RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19442 - SOFTWARE ENGINEERING CONCEPTS

LABORATORY RECORD NOTEBOOK

Name	**:**	SANJANA SHREE S
Year / Branch / Section	**:**	II/CSE/D
University Register No.	**:**	2116220701245
College Roll No.	**:**	220701245
Semester	**:**	IV
Academic Year	**:**	2023 - 2024

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name: SANJANA SHREE S

Academic Year: 2023 – 2024 Semester: 4 Branch: CSE

Register Number: 2116220701245

Certified that this is a bonafide record of work done by the above student in the CS19442 – Software Engineering Concepts Laboratory during the year 2023-2024.

Signature of Faculty in-charge

Submitted for the Practical Examination held on __________

Internal Examiner External Examiner

CO PO Attainment

Table of Contents

Overview of the Project

Problem it is trying to Resolve:

The primary problem this software aims to resolve is the lack of a systematic and efficient method for assessing and measuring the attainment of Course Outcomes (CO) and Program Outcomes (PO) in educational institutions. Currently, many colleges rely on manual methods to track and evaluate these outcomes, which can be time-consuming, prone to errors, and inefficient. This manual process makes it challenging to:

- Accurately measure the attainment levels of COs and POs.
- Identify areas where students or programs are underperforming.
- Provide actionable insights to faculty and administrators for continuous improvement.
- Efficiently communicate performance data to students and other stakeholders.

Explanation with Respect to Data:

The CO PO Attainment Software addresses these issues by leveraging data in the following ways:

- **Data Entry:** Admin users manually enter relevant data, including student performance metrics, COs, POs, and PSOs. This data forms the foundation for the assessment process.
- **Quantitative Assessment:** The software uses predefined performance evaluation criteria to analyse the entered data. These criteria are used to calculate the attainment levels of COs for each PO and PSO. This involves aggregating student performance data and mapping it to specific educational outcomes.
- **Data Visualization:** The software generates visual graphs and

charts to compare the attainment levels of each PO and PSO. These visual representations make it easier to identify trends, strengths, and weaknesses in the educational programs.

- **Reporting:** Detailed reports are created for different stakeholders, including faculty, administrators, and students. These reports provide insights into performance at various levels, from individual students to entire classes or programs.

How it, would Help Users When Implemented:

Implementing this system will provide several benefits to different user groups:

For Faculty:

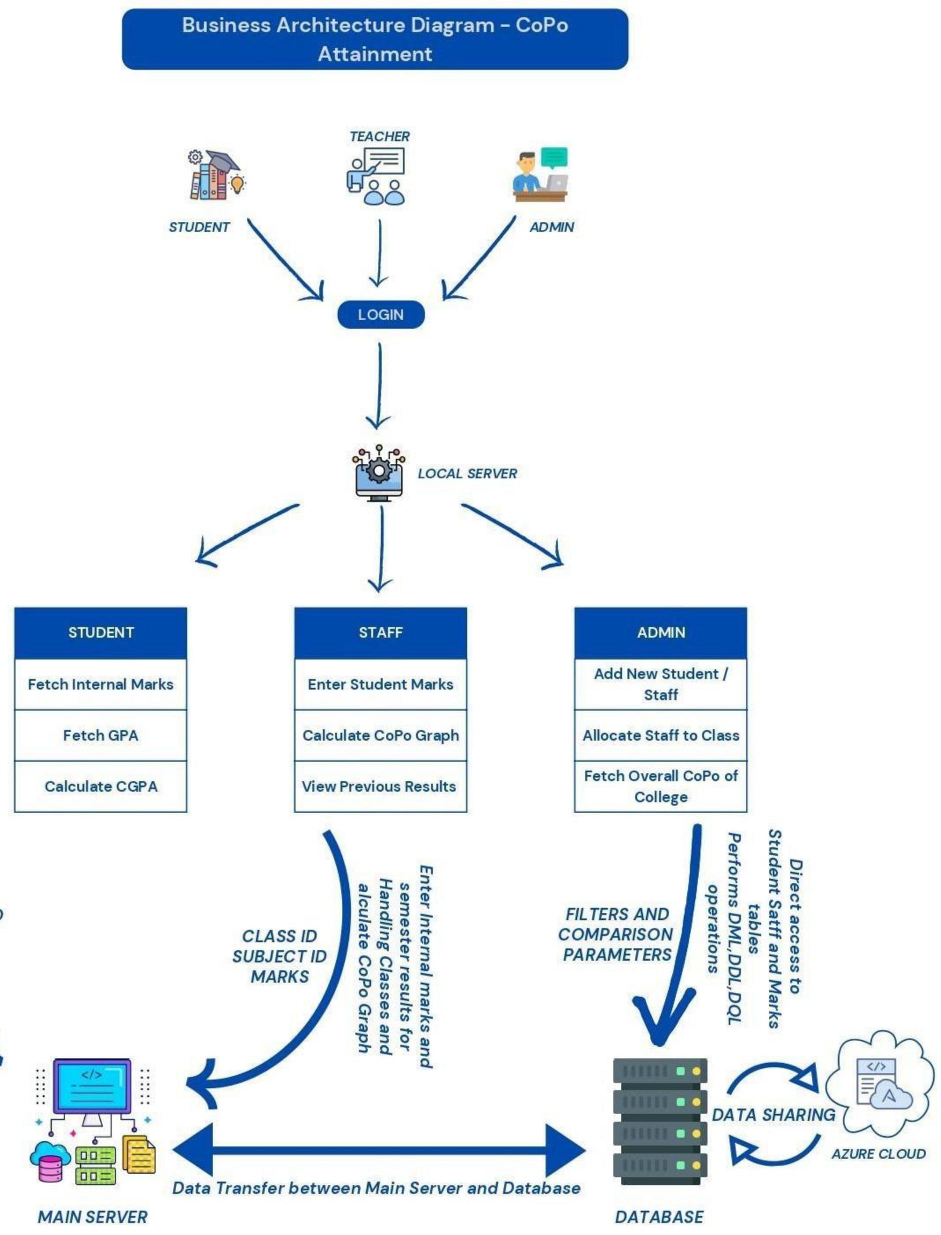
- **Improved Assessment:** Faculty can accurately assess the attainment of COs and POs, identifying specific areas where students are excelling or struggling.
- **Targeted Interventions:** With clear insights into performance data, faculty can design targeted interventions to help students improve in areas where they are underperforming.
- **Data-Driven Teaching:** Faculty can use data to inform their teaching strategies, ensuring they are meeting educational objectives effectively.

For Administrators:

- **Holistic Overview:** Administrators can obtain a comprehensive view of student performance across different courses and programs. This helps in making informed decisions regarding curriculum design and resource allocation.
- **Continuous Improvement:** By identifying trends and patterns in attainment data, administrators can implement policies and initiatives aimed at continuous improvement of educational quality.
- **Accountability and Reporting:** The system provides robust reporting capabilities that can be used for

accreditation purposes and to demonstrate accountability to external stakeholders.

Business Architecture Diagrams


Business Architecture Diagram - CoPo Attainment
TEACHER
STUDENT
ADMIN
LOGIN
LOCAL SERVER
STUDENT
Fetch Internal Marks
Fetch GPA
Calculate CGPA
STAFF
Enter Student Marks
Calculate CoPo Graph
View Previous Results
ADMIN
Add New Student / Staff
Allocate Staff to Class
Fetch Overall CoPo of College
Request to fetch internal marks and semester results with Unique Roll No
ROLL NO
CLASS ID SUBJECT ID MARKS
Enter Internal marks and semester results for Handling Classes and alculate CoPo Graph
FILTERS AND COMPARISON PARAMETERS
Direct access to Student Satff and Marks tables Performs DML,DDL,DQL operations
DATA SHARING
AZURE CLOUD
Data Transfer between Main Server and Database
MAIN SERVER
DATABASE

Business need of the project:

The primary business need for this project is to streamline and enhance the management of academic data within an educational institution. The key goal of our project is to use the system to enhance student performance and identify areas where they may be lagging.

Business problems:

- Time-Consuming Manual Entry: Teachers and administrators spend significant time entering and managing data manually.
- Growing Student Population: As institutions grow, manually managing academic data becomes increasingly unsustainable.

Enhanced Analytical Capabilities:

- Data Analysis: The system can perform complex data analysis, such as CoPo graphs, helping to assess and improve academic programs.
- Performance Tracking: Administrators and teachers can track student performance over time, identify trends, and make informed decisions.
- Comparative Analysis: Compare a student's performance with their peers, class averages, or historical performance data to identify trends and anomalies.
- CoPo Graphs: Specifically, CoPo (Course Outcomes - Program Outcomes) graphs can map student achievements against expected outcomes, providing a clear picture of where students meet or fall short of educational objectives
- Adjusting Curricula: Use data insights to refine and adjust the curriculum to better meet the needs of students.
- Performance Gaps: Identify gaps where student performance on certain COs is not meeting expectations or where there is a weak alignment with POs.

Perspective insights from different users:

1. Students
   - Accessing Data:
     - Login: Students log into the system using their credentials.
     - View Performance: Access individual dashboards showing grades, internal marks, GPA, and CGPA

2. Teachers

- Entering Data:
  - Login: Teachers log into the system.
  - Data Entry: Enter student marks and assessment results into the system.
- Using CoPo Graphs:
  - View Class Performance: Access CoPo graphs to see how well the class is meeting course outcomes and contributing to program outcomes.
  - Identify Trends: Spot trends and gaps in student learning and performance.
- Action Steps:
  - Adjust Teaching Methods: Tailor teaching strategies and materials to address identified gaps.

3. Administrators

- Managing Data:
  - Login: Administrators log into the system.
  - User Management: Add new students and staff, allocate staff to classes.
- Using CoPo Graphs:
  - Overall Analysis: View aggregated CoPo graphs to assess overall program effectiveness.
  - Track Performance: Monitor how different classes and courses contribute to achieving program outcomes.
- Action Steps:
  - Curriculum Development: Use insights to refine and improve the curriculum.

Requirements as User Stories

User Stories:

1. Course Outcome Mapping

- As an admin, I want to map course outcomes to program outcomes so that the curriculum is aligned with educational goals.

- **Estimate:** 3 points

2. **Assessment Creation**

- As a faculty member, I want to create and administer assessments that measure COPO so that student performance can be evaluated consistently.
- **Estimate:** 5 points

3. **Data Collection**

- As an admin, I want to collect data from various assessments and activities so that we can analyse COPO attainment levels.
- **Estimate:** 3 points

4. **Rubric Development**

- As a faculty member, I want to develop rubrics for consistent evaluation of outcomes so that assessments are standardized across courses.
- **Estimate:** 2 points

5. **Report Generation**

- As an admin, I want to generate detailed reports on COPO attainment for stakeholders so that performance can be reviewed and acted upon.
- **Estimate:** 5 points

6. **Interactive Dashboards**

- As a faculty member, I want to access interactive dashboards for real-time monitoring of attainment levels so that I can quickly identify areas needing improvement.

- **Estimate:** 8 points

7. **Feedback Collection**
   - As a faculty member, I want to gather feedback from students, faculty, and other stakeholders so that continuous improvement plans can be implemented.
   - **Estimate:** 5 points

8. **Compliance Documentation**
   - As an admin, I want to maintain comprehensive documentation for accreditation purposes so that we meet educational standards and requirements**.**
   - **Estimate:** 3 points

9. **User Authentication**
   - As a user, I want secure login and authentication mechanisms so that my data is protected.
   - **Estimate:** 2 points

10. **Data Security**
    - As an admin, I want to protect sensitive data from unauthorized access and breaches so that the integrity and confidentiality of data are maintained.
    - **Estimate:** 8 points

11. **Performance Analytics**
    - As an admin, I want the system to analyse data and provide insights into COPO attainment levels so that we can make data-driven decisions.
    - **Estimate:** 5 points

Summary of Poker Planning Estimates

1 point: Very small effort

2 points: Small effort

3 points: Moderate effort

5 points: Significant effort

8 points: Large effort

Non-Functional Requirements (NFRs)

1. **Performance:**
    - **Efficiency:** The system should process data and generate reports quickly and efficiently to ensure timely decision-making.
    - **Scalability:** The system must handle an increasing number of students, courses, and data points without performance degradation.
2. **Usability:**
    - **User-Friendly Interface:** The interface should be intuitive and easy to use for faculty and administrators to reduce the learning curve and increase productivity.
    - **Accessibility:** The system should be accessible to users with disabilities, complying with accessibility standards.
3. **Reliability:**
    - **Data Integrity:** Ensure the accuracy and consistency of data to maintain trust in the system's outputs.
    - **System Uptime:** The system should have high availability with minimal downtime to ensure continuous access for all users.

4. **Security:**

    - **Data Security:** Protect sensitive data from unauthorized access and breaches.
    - **User Authentication:** Secure login and authentication mechanisms.

5. **Interoperability:**

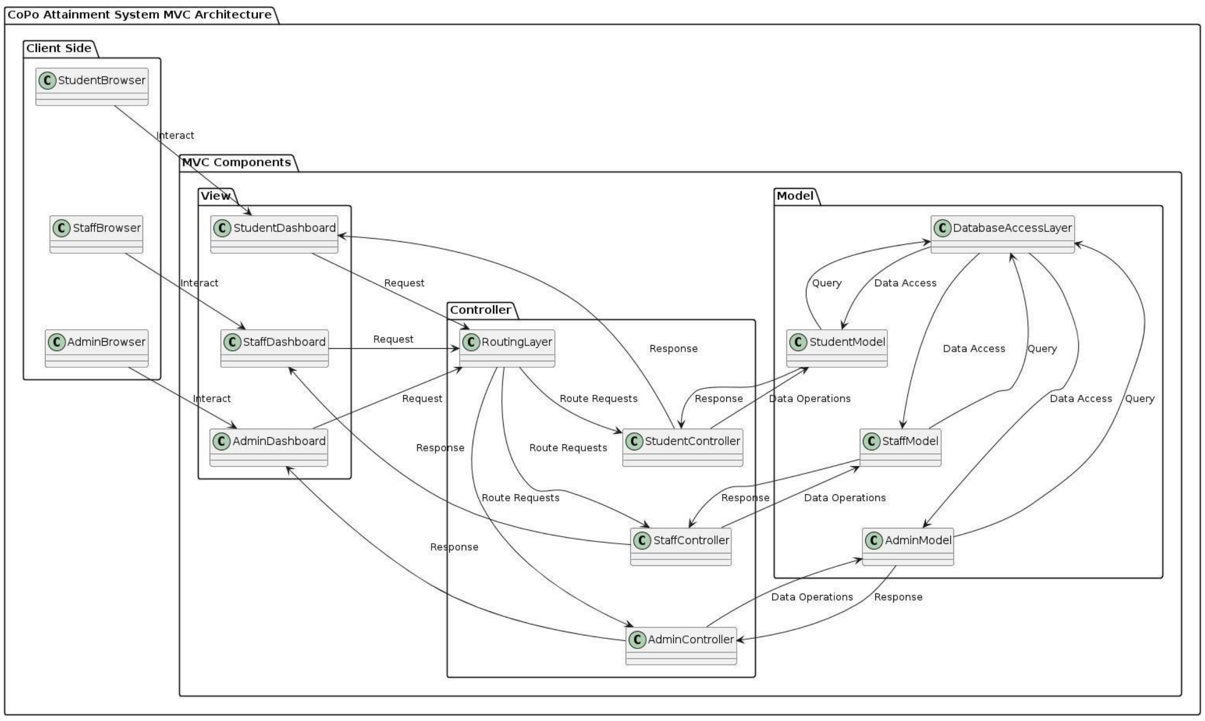
    - **Integration:** Ability to integrate with existing educational management systems and tools.
    - **Standard Compliance:** Adherence to data exchange standards for compatibility with other systems.

6. **Maintainability:**

    - **Modular Design:** System designed for easy updates and maintenance.
    - **Documentation**: Comprehensive documentation for system usage and maintenance.

Architecture Diagrams

MVC Diagram:



Model:

Represents the application's core data and business logic. It encapsulates data access and manipulation, ensuring data integrity.

Components in the Work Planner Model:

- Task Service: Creates, modifies, retrieves, and deletes tasks. Manages task-related data and business logic.
- Reporting Service: Generates various reports based on work planner data (e.g., task progress, user activity).
- Notification Service: Manages notifications and reminders related to tasks and deadlines.
- Authentication Service: Validates user credentials and manages user access levels.
- User Service: Manages user accounts and profiles, including access permissions.

- Calendar Service: Integrates with calendar applications to schedule tasks for better visibility and time management.
- Data Access Objects (DAOs): Classes that handle database interactions, providing a layer of abstraction for services. DAOs for Tasks, Reports, Notifications, Users, and Calendars are included.

View:

Handles how the application presents information to the user, translating the model's data into a user-friendly format on different interfaces (mobile, web, desktop).

Components in the Work Planner View:

- User Interface (UI): Visual components for user interaction. Separate UIs for mobile, web, and desktop devices. Includes functionalities for creating, viewing, modifying, and managing tasks, reports, notifications, and user accounts.

Potential Additional Components within the View:

- Task Lists: Displays assigned tasks, team tasks, and upcoming deadlines.
- Task Details View: Shows details of specific tasks, including descriptions, attachments, progress updates, and collaborators.
- Calendar View: Visualizes scheduled tasks and deadlines.
- Reports View: Presents generated reports in user-friendly formats like charts and tables.

Controller:

Acts as an intermediary between the user and the model, receiving user input from the view, interacting with the model to perform actions based on that input, and updating the view with any changes.

Components in the Work Planner Controller:

- Task Controller:
    - Manages user actions related to tasks, such as creating, editing, deleting, and retrieving tasks.
    - Interacts with the Task Service and updates the UI accordingly.
- Report Controller:
    - Handles user actions related to generating reports by date range, user, or task status.
    - Interacts with the Reporting Service and updates the UI with the generated reports.
- Notification Controller:
    - Manages user actions related to notifications, such as setting reminders.
    - Interacts with the Notification Service and updates the UI to reflect these changes.
- User Controller:
    - Handles user actions related to user accounts, including login, logout, and profile management.
    - Interacts with the Authentication Service and User Service to process these actions and updates the UI accordingly.
- Calendar Controller:
    - Manages user actions related to calendars, such as scheduling tasks or syncing with calendar applications.
    - Interacts with the Calendar Service and updates the UI to reflect changes (e.g., task added to the calendar).

Design Principles

1. Separation of Concerns

- Principle: Each class is responsible for a specific aspect of the application.
- Application:
  - Model Classes: Student, Course, Outcome, and Assessment represent the data structure.
  - View Classes: DashboardView, ReportView, ChartView handle the presentation logic.
  - Controller Classes: AdminController, FacultyController, ReportController manage the business logic and user interactions.
- Why: This principle helps in reducing complexity by dividing the application into distinct sections, making the system easier to manage, understand, and maintain.

2. Single Responsibility Principle (SRP)

- Principle: Each class should have only one reason to change, meaning each class should only have one job or responsibility.
- Application:
  - AdminController manages adding students, courses, outcomes, and entering assessment data.
  - FacultyController is responsible for viewing dashboards and reports.
  - ReportController handles generating reports and charts.
- Why: By adhering to SRP, the codebase becomes more modular and easier to test. Changes in one part of the system are less likely to affect other parts.

3. Encapsulation

- Principle: Encapsulation involves bundling the data (attributes) and methods (functions) that operate on the data into a single unit or class, and restricting access to some of the object's components.
- Application: The attributes of classes like Student, Course, Outcome, and Assessment are encapsulated within their

respective classes, with only relevant methods exposed.

- Why: Encapsulation protects the internal state of an object from unintended modification and misuse, ensuring data integrity and security.

4. Use of UML for Design Documentation
    - Principle: Unified Modeling Language (UML) is used to visually represent the design of a system.
    - Application: The provided diagram uses PlantUML to depict the structure and relationships of the system's classes.
    - Why: UML helps in visualizing the system architecture, making it easier to communicate the design to stakeholders and team members. It also aids in identifying potential issues early in the development process.

5. Composition Over Inheritance
    - Principle: Favor composition over inheritance to achieve code reuse.
    - Application: Classes like Student, Course, and Outcome are composed of lists of other objects rather than inheriting from them.
    - Why: Composition provides more flexibility than inheritance. It allows for the dynamic creation of complex objects by combining simpler objects, promoting loose coupling and easier code maintenance.

6. Controller-View Interaction
    - Principle: Controllers should handle the user inputs and interactions, while views should focus on presenting data.
    - Application:
        - FacultyController interacts with DashboardView, ReportView, and ChartView to display information to users.
        - ReportController generates the data needed for ReportView and ChartView.

- Why: This separation ensures that views are kept free of business logic, making it easier to modify the user interface independently of the underlying login

MODULES

User Interface Module:

- Function: Handles user interactions, collects inputs, and displays outputs.
- Interaction: Communicates with the Application Logic module for processing user commands.
- Error Handling: Validates user inputs, displays error messages for invalid inputs.

Application Logic Module:

- Function: Core processing logic of the application, implements business rules.
- Interaction: Interacts with both the User Interface for inputs/outputs and the Data Storage module for data access and storage.
- Error Handling: Manages errors from data processing, ensures business rules are followed.

Data Storage Module:

- Function: Manages data persistence, interacts with databases or file systems.
- Interaction: Provides data to and receives data from the Application Logic module.
- Error Handling: Handles database errors, ensures data integrity.

Logging Module:

- Function: Records application events, errors, and other significant occurrences.
- Interaction: Integrated across all modules to log activities and errors.
- Error Handling: Logs errors from all modules, provides information for debugging and monitoring.

Interactions

- User Interface ↔ Application Logic: Sends user inputs to the Application Logic, receives processed data to display.
- Application Logic ↔ Data Storage: Requests data from the Data Storage, stores processed data.
- Application Logic ↔ Logging: Logs significant events and errors during processing.
- Data Storage ↔ Logging: Logs data access and storage operations.

Error Handling

- User Interface: Checks for invalid inputs, shows error messages.
- Application Logic: Catches exceptions during processing, maintains application flow.
- Data Storage: Manages database connection issues, retries or reports errors.
- Logging: Captures errors from all modules, aids in troubleshooting.

Logging

- Purpose: Tracks application behavior, assists in debugging, monitors performance.
- Types: Info, Debug, Error, Warning.
- Storage: Log files, centralized logging services.

Data Storage

- Types: Relational databases (SQL), NoSQL databases, file systems.
- Access Methods: SQL queries, ORM (Object-Relational Mapping), file read/write operations.
- Backup and Recovery: Regular backups, data recovery plans to handle data loss.

Example Flow

User Action: User submits a form.

User Interface: Validates input, sends data to Application Logic.

Application Logic: Processes data, performs business logic, interacts with Data Storage for data retrieval/storage.

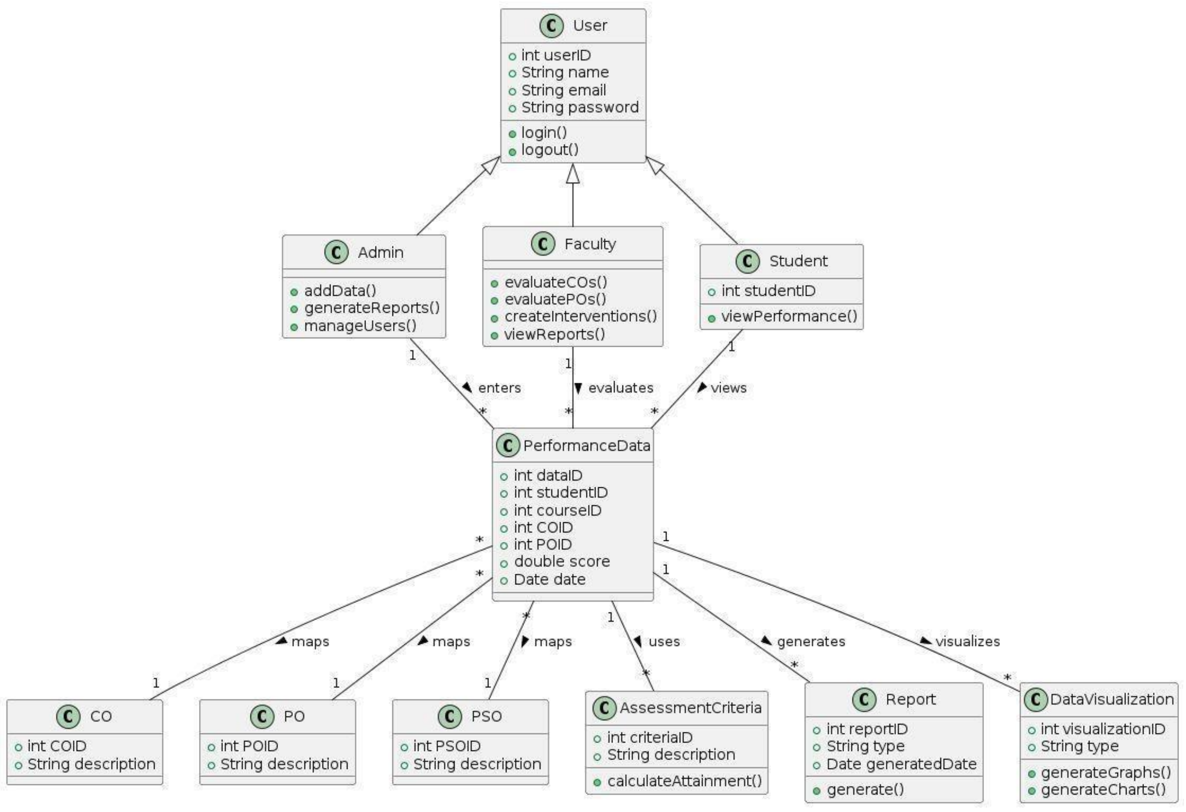
Data Storage: Executes database operations, returns results to Application Logic.

Application Logic: Processes results, sends response to User Interface.

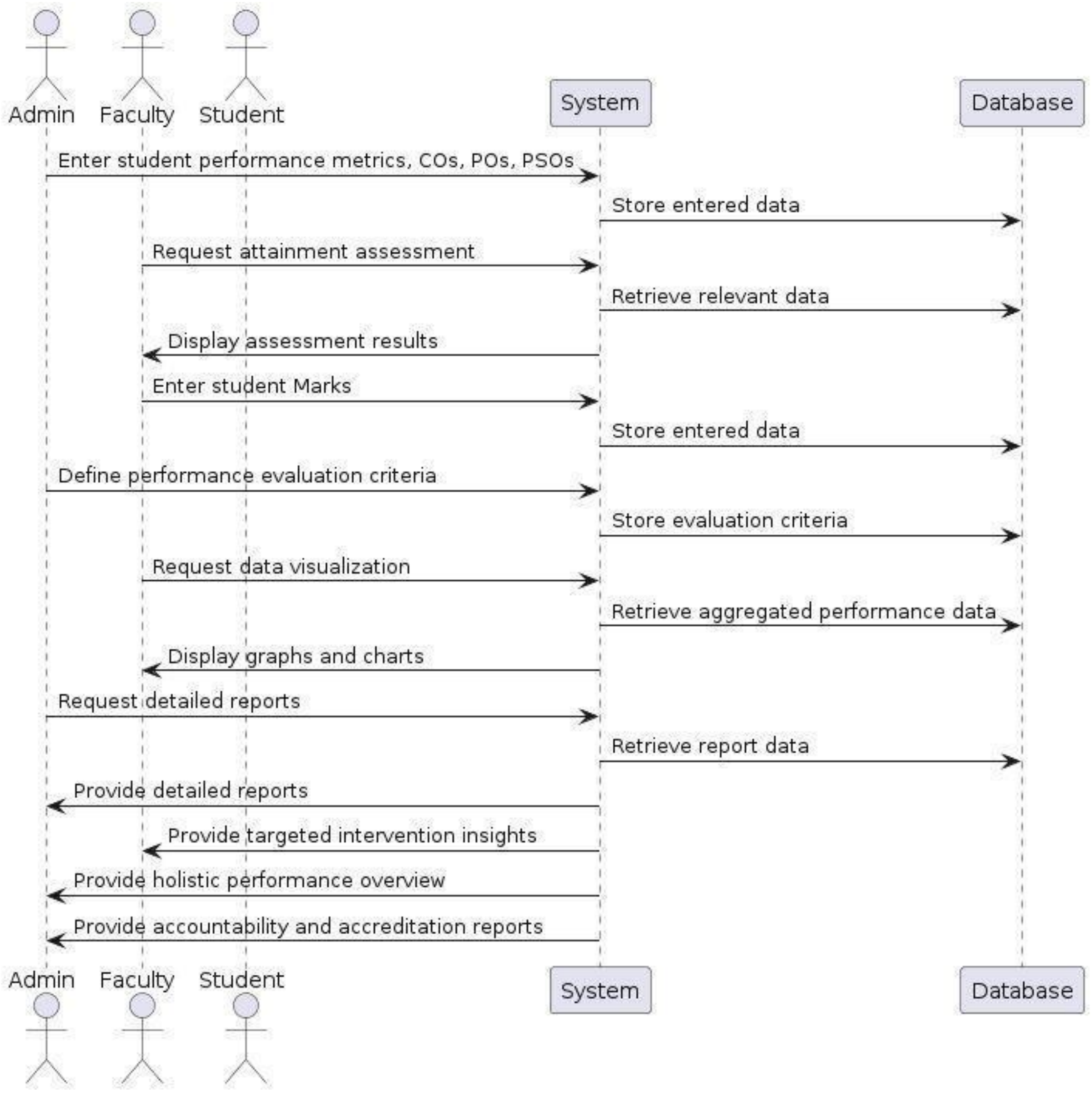
User Interface: Displays results to the user.

Logging: Logs each step and any errors encountered.

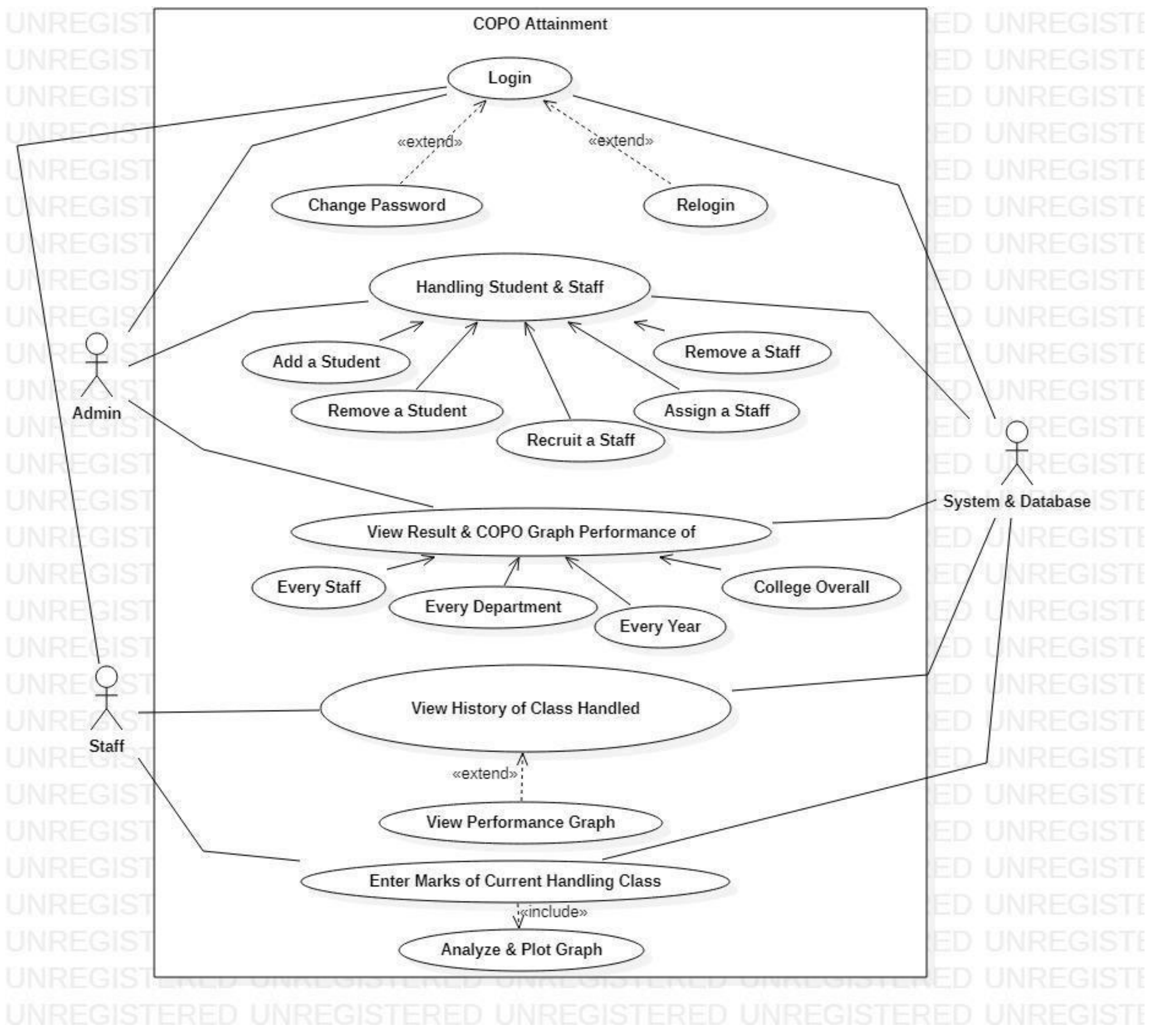
Class Diagram:



Sequence Diagram:


Admin
Faculty
Student
System
Database
Enter student performance metrics, COs, POs, PSOs
Store entered data
Request attainment assessment
Retrieve relevant data
Display assessment results
Enter student Marks
Store entered data
Define performance evaluation criteria
Store evaluation criteria
Request data visualization
Retrieve aggregated performance data
Display graphs and charts
Request detailed reports
Retrieve report data
Provide detailed reports
Provide targeted intervention insights
Provide holistic performance overview
Provide accountability and accreditation reports
Admin
Faculty
Student
System
Database

Use Case Diagram:



Conclusion

CoPo graphs are powerful tools for mapping and analyzing the relationship between course-specific learning outcomes and overarching program outcomes. By systematically collecting, processing, and visualizing this data, educational institutions can make informed decisions that enhance both teaching effectiveness and student performance. This approach not only helps in achieving educational excellence but also supports continuous improvement and accountability in academic programs.