SOFTWARE ENGINEERING CONCEPTS - LAB MANUAL

TEAM 5
COUSELLING SYSTEM

<<SHINY A>>	<<220701267>>
<<SHRI RAM SETHU S>>	<<220701273>>
<<SIVAKALASULINGAM A>>	<<220701279>>
<<SREENIDHI K>>	<<220701285>>
<<SUKESH RAJ V>>	<<220701291>>
<<SWETHA R>>	<<220701297>>

<<BE Computer Science and Engineering>>

<<2023 - 2024>>   <<E SECTION>>

Software Concepts & Engineering - Lab Manual

Problem Statement:

The primary problem a Student Information Management System (SIMS) aims to resolve is the efficient and organized management of student data. Specifically, the system addresses:

Data Fragmentation: Student information is often spread across various platforms, making it difficult to access and manage efficiently.

Manual Errors: Manual handling of student records can lead to errors in data entry, which can affect attendance, grades, and other critical student information.
Accessibility: Students and staff may find it difficult to access their information quickly and efficiently.
Communication Gaps: There can be a lack of effective communication between students, counsellors, and admin staff.
Administrative Overload: Handling large volumes of student data manually is time-consuming and labor-intensive for the administration.

Data Perspective:

Data Collection: The SIMS collects data from various sources, including attendance records, grades, and activities. This data is standardized and stored in a central database.
Data Management: Counsellors can update and maintain student records efficiently. This includes adding attendance data, updating grades, and logging other activities.
Data Retrieval: Students can access their data to view attendance reports, grades, and other records in real-time.
Data Accuracy: Automated data handling reduces errors associated with manual data entry, ensuring that the data is accurate and up-to-date.
Data Security: The system ensures that sensitive student information is secure and accessible only to authorized users.

How it would help users when we implement the system-

For students:

Easy Access to Information: Students can log into the system to view their attendance, grades, and other activity records at any time, providing transparency and enabling them to stay informed about their academic performance.
Improved Communication: Students can easily communicate with their counsellors regarding their academic performance and other concerns through the system.
Personalized Insights: Students receive timely updates and insights about their performance, helping them to take necessary actions for improvement.

For counsellors:

Efficient Data Management: Counsellors can manage student records more efficiently, reducing the time spent on administrative tasks.

Accurate Reporting: With accurate and up-to-date data, counsellors can generate reports and track student performance more effectively.
Enhanced Communication:Counsellors can communicate with students more effectively through the system, providing timely feedback and support.

For admins:

Centralized Data: Admin staff have access to a centralized repository of student data, making it easier to manage and retrieve information.
Reduced Workload: Automation of routine tasks, such as attendance tracking and grade recording, reduces the workload on admin staff.
Improved Decision Making: Access to comprehensive and accurate data helps the administration in making informed decisions related to academic policies and student welfare.

By incorporating testing throughout the project lifecycle, from initial development to ongoing maintenance and support, the inventory management system can consistently deliver value to users, optimize operational efficiency, and adapt to evolving business requirements.

Business Need:

Implementing a Counselling System at Rajalakshmi Engineering addresses several critical business needs. The primary objective is to enhance the efficiency, accuracy, and accessibility of managing student data. This includes streamlining processes for attendance tracking, grade management, and communication between students, counsellors, and administrative staff.

Current Process:

Manual Process:

1. Data Collection:
    - Attendance is manually recorded by teachers using paper logs or spreadsheets.
    - Grades are calculated and recorded manually by each faculty member, often resulting in delays and errors.
    - Student activities and achievements are logged separately, usually on paper or in disparate digital documents.

2. Data Management:
    - Counsellors manually update and maintain student records, which are stored in physical files or various unconnected digital formats.
    - Data consolidation is labor-intensive, requiring significant effort to ensure all records are accurate and up-to-date.

3. Data Retrieval:
    - Students must request their attendance and grade records from counsellors or administrative staff, leading to delays.
    - Information retrieval is time-consuming and often inefficient, involving multiple steps and manual verification.

4. Communication:
    - Communication between students, counsellors, and administrative staff is conducted via email, phone, or in-person meetings.
    - This process is slow, leading to potential miscommunications and delays in addressing student concerns.
    -

Different Personas and Their Use of the System:

Students:
Current Process: Students face delays in accessing their attendance records, grades, and other academic information. They must request data from counsellors or administrative offices and wait for manual processing.
With Counselling System: Students can log into a unified portal to view real-time updates on their attendance, grades, and activities. The system provides instant notifications and allows for seamless communication with counsellors.

Counsellors:
Current Process: Counsellors spend considerable time manually updating student records, handling paper documents, and consolidating data from various sources. Responding to student queries involves checking multiple records, which is time-consuming and prone to errors.
With Counselling System: Counsellors use a centralized system to manage student records, update attendance and grades, and log activities efficiently. The system generates reports automatically and facilitates direct communication with students.

Administrative Staff:
Current Process: Administrative staff are responsible for collecting and managing student data from various departments. This involves manual data entry, consolidation, and reporting, which is labor-intensive and error-prone.
With Counselling System: Administrative staff benefit from an integrated system that centralizes all student data. This reduces the workload associated with data collection and reporting. They can easily generate accurate reports and ensure data security and compliance.

Business problems

Inefficiency: The current manual processes are time-consuming and labor-intensive, leading to inefficiencies in managing student data. This affects the overall productivity of counsellors and administrative staff.

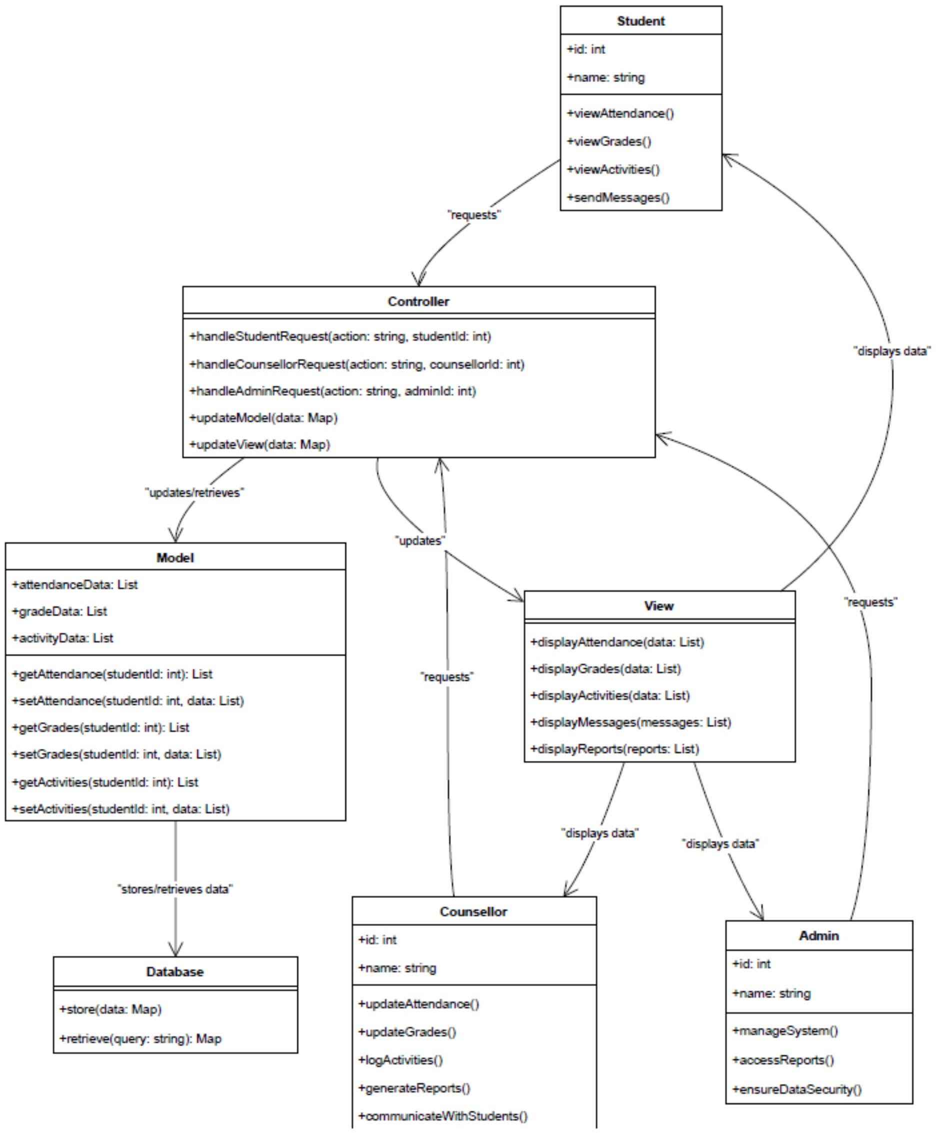
Data Inaccuracy: Manual data entry and management are prone to errors, leading to inaccurate records. This can impact student grades, attendance tracking, and overall academic performance.

Delayed Access to Information: Students, counsellors, and administrative staff experience delays in accessing and updating information. This affects timely decision-making and student support.

Poor Communication: The reliance on traditional communication methods (email, phone, in-person) leads to delays and potential miscommunications, hindering effective student-counsellor interactions.

Administrative Burden: The manual process imposes a heavy administrative burden on staff, requiring significant time and effort to manage student records, prepare reports, and ensure data accuracy.

MVC DIAGRAM


Student
+id: int
+name: string
+viewAttendance()
+viewGrades()
+viewActivities()
+sendMessages()
"requests"
Controller
+handleStudentRequest(action: string, studentId: int)
+handleCounsellorRequest(action: string, counsellorId: int)
+handleAdminRequest(action: string, adminId: int)
+updateModel(data: Map)
+updateView(data: Map)
"displays data"
"updates/retrieves"
"updates"
"requests"
Model
+attendanceData: List
+gradeData: List
+activityData: List
+getAttendance(studentId: int): List
+setAttendance(studentId: int, data: List)
+getGrades(studentId: int): List
+setGrades(studentId: int, data: List)
+getActivities(studentId: int): List
+setActivities(studentId: int, data: List)
View
+displayAttendance(data: List)
+displayGrades(data: List)
+displayActivities(data: List)
+displayMessages(messages: List)
+displayReports(reports: List)
"requests"
"displays data"
"displays data"
"stores/retrieves data"
Database
+store(data: Map)
+retrieve(query: string): Map
Counsellor
+id: int
+name: string
+updateAttendance()
+updateGrades()
+logActivities()
+generateReports()
+communicateWithStudents()
Admin
+id: int
+name: string
+manageSystem()
+accessReports()
+ensureDataSecurity()

User Stories for a Software Testing and Management System

1. Add Student
As an admin, I want to add new students to the system so that their details are recorded.
Estimate: 5 points

2. Edit Student Details
As an admin, I want to edit student details so that I can keep the information up-to-date.
Estimate: 3 points

3. View Student List
As an admin, I want to view a list of all students so that I can manage them easily.
Estimate: 2 points

4. Delete Student
As an admin, I want to delete student records so that I can remove outdated or incorrect information.
Estimate:3 points

5. Search Student
As an admin, I want to search for a student by name or ID so that I can quickly find their details.
Estimate: 2 points

6. Add Course
As an admin, I want to add new courses to the system so that they can be assigned to students.
Estimate: 5 points

7. Assign Course to Student
As an admin, I want to assign courses to students so that their academic plans are recorded.
Estimate: 3 points

8. View Student Grades
As a student, I want to view my grades so that I can track my academic performance.

Estimate: 2 points

9. Update Grades
As an instructor, I want to update student grades so that their academic performance is recorded accurately.
Estimate: 3 points

10.Generate Reports
As an admin, I want to generate reports on student performance so that I can analyze academic progress.
Estimate: 8 points

Poker planning methodology

Each user story should be estimated using the Fibonacci sequence (1, 2, 3, 5, 8, 13, etc.).

- Gather the team for a Poker planning session.
- Discuss each user story and estimate effort required collaboratively.
- Assign story points based on team consensus

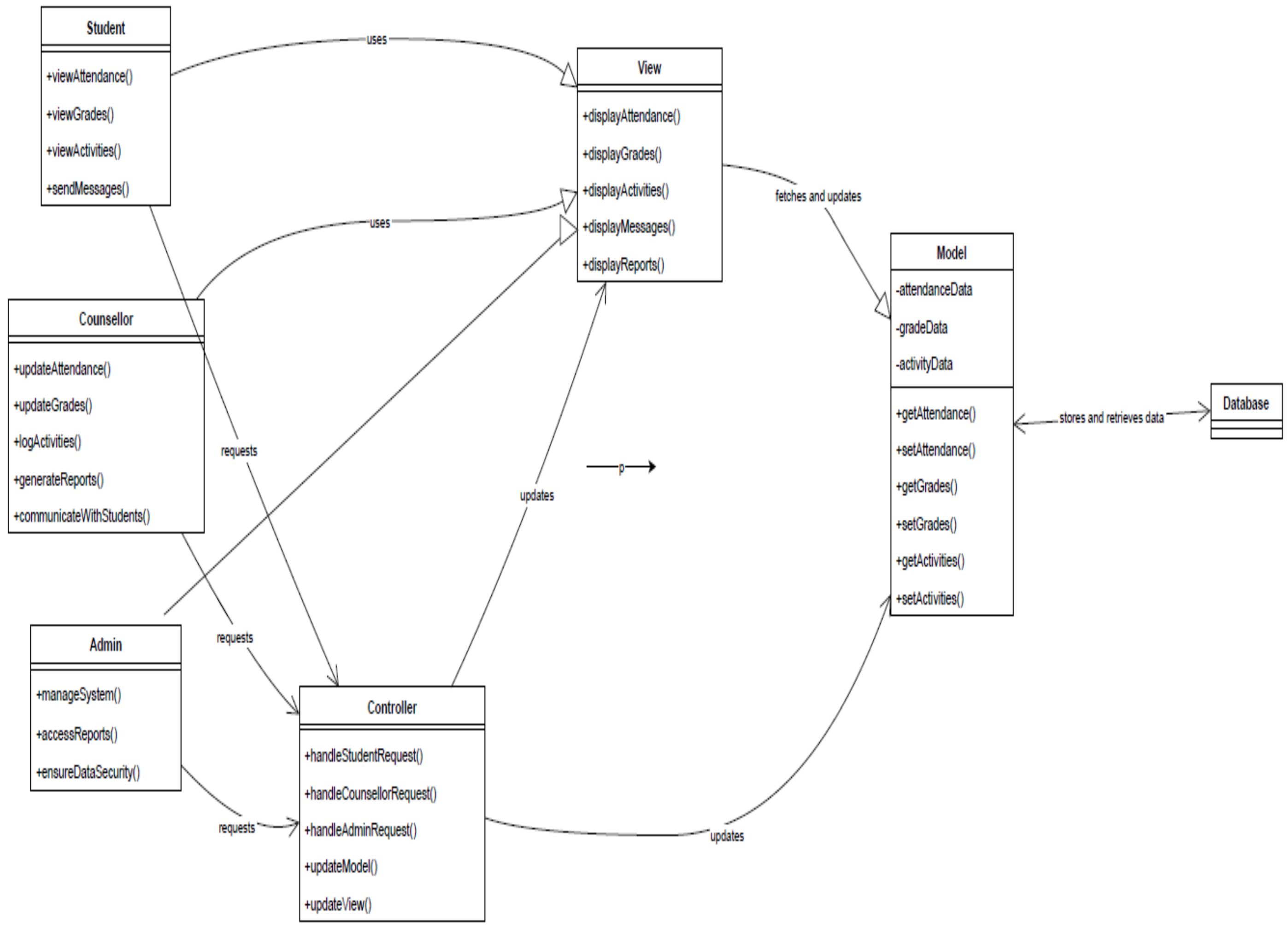
Non-Functional Requirements (NFRs)

Performance
- The system should be able to handle up to 1,000 concurrent users without performance degradation.

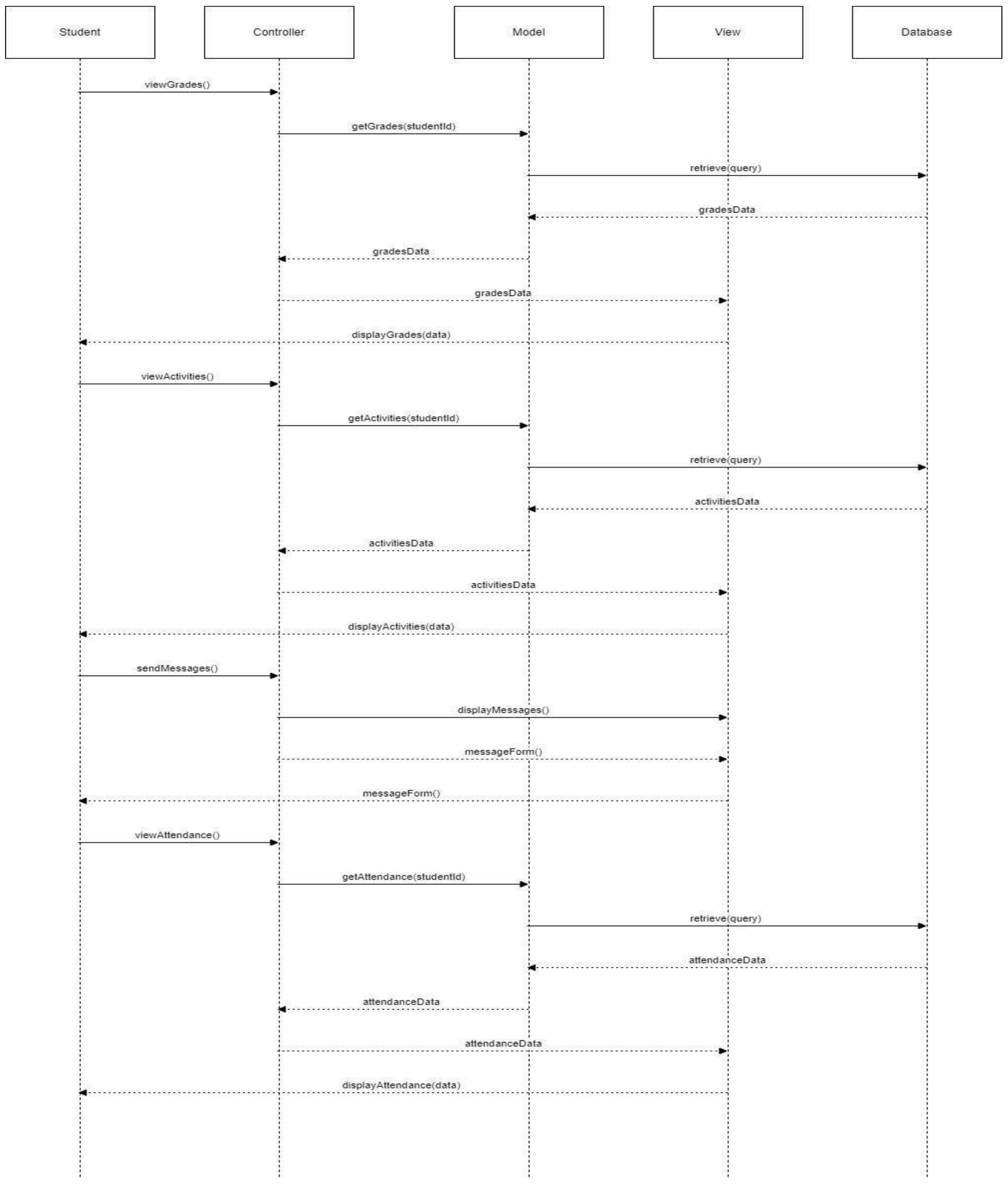
Security
- All user data must be encrypted at rest and in transit to protect student privacy.

Usability
- The user interface should be intuitive and accessible, with a maximum of 3 clicks to reach any primary function.

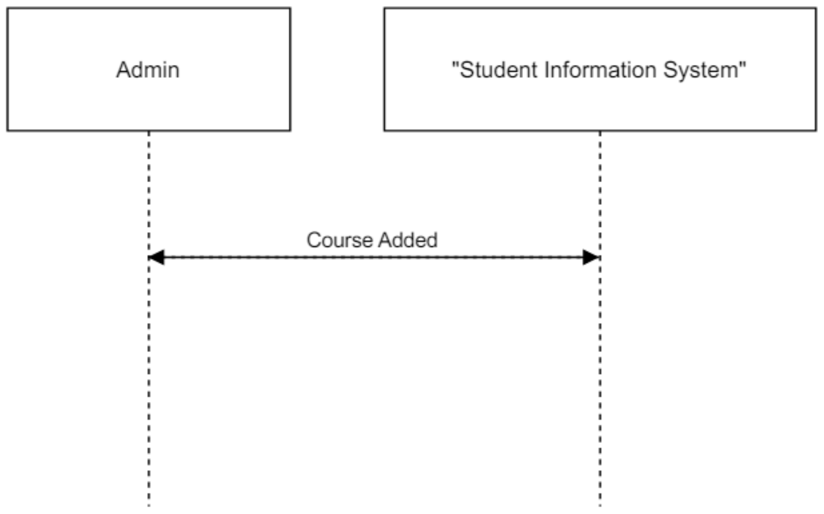
CLASS DIAGRAM



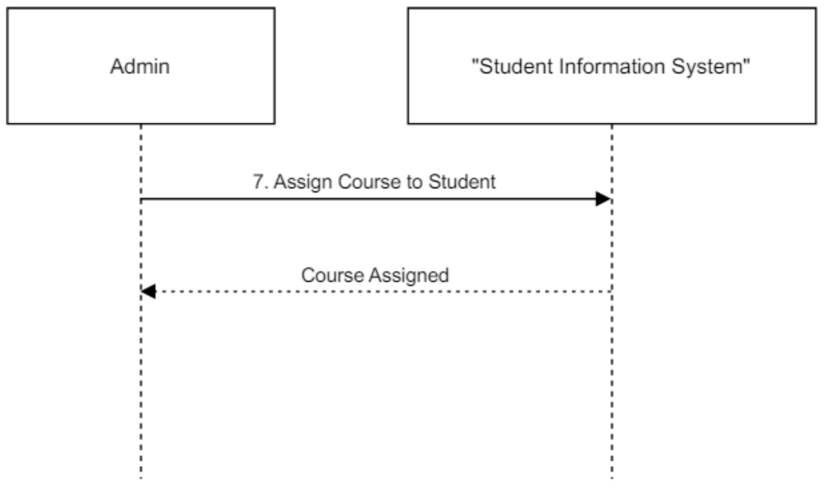
SEQUENCE DIAGRAMS



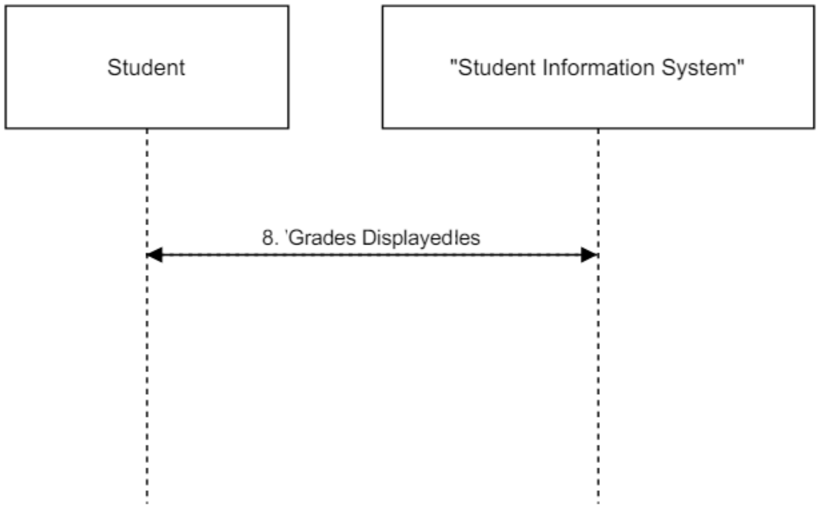
Sequence Diagram for "Add Course" User Story



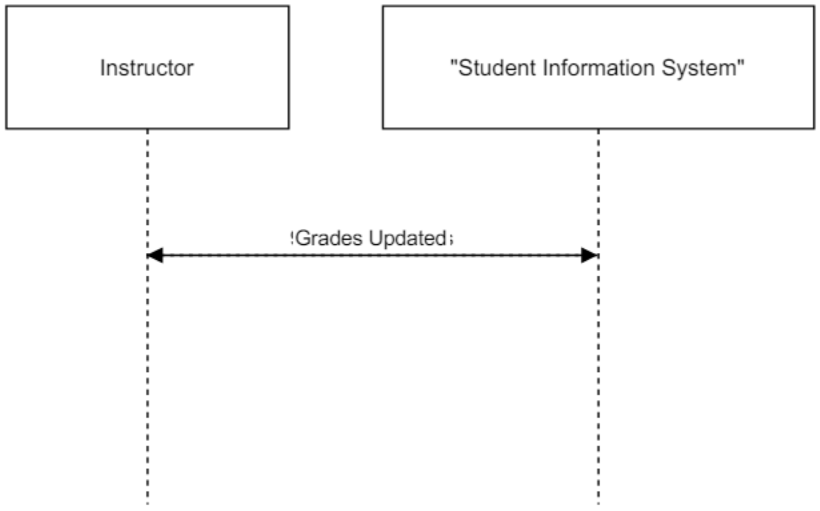
Sequence Diagram for "Assign Course to Student" User Story:



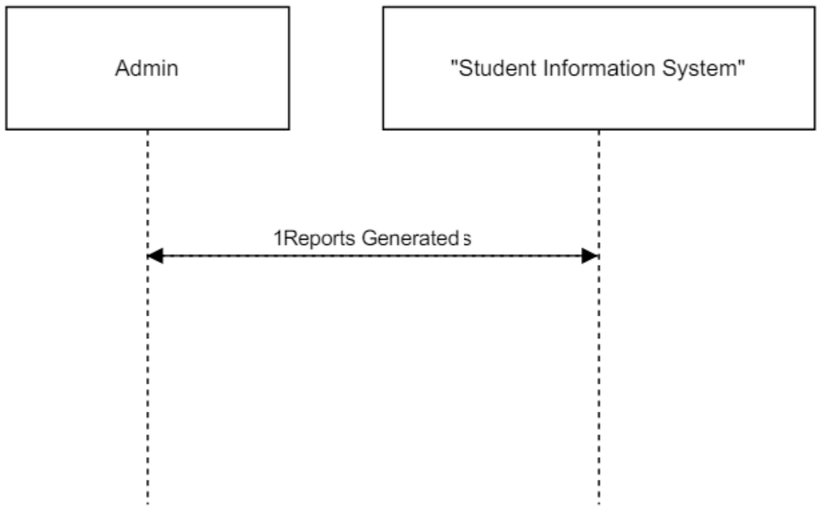
Sequence Diagram for "View Student Grades" User Story:



Sequence Diagram for "Update Grades" User Story:



Sequence Diagram for "Generate Reports" User Story:



Test Strategy

Test Plans

Objective: To ensure the software testing and management system functions as expected, is user-friendly, secure, and scalable.

Scope:

- Functional Testing
- Non-Functional Testing (Performance, Security, Usability)

- Integration Testing
- System Testing
- Regression Testing

Test Environments:

- Development Environment
- Staging Environment
- Production Environment

Tools:

- **Test Management**: Jira
- **Automation**: Selenium, JUnit
- **Performance Testing**: JMeter
- **Security Testing**: OWASP ZAP
- **CI/CD**: Azure DevOps
- **Version Control**: GitHub

Test Cases

User Story 1: Create a New Test Project

Happy Path:

- **Test Case ID**: TC001
- **Description**: Verify that a new test project can be created successfully.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the "Create Project" page.
    2. Enter valid project details.
    3. Click "Submit".
- **Expected Result**: Project is created and visible in the project list.

Error Scenario:

- **Test Case ID**: TC002

- **Description**: Verify that an error message is displayed when creating a project with missing required fields.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the "Create Project" page.
    2. Leave required fields empty.
    3. Click "Submit".
- **Expected Result**: Error message "Required fields cannot be empty" is displayed.

User Story 2: Create and Assign Test Cases

Happy Path:

- **Test Case ID**: TC003
- **Description**: Verify that a test case can be created and assigned to a project.
- **Preconditions**: User is logged in, and a project exists.
- **Steps**:
    1. Navigate to the "Create Test Case" page.
    2. Enter valid test case details.
    3. Select a project to assign.
    4. Click "Submit".
- **Expected Result**: Test case is created and assigned to the selected project.

Error Scenario:

- **Test Case ID**: TC004
- **Description**: Verify that an error message is displayed when creating a test case with invalid data.
- **Preconditions**: User is logged in, and a project exists.
- **Steps**:
    1. Navigate to the "Create Test Case" page.
    2. Enter invalid data (e.g., extremely long name).
    3. Click "Submit".

- **Expected Result**: Error message "Invalid data entered" is displayed. **User**

  Story 3: Execute Test Cases and Record Results

Happy Path:

- **Test Case ID**: TC005
- **Description**: Verify that a test case can be executed and the result recorded.
- **Preconditions**: User is logged in, and a test case exists.
- **Steps**:
    1. Navigate to the test case.
    2. Click "Execute".
    3. Enter results.
    4. Click "Submit".
- **Expected Result**: Test case result is recorded and updated.

Error Scenario:

- **Test Case ID**: TC006
- **Description**: Verify that an error message is displayed when submitting results with missing fields.
- **Preconditions**: User is logged in, and a test case exists.
- **Steps**:
    1. Navigate to the test case.
    2. Click "Execute".
    3. Leave result fields empty.
    4. Click "Submit".
- **Expected Result**: Error message "Results cannot be empty" is displayed.

  User Story 4: Generate Detailed Test Reports

Happy Path:

- **Test Case ID**: TC007
- **Description**: Verify that a detailed test report can be generated.

- **Preconditions**: User is logged in, and test cases have been executed.
- **Steps**:
    1. Navigate to the "Reports" page.
    2. Select a project.
    3. Click "Generate Report".
- **Expected Result**: Detailed report is generated and displayed.

Error Scenario:

- **Test Case ID**: TC008
- **Description**: Verify that an error message is displayed when generating a report for a project with no test cases.
- **Preconditions**: User is logged in.
- **Steps**:
    1. Navigate to the "Reports" page.
    2. Select a project with no test cases.
    3. Click "Generate Report".
- **Expected Result**: Error message "No test cases found for this project" is displayed.

User Story 5: Receive Notifications on Test Case Failures

Happy Path:

- **Test Case ID**: TC009
- **Description**: Verify that a notification is sent when a test case fails.
- **Preconditions**: User is logged in, and a test case exists.
- **Steps**:
    1. Execute a test case and mark it as failed.
- **Expected Result**: Notification is sent to the assigned users.

Error Scenario:

- **Test Case ID**: TC010
- **Description**: Verify that no notification is sent when a test case passes.
- **Preconditions**: User is logged in, and a test case exists.

- **Steps**:
    1. Execute a test case and mark it as passed. ● **Expected Result**: No notification is sent.

Test Strategy

Test Plans

This test plan outlines the testing strategy and approach for the Student Information Management System (SIMS) to ensure it meets the specified requirements and functions correctly. The SIMS is designed for students, counsellors, and administrative staff at Rajalakshmi Engineering.

Objectives

- Verify that all features and functionalities of SIMS work as expected.
- Ensure data accuracy, security, and integrity.
- Validate user interfaces and user experience.
- Identify and fix any defects or issues.

Scope

- **Modules to be Tested**:
    - Student Module
    - Counsellor Module
    - Admin Module
    - Data Management Module
    - Communication Module (Messaging and Notifications)

Test Items

- User authentication and authorization
- Attendance management
- Grade management
- Activity logging

- Real-time data access
- Messaging system
- Report generation
- Data security and integrity

Testing Approach

1. **Unit Testing**:
    - Test individual components for functionality.
    - Ensure each module performs as intended in isolation.
2. **Integration Testing**:
    - Test interactions between integrated components.
    - Validate data flow and communication between modules.
3. **System Testing**:
    - Test the complete system for end-to-end functionality.
    - Verify all features work together seamlessly.
4. **User Acceptance Testing (UAT)**:
    - Conduct testing with real users (students, counsellors, admin staff).
    - Ensure the system meets user requirements and expectations.
5. **Performance Testing**:
    - Test system performance under load.
    - Validate response times and system stability.
6. **Security Testing**:
    - Test for vulnerabilities and ensure data protection measures are effective.

    - Validate access controls and data encryption.

Test Environment

- **Hardware**: Standard computers and mobile devices.
- **Software**: Latest web browsers, operating systems.
- **Database**: Test database with sample data.
- **Network**: Stable internet connection.

Test Schedule

- **Unit Testing**: Week 1-2
- **Integration Testing**: Week 3-4
- **System Testing**: Week 5-6
- **UAT**: Week 7-8
- **Performance and Security Testing**: Week 9

Test Cases for User Stories
User Story 1: Student Login

Description: As a student, I want to log in to the system so that I can access my academic information.

Test Case 1.1: Happy Path

- **Test Case ID**: TC001
- **Description**: Verify student can log in with valid credentials.
- **Preconditions**: Student account exists with valid credentials.
- **Steps**:
    1. Navigate to the login page.
    2. Enter valid username and password.
    3. Click on the "Login" button.
- **Expected Result**: Student is successfully logged in and redirected to the dashboard.

Test Case 1.2: Error Scenario

- **Test Case ID**: TC002
- **Description**: Verify error message for invalid credentials.

- **Preconditions**: Student account exists.
- **Steps**:
  1. Navigate to the login page.
  2. Enter invalid username and/or password.
  3. Click on the "Login" button.
- **Expected Result**: Error message "Invalid username or password" is displayed.

User Story 2: View Attendance

Description: As a student, I want to view my attendance records so that I can keep track of my attendance.

Test Case 2.1: Happy Path

- **Test Case ID**: TC003
- **Description**: Verify student can view attendance records.
- **Preconditions**: Student is logged in.
- **Steps**:
  1. Navigate to the "Attendance" section.
  2. View the attendance records.
- **Expected Result**: Attendance records are displayed correctly.

Test Case 2.2: Error Scenario

- **Test Case ID**: TC004
- **Description**: Verify message when there are no attendance records.
- **Preconditions**: Student is logged in. No attendance records exist.
- **Steps**:
  1. Navigate to the "Attendance" section.
- **Expected Result**: Message "No attendance records found" is displayed.

User Story 3: Counsellor Update Grades

Description: As a counsellor, I want to update student grades so that I can maintain accurate academic records.

Test Case 3.1: Happy Path

- **Test Case ID**: TC005
- **Description**: Verify counsellor can update student grades.
- **Preconditions**: Counsellor is logged in.
- **Steps**:
  1. Navigate to the "Grades" section.
  2. Select a student.
  3. Enter the grades.
  4. Click on "Save".
- **Expected Result**: Grades are successfully updated.

Test Case 3.2: Error Scenario

- **Test Case ID**: TC006
- **Description**: Verify error message for invalid grade input.
- **Preconditions**: Counsellor is logged in.
- **Steps**:
  1. Navigate to the "Grades" section.
  2. Select a student.
  3. Enter an invalid grade (e.g., a letter instead of a number).
  4. Click on "Save".
- **Expected Result**: Error message "Invalid grade input" is displayed.

User Story 4: Admin Manage System

Description: As an admin, I want to manage system settings so that the system operates smoothly.

Test Case 4.1: Happy Path

- **Test Case ID**: TC007
- **Description**: Verify admin can access system settings.
- **Preconditions**: Admin is logged in.

- **Steps**:
    1. Navigate to the "System Settings" section.
    2. View and modify settings.
    3. Click on "Save".
- **Expected Result**: System settings are successfully updated.

Test Case 4.2: Error Scenario

- **Test Case ID**: TC008
- **Description**: Verify error message when required settings are missing.
- **Preconditions**: Admin is logged in.
- **Steps**:
    1. Navigate to the "System Settings" section.
    2. Leave required settings blank.
    3. Click on "Save".
- **Expected Result**: Error message "Required fields cannot be empty" is displayed.

User Story 5: Messaging System

Description: As a student, I want to send messages to my counsellor so that I can ask for advice.

Test Case 5.1: Happy Path

- **Test Case ID**: TC009
- **Description**: Verify student can send a message to the counsellor.
- **Preconditions**: Student is logged in.
- **Steps**:
    1. Navigate to the "Messages" section.
    2. Select counsellor.
    3. Enter message text.
    4. Click on "Send".
- **Expected Result**: Message is successfully sent.

Test Case 5.2: Error Scenario

- **Test Case ID**: TC010
- **Description**: Verify error message for empty message text.
- **Preconditions**: Student is logged in.
- **Steps**:
    1. Navigate to the "Messages" section.
    2. Select counsellor.
    3. Leave the message text empty.
    4. Click on "Send".
- **Expected Result**: Error message "Message text cannot be empty" is displayed.

COUNSELLING SYSTEM-Project/

```
│
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── com/
│   │   │   │   ├── sims/
│   │   │   │   │   ├── controller/
│   │   │   │   │   ├── model/
│   │   │   │   │   ├── repository/
│   │   │   │   │   ├── service/
│   │   │   │   │   └── utils/
│   │   ├── resources/
│   │   │   ├── application.properties
│   │   │   ├── templates/
│   │   │   └── static/
│   ├── test/
│   │   ├── java/
│   │   │   ├── com/
│   │   │   │   ├── sims/
│   │   │   │   │   ├── controller/
│   │   │   │   │   ├── service/
│   │   │   │   │   └── repository/
│   │   └── resources/
│
├── .github/
│   ├── workflows/
│   │   ├── ci.yml
│   │   └── cd.yml
│
├── docs/
│   ├── requirements/
│   ├── design/
│   ├── test-plan/
```

```
│
├── scripts/
│   ├── build.sh
│   └── deploy.sh
│
├── README.md
├── pom.xml (or build.gradle)
└── LICENSE
```

DevOps Architecture using Azure

1. **Source Code Management**:
   - **Tool**: GitHub
   - **Description**: Code repository and version control.
2. **Continuous Integration (CI)**:
   - **Tool**: GitHub Actions
   - **Description**: Automates building and testing of the application.
   - **Pipeline**: Defined in **ci.yml** workflow file.
3. **Continuous Deployment (CD)**:
   - **Tool**: GitHub Actions + Azure DevOps
   - **Description**: Automates deployment to Azure services.
   - **Pipeline**: Defined in **cd.yml** workflow file.
4. **Infrastructure as Code (IaC)**:
   - **Tool**: Azure Resource Manager (ARM) Templates or Terraform
   - **Description**: Manages and provisions infrastructure using code.
5. **Containerization**:

- **Tool**: Docker
- **Description**: Packages application into containers for consistency across environments.

6. **Orchestration**:

- **Tool**: Azure Kubernetes Service (AKS)
- **Description**: Manages deployment, scaling, and operations of containerized applications.

7. **Monitoring and Logging**:

- **Tool**: Azure Monitor, Azure Log Analytics
- **Description**: Provides monitoring, logging, and alerting.

8. **Database**:

- **Tool**: Azure SQL Database or Cosmos DB
- **Description**: Manages application data.

9. **Security**:

- **Tool**: Azure Security Center
- **Description**: Ensures security compliance and protects against threats.

Deployment Architecture

Components:

1. **User Interfaces**:

- **Student Portal**: Web-based interface for students to access their information.
- **Admin Portal**: Web-based interface for administrators to manage student information.
- **Counsellor Portal**: Web-based interface for counsellors to update and view student records.

2. **Backend Services**:

- **API Gateway**: Central entry point for all API requests.

- **Authentication Service**: Manages user authentication and authorization.
- **Student Management Service**: Handles CRUD operations for student data.
- **Course Management Service**: Manages course-related data.
- **Grade Management Service**: Manages student grades and academic records.
- **Notification Service**: Sends notifications and updates to users.

3. **Data Storage**:

- **Relational Database**: Stores structured data for students, courses, grades, etc.
- **Object Storage**: Stores files and documents related to student information.
- **Cache**: Stores frequently accessed data to improve performance.

4. **Monitoring and Logging**:

- **Monitoring Service**: Tracks system performance and health.
- **Logging Service**: Collects and stores logs for troubleshooting and analysis.

5. **Deployment and CI/CD**:

- **CI/CD Pipeline**: Automates build, test, and deployment processes.
- **Container Registry**: Stores container images for deployment.

Cloud Services:

- **Cloud Provider**: Azure, AWS, or GCP (Example: Azure used here)
- **Kubernetes Service**: Azure Kubernetes Service (AKS) for container orchestration.
- **Database Service**: Azure SQL Database for relational data.
- **Storage Service**: Azure Blob Storage for object storage.
- **Cache Service**: Azure Cache for Redis.
- **Monitoring and Logging**: Azure Monitor and Log Analytics.
- **CI/CD**: Azure DevOps for CI/CD pipeline.

Explanation

1. **User Interfaces**: Separate portals for students, admins, and counsellors to interact with the system.

2. **Backend Services**: Modular services handling different aspects of the system, all accessed via the API Gateway.
3. **Data Storage**: Relational database for structured data, object storage for files, and cache for performance.
4. **Monitoring and Logging**: Services to monitor the system and collect logs for analysis.
5. **Deployment and CI/CD**: Automated processes for building, testing, and deploying applications.
6. **Cloud Infrastructure**: Azure Kubernetes Service for container orchestration, ensuring scalability and high availability.

This architecture ensures a robust, scalable, and secure deployment of the Counselling System.