Tribhuvan University

Faculty of Humanities and Social Science

A Project Report on

"Smart College Leave Management System (SCLMS)"

Submitted to

Department of Computer Application

Academia International College

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by

Sandeep Maharjan

(6-2-346-33-2020)

Dev Aashish Gole

(6-2-346-19-2020)

Under the Supervision of

Ananda Adhikari

Nov 26, 2025

Tribhuvan University

Faculty of Humanities and Social Science

Academia Int'l College

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by "**Sandeep Maharjan"** and "**Dev Aashish Gole"** entitled "**Smart College Leave Management System** " in partial fulfillment of the requirements of the degree of Bachelor of Computer Application is recommended for the final evaluation.

…….

SIGNATURE

Ananda Adhikari

SUPERVISOR

Academia Int'l College

Gwarko, Lalitpur

Tribhuvan University

Faculty of Humanities and Social Science

Academia Int'l College

LETTER OF APPROVAL

This is to certify that this project is prepared by "**Sandeep Maharjan** " and " **Dev Aashish Gole** " entitled " **Smart College Leave Management System** " in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. **Mr. Ananda Adhikari** **Supervisor** **Academia International College** **Gwarko, Lalitpur**	…….. **Mr. Bishwas Mathema** **Coordinator** **Academia International College** **Gwarko, Lalitpur**
…….. **Internal Examiner**	…….. **External Examiner**

STUDENT'S DECLARATION

We hereby declare that we are the only author of this work and that no sources other than those listed have been used in this work.

…….	…….
Sandeep Maharjan	Dev Aashish Gole
Date:	Date:

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Academia International College** for providing us the opportunity to carry out this project titled **"Smart College Leave Management System (SCLMS)"** as part of the BCA 8th Semester curriculum under Tribhuvan University. This project has been an invaluable learning experience that allowed us to apply theoretical knowledge to solve a real-world challenge faced by academic institutions.

We extend our sincere appreciation to our project supervisor, **Mr. Ananda Adhikari** for their continuous guidance, constructive feedback, and encouragement throughout the project. Their expertise and insightful suggestions were instrumental in shaping the direction of this work.

We would also like to thank all faculty members of the Department of Computer Science and Information Technology for creating a supportive learning environment. Our special thanks go to our classmates, friends, and families for their constant motivation and cooperation during the research, development, and documentation phases of this project.

Sandeep Maharjan

TU Exam Roll No: 6-2-346-33-2020

Dev Aashish Gole

TU Exam Roll No: 6-2-346-19-2

ABSTRACT

The **Smart College Leave Management System (SCLMS)** is a modern, web-based application designed to digitize, automate, and optimize the leave management process within academic institutions. Traditional leave management in colleges relies heavily on handwritten applications, verbal communication, and manual approval workflows. Such outdated methods often result in delays, missing records, inconsistencies, administrative overload, and a complete lack of transparency. In many Nepali academic institutions, no centralized mechanism exists to track leave requests, verify eligibility, or maintain departmental leave statistics. This project addresses these challenges by proposing an intelligent, automated, and user-friendly leave management system.

SCLMS offers an integrated platform for students, teachers, and administrators to submit, review, approve, or reject leave applications efficiently. The system is developed using Laravel for backend logic and authentication, Vue.js/Flutter for responsive and interactive user interfaces, MySQL for structured data storage, and Python (Flask) for implementing machine learning modules such as anomaly detection and leave approval prediction. Agile

development methodology ensures iterative progress, continuous testing, and user-centered refinement throughout the project's lifecycle.

Keywords: Leave Management, Automation, Machine Learning, Laravel, MySQL, Academic System, Predictive Analytics.

Table of Content

LIST OF ABBREVIATIONS

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DBMS	Database Management System
HOD	Head of Department
HTML	HyperText Markup Language
LMS	Learning Management System
ML	Machine Learning
PHP	Hypertext Preprocessor
UI	User Interface

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Introduction

Academic institutions process a large number of leave applications every semester from both students and faculty across different departments. However, many colleges still rely on outdated manual methods such as handwritten forms, verbal communication, basic spreadsheets, and informal messaging. These traditional approaches often lead to missing documents, inconsistent approval decisions, delayed responses, and a complete absence of proper records or analytics. As the student population grows and academic responsibilities increase, these manual systems struggle to handle the rising workload, resulting in human errors, administrative delays, and disruptions in class planning and overall campus discipline.

The Smart College Leave Management System is designed to overcome these challenges by introducing a fully digital, automated, and intelligent platform tailored for academic environments. It streamlines the entire leave process from submission to multi-level approval through user-friendly dashboards for students, teachers, heads of departments, and administrators. Features such as automated notifications, real-time status tracking, and organized digital records enhance transparency and speed up decision-making. Additionally, advanced components like machine-learning-based approval prediction, fraud detection, department-wise leave load analysis, and smart conflict checks further strengthen the system's reliability. Built using modern technologies like Laravel for backend security, Flutter/Vue.js for responsive interfaces, MySQL for structured databases, and Python modules for predictive analytics, the system ensures scalability, accuracy, and efficient data-driven management across the institution.

1.2 Problem Statement

Traditional leave management systems in academic institutions suffer from a range of inefficiencies that significantly hinder smooth administrative operations. Most colleges still depend on paper-based processes, handwritten forms, or informal verbal communication, which leads to lost applications, delayed responses, and inconsistent record-keeping. Without a centralized digital platform, students and faculty often struggle to submit

requests properly, and administrators lack a unified system to view, verify, and process the increasing volume of leave submissions.

Another major limitation is the complete absence of real-time visibility and intelligent monitoring. Stakeholders cannot track live leave status, leading to confusion and repeated follow-ups. During crucial academic periods such as examinations or project evaluations, overlapping leaves become frequent due to the lack of automated schedules and conflict checks. Moreover, institutions using manual systems do not have mechanisms to identify suspicious leave patterns, detect anomalies, or generate insights from historical data. Teachers and HODs are further constrained because they lack proper tools to monitor departmental leave loads, trends, or potential shortages in staff availability.

1.3 Objectives

- To digitalize the entire leave process by enabling online submission, multi-level approvals, real-time notifications, secure centralized data storage, and a user-friendly responsive interface for all stakeholders.
- To improve decision-making and transparency through machine-learning support, detection of suspicious leave patterns, and generation of analytical reports with exportable summaries.

1.4 Scope and Limitation

1.4.1 Scopes:

Scopes are as follows:

- A web-based system with optional mobile access that provides role-specific dashboards, leave credit management, complete history tracking, real-time notifications, an interactive calendar view, and exportable reports for efficient leave handling across all user roles.
- Intelligent decision-support features including machine-learning-based approval prediction, fraud and anomaly detection, and department-level leave load analytics, helping administrators monitor patterns, prevent misuse, and manage staffing effectively.

1.4.2 Limitations:

Limitations are as follows:

- The system depends on internet connectivity and third-party notification services, so poor network conditions or external API issues can cause delays or temporary communication disruptions.
- Machine-learning accuracy may be limited in the early stages due to insufficient historical data, and the system currently does not include biometric integration or broader HRM features beyond leave management.

1.5 Report Organization

a) **Introduction:**
   This chapter gives an overview of the SCLMS, including the introduction, problem statement, objectives, scope, and system limitations. It explains why an automated leave management system is needed in academic institutions.

b) **Background Study and Literature Review:**
   This chapter reviews existing leave management systems and related studies. It highlights how current solutions work, their shortcomings, and how SCLMS improves upon them through automation and smart features.

c) **System Analysis and Design:**
   This chapter covers requirement analysis, feasibility study, system architecture, DFDs, ER diagrams, and the algorithms used in SCLMS. It explains how the system is planned and structured before development.

d) **Implementation and Testing:**
   This chapter describes the development process using Laravel, Vue.js/Flutter, and MySQL. It also includes module implementation, algorithm integration, and the testing methods used to verify system performance.

e) **Conclusion and Future Recommendations:**
   This chapter summarizes the project outcome and suggests future enhancements, such as biometric integration, improved ML models, and additional HRM features.

Chapter 2: Background Study and Literature Review

2.1 Background Study

Efficient leave management is an essential component of academic administration. Colleges must monitor student and faculty absences, maintain accurate records, ensure academic integrity, and avoid disruptions in teaching schedules. Traditionally, this process has relied on manual systems such as paper leave forms, verbal approvals, or basic spreadsheets. These approaches are prone to errors, delays, and lack of accountability.

In many Nepali academic institutions under Tribhuvan University and other boards, leave applications are submitted physically and moved through a chain of approvals—teacher, coordinator, and Head of Department (HOD). Misplacement of forms, delayed communication, difficulty in tracking leave status, and lack of historical data are recurring issues. Furthermore, the manual system offers no support for analytics, policy enforcement, or fraud detection.

With the rapid growth of digital platforms across industries, academic institutions are also transitioning toward automated solutions. However, existing platforms such as LMS tools (Moodle, Google Classroom) or ERP systems do not provide specialized leave management functionalities suited for academic workflows. These tools fail to consider academic constraints like semester schedules, examination periods, lecture conflicts, and attendance-based policies.

The **Smart College Leave Management System (SCLMS)** fills this gap by introducing an academic-specific, intelligent, and role-based leave management solution. Designed for Nepali educational structures, the system supports student–teacher–HOD workflows, smart analytics, automated approvals, and advanced monitoring mechanisms. By integrating machine learning for prediction and anomaly detection, SCLMS transforms leave administration from a manual, reactive process into a smart and proactive system.

2.2 Literature Review

Digital transformation in educational administration has influenced various areas such as student enrolment, examination systems, evaluation, and attendance monitoring. Leave

management, however, has remained largely untouched in many institutions. Several studies highlight the significance of digitization in improving transparency, reducing workload, and enhancing decision-making in academic administration.

Zhou et al. (2018) [1] discuss how digital platforms, when aligned with academic structures, significantly improve operational efficiency and accuracy. Their findings support the introduction of automated workflows in place of manual administrative tasks.

Patel and Kumar (2019) [2] identify that manual leave systems are highly prone to errors, lack historical data, and fail to provide real-time visibility to stakeholders. Their work emphasizes the necessity for automated solutions that can support structured approval flows and documentation.

Decision-support technologies have evolved from rule-based systems to data-driven models. Kaufman (2020) [3] discusses how machine learning algorithms like Logistic Regression, Random Forest, and Decision Trees enhance fairness and consistency in administrative decision-making.

Fraud and anomaly detection are crucial in any system involving approvals. Schneier (2021) [4] explains how anomaly detection algorithms such as Isolation Forest and statistical outliers can identify suspicious or recurring irregular patterns, such as frequent weekend-adjacent leaves.

Enterprise ERP systems often include leave modules, but Smith (2022) [5] argues that such tools are not suited for academic hierarchies. They focus primarily on employee workflows, which do not match academic structures where student–teacher–HOD relationships, semester cycles, and academic calendars must be considered.

Recent studies by Chen et al. (2023) [6] emphasize the importance of intelligent leave management systems that integrate predictive analytics, fraud detection, and real-time tracking.

The proposed **Smart College Leave Management System (SCLMS)** integrates these insights, offering a hybrid of automation, data-driven decision support, academic workflow alignment, and intelligent analytics. It represents an innovative approach tailored to the unique needs of colleges in Nepal and similar educational environments.

Chapter 3: System Analysis and Design

3.1 System Analysis

System analysis focuses on understanding how the current leave management process works, identifying its problems, and defining a solution that can address those issues effectively. In many academic institutions, leave applications are still handled manually. Students write letters by hand, submit them physically, and wait for teachers to review and forward them to the head of department. Approvals are often communicated informally, and there is no centralized system to store decisions or track the complete leave history of students and faculty. This manual process leads to slow approvals, inconsistent decisions, poor transparency, difficulty retrieving past records, and a high chance of misplaced information. As the number of students and faculty continues to grow, these problems become even more difficult to manage.

The proposed Smart College Leave Management System aims to solve these issues by introducing a structured and digital platform that clearly defines the roles of students, teachers, and administrators. It allows online submission of leave requests, faster review, safe record storage, and real-time updates, making the entire process more efficient and reliable.

The system also includes useful features such as approval prediction, anomaly detection, automatic conflict checking, and complete leave history tracking. These tools support better accuracy, faster processing, and more consistent decisions across the institution.

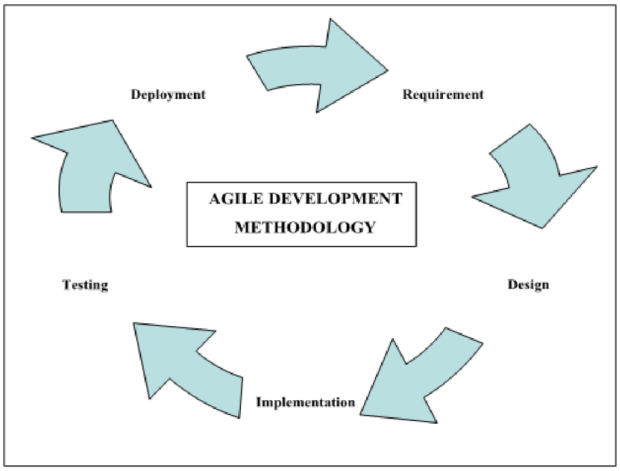


Figure 3.1: Agile Methodology

3.1.1 Requirement and Analysis

Requirement analysis is a critical step for determining the success of a system or software project. Requirement analysis is of two types:

- Functional Requirements
- Non-Functional Requirements.

i. Functional Requirement

This section provides the requirement overview of the system. Different functional requirement of the system has been listed below:

- The system must allow secure login with role-based access for Students, Teachers, HODs, and Admins.
- Users should be able to submit leave requests with leave type, dates, reason, and document uploads.
- Leave approval must follow a multi-level workflow: Student → Teacher → HOD, and Teacher → HOD.
- The system should manage leave credits, showing available, used, and remaining leaves.
- Real-time notifications (email or system alerts) must be sent for submission, approval, or rejection.
- The system must include conflict detection (recent leave count, blackout overlap, departmental collision).
- Smart approval prediction using machine learning should assist decision-making.
- Fraud/anomaly detection should identify suspicious leave behaviour.
- A calendar view should display approved leaves and overlapping schedules.
- The system must generate and export leave reports in PDF/Excel formats.
- Admin should be able to manage users, leave types, and system settings.

Use Case Diagram

The Use Case Diagram for the SCLM illustrates the main actions available to both Users and Admins within the system

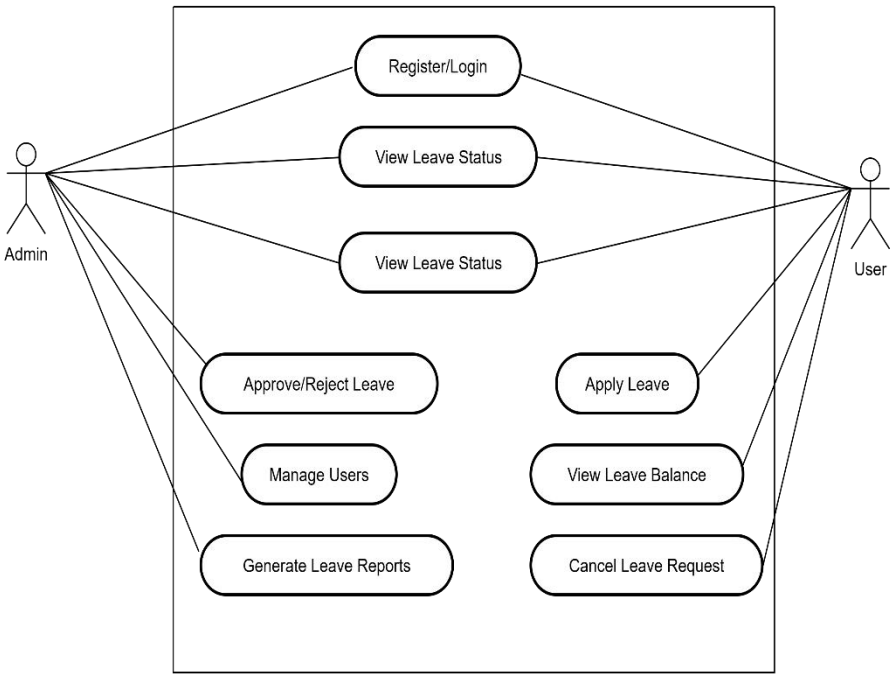


Figure 3.2: Use Case Diagram

ii. Non-Functional Requirement

Different non-functional requirement of the system has been listed below:

- The system should load dashboards and process most queries within 2 seconds to ensure fast performance.
- The system must enforce authentication, authorization, data encryption, and secure APIs to protect all user data.

  The system should be scalable, supporting multiple departments, semesters, and an increasing number of users without performance issues.
- The system must provide an intuitive, user-friendly interface with full mobile responsiveness.
- The system should maintain high reliability with minimal downtime and secure, consistent data backups.
- The system should follow a modular code structure to support easy updates, improvements, and future enhancements

- The system must be compatible with all major browsers and devices for seamless access.
- The system should support easy maintenance, allowing quick bug fixes and performance optimizations.

3.1.2 Feasibility Analysis

3.1.2.1 Technical Feasibility

The technical feasibility determines whether the required technologies are available and suitable for developing the system. The SCLMS uses open-source and reliable tools such as Laravel (PHP) for backend security and authentication, Vue.js / Flutter for a responsive interface, MySQL for structured data storage, and Python Flask for machine-learning integration. These technologies are widely supported, cost-effective, and easy to maintain. Therefore, the system is technically feasible with the resources available.

3.1.2.2 Operational Feasibility

Operational feasibility assesses how well the system functions in a real environment. The SCLMS is designed with a simple and intuitive interface, allowing students, teachers, and staff to use it with minimal training. The automated leave workflow matches existing academic processes, reduces paperwork, and improves accuracy and efficiency. Thus, the system is operationally feasible and easy for users to adopt.

3.1.2.3 Economic Feasibility

Economic feasibility examines the cost-effectiveness of the project. Since SCLMS is built using free and open-source tools like Laravel, MySQL, Vue.js, and Python, no licensing fees or expensive hardware are required. The system also reduces long-term operational costs by eliminating manual work and paperwork. Therefore, the project is economically feasible and offers significant long-term savings.

3.1.2.4 Schedule Feasibility

Schedule feasibility determines whether the project can be completed within the planned timeframe. The development tasks planning, design, coding, testing, and documentation are clearly divided and aligned with the academic schedule. With proper time management, the project can be completed on time, making it schedule feasible.

3.1.3 Gantt Chart

The Gantt chart visually presents the timeline of each phase and helps track progress throughout the project.

Table 3.1: Time Schedule

Task Name	Start Date	End Date	Duration
Requirement Analysis & System Design	1-Jun	30-Jun-25	28
Documentation	1-Jun	20-Nov-25	175
Database & Back-end Development	1-Jun	15-Aug-25	42
Front-end Development	15-Jul	31-Aug-25	42
Integration & Testing	1-Oct	31-Oct-25	28

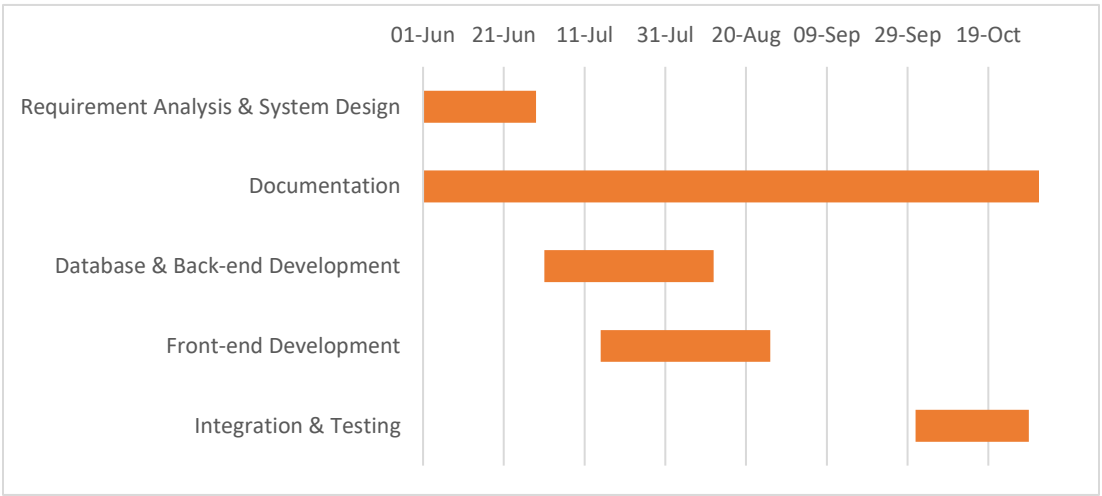


Figure 3.3: Gantt Chart

3.1.4 Object Modelling using Class and Object Diagram

Object Modelling using Class Diagram

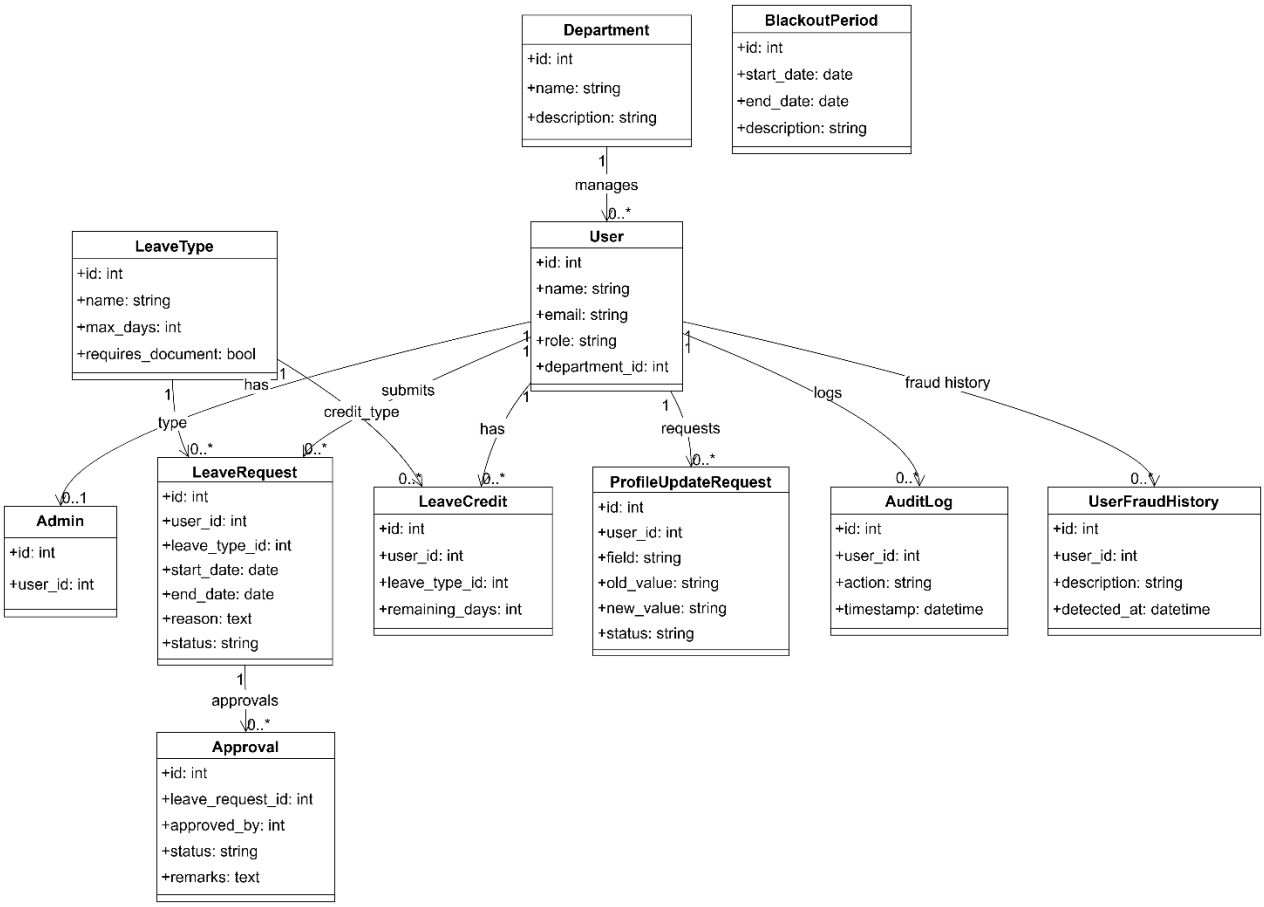


Figure 3.4: Object Modelling using Class Diagram

The object modelling of the Smart College Leave Management System identifies the key entities involved in handling leave activities and explains how they interact. The main entity is the User, which includes students, teachers, and administrators who use the system. Users can apply for leave, track their leave balance, update their profile information, and generate activity records. The LeaveRequest entity stores details of each leave application, such as dates, leave type, reason, and approval status. The LeaveType entity defines the rules and limits for different kinds of leave. The Approval entity records decisions made by administrators when reviewing leave requests. Additional entities such as LeaveCredit, BlackoutPeriod, AuditLog, and UserFraudHistory help maintain accuracy, security, and proper record-keeping in the system. Together, these objects create a clear structure that shows how information

moves through the system and helps ensure that the leave process is simple, transparent, and well organized.

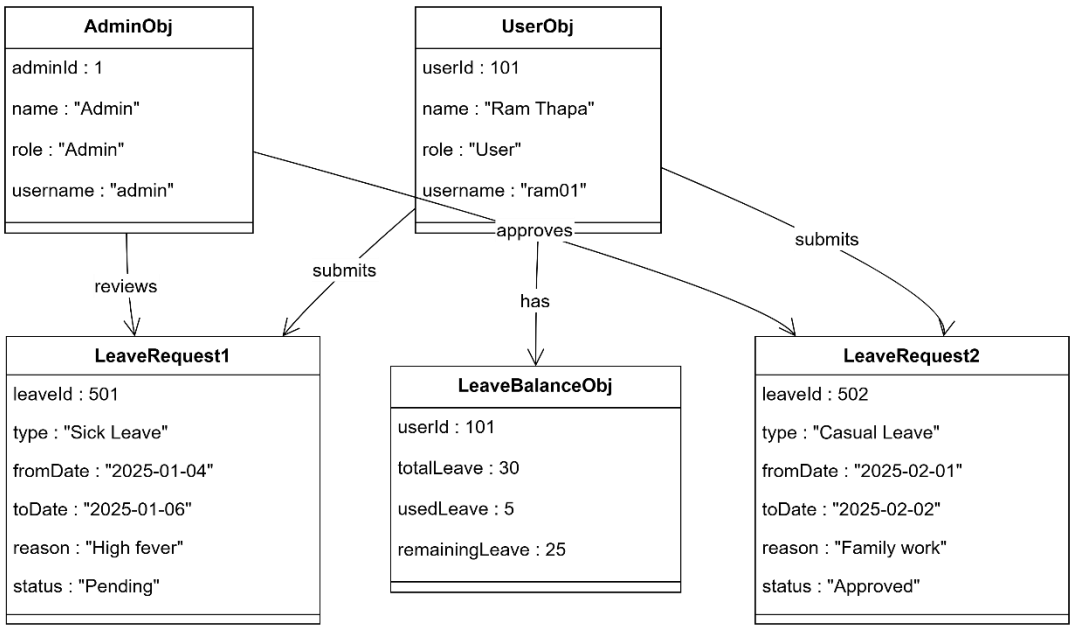


Figure 3.5: Object Modelling Using Object Diagram

The object diagram shows how actual instances of the system work together during runtime. It includes sample objects such as a user, an admin, leave requests, and a leave balance. The user object represents a student or teacher who interacts with the system. This user submits leave request objects, each containing details like leave type, dates, reason, and status. The admin object represents the person who checks and processes these leave requests. The admin is linked to the same leave request objects because they review or approve them. A leave balance object is also connected to the user, showing the user's total, used, and remaining leave. Overall, the diagram displays real examples of data in the system and how the user and admin interact with leave requests, helping to explain the flow of actions in a simple and clear way.

3.1.5 Dynamic Modeling using State and Sequence Diagram

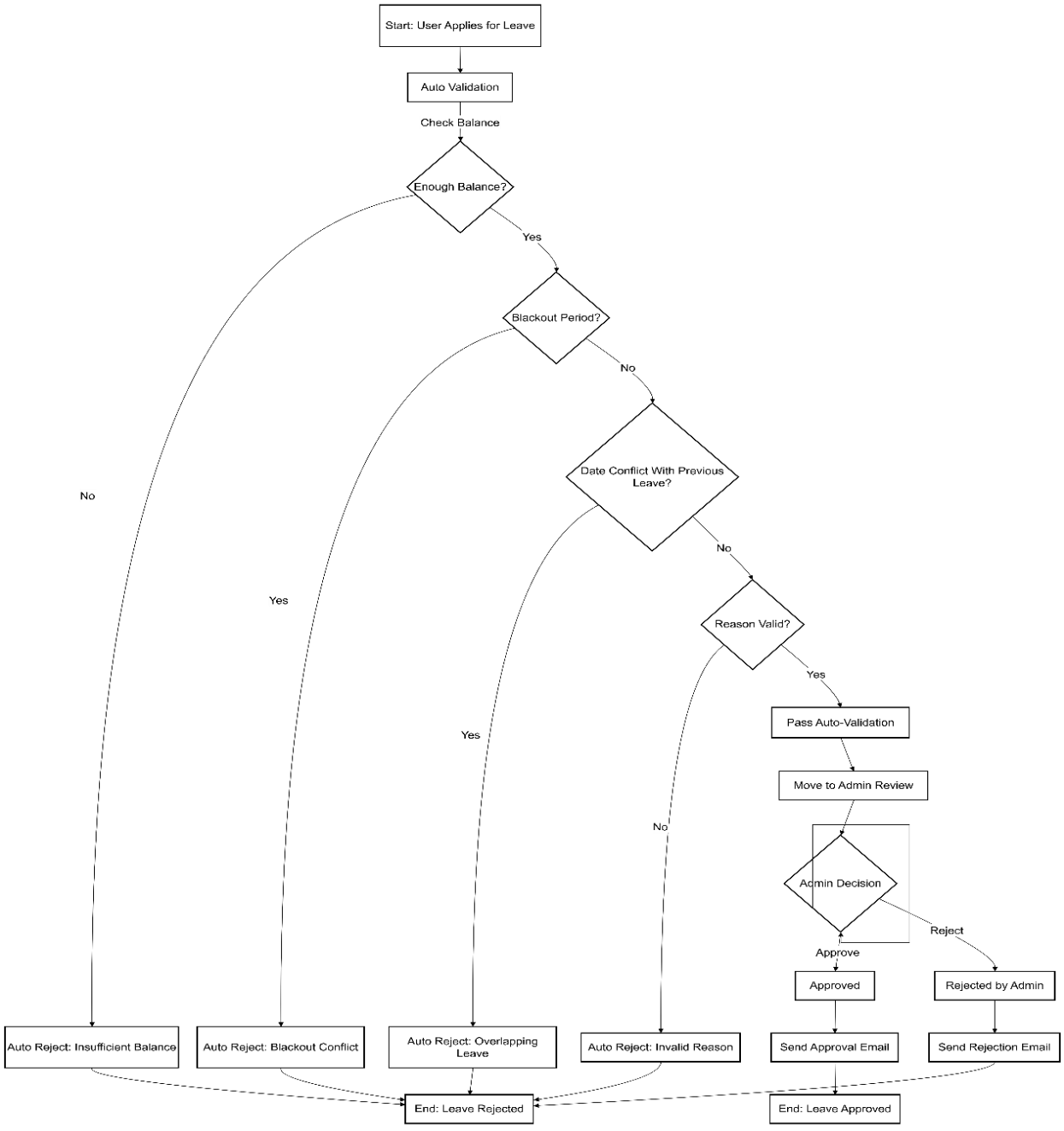


Figure 3.6: Dynamic Modeling using State Diagram

The system first performs automatic checks: balance, conflicts, dates, and reason. If any fail, the leave is auto-rejected. If all pass, the request enters the manual flow. The admin verifies details, approves or rejects, and updates status. The user receives the final decision through notification or email.

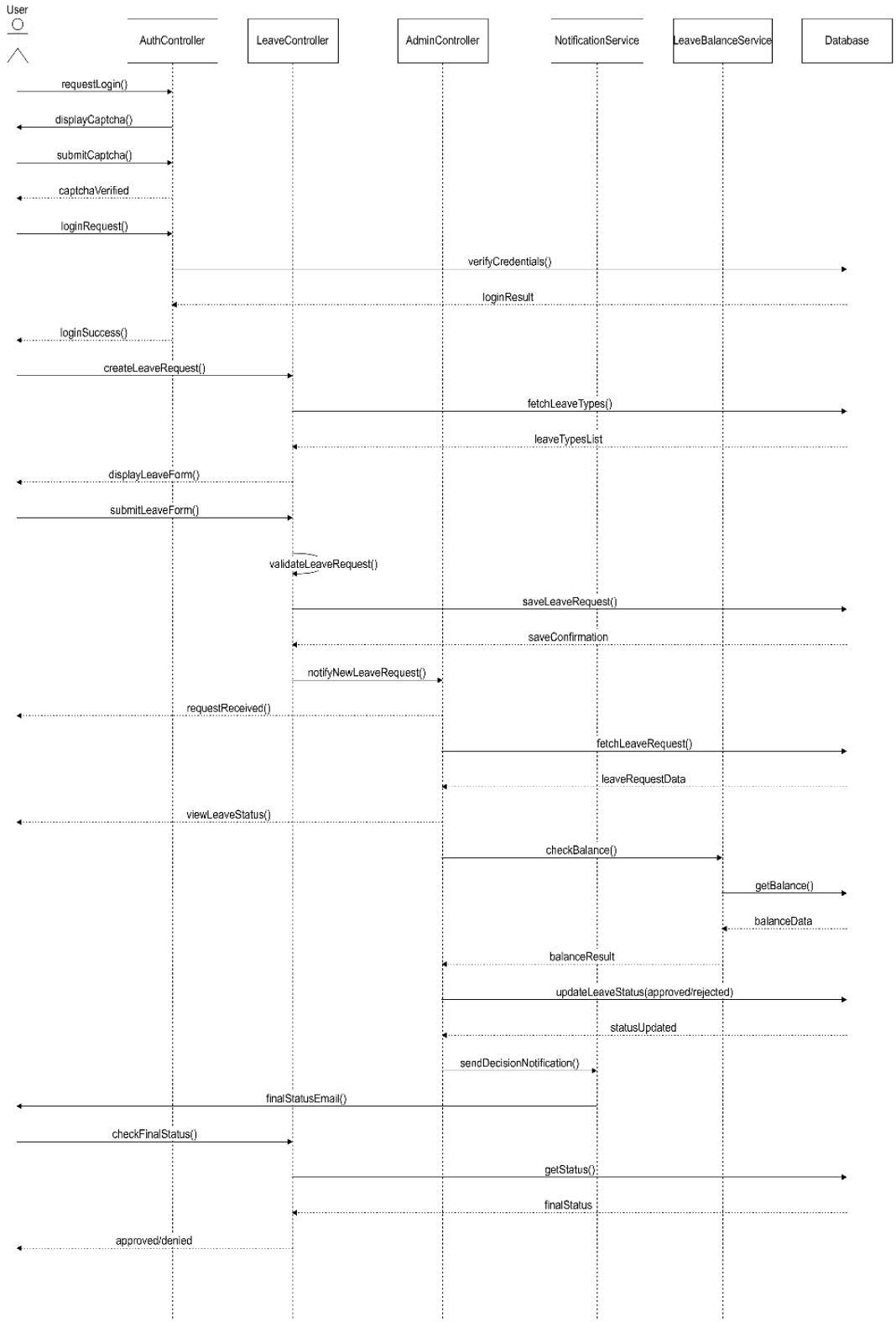


Figure 3.7: Dynamic Modeling using Sequence Diagram

The sequence diagram explains how the user, system, and admin interact step by step during the leave request process. It begins when the user opens the leave form, and the system responds by displaying the required fields. After the user fills in the details and submits the request, the system validates the information and stores the request with a pending status. This marks the end of the user's initial interaction. The admin then accesses the system to view all pending leave requests. The system presents the list, allowing the admin to review the details of each request. Based on the information provided, the admin either approves or rejects the request. Once the decision is made, the system updates the request status accordingly. Finally, the system sends a notification back to the user, informing them about the admin's decision. This diagram shows the full communication flow and highlights how both the user and the admin rely on the system to exchange information and complete the leave management process

3.1.6 Process Modeling using Activity Diagram

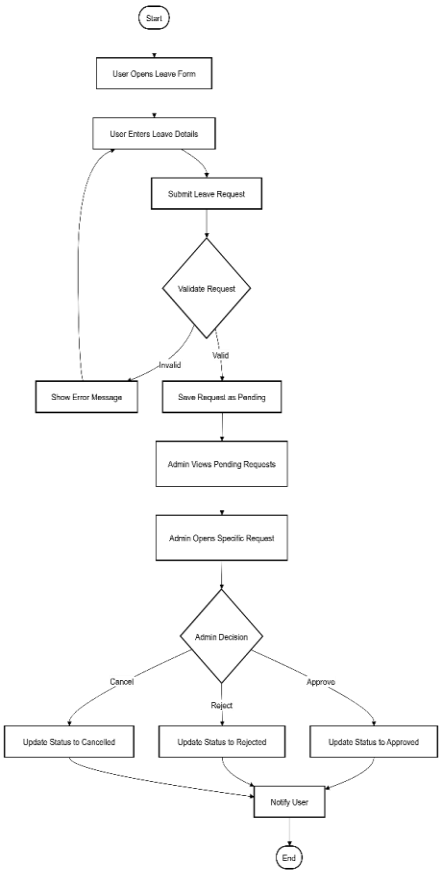


Figure 3.8: Process Modeling using Activity Diagram

The activity diagram shows the full workflow of a leave request. It begins when the user opens the form, fills in the details, and submits the request. The system validates the input; if the data is invalid, the user is asked to correct it. If valid, the system saves the request as pending. The admin then views all pending requests and selects one for review. The admin may approve, reject, or cancel the request. After the decision, the system updates the request status and sends a notification to the user. The process then ends. This diagram clearly represents the flow of actions, decision points, and responsibilities of both the user and the admin within the system.

3.2 System Design

3.2.1. Refinement of Class, Object, State and Sequence Activity Diagrams

3.2.1.1. Refinement of Class Diagram

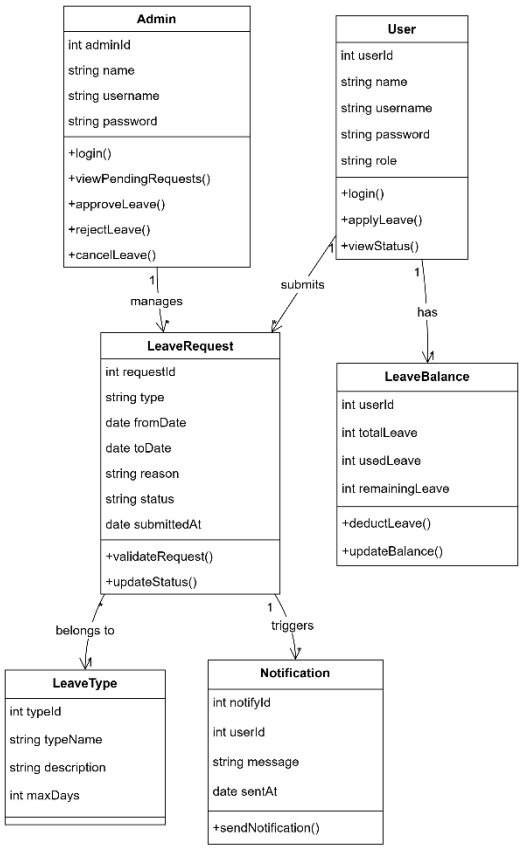


Figure 3.9: Refinement of Class Diagram

This refined class diagram includes more detailed attributes and methods for each class, making it closer to the real system. The User and Admin classes now include login and action methods. LeaveRequest is expanded with validation and status update operations. Additional supporting classes like LeaveType, LeaveBalance, and Notification show the full workflow of leave management. These refinements make the class diagram more realistic and aligned with the actual system behavior.

3.2.1.2. Refinement of Object Diagram

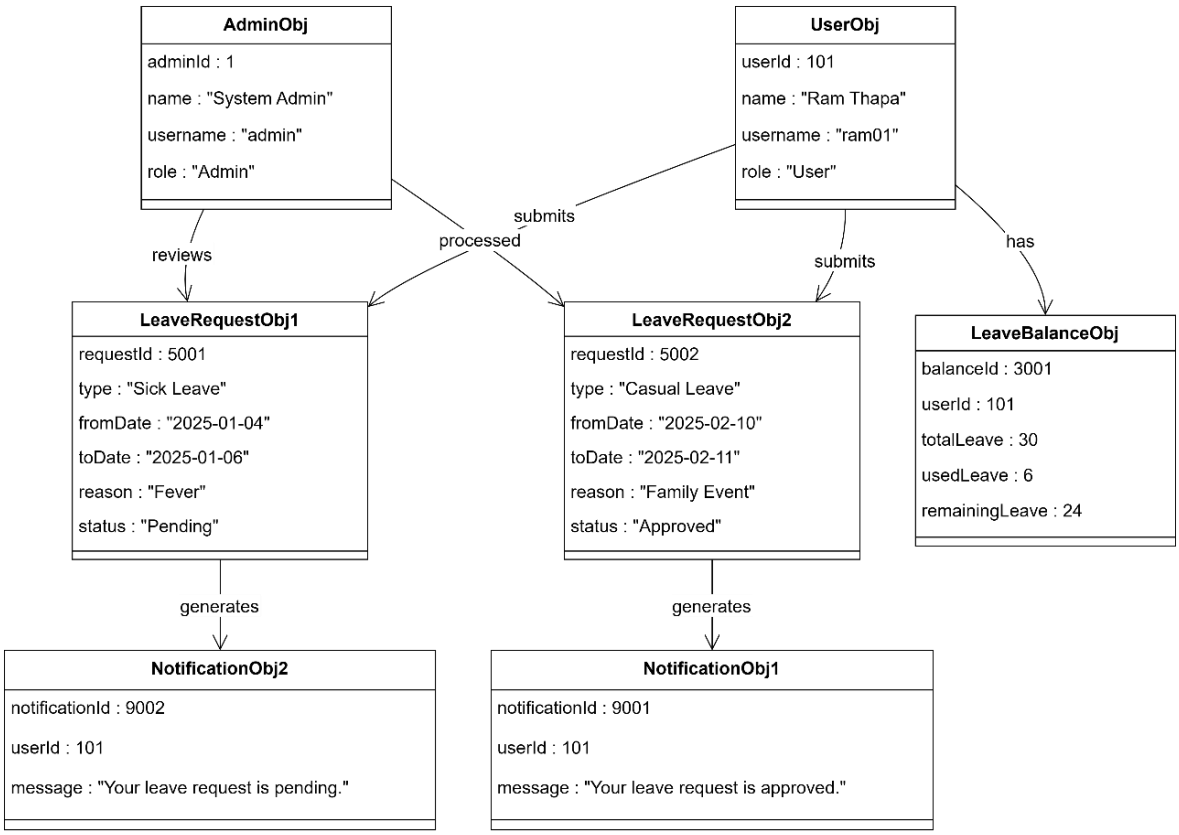


Figure 3.10: Refinement of Object Diagram

The refined object diagram shows concrete runtime objects that exist during the leave management process. It includes a user, an admin, two leave requests, a leave balance object, and related notification objects. Each object contains sample values that reflect real data. The user is linked to the leave requests they submitted and to their leave balance. The admin interacts with the same leave request objects when reviewing or approving them. Each leave

request produces a notification object that is delivered to the user. This refined diagram clearly shows how actual data instances are connected and how the system behaves during operation.

3.2.1.3. Refinement of State Diagram

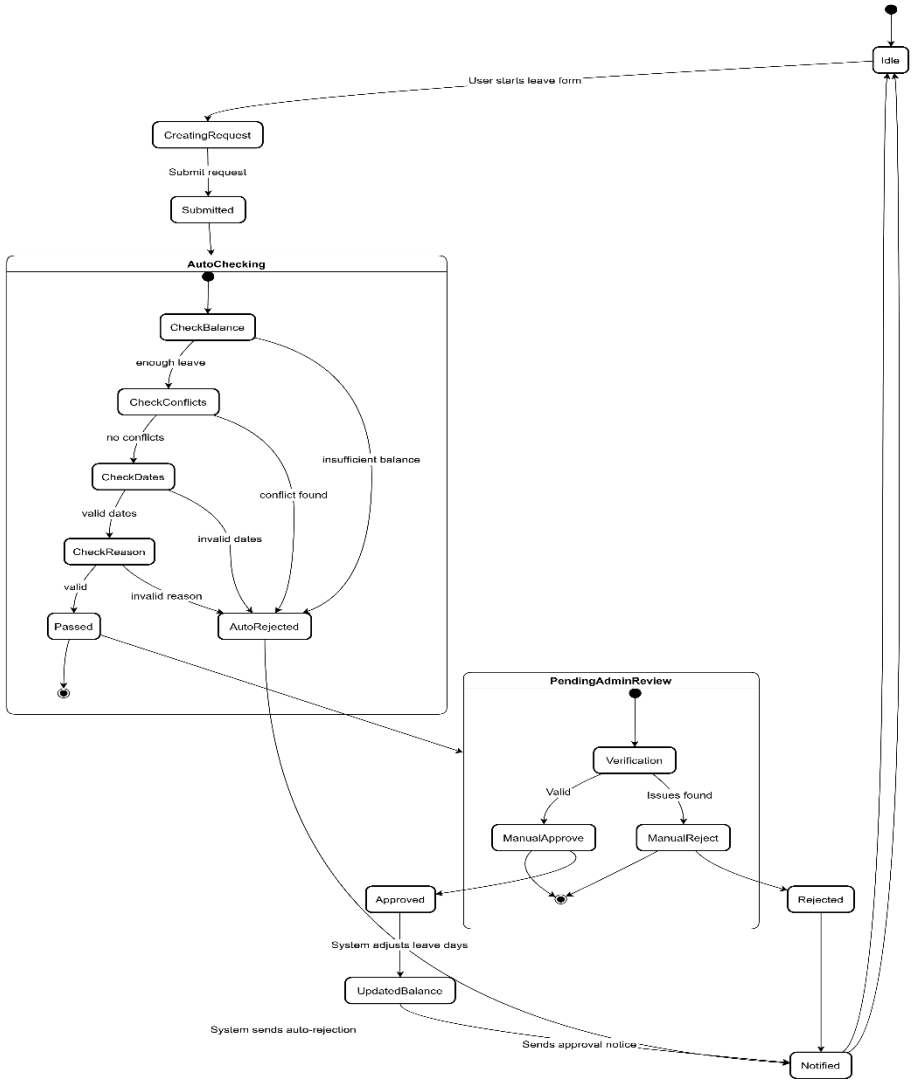


Figure 3.11: Refinement of State Diagram

The refined diagram expands the main leave-processing states. Auto-check is broken into balance, conflict, date, and reason checks. Admin review is also detailed with manual verification steps. These additions show *how* each decision happens while keeping the same overall flow of request, review, approval, or rejection.

3.2.1.4. Refinement of Sequence Diagram

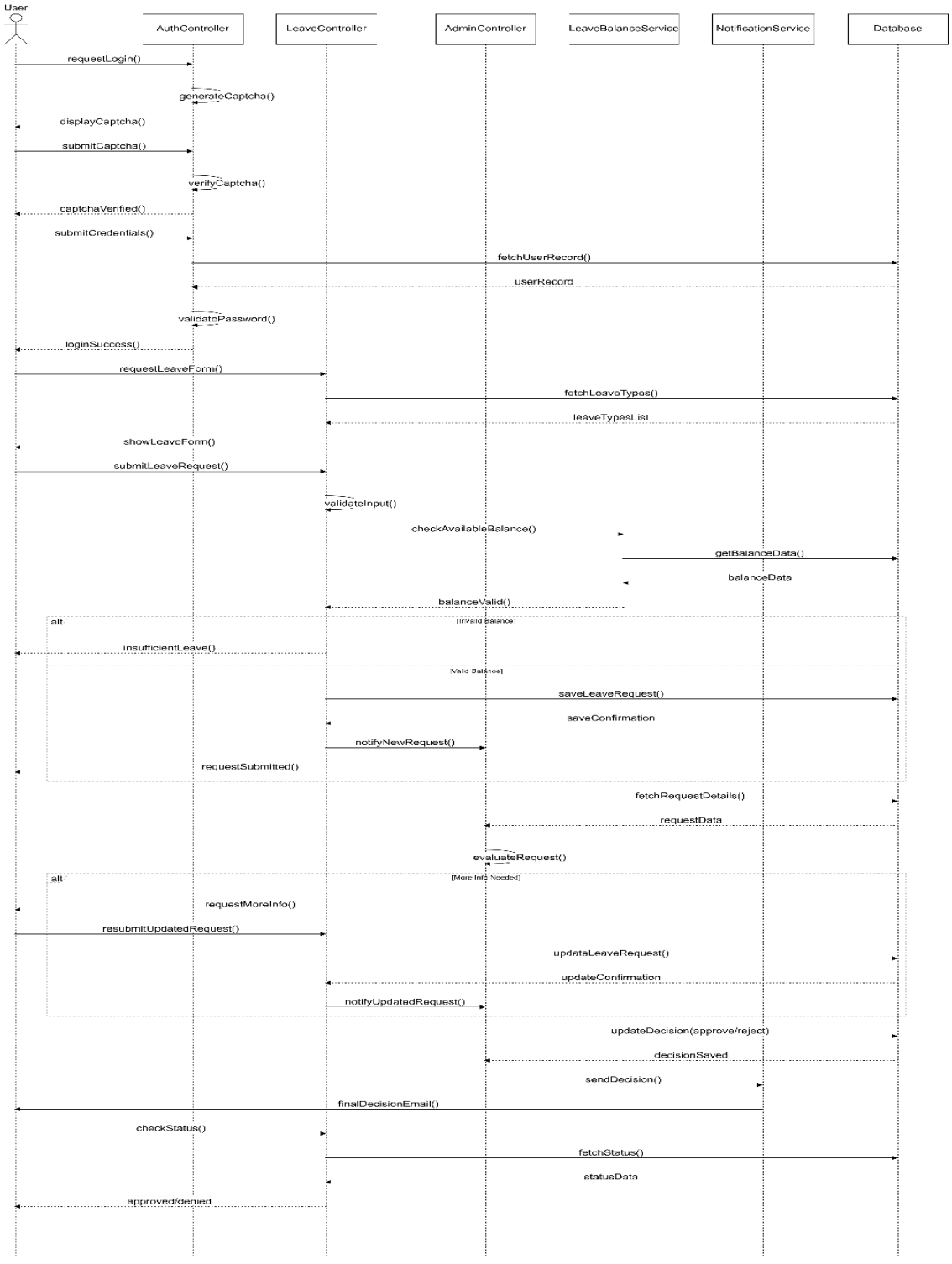


Figure 3.12: Refinement of Sequence Diagram

The refined sequence diagram shows a clear flow of how a leave request is processed in the system. It starts when the user logs in, creates a leave request, and submits it. The system validates the request and notifies the admin. The admin reviews the details and may approve, reject, or return the request for more information. If edits are needed, the user updates and resubmits the request, and the admin reviews it again. Once a final decision is made, the system updates the leave status and balance and sends a notification back to the user. This refined version includes optional steps, system validations, and internal updates, giving a more realistic and complete representation of the leave workflow.

3.2.1.5. Refinement of Activity diagram

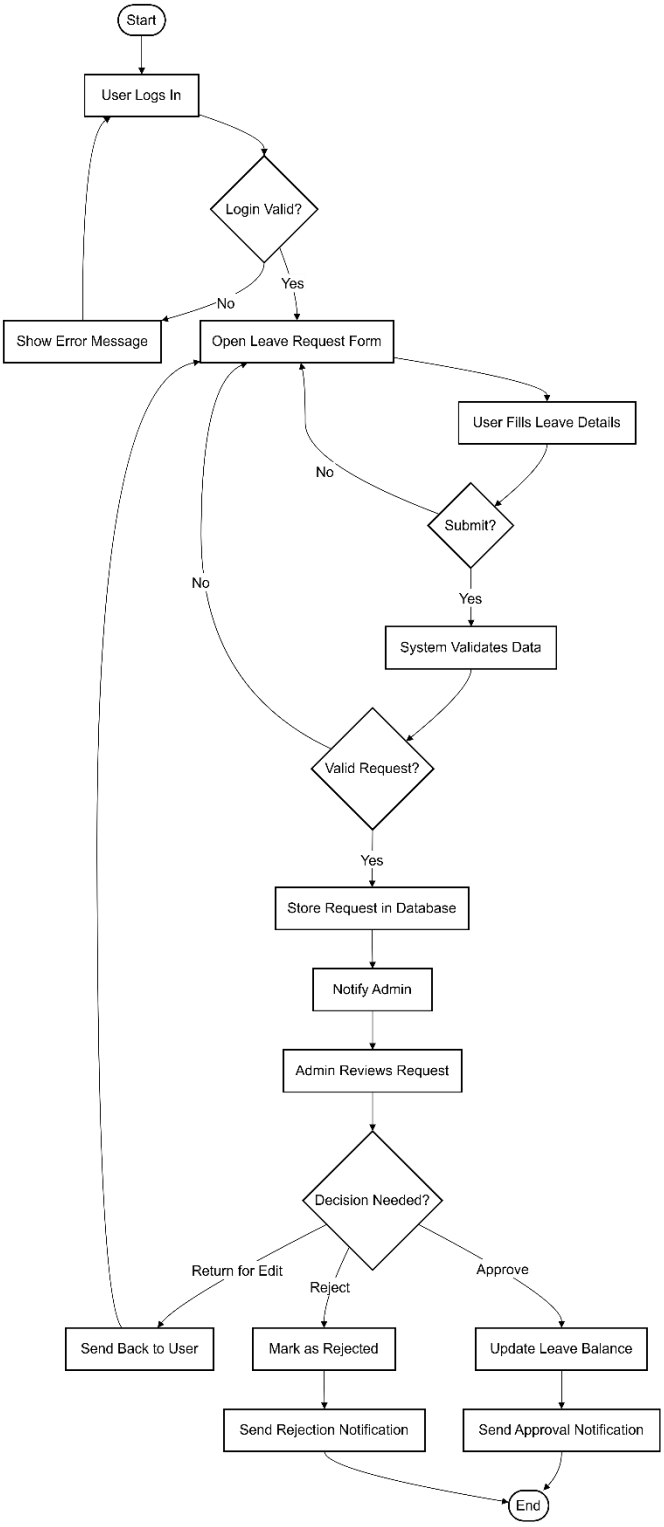


Figure 3.13: Refinement of Activity Diagram

The refined activity diagram shows a clearer flow of how a leave request is handled in the system. It starts with the user logging in, filling the leave form, and submitting it. The system checks the details and forwards the valid request to the admin. The admin reviews it and can approve, reject, or return it for editing. If returned, the user updates and resubmits the request. Approved requests update the leave balance, while rejected ones are marked as final. In both cases, the user receives a notification. This refined version highlights validation steps, decision points, and system actions in a clean and simple way.

3.2.2. Component Diagram

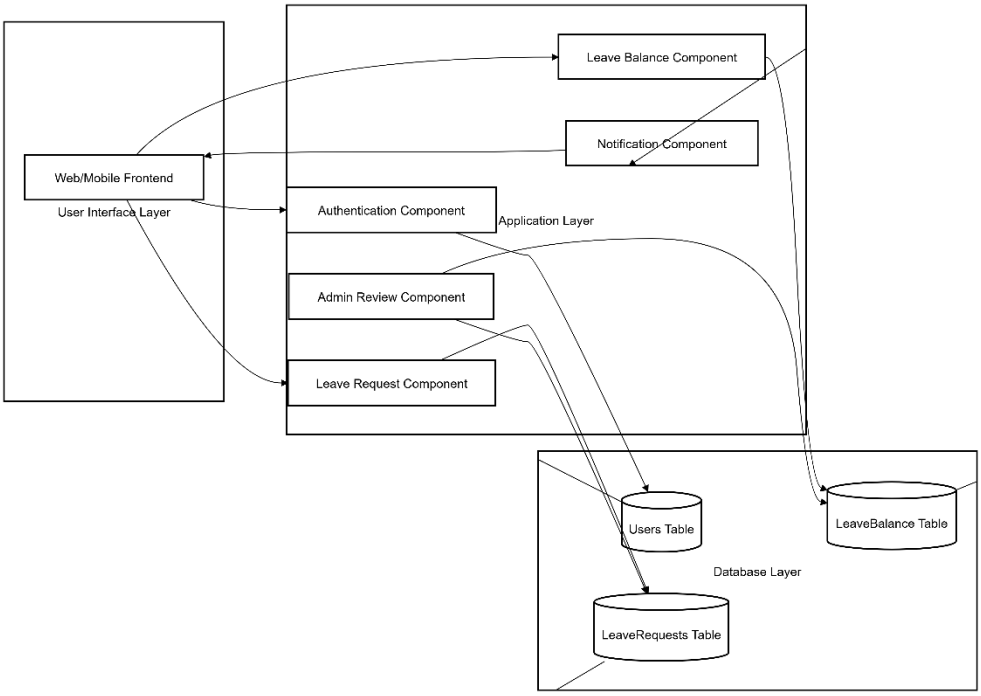


Figure 3.14: Component Diagram

The component diagram shows the main building blocks of the leave management system. The user interacts with the frontend, which connects to several application components such as authentication, leave request handling, leave balance management, admin review, and notification services. These components store and retrieve data from the database tables for users, leave requests, and leave balances. The notification component sends updates back to

the user. This diagram highlights how each major part of the system works together in a clear and structured way.

3.2.3. Deployment Diagram

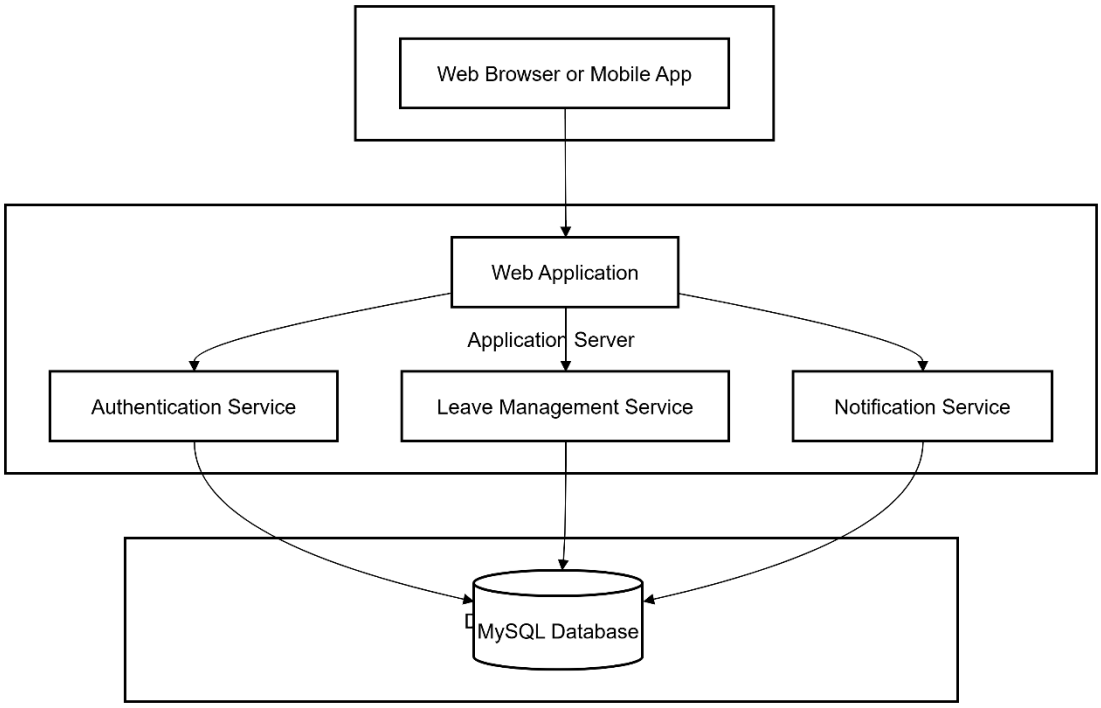


Figure 3.15: Deployment Diagram

The deployment diagram shows how the system runs across different hardware and server environments. Users access the system through a web browser or mobile app on their device. Their requests are processed by the application server, which hosts the main web application along with components for authentication, leave management, and notifications. All data such as users, leave requests, and leave balances are stored in a MySQL database on a separate database server. This setup provides a clear view of how the system's software components are deployed across client, server, and database layers.

Chapter 4: Implementation and Testing

4.1 Implementation

The Smart College Leave Management System (SCLMS) was implemented as a fully functional web-based application using modern, open-source technologies. The system integrates a combination of front-end, back-end, and database layers to ensure smooth performance, clear workflow management, and secure leave data handling.

The implementation focused on developing a user-friendly platform that supports multiple roles students, teachers, and department heads while maintaining data integrity and efficient automated leave workflows.

Each module was designed systematically, ensuring that the leave application process, approvals, document handling, machine learning predictions, and reporting features work seamlessly throughout the system.

4.1.1 Tools Used (CASE tools, Programming languages, Database platforms)

Tools used for the development of this project are as follows:

Front-End

- HTML

  HTML was used to structure all web pages, defining layouts for login pages, leave forms, dashboards, calendars, and approval sections. Buttons, tables, form fields, and navigation elements were created using HTML, making it the foundation of the system's interface.

- CSS

  CSS was used to style and enhance the visual appearance of the system. It ensured consistency in colors, spacing, fonts, and responsive design. CSS made the pages look modern and professional, ensuring ease of use for all users.

- JavaScript

  JavaScript was used to add dynamic behavior to the system. It enabled features like form validation, dynamic dashboard updates, calendar rendering, notifications, and asynchronous API calls to fetch leave data without reloading pages. This improved user experience significantly.

- Blade

  Blade is Laravel's built-in templating engine used to create dynamic web pages. It allows combining HTML with simple tags for loops, conditions, and data display. Blade makes it easy to build reusable layouts and keeps the front-end clean and organized. In this project, Blade was used for designing pages like login, leave forms, dashboards, and approval views.

Back-End

- Laravel

  Laravel acted as the primary server-side framework. It handled user authentication, business logic, API routing, form processing, and communication with the database. Laravel's built-in security features ensured encrypted passwords, protected routes, and secure session management.

Database

- MySQL

  MySQL served as the central database for storing users, leave requests, approval logs, notifications, departments, and ML flags. It offered fast data retrieval, secure storage, and relational structure to maintain data integrity.

Development Tools

- VS Code

  VS Code was used as the main code editor, providing extensions for PHP, HTML, CSS and JavaScript. It helped manage source code efficiently and improve development workflow.

4.1.2 Implementation Details of Modules (Description of procedures/functions)

Admin Module:

The Admin Module serves as the main control center of the Smart College Leave Management System. The administrator manages students, teachers, departments, and leave categories. This module allows the admin to add or update user accounts, define leave rules, and configure approval hierarchies.

The admin also monitors all leave activities across the system and can view department-wise reports, suspicious leave patterns, and system notifications. It ensures that overall operations remain organized, secure, and aligned with college policies.

Student Module

The User Module allows users to submit leave applications digitally. Users can enter their leave details, attach documents, and choose the leave type. The system automatically checks for overlapping dates or invalid entries before submitting the request. Users can track the progress of their leave in real time and view their full leave history. This module provides a simple and convenient way for users to communicate their absence without paperwork.

HOD/Approver Module

The HOD Module provides the final approval authority for both student and teacher leave requests. HODs view all relevant information, including ML-generated predictions and flagged suspicious patterns.

They can approve, reject, or return applications with comments. The module also includes department-level calendars and summaries, helping HODs manage academic load effectively.

Notification Module

This module manages all alerts and notifications, mails in the system. When a leave request is submitted, approved, or rejected, the related user gets an update. Notifications show on

dashboards and can be sent by email. This supports communication and reduces manual follow up.

Calendar Module

The Calendar Module visually displays approved and pending leaves. Students can see their personal leave record, while teachers and HODs can view department-wide leave schedules. It helps prevent overlapping leaves and supports smooth academic planning

Machine Learning Module

The ML module enhances system intelligence through:

1. Approval Prediction: Analyzes past patterns to estimate the likelihood of approval.
2. Anomaly Detection: Identifies suspicious leave patterns such as frequent weekend leaves or repeated short absences.

These insights support fairer decision-making and strengthen administrative supervision.

4.2 Algorithm Details

To make the Smart College Leave Management System intelligent and reliable, a set of algorithms was implemented. These algorithms help improve decision-making, detect unusual leave patterns, and support administrators with automated insights. The algorithms are integrated through a Python-based ML service, while the main system runs on Laravel.

Description of Algorithm:

A. Heuristic Rule-Based Fraud Detection

To safeguard fairness and prevent misuse, the fraud detection engine continuously inspects behavioral anomalies. The algorithm examines patterns such as repeated Friday or Monday leave submissions intended to extend weekends, excessive emergency leaves within a sixty-day period, unusually long leave durations compared to personal history, high frequency of recent leaves, and suspicious documents previously flagged by administrators. Each suspicious characteristic increases the fraud

score. This score accumulates over time and provides a transparent indicator of behavioral irregularities. By applying bounded lookback windows and statistical comparisons, the module efficiently identifies high-risk activity without affecting system performance**.**

B. Constraint Satisfaction via Date-Range Sweep

This algorithm checks how many people from the same department, role, and semester are already on approved leave during the dates requested by a user. It does this by scanning each day in the requested range and counting the number of overlapping approved leaves for that date. As it moves through each day, it keeps track of the highest number of individuals who are already absent. After completing the scan, it compares this maximum value with a predefined threshold teachers are allowed only one concurrent absence, while students can have up to three. The ratio of the actual absences to this threshold helps the system evaluate how risky or disruptive the requested leave would be. This design ensures predictable execution and accurately identifies leave conflicts across the selected date range.

C. Deterministic Penalty State Automaton

The penalty engine operates as a deterministic finite-state model responsible for administering corrective actions based on the accumulated fraud score. The system transitions through increasingly severe states as the score rises. A score of five triggers a formal warning. A score of ten forces all future leave requests into mandatory manual review. A score of fifteen reduces the user's available leave credits. Upon reaching a score of twenty, the system restricts the user from applying for any leave for the next thirty days. Because this state machine uses fixed thresholds and predefined transitions, its execution is constant-time and guarantees consistent administrative behavior.

4.3 Testing

4.2.1 Test Cases for Unit Testing

Unit testing focuses on verifying the smallest, independent components of the Leave Management System to ensure each part performs correctly on its own. In this system,

these units include login and authentication functions, leave application handlers, database interaction modules for storing and updating leave records, and the logic for automatic leave balance validation. By testing each component individually, issues can be detected early before the modules are combined into the complete workflow. This ensures reliability, confirms that core features such as user login, leave submission, admin approval, and status updates function as expected, and creates a stable foundation for integration and full system testing.

Table 4.1: Test Result of Login

ID	**Test Case Description**	**Test Data**	**Expected Result**	**Actual Result**	**Pass/Fail**
T1	User Login	Username:*ram01*, Password: *ram123*	Login successful, redirect to dashboard	Works as expected	Pass
T2	User Login (Invalid)	Username: *ram01*, Password: *wrong*	Show "Invalid Credentials"	Message displayed	Pass

Table 4.2: Test Result of apply Leave

ID	**Test Case Description**	**Test Data**	**Expected Result**	**Actual Result**	**Pass/Fail**
T3	Apply Leave	Type: Sick, From: 2025-01-05, To: 2025-01-07, Reason: Fever	Leave request stored	Record saved correctly	Pass
T4	Apply Leave (Missing Reason)	Reason: *empty*	Form validation error	"Reason Required" shown	Pass

Table 4.3: Test result of Auto Check

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
T5	Auto Check Leave Balance	UserID:*101*, Balance:*10*, Requested: *3*	Auto-check passes	Request forwarded to admin	Pass

Table 4.4: Test Result of Admin Approve

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
T6	Admin Approves Leave	LeaveID:*501*	Status becomes "Approved"	Status updated	Pass
T7	Admin Rejects Leave	LeaveID:*502*	Status becomes "Rejected"	Update successful	Pass

Table 4.5: Test Result of view leave

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
T8	View Leave Status	LeaveID:*501*	Show: Approved/Rejected/Pending	Correct status displayed	Pass

Table 4.6: Test Result of Cancel pending Leave Request

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
T8	Cancel Leave Request	LeaveID:*503 (Pending)*	Status = "Cancelled	Cancel successful	Pass

Table 4.7: Test Result of Update user Profile

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
T8	Update Profile	Name: *Ram Thapa (updated)*	Profile saved	Updated info stored	Pass

Chapter 5: Conclusion and Future Recommendation

5.1 Lesson learnt / Outcome

Developing the Smart College Leave Management System (SCLMS) has been an important learning journey that strengthened both my technical and analytical skills. Working on this project allowed me to translate theoretical concepts into a practical system that addresses a real-world challenge faced by educational institutions. Throughout the development process, I gained experience in analyzing college workflows, understanding user roles, gathering system requirements, and designing structured models such as ER diagrams and DFDs.

Building SCLMS also enhanced my full-stack development skills, particularly with Laravel, Vue.js, MySQL, and Python-based machine learning integration. Implementing features like leave approval prediction and anomaly detection helped me understand API communication, model deployment, and feature engineering in a practical context.

Additionally, this project emphasized the importance of secure coding practices, proper database design, and system scalability especially given the system's role in handling sensitive user information. Working through the phases of design, implementation, testing, and refinement improved my debugging abilities and reinforced the need for thorough testing to ensure reliability and a smooth user experience.

Overall, this project was both challenging and rewarding. It not only improved my technical capabilities but also prepared me for future academic and professional projects by boosting my confidence in developing complete, functional systems from the ground up.

5.2 Conclusion

The Smart College Leave Management System (SCLMS) was developed to replace the traditional manual leave-handling process commonly used in colleges. Manual systems often lead to slow approvals, missing records, and frequent miscommunication. By introducing a digital platform, SCLMS streamlines these tasks and ensures that leave processing becomes faster, more accurate, and more transparent.

The system is built around real academic workflows through a clearly defined role-based structure. Students can submit their leave requests online, teachers can review and approve them, and HODs make the final decision. This structured flow ensures that every request moves through the proper hierarchy, reducing confusion and maintaining accountability.

One of the notable features of SCLMS is the integration of machine learning tools such as leave approval prediction and anomaly detection. These intelligent components help reviewers make informed decisions by analyzing past leave trends, student behavior, and request patterns. The anomaly detection module also strengthens fairness by identifying suspicious or repetitive leave submissions that may need attention.

Along with these intelligent features, the system provides an easy-to-use interface, a visual leave calendar, document upload support, automated notifications, and detailed reports. The use of modern technologies like Laravel, Vue.js, MySQL, and Python-based ML services ensures that the platform remains secure, efficient, and scalable for future enhancements.

In conclusion, the Smart College Leave Management System successfully achieves its aim of digitizing and optimizing college leave workflows. It significantly reduces administrative burden while improving communication, record accuracy, and overall operational efficiency

5.3 Future recommendation

- Develop a dedicated mobile application (Android/iOS) using Flutter for easier leave submission and approval.
- Integrate biometric or RFID attendance systems for automatic absence tracking and accurate leave validation.
- Implement advanced AI models to provide intelligent leave approval recommendations.
- Add SMS and mobile push notifications for faster and more reliable communication.
- Extend the system to a multi-tenant architecture to support multiple colleges or campuses.
- Enhance the reporting module with interactive dashboards, charts, and analytics.

References

[1] Y. Zhou, L. Wang, and H. Li, “Digital transformation in academic administrative systems: Improving efficiency through automated platforms,” *Journal of Educational Technology Systems*, vol. 47, no. 4, pp. 567–582, 2018.

[2] R. Patel and S. Kumar, “A study on automation needs in institutional leave processing,” *International Journal of Information Management Systems*, vol. 12, no. 2, pp. 101–112, 2019.

[3] T. Kaufman, “Decision-support technologies in modern administrative workflows: A machine-learning perspective,” *IEEE Transactions on Computational Social Systems*, vol. 7, no. 3, pp. 812–820, 2020.

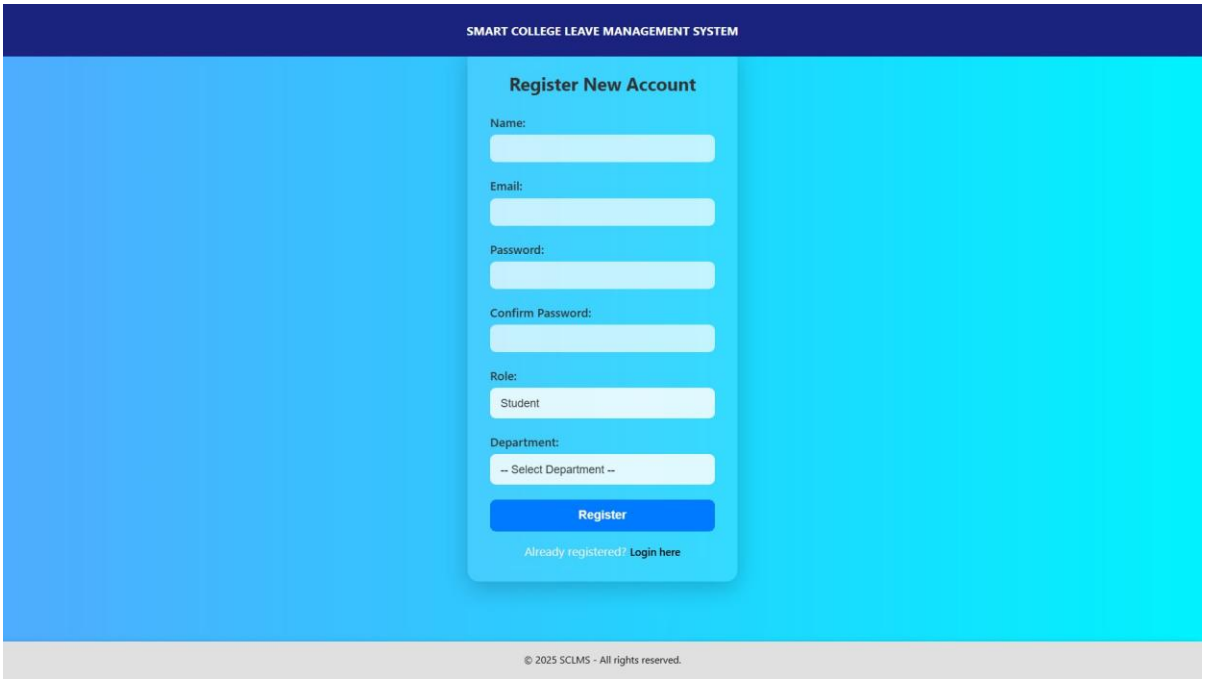
[4] B. Schneier, “Anomaly detection techniques for administrative fraud prevention,” *IEEE Security & Privacy*, vol. 19, no. 1, pp. 45–53, 2021.

[5] A. Smith, “Limitations of ERP-based leave modules for academic institutions,” *International Journal of Enterprise Computing*, vol. 15, no. 1, pp. 22–30, 2022.

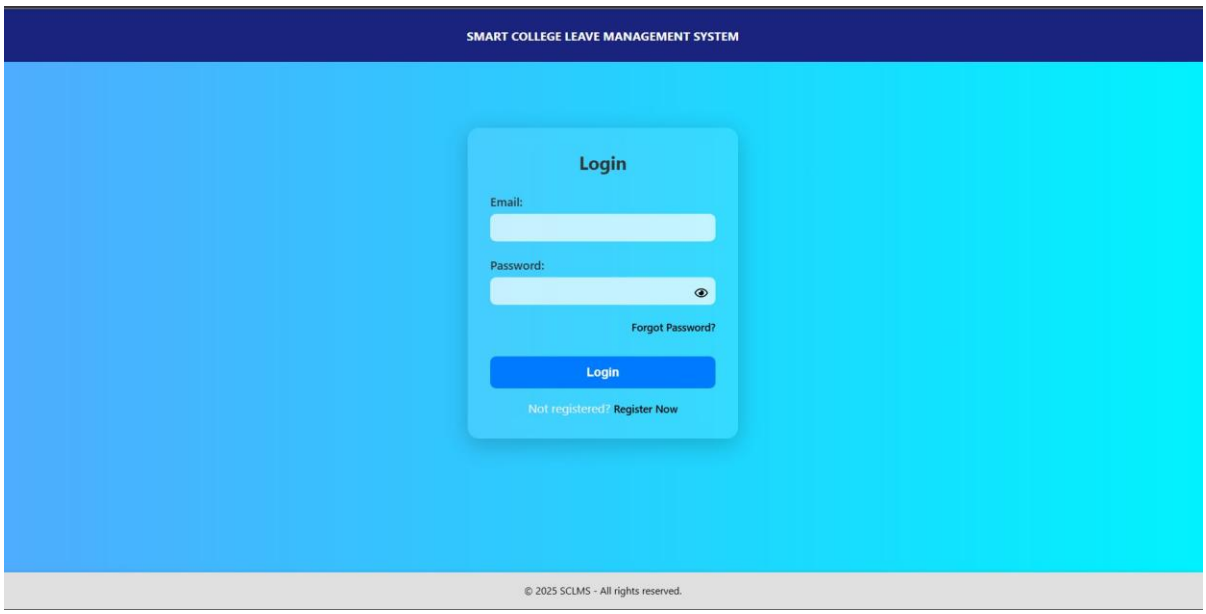
[6] F. Chen, Y. Liu, and M. Zhang, “Intelligent leave management systems: Integrating predictive analytics and anomaly detection,” *IEEE Access*, vol. 11, pp. 14532–14545, 2023.

Appendices

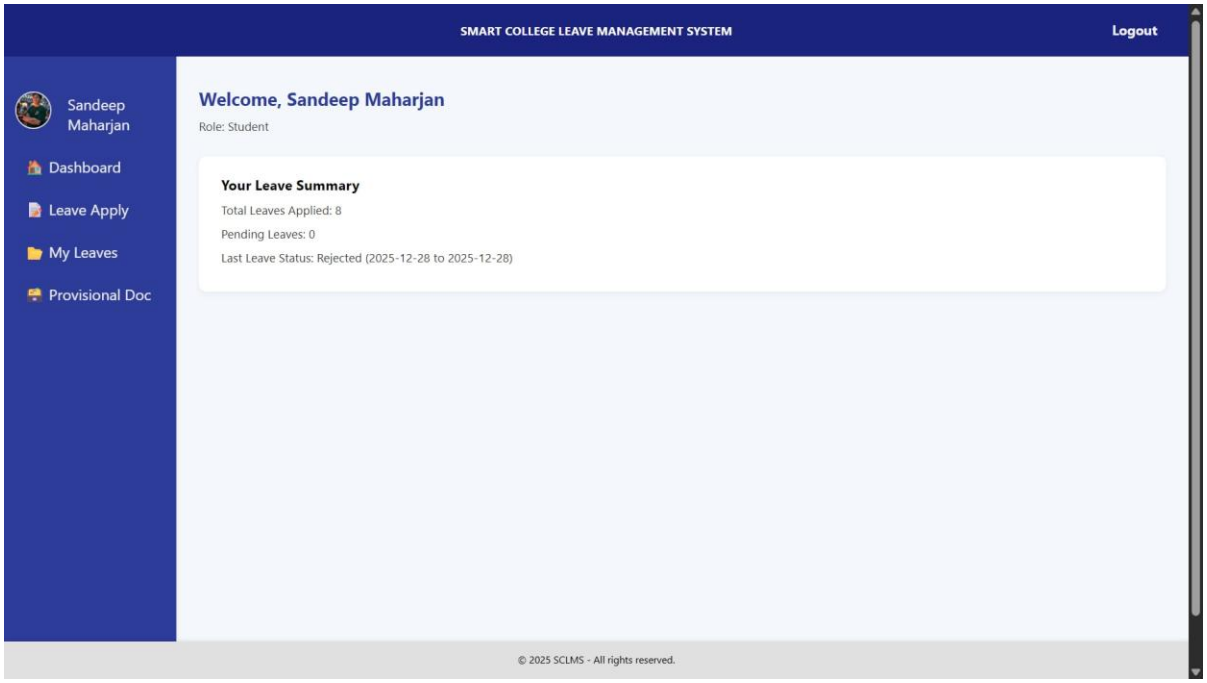
User Registration Page



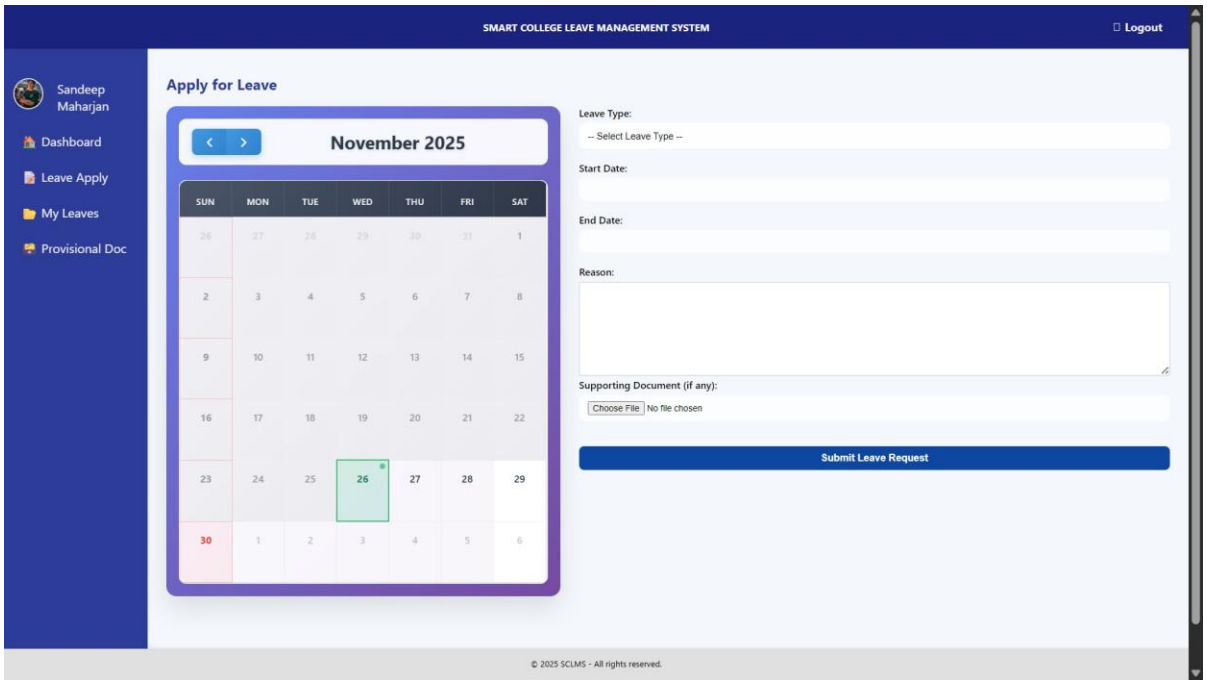
User Login Page



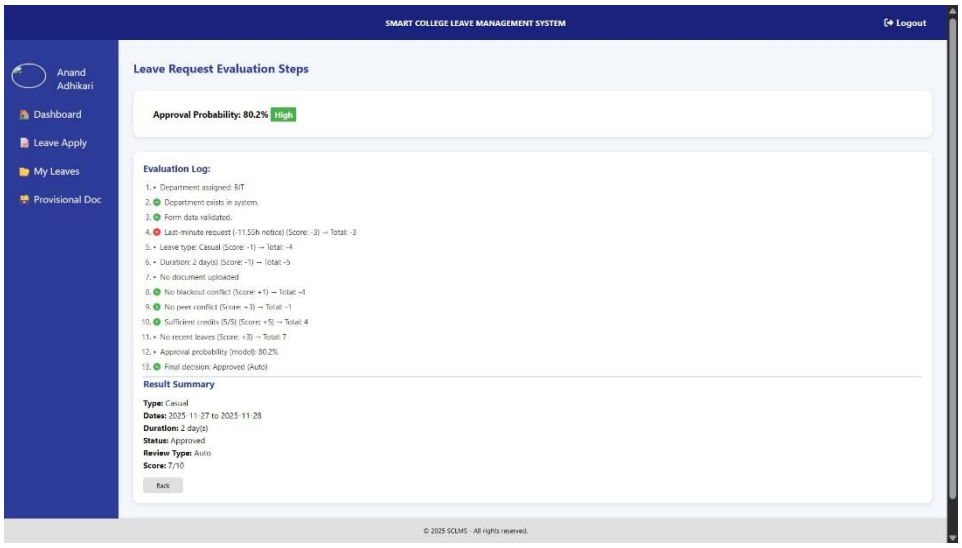
User Dashboard

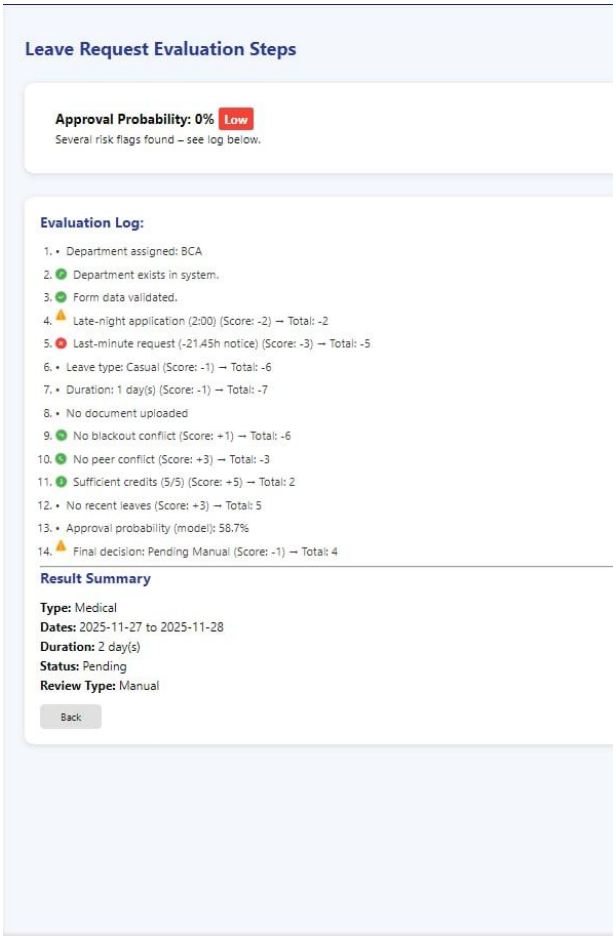


Apply Leave

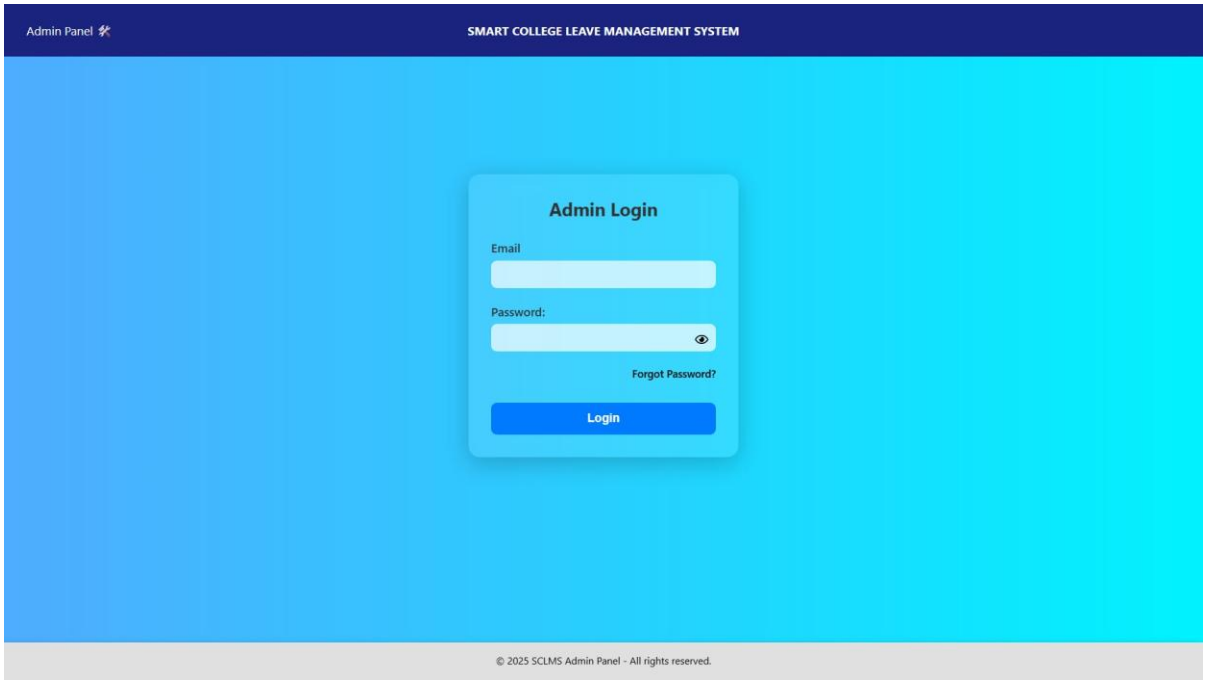


Process View

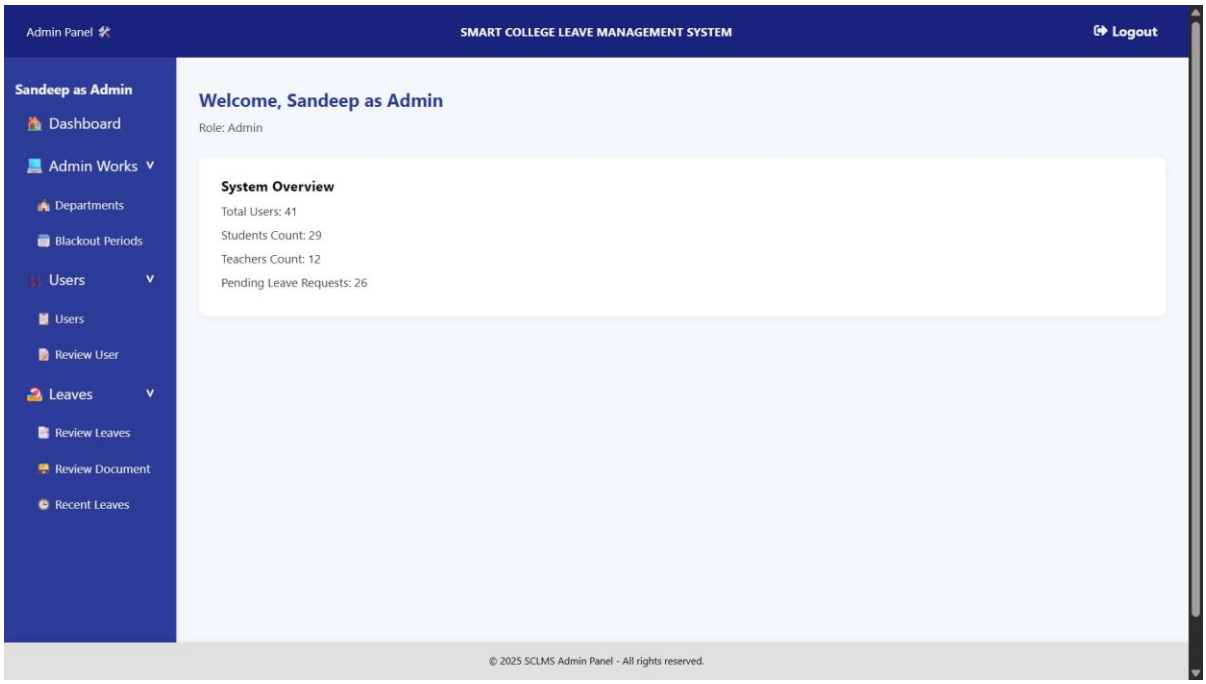




Admin Login



Admin Dashboard



Review Leave