusts course timings to resolve conflicts).*

1. **Course Coordinator**

* *Login (Authenticates Course Coordinator access to the system).*
* *Add New Semester (Initializes a new academic period).*
* *Extends: Edit semester details after semester has been added.*
* *Offer Courses (Adds new courses for the semester).*
* *Upload Scheme of Study (Defines course weightage and prerequisites for student batches).*
* *Edit Courses (Change courses being offered)*
* *Includes: Change Prerequisites (modify course dependency rules).*
* *View Registrations (Checks enrollment statistics).*
* *View Student Progress (Monitors individual student performance (passed/failed courses)).*
* *Includes: View Pass/Fail status of courses.*
* *Generate Student Academic Report (Creates CSV/pdf reports of student’s academics)*

**Roles:**

* *Students: focus on enrollment and progress tracking.*
* *Timetable Coordinators: manage scheduling conflicts.*
* *Course Coordinators: handle academic rules, prerequisites, and reporting.* 
  1. **Specific Requirements**

**Use Case 1: Login**

**Primary Actor(s):**

* *Student*
* *Course Coordinator*
* *TimeTable Coordinator*

**Use Case Description**:

*This use case allows users (students, course coordinators, and timetable coordinators) to log into the system to access their respective functionalities.*

**Stakeholders and Interests:**

* *Student: Requires access to academic features.*
* *Course Coordinator: Needs system access for course-related tasks.*
* *TimeTable Coordinator: Manages class schedules and prerequisites.*

**Preconditions:**

* *The user must have valid login credentials.*

**Postconditions:**

* *The user is authenticated and granted access.*

**Inputs:**

* *Username*
* *Password*

**Outputs:**

* *LoginStatus (Success/Failure)*

**Main Success Scenario:**

1. *The user navigates to the login page.*
2. *The user enters a valid username and password.*
3. *The system verifies the credentials.*
4. *The system grants access and redirects the user to their dashboard.*

**Alternative Scenarios:**

1. *Invalid Credentials:*  
   *If the entered credentials are incorrect, the system displays an error message and the user is prompted to re-enter valid credentials.*
2. *System Error:*

*If an unexpected issue occurs, the system notifies the user and suggests retrying later*

**Use Case 2: Register Courses**

**Primary Actor(s):**

* *Student*

**Use Case Description:**

*This use case allows a student to register for courses during the course registration period. The system ensures that the student meets prerequisites before enrolling them in a course.*

**Stakeholders and Interests:**

* *Student: Wants to successfully register for courses without conflicts.*
* *University Administration: Ensures students enroll in the correct courses.*
* *Course Coordinator: Monitors course enrollments and prerequisite compliance.*
* *Timetable Coordinator: Manages scheduling conflicts and class capacities.*

**Preconditions:**

* *The student must be logged into the system.*
* *The course registration period must be active.*

**Postconditions:**

* *The registered course is stored in the database.*
* *The system updates the student's registered courses.*

**Inputs:**

* *StudentID*
* *CourseID*

**Outputs:**

* *RegistrationStatus (Confirmed or Denied)*
* *UpdatedCourseList*

**Main Success Scenario:**

1. *The student navigates to the course registration section.*
2. *The student enters the course ID for the desired course.*
3. *The system checks if the student meets the course prerequisites.*
4. *If prerequisites are met, the system registers the student.*
5. *The system updates the student’s registered courses.*
6. *The system confirms the registration to the student by displaying a ✔ (checkmark) or 'True' status next to the registered course ID.*

**Alternative Scenarios:**

1. *Prerequisites Not Met  
   The system notifies the student that they do not meet the prerequisites.The student is unable to register for the course.*
2. *Technical Failure*

*If the system crashes or encounters an error, the student cannot complete registration.The system logs the failure, and the student is advised to retry later.*

**Use Case 3: View Courses**

**Primary Actor(s):**

* *Student*

**Use Case Description:**

*This use case allows students to view available courses for registration and academic tracking.*

**Stakeholders and Interests:**

* *Student: Requires course information for registration and planning.*
* *Course Coordinator: Ensures course details are correctly displayed.*

**Preconditions:**

* *The student must be logged into the system.*

**Postconditions:**

* *The student can view the list of available courses.*

**Inputs:**

* *StudentID*

**Outputs:**

* *CourseList*

**Main Success Scenario:**

1. *The student navigates to the "View Courses" section.*
2. *The system retrieves available courses for the student.*
3. *The system displays the course list.*

**Alternative Scenarios:**

1. *No Courses Available:*

*If no courses are found, the system informs the student that no courses are currently available.*

1. *System Error:*

*If a technical issue occurs, the system notifies the student to try again later.*

**Use Case 4: View Academic Progress**

**Primary Actor(s):**

* *Student*

**Use Case Description:**

*This use case allows a student to view their academic progress, including courses they have passed, failed, and pending requirements.*

**Stakeholders and Interests:**

* *Student: Wants to track their academic performance and plan future registrations.*
* *University Administration: Ensures accurate record-keeping and compliance with academic policies.*
* *Course Coordinator: Uses progress data to advise students and update curriculum requirements.*

**Preconditions:**

* *The student must be logged into the system.*
* *The student’s academic records (course registrations, grades) must exist in the system.*

**Postconditions:**

* *The student’s academic progress is displayed.*
* *No changes are made to the database (read-only action).*

**Inputs:**

* *StudentID*

**Outputs:**

* *AcademicProgressDetails (passed courses, failed courses, pending prerequisites, GPA).*

**Main Success Scenario:**

1. *The student navigates to the Academic Progress section.*
2. *The system retrieves the student’s academic records (registrations, grades, prerequisites).*
3. *The system generates a summary of:*

* *Passed Courses: Courses marked as "Pass."*
* *Failed Courses: Courses marked as "Fail."*
* *Pending Requirements: Prerequisites not yet completed.*

1. *The system displays the academic progress summary to the student in a tabular or visual format.*
2. *The student reviews the information and exits the section.*

**Alternative Scenarios:**

1. *No Academic Records Found:*

*The system displays a message: “No academic records available. Please register for courses first.”*

1. *Technical Failure:*

*If the system fails to retrieve data, an error message is displayed: “Unable to load academic progress. Please try again later.”*

**Use Case 5: Add New Semester**

**Primary Actor(s):**

* *Course Coordinator*

**Use Case Description:**

*This use case allows the Course Coordinator to initialize a new academic period by defining its start and end dates and then offering courses for that semester.*

**Stakeholders and Interests:**

* *Course Coordinator: Responsible for setting up each academic semester.*
* *Students & Faculty: Need an active semester for course registration and academic activities.*

**Preconditions:**

* *The user must be logged in as a Course Coordinator.*
* *No active semester should exist for the same time period.*

**Postconditions:**

* *A new semester is successfully created and available for course registration.*

**Inputs:**

* *Semester ID*
* *Start Date*
* *End Date*

**Outputs:**

* *Confirmation of Semester Creation*
* *Error Message (if semester overlaps with an existing one)*

**Main Success Scenario:**

1. *The Course Coordinator navigates to the "Add Semester" page.*
2. *They enter semester details (Semester ID, Start Date, End Date).*
3. *The system checks if the new semester conflicts with an existing one.*
4. *If valid, the system creates the new semester.*
5. *The system confirms the addition of the semester.*

**Alternative Scenarios:**

1. *Overlapping Semester:*

*If the new semester overlaps with an existing one, the system prevents creation and displays an error message.*

1. *Missing Required Information:*

*If any field is left blank, the system prompts the user to complete all fields.*

**Use Case 6: Offer Course**

**Primary Actor(s):**

* *Course Coordinator*

**Use Case Description:**

*This use case allows the Course Coordinator to offer new courses for an academic semester. The system ensures that course details are validated and stored in the database.*

**Stakeholders and Interests:**

* *Course Coordinator: Wants to offer new courses for students.*
* *Students: Need courses to be available for registration.*
* *University Administration: Ensures proper academic offerings.*

**Preconditions:**

* *The Course Coordinator must be logged into the system.*
* *The new course details (name, code, credits, prerequisites) must be available.*

**Postconditions:**

* *• The new course is added.*
* *• The course becomes available for student registration.*

**Inputs:**

* *Course Name*
* *Course Code*
* *Credit Hours*
* *Prerequisites (if any)*

**Outputs:**

*Course Offering Status (Success or Failure)*

**Main Success Scenario:**

1. *The Course Coordinator navigates to the "Offer Course" section.*
2. *The Course Coordinator enters the course details.*
3. *The system validates the details (e.g., unique course code, credit hour limits).*
4. *If valid, the system stores the course.*
5. *The system confirms the course has been successfully added.*

**Alternative Scenarios:**

1. *Validation Failure:*

*If course details are incorrect, the system prompts for corrections.*

1. *Technical Failure:*

*If the system crashes, the course cannot be added, and the Coordinator is advised to try later.*

**Use Case 7: Upload Scheme of Study**

**Primary Actor(s):**

* *Course Coordinator*

**Use Case Description:**

*This use case allows the Course Coordinator to upload a Scheme of Study, defining course weightage and prerequisites for different student batches.*

**Stakeholders and Interests:**

* *Course Coordinator: Ensures course structure is correctly defined.*
* *Students: Depend on an accurate scheme of study for academic planning.*
* *University Administration: Requires updated curriculum information.*

**Preconditions:**

* *The Course Coordinator must be logged into the system.*

**Postconditions:**

* *The Scheme of Study is recorded in the system.*
* *Students can view updated course prerequisites and weightage.*

**Inputs:**

* *SchemeOfStudyFile*

**Outputs:**

* *UploadStatus (Successful / Failed)*

**Main Success Scenario:**

1. *The Course Coordinator navigates to the "Upload Scheme of Study" section.*
2. *The Course Coordinator selects and uploads the Scheme of Study file.*
3. *The system validates the file format and content.*
4. *If the file is valid, the system records it.*
5. *The system updates course prerequisites and weightage.*
6. *The system confirms the successful upload.*

**Alternative Scenarios:**

1. *File Validation Failed:*

*The uploaded file format is invalid.The system informs the Course Coordinator to upload a valid Scheme of Study.*

1. *Technical Failure:*

*A system error occurs while processing the upload.The system informs the Course Coordinator to try again later.*

**Use Case 8: Edit Courses**

**Primary Actor(s):**

* *Course Coordinator*

**Use Case Description:**

*This use case allows the Course Coordinator to edit the details of a course, such as course name, prerequisites, or other relevant information. The system ensures that the changes are validated and updated in the database.*

**Stakeholders and Interests:**

* *Course Coordinator: Wants to update course details accurately and efficiently.*
* *University Administration: Ensures that course information is up-to-date and accurate.*
* *Students: Rely on accurate course information for registration and academic planning.*
* *Timetable Coordinator: Needs updated course details to manage scheduling conflicts.*

**Preconditions:**

* *The Course Coordinator must be logged into the system.*
* *The course to be edited must exist in the system.*

**Postconditions:**

* *The course details are updated in the database.*
* *The system reflects the updated course information for all users.*

**Inputs:**

* *CourseID*
* *Updated course details (e.g., course name, prerequisites, etc.)*

**Outputs:**

* *EditStatus (Confirmed or Denied)*
* *UpdatedCourseDetails.*

**Main Success Scenario:**

1. *The Course Coordinator navigates to the course management section.*
2. *The Course Coordinator selects the course to be edited by entering the CourseID.*
3. *The system retrieves the current course details.*
4. *The Course Coordinator updates the course details (e.g., course name, prerequisites, etc.).*
5. *The system validates the updated course details.*
6. *If the validation is successful, the system updates the course details in the database.*

**Alternative Scenarios:**

* 1. *Validation Failure:*

*If the updated course details fail validation (e.g., invalid prerequisites or missing information), the system notifies the Course Coordinator. The Course Coordinator is prompted to correct the details and resubmit.*

* 1. *Technical Failure:*

*If the system crashes or encounters an error during the update process, the Course Coordinator cannot complete the edit. The system logs the failure, and the Course Coordinator is advised to retry later.*

**Use Case 9: View Course Registrations**

**Primary Actor(s):**

* *Course Coordinator*
* *Timetable Coordinator*

**Use Case Description:**

*This use case allows the course and timetable coordinator to view student course registrations to assist in class scheduling and avoid conflicts. The system retrieves and displays enrollment data for different courses.*

**Stakeholders and Interests:**

* *Timetable Coordinator: Needs registration data to plan class schedules efficiently.*
* *Students: Benefit from properly managed class schedules with no conflicts.*
* *Course Coordinator: May need to adjust course offerings based on registration trends.*

**Preconditions:**

* *The coordinator must be logged into the system.*
* *Course registration data must be available.*

**Postconditions:**

* *The system displays up-to-date student enrollment data for different courses.*

**Inputs:**

* *Semester*
* *Course Name*

**Outputs:**

* *List of students registered for the course.*

**Main Success Scenario:**

1. *The coordinator navigates to the "View Course Registrations" section.*
2. *The coordinator enters search criteria (e.g., semester or specific courses).*
3. *The system retrieves and displays a list of students registered in each course.*
4. *The timetable coordinator reviews the data for scheduling adjustments.*

**Alternative Scenarios:**

1. *No Registrations Found*

*The system notifies the coordinator that no registrations exist for the selected criteria.*

1. *Data Retrieval Error*

*The system displays an error message, and the coordinator is advised to try again later.*

**Use Case 10: View Student Progress**

**Primary Actor(s):**

* *Course Coordinator*

**Use Case Description:**

*This use case allows the Course Coordinator to monitor individual student performance, including passed and failed courses, pending prerequisites, and GPA.*

**Stakeholders and Interests:**

* *Course Coordinator: Monitors student progress to provide academic guidance.*
* *Student: Receives academic feedback and course planning support.*
* *University Administration: Ensures accurate record-keeping and compliance with academic policies.*

**Preconditions:**

* *The Course Coordinator must be logged into the system.*
* *The student must be enrolled in at least one semester.*

**Postconditions:**

* *The student's academic progress is displayed.*
* *No modifications are made to student records (view-only action).*

**Inputs:**

* *StudentID*

**Outputs:**

* *AcademicProgressDetails (Passed Courses, Failed Courses, Pending Prerequisites, GPA)*

**Main Success Scenario:**

1. *The Course Coordinator navigates to the "View Student Progress" section.*
2. *The System retrieves the student’s academic records (registrations, grades, prerequisites).*
3. *The System generates a summary of:*

* *Passed Courses*
* *Failed Courses*
* *Pending Prerequisites*
* *GPA*

1. *The System displays the academic progress summary to the Course Coordinator.*
2. *The Course Coordinator reviews the information for advising purposes.*

**Alternative Scenarios:**

1. *No Academic Records Found:*

*If no academic records are available for the student, the system informs the Course Coordinator that no academic records are available for this student.*

1. *Technical Failure:*

*If a system error occurs while retrieving academic records, the system informs the Course Coordinator that it is unable to load academic progress and to try again later*

**Use Case 11: Generate Student Academic Report**

**Primary Actor(s):**

* *Course Coordinator*

**Use Case Description:**

*This use case allows the Course Coordinator to generate a Student Academic Report for a specific student and semester.*

**Stakeholders and Interests:**

* *Course Coordinator: Needs student reports for academic monitoring.*
* *University Administration: Uses reports for record-keeping and analysis.*
* *Students: May request reports for personal academic tracking.*

**Preconditions:**

* *The Course Coordinator must be logged into the system.*

**Postconditions:**

* *The system generates a Student Academic Report.*
* *The report is available for viewing.*

**Inputs:**

* *StudentID*
* *Semester*

**Outputs:**

* *GeneratedStudentReport*

**Main Success Scenario:**

1. *The Course Coordinator navigates to the "Generate Student Report" section.*
2. *The Course Coordinator enters the StudentID and Semester for which the report is required.*
3. *The system retrieves the academic data of the specified student for the given semester.*
4. *The system generates the Student Academic Report.*
5. *The system confirms report generation and provides a viewing option.*

**Alternative Scenarios:**

1. *No Student Data Available:*

*No student data is available for the selected student and semester.The system informs the Course Coordinator that no records exist for the specified student and semester.*

1. *Technical Failure:*

*A system error occurs while generating the report. The system informs the Course Coordinator to try again later due to the error.*

**Use Case 12: Check Timetable Conflicts**

**Primary Actor(s):**

* *Timetable Coordinator*

**Use Case Description:**

*This use case allows the Timetable Coordinator to check for conflicts in the timetable, such as overlapping courses, room double-bookings, and instructor scheduling conflicts.*

**Stakeholders and Interests:**

* *Timetable Coordinator: Ensures smooth scheduling without conflicts.*
* *Students: Avoids overlapping class schedules.*
* *Faculty: Ensures they do not have overlapping teaching assignments.*
* *University Administration: Maintains efficient resource allocation.*

**Preconditions:**

* *A timetable must exist in the system.*
* *Course and instructor details must be entered into the system.*

**Postconditions:**

* *Identified conflicts are flagged for resolution.*
* *The timetable is validated for conflicts before finalization.*

**Inputs:**

* *Semester timetable*
* *Instructor schedules*
* *Room availability*

**Outputs:**

* *List of conflicts (if any)*
* *Suggestions for resolving conflicts*

**Main Success Scenario:**

1. *The Timetable Coordinator navigates to the "Check Timetable Conflicts" section.*
2. *The system scans the timetable for: • Overlapping courses in the same room. • Instructors assigned to multiple courses at the same time. • Students enrolled in overlapping courses.*
3. *The system generates a report listing detected conflicts.*
4. *The Coordinator makes adjustments to resolve conflicts.*
5. *The system confirms a conflict-free timetable.*

**Alternative Scenarios:**

1. *Multiple Conflicts Found:*

*The system displays multiple conflict warnings. The Coordinator prioritizes and resolves conflicts one by one.*

1. *Technical Error:*

*If the system fails to process the timetable check, the Coordinator retries after system maintenance*

**Use Case 13: Change Class Schedule**

**Primary Actor(s):**

* *Timetable Coordinator*

**Use Case Description:**

*This use case allows the Timetable Coordinator to modify the class schedule, such as changing time slots or instructors. The system ensures updates do not conflict with existing schedules.*

**Stakeholders and Interests:**

* *Timetable Coordinator: Responsible for scheduling courses efficiently.*
* *Students: Need a clear schedule to plan their studies.*
* *Faculty: Require confirmed schedules to conduct classes.*
* *University Administration: Ensures proper resource allocation for lectures.*

**Preconditions:**

* *The Timetable Coordinator must be logged into the system.*
* *Course offerings for the semester must be finalized.*

**Postconditions:**

* *A new timetable is added to the system.*
* *The timetable is accessible to students and faculty.*

**Inputs:**

* *Course ID*
* *New Schedule Details (Time Slot, Instructor, Classroom)*

**Outputs:**

* *Schedule Update Status (Success or Conflict)*

**Main Success Scenario:**

1. *The Timetable Coordinator navigates to the "Change Class Schedule" section.*
2. *The Coordinator selects a course and enters the new schedule details.*
3. *The system checks for conflicts (e.g., overlapping classes, instructor availability).*
4. *If no conflicts, the system updates the schedule in the database.*
5. *The system confirms the schedule has been successfully changed.*

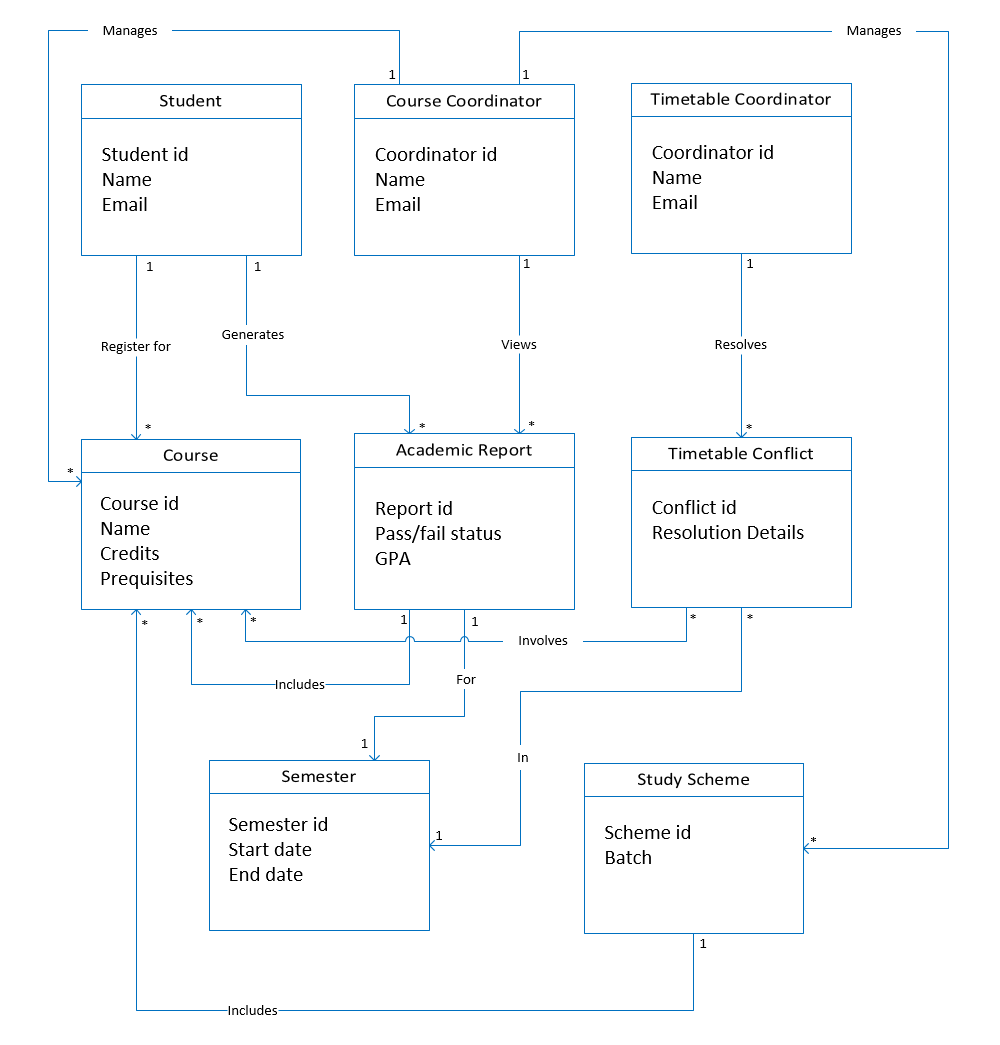
**Alternative Scenarios:**

1. *Room Unavailability:*

*The system notifies the Coordinator that a room is unavailable and suggests alternatives. The Coordinator selects a different room or time slot.*

1. *Instructor Scheduling Conflict:*

*The system detects a scheduling conflict for an instructor. The Coordinator revises the schedule to resolve the conflict.*

1. **Domain Model**

1. **Data Dictionary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Field Name*** | ***Description*** | ***Type*** | ***Length*** | ***Conditions/Constraints*** |
| **Student\_ID** | Unique identifier for students | String | 10 | Must be unique for each student |
| **Student\_Name** | Full name of the student | String | 50 | Must contain only alphabets |
| **Email** | Student email address | String | 50 | Must be a valid email format |
| **Coordinator\_ID** | Unique identifier for a course or timetable coordinator | Integer | 10 | Must be unique. |
| **Coordinator\_Name** | Full name of the coordinator | String | 50 | Required. |
| **Coordinator\_Email** | Email address of the coordinator | String | 50 | Must be a valid email format. |
| **Course\_ID** | Unique identifier for a course | String | 10 | Must be unique for each course |
| **Course\_Name** | Name of the course | String | 20 | Cannot be null |
| **Course\_Credits** | Number of credit hours for the course | Integer | 2 | Must be between 1 and 6. |
| **Prerequisites** | List of prerequisite courses. | List<String> | - | Valid course IDs only. |
| **Semester\_ID** | Unique identifier for a semester. | String | 6 | Format: YYMM (e.g., 2501 for Jan 2025). |
| **Start\_Date** | Semester Start date | Date | - | MM/DD/YYYY format. |
| **End\_Date** | Semester End date | Date | - | MM/DD/YYYY format. |
| **Report\_ID** | Unique identifier for an academic report | Integer | 10 | Must be unique. |
| **Pass\_Fail\_Status** | Pass or fail status of the student’s performance | String | 10 | Values: “Pass” or “Fail.” |
| **GPA** | Grade Point Average | Decimal | 4,2 | Must be within 0.00–4.00 (or your institutional range). |
| **Conflict\_ID** | Unique identifier for a timetable conflict | Integer | 10 | Must be unique. |
| **Resolution\_Details** | Details on how a timetable conflict is resolved | String | 255 | Optional/May be empty if not resolved. |
| **Student\_ID (FK)** | Student receiving the grade | String | 10 | Must exist in Student table |
| **Scheme\_ID** | Unique identifier for a study scheme | Integer | 10 | Must be unique. |
| **Batch** | Represents the batch/year group for the study scheme | String | 10 | Example: “Fall2025,” “2025-2029.” |

**Data Structures**

1. **Student**

*Student = Student\_ID + Student\_Name + Email*

1. **Course Coordinator**

*Coordinator\_ID + Coordinator\_Name + Coordinator\_Email*

1. **Timetable Coordinator**

*Coordinator\_ID + Coordinator\_Name + Coordinator\_Email*

1. **Course**

*Course\_ID + Course\_Name + Course\_Credits + Prerequisites*

1. **Academic Report**

*Report\_ID + Pass\_Fail\_Status + GPA*

1. **Semester**

*Semester\_ID + Start\_Date + End\_Date*

1. **Timetable Conflict**

*Conflict\_ID + Resolution\_Details*

1. **Study Scheme**

*Scheme\_ID + Batch*