# SEC LAB RECORD

**SUBJECT CODE:**

CS19442

**SUBJECT NAME:**

SOFTWARE ENGINEERING CONCEPTS

**PROJECT TITLE:**

LEAVE MANAGEMENT SYSTEM

NAME: Rishi Karthick P

REGISTER NO.: 220701225

DEPARTMENT: CSE-D

ACADEMIC YEAR: 2023-2024

YEAR/SEMESTER: II Year IV Sem

# LEAVE MANAGEMENT SYSTEM

## OVERVIEW OF THE PROJECT:

### PROBLEM STATEMENT:

Nowadays, the most common problem faced by the students is related to attendance, they get shortage of attendance due to several factors beyond the control as in case of emergency where they are not able to send their leave application. The aim of this project is to create a leave management system that can be used by both students and staff that will help to clear any chaos during validation of any leave. This application monitors the leave application of the applicants. Once the user applied for a leave, the application is validated by the Admin(counseller, HOD, class incharge, Dean).

### OBJECTIVE

1. **Automation of Leave Requests:** Allow students to submit leave requests online, reducing paperwork and manual processing.
2. **Streamlined Approval Process:** Facilitate a systematic approval workflow involving class teachers, faculty advisors, and department heads.
3. **Transparency and Accessibility:** Provide real-time access to leave information and status for students and faculty.
4. **Compliance and Reporting:** Ensure adherence to institutional policies and provide robust reporting capabilities for administrative purposes.
5. **Record Management:** Maintain accurate and up-to-date records of student leaves for academic and administrative review.

## KEY FEATURES

1. **User Authentication and Roles:** Secure login with role-based access (Student, Faculty, Administrator).
2. **Leave Request Submission:** Enable students to submit leave requests with details such as leave type (e.g., medical, personal), dates, and reasons.
3. **Approval Workflow:** Implement a workflow for faculty and administration to review, approve, or reject leave requests.
4. **Notification System:** Send notifications and alerts for key actions (e.g., request submission, approval, reminders).
5. **Calendar Integration:** Display leave schedules to avoid conflicts with academic activities.
6. **Reporting and Analytics:** Generate reports on leave trends, usage, and balances for academic and administrative purposes.
7. **Policy Enforcement:** Ensure the system enforces institutional leave policies.
8. **Audit Trail:** Maintain logs of all leave transactions for audit and review purposes.

## USER BENEFITS

### Student benefits:

1. **Ease of Submission:** Students can submit leave requests online anytime, from anywhere, eliminating the need for physical forms and in-person submissions.
2. **Transparency:** Students can track the status of their leave requests in real-time, knowing whether it has been approved, denied, or is pending.
3. **Reduced Errors:** Automated processes reduce the risk of errors in leave applications and approvals.
4. **Time-saving:** Quick and easy submission process saves time and effort compared to manual methods.
5. **Notifications and Reminders:** Students receive notifications about the status of their requests and reminders for pending approvals or necessary actions.

## Admin benefits:

1. **Streamlined Approvals:** can review and approve or reject leave requests online, which is more efficient than handling paper forms.
2. **Real-time Information**: Faculty have immediate access to students' leave status and history, helping in planning and managing academic responsibilities.
3. **Consistency and Compliance:** Ensures that all leave requests are handled according to institutional policies, maintaining consistency and fairness.
4. **Reduced Administrative Burden:** Faculty can focus more on teaching and mentoring rather than handling leave paperwork.

## SYSTEM IMPLEMENTATION BENEFITS:

1. **Efficiency:** Reduced administrative workload and faster processing of leave requests.
2. **Accuracy:** Improved accuracy in tracking and managing student leave records.
3. **Transparency:** Enhanced visibility into the leave process for all stakeholders.
4. **Compliance:** Better adherence to institutional policies and regulations.
5. **User Satisfaction:** Increased student and faculty satisfaction due to a streamlined and transparent process.

## BUSINESS NEED OF THE PROJECT

## CURRENT PROCESS

## AUTOMATED PROCESS AND ITS WORKING:

**Online Leave Request Submission:**

Students access the Student Leave Management System through a web portal or mobile app.

They log in using their credentials and submit leave requests electronically, specifying the type of leave, duration, and any supporting documentation.

**Automated Approval Workflow:**

Upon submission, the system automatically routes the leave request to the appropriate authority based on predefined rules and workflows.

The system notifies the designated approver(s) via email or through the system's interface, prompting them to review and take action on the request.

**Automatic Notification:**

Once the leave request is approved or rejected, the system sends an automated notification to the student, informing them of the decision.

The notification includes details such as the approved leave dates, any conditions or remarks from the approver, and instructions for further action if necessary.

**Real-time Leave Balances and Tracking:**

The system automatically updates the student's leave balances in real-time based on approved leave requests.

Students can view their current leave balances and track their leave history through the system's dashboard or reporting features.

## BUSINESS NEEDS, CURRENT PROCESS AND PROBLEMS FOR DIFFERENT PERSONAS:

**ADMIN:**

1. **BUSINESS NEEDS:** An automated system to streamline the submission, approval, and tracking of leave requests, thereby freeing up administrative and faculty time for more productive tasks. To provide real-time visibility into the status of leave requests for students, faculty, and administrators, enhancing transparency.
2. **CURRENT PROCESS:** The current process for administering student leave requests is characterized by manual steps, paperwork, and limited automation. This manual approach results in inefficiencies, errors, and challenges in ensuring compliance and transparency.
3. **BUSINESS PROBLEMS:** The current manual process for managing student leave requests in educational institutions is inefficient, error-prone, and lacks transparency. This creates significant administrative burdens and compliance challenges.

**STUDENT:**

1. **BUSINESS NEEDS:** Students need a simple and convenient way to submit leave requests without unnecessary paperwork or bureaucracy. Students require real-time updates on the status of their leave requests, including approvals, rejections, and pending reviews. Ability to request different types of leave, such as sick leave, personal leave, or academic leave, with varying durations and purposes.
2. **CURRENT PROCESS:** Students typically fill out paper forms or send emails to request leave. These forms may require details such as leave dates, reason for leave, and any supporting documentation. Approved leave requests are manually entered into spreadsheets, databases, or paper logs to maintain records.
3. **BUSINESS PROBLEM:** Students often have limited visibility into the status of their leave requests and the approval process. Students may have limited access to information about their current leave balances, usage history, or available options.

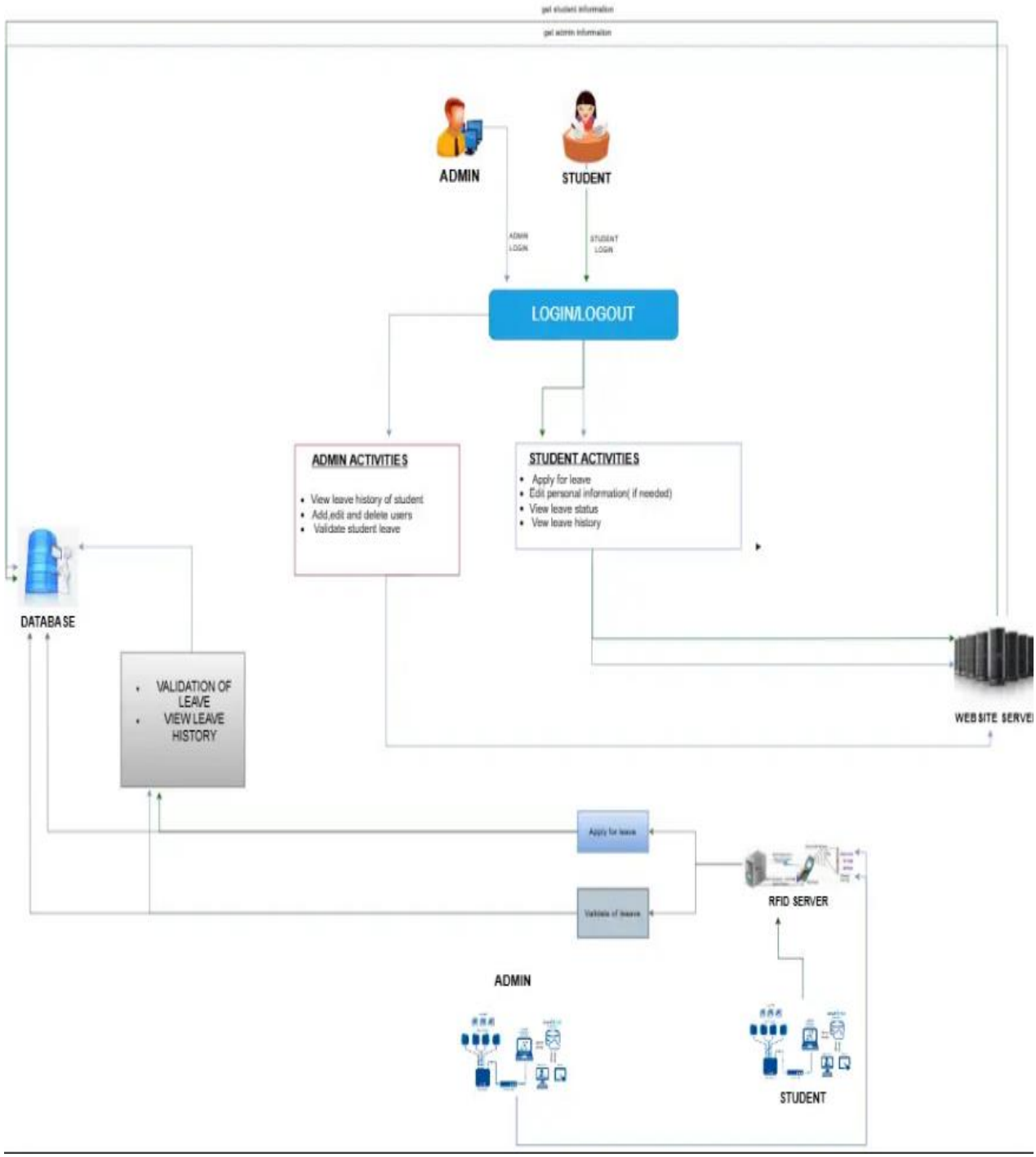
**ADDRESSING BUSINESS PROBLEMS:**

**Inefficiency in Leave Processing:**
**Solution:** Implement an automated leave management system that streamlines the entire process, from request submission to approval and tracking. Automation reduces processing time, eliminates paperwork, and ensures consistency in approval decisions.

**Lack of Transparency and Communication:**
**Solution:** Implement automated notifications to keep students informed at each stage of the leave approval process. Notifications should provide real-time updates on the status of their requests, including approval/rejection decisions and any additional actions required.

**Difficulty in Tracking Leave Balances and History:**
**Solution:** Provide students with access to a user-friendly dashboard within the leave management system where they can easily view their current leave balances, upcoming leave schedules, and historical leave records.

# BUSINESS ARCHITECTURE DIAGRAM:



## FUNCTIONAL REQUIREMENTS/USER STORIES

- As a student, I want to submit a leave request online so that I can easily request time off for personal or academic reasons without having to fill out paper forms.

- As a Student, I want to check my leave balance so that I know how many days I have left.

- As a Student, I want to upload supporting documents for my leave application so that I can provide evidence for medical or other types of leave.

- As an Admin, I want to view and manage all leave requests submitted by students so that I can ensure compliance with policies.

- As an Admin, I want to approve or reject leave applications based on institution's guidelines so that leave is granted appropriately.

- As an Admin, I want to access and verify the supporting documents provided by students for leave applications so that I can validate their requests.

- As an Admin, I want to generate reports on student leaves so that I can analyze leave patterns and provide insights to the management.

- As a Student, I want to view the status of my leave applications so that I am aware of their approval or rejection.

- As a Student, I want to see the leave history of so that I can take leave accordingly

- As an Admin, I want to track the leave history of students so that I can identify any patterns of frequent absenteeism.

## NON-FUNCTIONAL REQUIREMENTS:

1. **Performance:**

   **Response Time:** The system should respond to user interactions within a reasonable time frame, typically within 2-3 seconds.

   **Scalability:** The system should be able to handle an increasing number of users and data without significant degradation in performance.

   **Throughput:** The system should be able to process a certain number of leave requests per unit time.

2. **Security:**

   **Authentication:** Users should be required to authenticate themselves before accessing the system, using secure methods such as passwords or biometrics.

   **Authorization:** Different user roles should have appropriate levels of access to the system and its functionalities.

   **Data Encryption:** Sensitive data such as personal information and leave records should be encrypted to prevent unauthorized access.

3. **Reliability:**

   **Availability:** The system should be available for use during peak times, with minimal downtime for maintenance or updates.

   **Fault Tolerance:** The system should be resilient to failures, ensuring that data integrity is maintained even in case of hardware or software failures.

   **Backup and Recovery:** Regular backups of data should be taken, and there should be a mechanism in place to restore the system to a previous state in case of data loss.

# POKER PLANNING ESTIMATE

**User Story: Apply for Leave**

- **Estimate:** 3 points
- **Explanation:** This task involves implementing the functionality for students to apply for leave, including form creation, validation, and database integration. It's moderately complex but manageable.

**User Story: Approve Leave**

- **Estimate:** 5 points
- **Explanation:** Implementing the approval process for leave requests requires handling authorization, notifications, and status updates. It involves coordination between different components and stakeholders, making it more complex.

**User Story: View Leave History**

- **Estimate:** 2 points
- **Explanation:** Creating a simple interface for students to view their leave history is relatively straightforward. It doesn't involve complex logic or interactions, resulting in a lower estimate.

**User Story: Cancel Leave Request**

- **Estimate:** 3 points
- **Explanation:** Implementing the functionality for students to cancel their leave requests involves updating request statuses and handling notifications. While not overly complex, it requires some coordination and logic.

**User Story: Upload Supporting Documents**

- **Estimate:** 5 points
- **Explanation:** Developing the feature for students to upload supporting documents adds complexity due to file handling, validation, and

security measures. It requires implementing file upload functionality and handling various file types.

**User Story: View Leave Application Status**

- **Estimate:** 2 points
- **Explanation:** Creating the feature for students to view the status of their leave applications is relatively straightforward. It involves retrieving and displaying information from the database without significant complexity.

**User Story: Manage Leave Types**

- **Estimate:** 3 points
- **Explanation:** Implementing the functionality for administrators to manage different types of leave (e.g., sick leave, vacation) involves creating an interface for adding, editing, and deleting leave types. It requires database updates and validation logic.

**User Story: Set Leave Policies**

- **Estimate:** 5 points
- **Explanation:** Developing the feature for administrators to set leave policies involves defining rules and constraints for leave eligibility, accrual rates, and maximum balances. It requires careful consideration of business rules and implementation of complex logic.

**User Story: Notify Users of Leave Status Changes**

- **Estimate:** 4 points
- **Explanation:** Implementing notification functionality to inform students and administrators of leave status changes (e.g., approval, rejection) involves integrating with email or notification services. It requires handling different notification scenarios and user preferences.

**User Story: Generate Leave Reports**

- **Estimate:** 3 points
- **Explanation:** Creating the feature for administrators to generate leave reports involves querying the database and formatting the data into reports.

## ARCHITECTURE DIAGRAM:

## MVC ARCHITECTURE:

**Model:** This module represents the data and business logic of the application. It includes entities such as classrooms, students, faculty, and allocation details. The model interacts with the database for CRUD operations on data entities.

**View:** The view module represents the user interface components of the application. It includes screens for admin, faculty, and student interfaces to view allocation details, request modifications, and view schedules.

**Controller:** The controller module acts as an intermediary between the model and the view. It processes user requests, interacts with the model to fetch or update data, and sends the appropriate response to the view. Controllers handle authentication, allocation processing, and error handling logic.

## Error Handling and Logging:

**Error Handling:** Exception handling mechanisms would be implemented within the controller layer to catch and manage runtime errors gracefully. Specific error messages can be returned to users through the view layer for informative feedback.

**Logging:** Logging mechanisms can be implemented within the controller layer to record important events, errors, and user actions. This helps in debugging, auditing, and monitoring the application's behavior.

## Data Storage:

Data storage would typically involve a relational database management system (RDBMS) such as MySQL or PostgreSQL to store information about

classrooms, students, faculty, allocation details, and schedules. The model layer interacts with the database through an ORM (Object-Relational Mapping) framework like SQLAlchemy in Python.

## ARCHITECTURE PATTERN USED:

MVC Architecture: The MVC architecture pattern is chosen for its ability to separate concerns, making the application easier to understand, maintain, and scale. It promotes modularity, reusability, and testability by clearly defining the roles of models, views, and controllers.

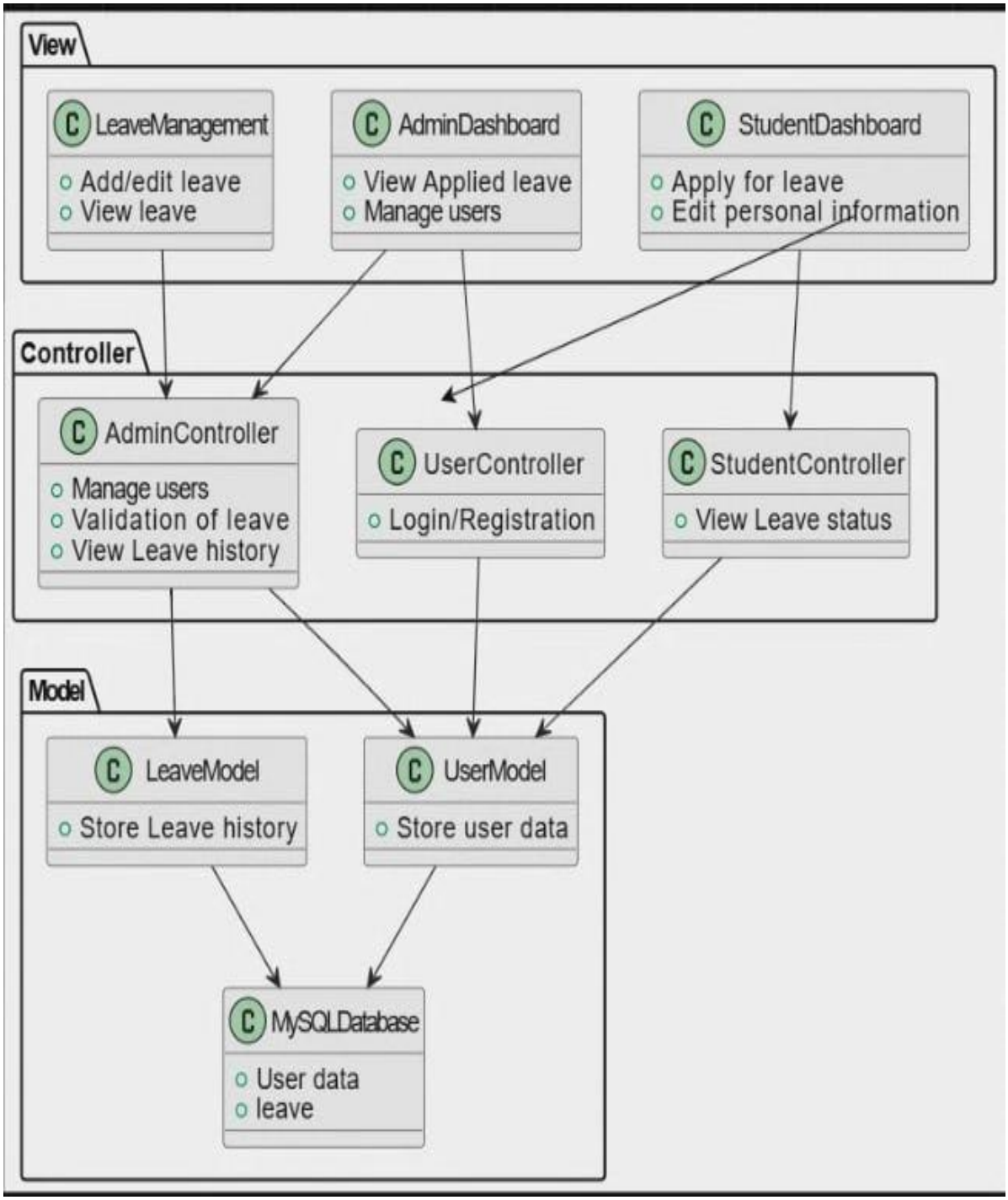
## DESIGN PRINCIPLES USED:

**Separation of Concerns (SoC):** MVC adheres to the SoC principle by dividing the application into distinct modules responsible for handling different aspects of the system (data, presentation, and control logic). This makes the codebase more modular, easier to maintain, and less prone to unintended side effects.

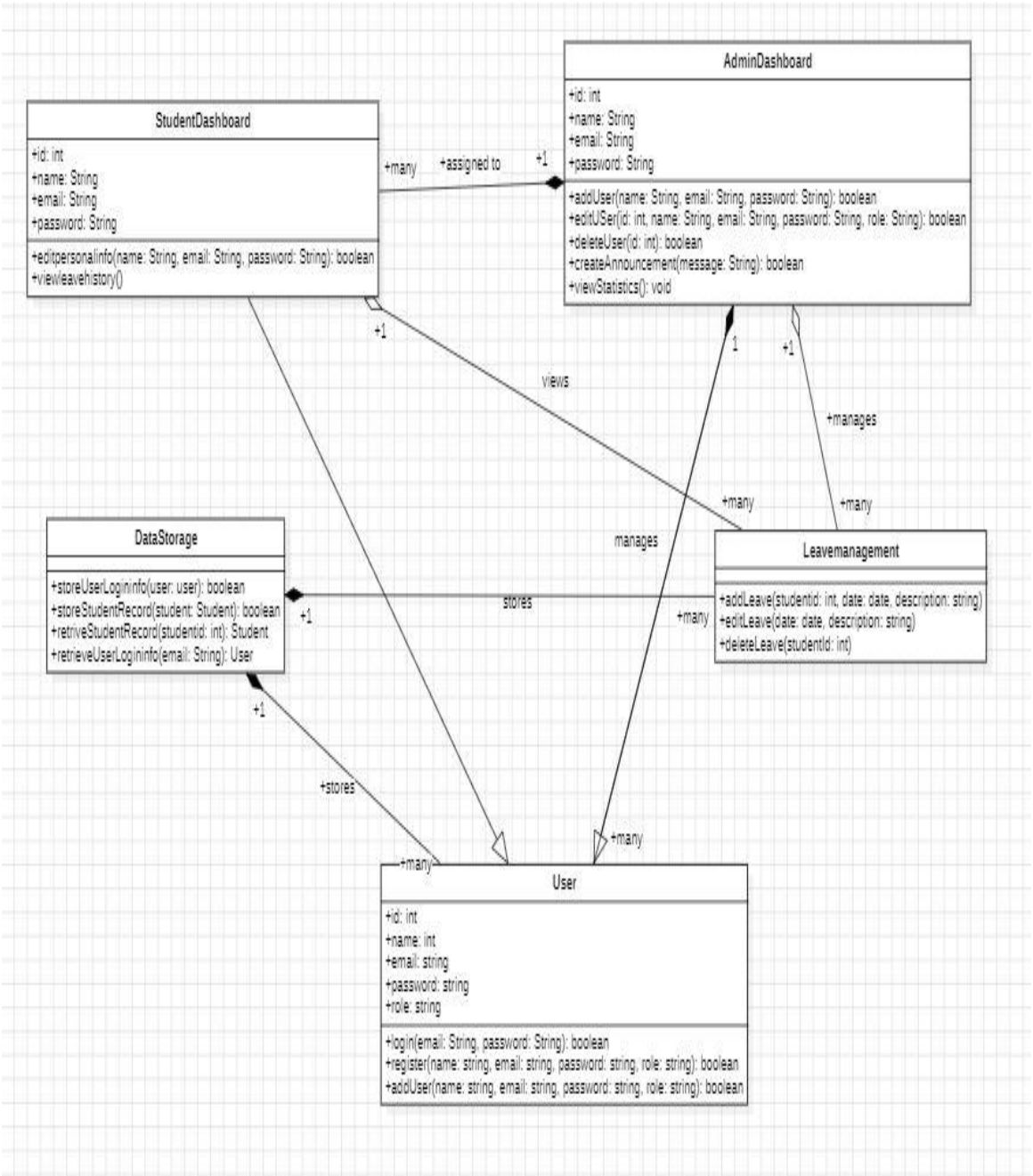
**Single Responsibility Principle (SRP):** Each component in the MVC architecture follows the SRP by having a single responsibility. Models handle data and business logic, views handle presentation logic, and controllers handle user input and application flow.

**Loose Coupling:** MVC promotes loose coupling between modules, allowing changes in one component to have minimal impact on others. This enhances flexibility, scalability, and maintainability of the application

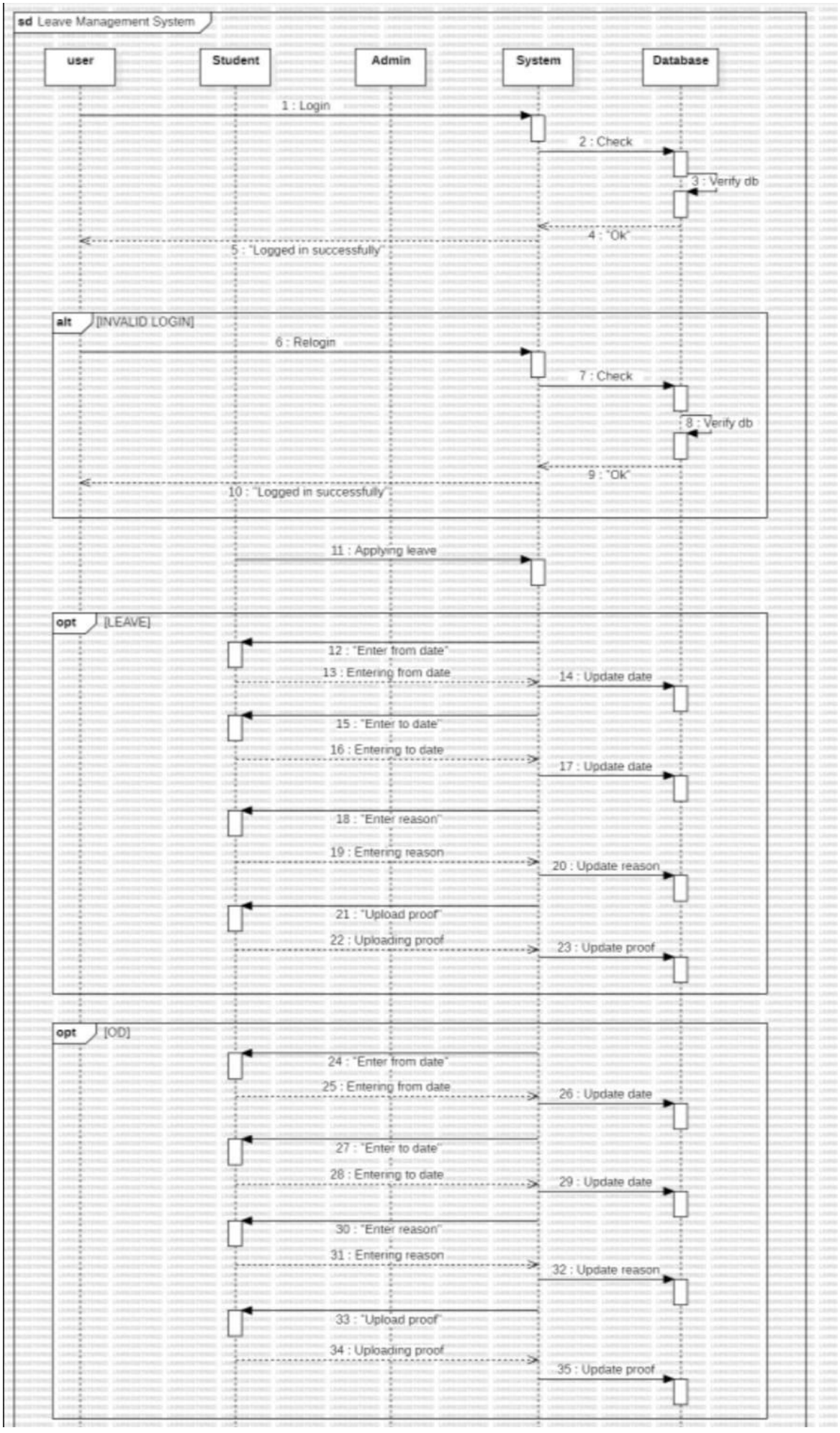
# MVC ARCHITECTURE DIAGRAM

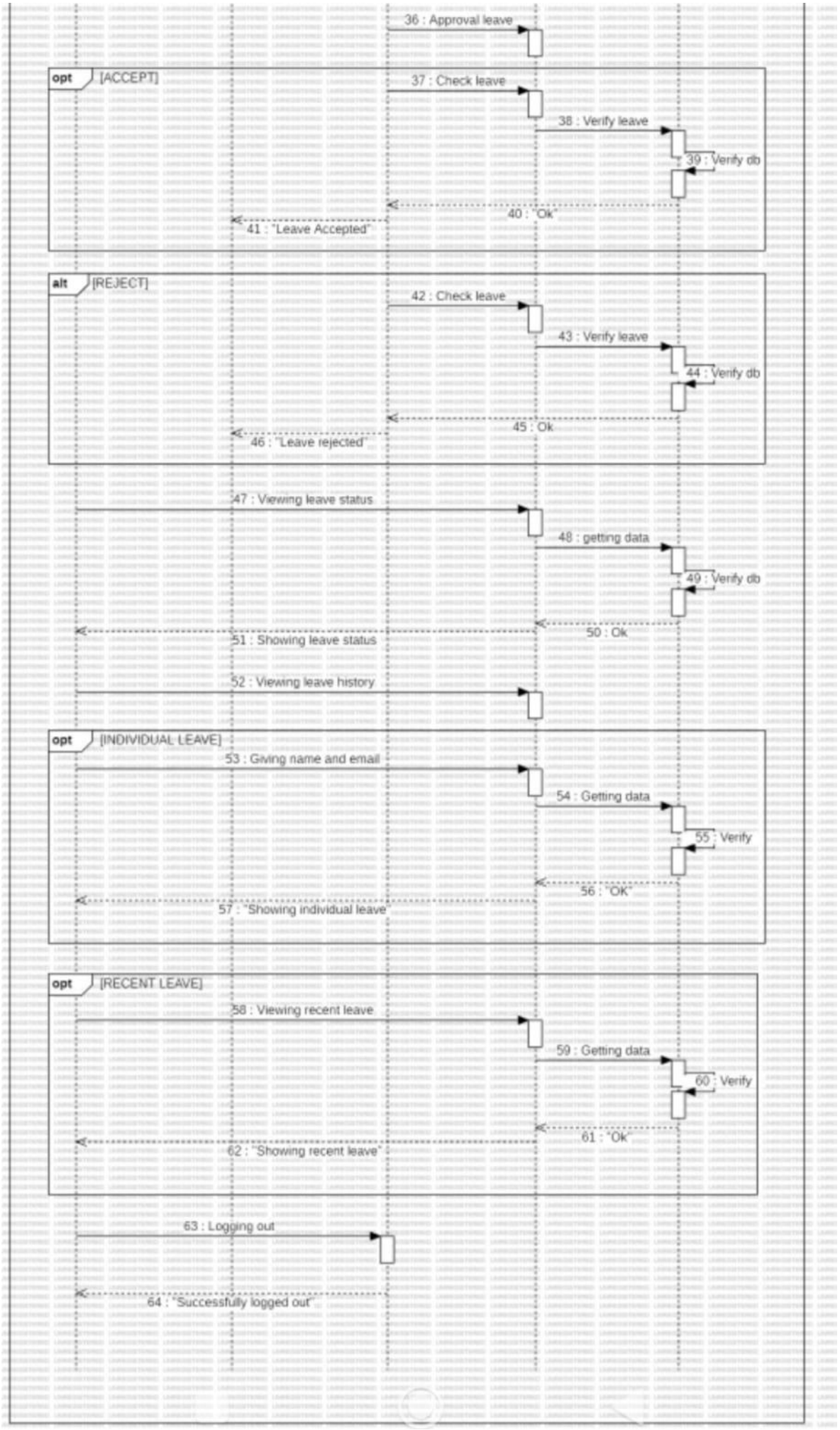


# CLASS DIAGRAM:



# SEQUENCE DIAGRAM




36 : Approval leave
opt
[ACCEPT]
37 : Check leave
38 : Verify leave
39 : Verify db
40 : "Ok"
41 : "Leave Accepted"
alt
[REJECT]
42 : Check leave
43 : Verify leave
44 : Verify db
45 : Ok
46 : "Leave rejected"
47 : Viewing leave status
48 : getting data
49 : Verify db
50 : Ok
51 : Showing leave status
52 : Viewing leave history
opt
[INDIVIDUAL LEAVE]
53 : Giving name and email
54 : Getting data
55 : Verify
56 : "OK"
57 : "Showing individual leave"
opt
[RECENT LEAVE]
58 : Viewing recent leave
59 : Getting data
60 : Verify
61 : "Ok"
62 : "Showing recent leave"
63 : Logging out
64 : "Successfully logged out"

# TEST STRATEGY:

## Testing Approach

### Unit Testing

Scope: Individual components/modules.
Responsibility: Developers.
Tools: JUnit, pytest.
Automation: High.

### Integration Testing

Scope: Interaction between integrated components.
Responsibility: QA team.
Tools: Selenium, Postman.
Automation: Medium.

### System Testing

Scope: Complete integrated system.
Responsibility: QA team.
Tools: JMeter, OWASP ZAP.
Automation: Medium to High.
Types: Functional, Performance, Security.

### Acceptance Testing

Scope: Validation against business requirements.
Responsibility: QA team and stakeholders.
Tools: Manual.
Automation: Low.

### Regression Testing

Scope: Re-testing after changes.
Responsibility: QA team.
Tools: Selenium, Jenkins.
Automation: High.

## TEST CASES

1. **User Story**: As a student, I want to submit a leave request online so that I can easily request time off for personal or academic reasons without having to fill out paper forms.

**Happy Path:**

- **Scenario:** Student applies for leave within the allowed time frame.

**Test Case:**

- **Input:** Student submits leave request with valid start and end dates.
- **Expected Output:** Leave request is successfully submitted and pending approval.

**Error Scenarios:**

- **Scenario:** Student applies for leave with invalid start or end date.

**Test Case:**

- **Input:** Student submits leave request with an end date earlier than the start date.
- **Expected Output:** System displays an error message prompting the user to enter valid dates.

2. **User story:** As an Admin, I want to approve or reject leave applications based on institution's guidelines so that leave is granted appropriately.

**Happy Path:**

- **Scenario:** Faculty approves leave request.

**Test Case:**

- **Input:** Faculty member reviews and approves a pending leave request.
- **Expected Output:** Leave request status changes to approved.

**Error Scenarios:**

- **Scenario:** Faculty member tries to approve an already approved leave request.

**Test Case:**

- **Input:** Faculty member attempts to approve a leave request that is already approved.

- **Expected Output:** System displays an error message indicating that the leave request has already been approved.

3. **User Story:** As a Student, I want to see the leave history of so that I can take leave accordingly

**Happy Path:**

- **Scenario:** Student views their leave history.

**Test Case:**

- **Input:** Student navigates to the leave history section.
- **Expected Output:** System displays a list of all past leave requests along with their statuses.

**Error Scenarios:**

- **Scenario:** Student tries to view leave history without any previous leave requests.

**Test Case:**

- **Input:** Student with no previous leave requests navigates to the leave history section.
- **Expected Output:** System displays a message indicating that the student has no past leave requests.

4. **User Story:** As a Student, I want to upload supporting documents for my leave application so that I can provide evidence for medical or other types of leave.

**Happy Path:**

- **Scenario:** Student successfully uploads a supporting document.

**Test Case:**

- **Input:** Student selects a file to upload as a supporting document for their leave request.
- **Expected Output:** System successfully uploads the document and associates it with the corresponding leave request.

**Error Scenarios:**

- **Scenario:** Student tries to upload a file exceeding the size limit.

**Test Case:**

- **Input:** Student attempts to upload a file larger than the allowed size limit.
- **Expected Output:** System displays an error message indicating that the file exceeds the size limit and prompts the student to upload a smaller file.

5. **User Story:** View Leave Application Status

**Happy Path:**

- **Scenario:** Student checks the status of a pending leave application.

**Test Case:**

- **Input:** Student navigates to the leave status page and selects a pending leave application.
- **Expected Output:** System displays the current status of the leave application as "Pending Approval".

**Error Scenarios:**

- **Scenario:** Student checks the status of a leave application that does not exist.

**Test Case:**

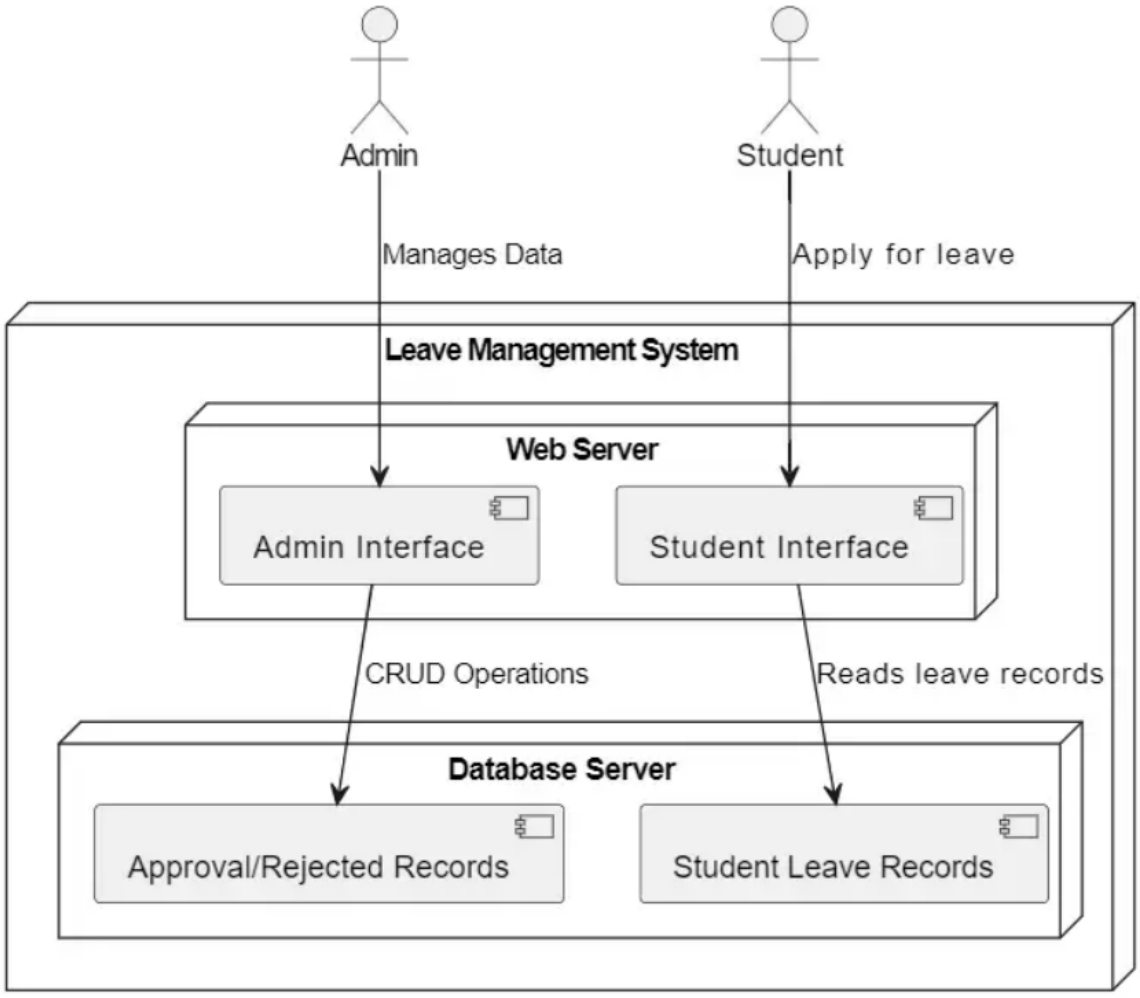
- **Input:** Student navigates to the leave status page and tries to select a non-existent leave application.
- **Expected Output:** System displays an error message indicating that the leave application does not exist.

**Test Environment**

Development Environment: For unit testing.

QA Environment: For integration, system, and regression testing.

UAT Environment: For user acceptance testing.

Production Environment: For final validation.

# DEPLOYMENT DIAGRAM