© June 2026 | IJIRT | Volume 13 Issue 1 | ISSN: 2349-6002

Internal Mark Allocation System

Soundarya S[1], Dr. R. Vijayalakshmi[2]
[1]*Department of Computer Applications (MCA), Krishnasamy College of Engineering & Technology*
[2]*MCA., M.Phil., Ph.D., Project Guide & Head of The Department (Associate Professor), Department of Computer Applications, Krishnasamy College of Engineering & Technology*

***Abstract*— The Internal Mark Allocation System is designed to automate and streamline the process of managing and evaluating internal assessment marks in educational institutions. Traditional methods of recording, calculating, and maintaining student marks are often manual, time-consuming, and highly prone to human errors, which can lead to inaccuracies, inconsistencies, and a lack of transparency in the evaluation process. This system provides a centralized and user-friendly digital platform where faculty members can efficiently enter, update, and manage marks for various academic components such as assignments, unit tests, attendance, seminars, and practical examinations. It ensures that all marks are systematically stored and processed, eliminating the need for manual calculations and paperwork. The system automatically computes the total internal marks based on predefined criteria, weightage, and institutional policies, thereby ensuring accuracy, consistency, and fairness in student evaluation. Furthermore, the system incorporates a secure role-based access mechanism that allows administrators, faculty, and students to interact with the platform according to their permissions. Administrators can manage user roles, subjects, and evaluation structures; faculty can handle mark entry and updates; and students can view their internal marks and performance details, thereby promoting transparency and accountability. The system also supports real-time data updates, enabling instant reflection of changes and reducing delays in information access. In addition, it provides comprehensive features such as report generation, data analysis, and performance tracking, which help educators monitor student progress, identify strengths and weaknesses, and make informed academic decisions. By digitizing the entire internal mark allocation process, the system significantly reduces administrative workload, minimizes errors, enhances data security, and improves overall efficiency. Ultimately, it serves as a reliable, scalable, and effective solution for modern educational institutions seeking to improve the quality, transparency, and management of academic evaluations.**

LIST OF ABBREVIATIONS

Abbreviation	Expansion
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
API	Application Programming Interface
REST	Representational State Transfer
SQL	Structured Query Language
URL	Uniform Resource Locator
HTTP	Hyper Text Transfer Protocol
PDF	Portable Document Format
CRUD	Create, Read, Update, Delete
UI	User Interface
DBMS	Database Management System
JSON	JavaScript Object Notation
DFD	Data Flow Diagram
UML	Unified Modeling Language
DOM	Document Object Model
NPM	Node Package Manager
GUI	Graphical User Interface
REACT	JavaScript Library for Building User Interfaces
BOOTSTRAP	Frontend CSS Framework for Responsive Design
PYTHON	High-Level Programming Language
FASTAPI	High-Performance Python Web Framework for APIs

I. INTRODUCTION

1.1. PROJECT DESCRIPTION

In addition to basic mark entry and calculation, the system is designed to handle multiple subjects and different types of assessments simultaneously. Each subject may have different evaluation criteria, such as

varying weightage for assignments, tests, attendance, and practical work. The system provides flexibility to configure these weightages according to institutional requirements, ensuring that the evaluation process aligns with academic policies.

The platform also supports role-based access control, where different users such as administrators, faculty members, and students have distinct functionalities. Administrators can manage student records, subject details, and user roles, while faculty members are responsible for entering and updating internal marks. Students are given access to view their marks and performance reports, ensuring transparency in the evaluation process.

Another important feature of the system is data validation and error handling. The system ensures that only valid marks are entered within predefined limits, preventing incorrect data entry. It also reduces redundancy by maintaining a centralized database where all information is stored and managed efficiently. This eliminates the need for maintaining multiple records across different files or systems.

The system provides real-time updates, meaning that any changes made by faculty members are instantly reflected in the database and visible to students. This improves communication between students and faculty and allows students to stay updated on their academic performance without delays.

Furthermore, the system includes reporting and analytics features that help faculty and administrators analyze student performance. Reports can be generated based on subject-wise performance, overall class performance, and individual student progress. These reports assist in identifying weak areas and help in taking corrective measures to improve academic outcomes.

Security is a critical aspect of the system. User authentication mechanisms ensure that only authorized users are able to access the system. Sensitive data, such as student marks and personal information, is protected through secure access controls. This helps maintain data privacy and prevents any unauthorized modifications.

The system is designed with scalability in mind, enabling it to efficiently manage a large number of users and records. As the institution continues to grow, the system can be extended to include additional features such as grade prediction, attendance tracking, and integration with other academic systems.

Overall, the Internal Mark Allocation System streamlines the process of managing internal assessments by automating calculations, reducing errors, and improving transparency. It offers a reliable and efficient solution that enhances the academic evaluation process and supports better decision-making for both faculty and students.

1.2. SCOPE OF THE PROJECT

The scope of this project encompasses the complete digital transformation of the internal mark management process at Krishnasamy College of Engineering and Technology, Cuddalore. The system focuses on automating the process of recording, calculating, and monitoring internal marks for students in an efficient and transparent manner. The system specifically covers the following areas:

Student Data Management:

The system maintains comprehensive student records including registration number, name, department, and subject details. It ensures that only valid and authenticated student data is used for internal mark processing. All data is stored in a centralized database, enabling easy access and management.

Faculty Mark Entry:

The system provides a dedicated interface for faculty members to enter internal marks for various assessment components such as assignments, internal tests, attendance, and practical examinations. Proper validation mechanisms are implemented to ensure accuracy and prevent invalid data entry.

Automated Mark Calculation:

The system automatically calculates internal marks based on predefined weightage criteria assigned to different evaluation components. This eliminates manual calculations and ensures consistency and fairness in mark allocation.

Performance Tracking:

Students can access the system to view their internal marks and track their academic performance. This promotes transparency and helps students identify their strengths and area for improvement.

Admin Dashboard:

A secure administrative dashboard is provided for

managing students, subjects, and faculty data. The admin can monitor overall performance, manage user roles, and generate reports. The dashboard also provides insights into student performance through basic analytics.

RESTful API Integration:
The backend is developed using Fast API, providing RESTful APIs for efficient communication between the frontend and backend systems. This ensures smooth data exchange and scalability of the application.

Responsive User Interface:
The frontend developed using React and Bootstrap ensures that the system is accessible across multiple devices including desktops, tablets, and smartphones, providing a seamless user experience.

Security and Access Control:
The system implements authentication and role-based access control to ensure that only authorized users (admin, faculty, students) can access relevant functionalities. This ensures data privacy and system security.

1.3. ORGANIZATION PROFILE

Krishnasamy College of Engineering and Technology (KCET) is a reputed engineering institution located in Cuddalore, Tamil Nadu, India. The college is affiliated with Anna University, Chennai, and is approved by the All-India Council for Technical Education (AICTE). It is also accredited by the National Assessment and Accreditation Council (NAAC), reflecting its commitment to quality education and academic excellence.

The institution offers a wide range of undergraduate, postgraduate, and doctoral programs in various disciplines of engineering, technology, and computer applications. KCET is known for its strong academic framework, well-qualified faculty members, and modern infrastructure that supports both theoretical and practical learning.

The Department of Computer Applications offers the Master of Computer Applications (MCA) program, which aims to provide students with strong foundations in software development, programming, and emerging technologies. The department is guided by experienced faculty members with excellent academic and research backgrounds, ensuring quality education and mentorship.

The institution encourages innovation, research, and industry-oriented learning. It promotes collaboration between academia and industry to help students develop real-world problem-solving skills. Students are motivated to undertake projects that address practical challenges, thereby enhancing their technical knowledge and professional competence.

Overall, Krishnasamy College of Engineering and Technology provides a conducive environment for academic growth, skill development, and career advancement, making it a preferred choice for higher education in the field of engineering and computer applications

II. LITERATURE REVIEW

[1] Smith et al. (2018)

Stated that traditional systems were manual and time-consuming, leading to errors in data management. Smith et al. (2018) analyzed traditional data management systems and found that most of them relied on manual processes such as paper-based records and basic spreadsheets. These methods were time-consuming and highly prone to human errors, especially when handling large volumes of data. The study emphasized that lack of automation and real-time data processing reduced overall efficiency. Therefore, the authors suggested the need for computerized systems to improve accuracy, speed, and data management.

[2] Johnson and Kumar (2019)

Developed a web-based system that improved accessibility but lacked proper security features. Johnson and kumar (2019) proposed a web-based management system that improved data accessibility and reduced manual effort. The system allowed multiple users to access and update data efficiently. However, it lacked strong security mechanisms and had limitations in handling large-scale data. The study highlighted the need for more secure and scalable solutions.

[3] Brown (2017)

Discussed the use of HTML, CSS, and javascript for frontend and mysql for database management in such system. Brown (2017) discussed the use of web

technologies such as HTML, CSS, and JavaScript for developing user-friendly interfaces, along with MySQL for efficient data storage. The study highlighted that these technologies improve system performance but require proper design for better usability.

[4] Garcia and Patel (2021)
Highlighted the importance of cloud-based solutions for better scalability and performance. Garcia and patel (2021) emphasized the use of cloud-based technologies to improve system scalability and performance. Their study showed that cloud solutions enable efficient data storage, faster access, and better handling of large-scale data. However, they also highlighted the need for proper security measures in cloud environments.

2.1. OVERVIEW OF LITERATURE REVIEW
The literature review focuses on analyzing previously developed systems and research works related to the project. It helps in understanding the existing methods, technologies, and their limitations. From the study of various authors, it is observed that traditional systems lack efficiency, security, and real-time processing. Modern systems have improved performance using web and cloud technologies, but still face challenges in scalability and data management. This analysis helps in identifying the gaps and developing a more efficient proposed system
The literature review examines existing systems and technologies based on studies by Smith et al. (2018), Johnson and Kumar (2019), and Lee et al. (2020). These studies highlight the advantages and limitations of current systems, such as lack of security and real-time processing. Further, Brown (2017) and Garcia and Patel (2021) emphasized the role of modern technologies and scalability. This analysis supports the need for an improved and efficient proposed system.

III. SYSTEM ANALYSIS

3.1. INTRODUCTION
System analysis is an important phase in project development that involves studying the existing system and identifying its problems and requirements. It helps in understanding how the current system works and what improvements are needed. Through system analysis, the limitations of the existing system such as lack of efficiency, security, and performance are identified. This process provides a clear idea for designing a better and more effective proposed system.

3.2. FEASIBILITY STUDY
Feasibility study is an important step in system development that evaluates whether the proposed system is practical and achievable. It helps in analyzing different aspects such as technical, economic, and operational feasibility.

3.2.1.ECONOMICAL FEASIBILITY
Economic feasibility evaluates whether the proposed system is cost-effective and financially viable. It considers the total cost required for development, implementation, and maintenance of the system. The proposed system uses commonly available technologies, which reduces development cost. The benefits of the system, such as improved efficiency, reduced manual work, and time savings, outweigh the overall cost. Therefore, the project is considered economically feasible and beneficial.

3.2.2.TECHNICAL FEASIBILITY
Technical feasibility evaluates whether the required technology, tools, and resources are available to develop the proposed system. The system can be developed using commonly available technologies such as programming languages, databases, and software tools.
The required hardware and software resources are easily accessible and do not require high investment. The development team has the necessary technical knowledge and skills to implement the system successfully. Therefore, the project is technically feasible.

3.2.3.SOCIAL FEASIBILITY
Social feasibility evaluates the acceptance of the proposed system by users and its impact on society. The system is designed to be user-friendly and easy to understand, which helps users adopt it without difficulty. It reduces manual work and improves efficiency, making it beneficial for users. The system does not create any negative social impact and supports better productivity. Therefore, the proposed system is socially feasible.

3.3. EXISTING SYSTEM

The existing system is mainly based on manual processes or basic digital tools such as spreadsheets. Data is recorded and maintained manually, which is time-consuming and prone to errors. The system lacks automation, real-time data updates, and proper security mechanisms. It is also difficult to handle large volumes of data efficiently. Due to these limitations, the existing system is less reliable and less efficient.

3.3.1.DISADVANTAGES

- Time-Consuming: Manual collection, verification, and processing is extremely slow, often taking several days to complete registrations for a single event.
- Error-Prone: Manual data entry causes frequent errors in student details such as incorrect registration numbers, misspelled names, and wrong dates.
- Lack of security: The system does not provide proper protection for data, leading to risk of unauthorized access.
- No real-time updates: The system does not update data instantly, causing delays in accessing current information.
- Low efficiency: The system performs tasks slowly and does not utilize resources effectively.
- Poor data management: The system does not organize and store data properly, making it difficult to access and maintain information.

3.4. PROPOSED SYSTEM

The proposed system is designed to overcome the limitations of the existing system by introducing an automated and efficient solution. It uses modern technologies to ensure better performance, security, and data management.

The system provides real-time data updates, secure data handling, and a user-friendly interface. It reduces manual work, improves accuracy, and allows easy access to information. Overall, the proposed system is more reliable, efficient, and scalable.

3.4.1 ADVANTAGES

- Saves time – Tasks are completed quickly through automation.
- Reduces errors– Minimizes human mistakes in data handling.
- Improves security– Protects data from unauthorized access.
- Provides real-time updates – Data is updated instantly.
- Increases efficiency– Performs tasks faster and effectively.
- Easy to use– Simple and user-friendly interface.
- Better data management– Organizes and stores data properly.
- Scalable and reliable– Can handle large data and works consistently.

3.5. SYSTEM REQUIREMENTS

3.5.1.HARDWARE REQUIREMENTS

- Processor: Intel Core i3 or above
- RAM: Minimum 4 GB
- Hard Disk: 500 GB or above
- Monitor: Standard display monitor
- Keyboard and Mouse
- Internet Connection (if required)

3.5.2.SOFTWARE REQUIREMENTS

- Operating System: Windows 10 or above
- Programming Language: Java / Python / PHP
- Frontend: HTML, CSS, JavaScript
- Backend: MySQL
- Development Tool: VS Code / Eclipse
- Browser: Google Chrome

3.6. LANGUAGE SPECIFICATION

3.6.1.TOOLS DESCRIPTION

HTML

HTML is a standard markup language used to design and create the structure of web pages. It is used to define elements such as headings, paragraphs, images, links, and tables. HTML provides the basic framework of a website and helps in organizing content in a structured format. It works along with CSS and JavaScript to develop interactive and user-friendly web applications.

MySQL

MySQL is an open-source relational database management system based on Structured Query Language (SQL). It is the world's most popular open-source database and is used in this project to store and manage student records, event data, and convocation registration details via the mysql2 NPM package.

VISUAL STUDIO

Visual Studio Code is a widely used source code editor developed by Microsoft. It is used for writing, editing, and managing code efficiently. VS Code supports multiple programming languages and provides features such as syntax highlighting, debugging, and extensions. It helps developers to build applications easily with a user-friendly interface and powerful tools.

WEB BROWSER

A web browser is a software application used to access and view web pages. It helps users to run and test web applications.

CSS

CSS is used to style and design web pages. It controls the layout, colors, fonts, and overall appearance of website.

IV. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

System architecture describes the overall structure and working of the system. It shows how different components such as user interface, application logic, and database interact with each other.

In this system, the user interacts with the frontend (HTML, CSS, JavaScript), which sends requests to the backend. The backend processes the data and communicates with the database (MySQL) to store and retrieve information. The processed data is then sent back to the user through the interface.

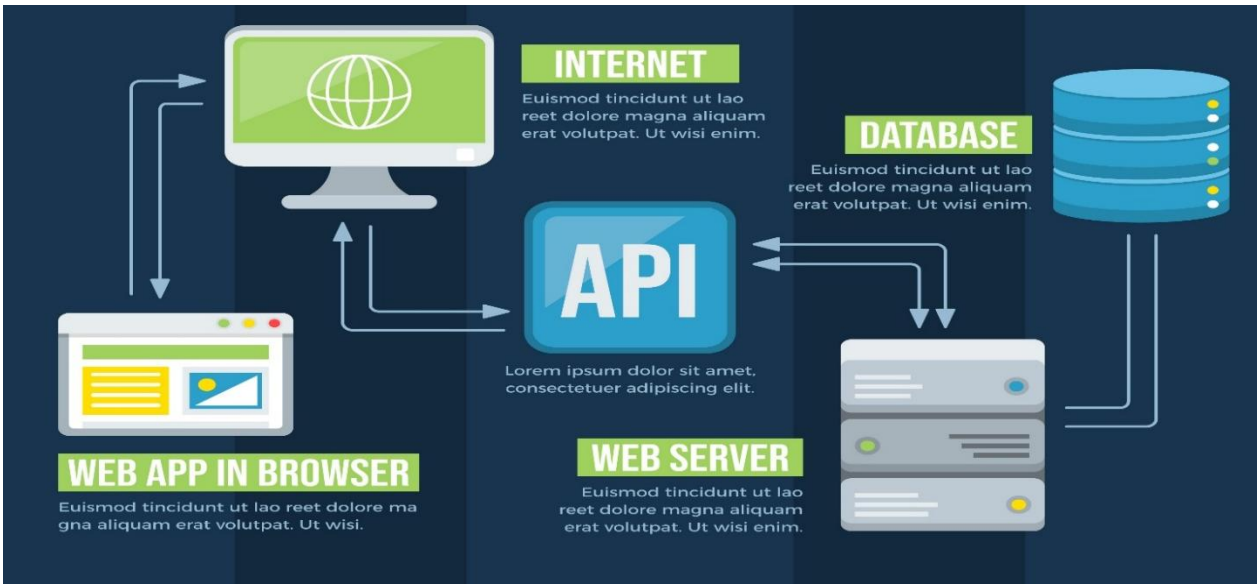

Figure No. 1: System Architecture

4.2 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is used to represent the flow of data within the system. It shows how data is input, processed, stored, and output in the system.

The main components of DFD include Process, Data Flow, Data Store, and External Entity. The user provides input to the system, which is processed and stored in the database. The processed data is then returned as output to the user.

Level 0 (Context Diagram) – Shows the overall system as a single process with input and output.

Level 1 – Shows detailed processes and data flow inside the system.

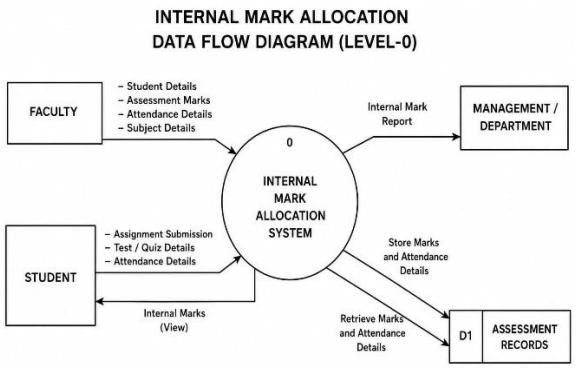

Figure No. 2a: Data Flow Diagram — Level 0 (Context Diagram)

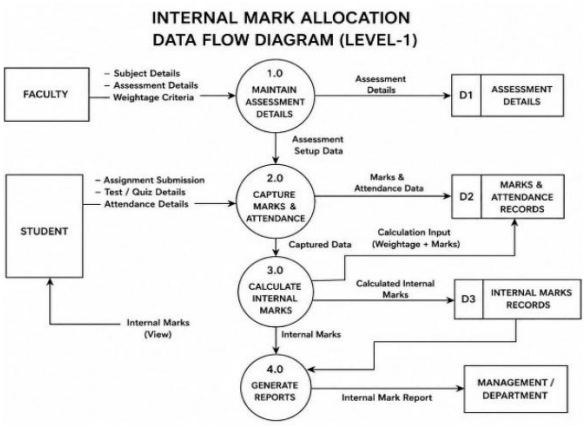

Figure No. 2b: Data Flow Diagram — Level 1

4.3 UML DIAGRAMS

UML diagrams are used to represent the design and structure of a system in a visual way. They help in understanding how the system works, how different components interact, and how users interact with the system.

4.3.1. USE CASE DIAGRAM

A Use Case Diagram is a type of UML diagram that shows the interaction between users (actors) and the system. It represents the different functionalities of the system and how users use them.

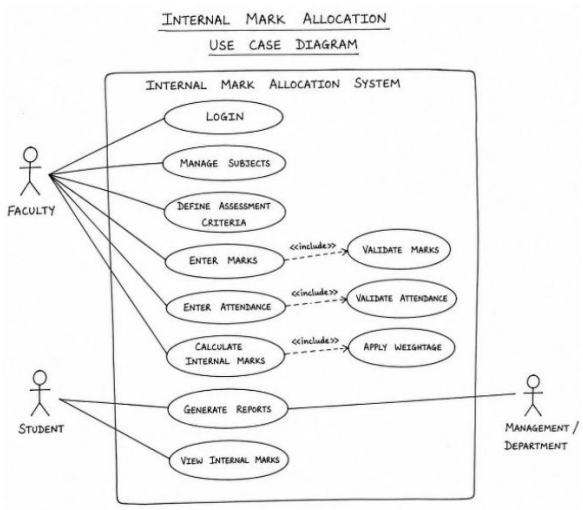

Figure No. 3: Use Case Diagram

4.3.2. CLASS DIAGRAM

A class diagram shows the structure of a system by displaying its classes, their attributes, methods, and the relationships between them. It helps the team understand how the system is organized and how components interact.

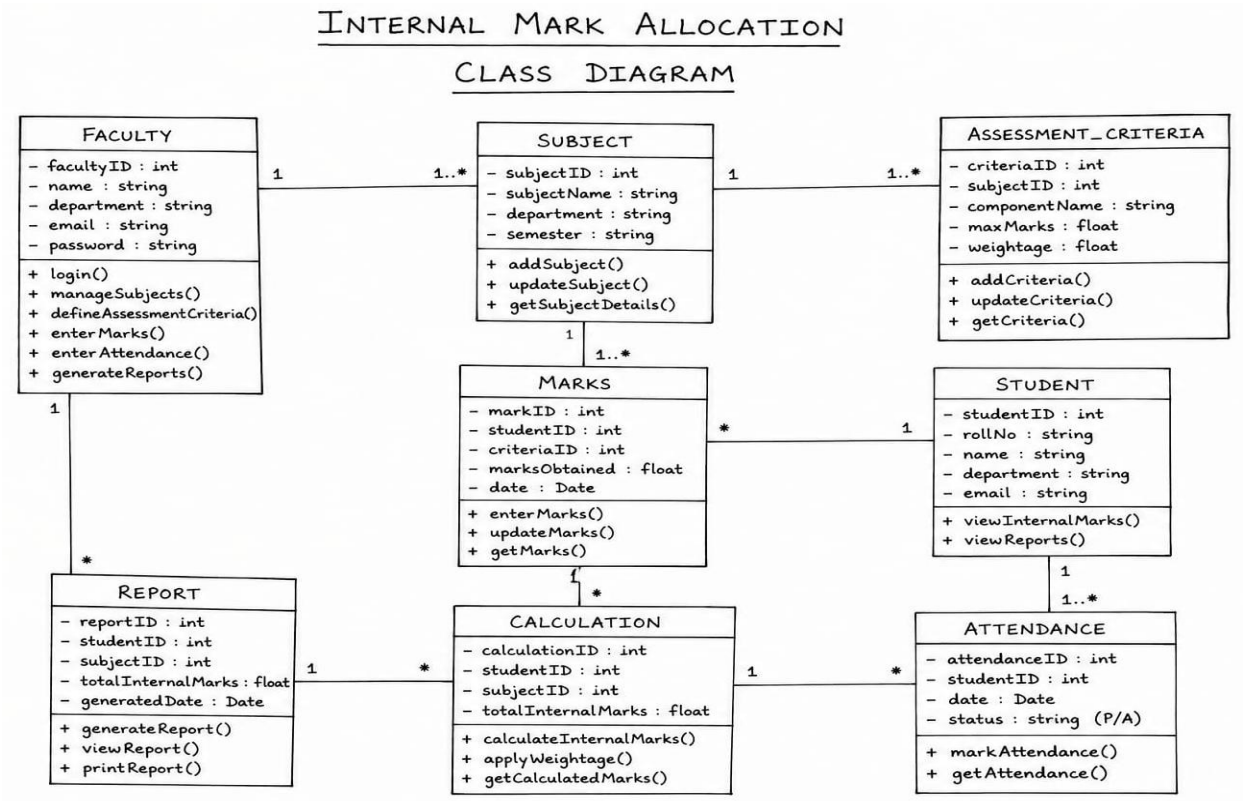

Figure No. 4: Class Diagram

4.3.3.SEQUENCE DIAGRAM

The Internal Mark Allocation System allows the admin to create marking schemes and assign subjects to faculty members. Faculty members enter and publish students' internal marks, which are stored in the database. Students can log in to the system and view their published internal marks. The database manages authentication, assignment details, and mark records throughout the process.

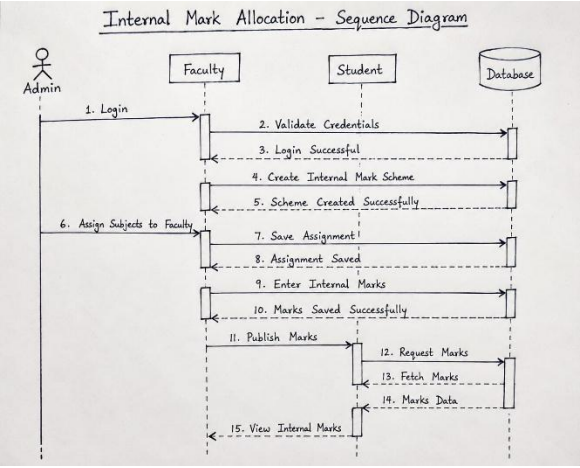

Figure No. 5: Sequence Diagram

V. SYSTEM REQUIREMENTS

5.1. MODULE DESCRIPTIONS

- Student Management Module
- Internal mark allocation module
- Mark calculation module
- Admin dashboard module
- Report generation module

5.1.1.STUDENT MANAGEMENT MODULE

The Student Management Module in an internal mark allocation system is used to manage all student-related information efficiently. It stores details such as student ID, name, department, and semester, and allows adding, updating, or deleting records. The module helps in assigning students to courses, recording attendance, and entering internal marks like tests and assignments. It also automatically calculates total marks and helps in tracking student performance. Overall, it simplifies data management and reduces manual work.

5.1.2.INTERNAL MARK ENTRY MODULE

The Internal Mark Entry Module in an internal mark allocation system is used to record and manage students' internal assessment scores. It allows faculty to enter marks for components such as assignments, unit tests, seminars, and attendance. The module ensures that marks are entered subject-wise and student-wise accurately, with options to edit or update when needed. It may also validate inputs to avoid errors and automatically calculate total internal marks based on predefined weightage. This module helps streamline the mark entry process, ensures consistency, and reduces manual calculation errors.

5.1.3.MARK CALCULATION MODULE

The Mark Calculation Module is responsible for computing students' internal marks based on predefined evaluation criteria such as assignments, tests, attendance, and practical performance. This module automates the calculation process to ensure accuracy, consistency, and fairness in mark distribution.

It collects input data from various sub-modules like test scores, assignment marks, and attendance records. Based on the configured weightage (for example: 40% tests, 30% assignments, 20% attendance, 10% practical), the system calculates the total internal mark for each student.

5.1.4.ADMIN DASHBOARD MODULE

The Admin Dashboard Module acts as the central control panel of the Internal Mark Allocation System. It allows the administrator to manage, monitor, and control all system activities efficiently through a single interface.

This module provides access to key functionalities such as student management, staff management, subject allocation, and internal mark overview. The admin can add, update, or delete student and faculty details, assign subjects to staff, and configure marking criteria.

5.1.5.REPORT GENERATION MODULE

The Report Generation Module is responsible for creating and displaying detailed reports of students' internal marks. It helps administrators and faculty easily analyse student performance and maintain proper academic records.

This module collects data from the mark calculation system and generates reports in a structured format. Reports can be viewed on-screen or exported for printing and documentation purposes.

VI. SYSTEM TESTING

6.1. INTRODUCTION

System Testing is the process of evaluating the complete and integrated system to ensure that it meets the specified requirements. In the Internal Mark Allocation System, system testing verifies whether all modules—such as student management, mark calculation, admin dashboard, and report generation—work together correctly as a whole.

This phase is performed after integration testing and focuses on validating the system's functionality, performance, security, and usability. The main goal is to identify any defects or issues before the system is deployed for actual use.

6.2. UNIT TESTING

Unit Testing is the process of testing individual components or modules of a system independently to ensure that each part functions correctly. In the Internal Mark Allocation System, unit testing focuses on verifying small units such as mark calculation functions, data input forms, and database operations.

Each module—like student data entry, mark calculation, or report generation—is tested separately to identify and fix errors at an early stage. This helps ensure that every component works as expected before integrating them into the complete system.

6.3. INTEGRATION TESTING

Integration Testing is the process of testing how different modules of a system work together as a group. In the Internal Mark Allocation System, it ensures that all individual modules—such as student management, mark calculation, admin dashboard, and report generation—interact correctly and exchange data without errors.

6.4. FUNCTIONAL TESTING

Functional Testing is the process of verifying whether the system's features and functionalities work according to the specified requirements. In the Internal Mark Allocation System, it ensures that each function—such as mark entry, calculation, report generation, and user login—operates correctly.

This type of testing focuses on the system's behavior by providing inputs and checking the expected outputs, without considering the internal code structure.

Key aspects of functional testing include:

- Validating all system features and functions
- Checking input and output correctness
- Ensuring business requirements are met
- Testing user interactions with the system

6.5. SYSTEM TESTING

System Testing is the process of testing the complete and fully integrated system to ensure that it meets all specified requirements. In the Internal Mark Allocation System, this testing verifies that all modules—such as student management, admin dashboard, mark calculation, and report generation—work together correctly as a whole.

It is performed after integration testing and focuses on validating the overall system behavior in a real-world environment.

6.6. WHITE BOX TESTING

White Box Testing is a testing technique that focuses on examining the internal structure, logic, and code of the system. In the Internal Mark Allocation System, it is used to verify that the internal operations—such as mark calculations, data processing, and control flow—are working correctly.

This type of testing is usually performed by developers who have knowledge of the system's code. It ensures that all paths, conditions, and loops in the program are properly tested.

6.7. BLACK BOX TESTING

Black Box Testing is a testing technique that focuses on verifying the functionality of a system without knowing its internal code or structure. In the Internal Mark Allocation System, it checks whether the system behaves correctly based on user inputs and expected outputs. Tester interacts with the system just like end users—by entering data and observing results—without any knowledge of how the system processes the data internally.

6.8. INTEGRATION TESTING

Integration Testing is a software testing phase where individual modules of a system are combined and tested as a group to ensure they work together correctly. In the Internal Mark Allocation System, this testing verifies the interaction between modules such as student management, mark entry, mark calculation, admin dashboard, and report generation.

6.9. ACCEPTANCE TESTING

Acceptance Testing is the final phase of software testing where the system is evaluated to determine whether it meets the user requirements and is ready for deployment. In the Internal Mark Allocation System, this testing ensures that the system works according to the expectations of end users such as administrators and faculty.

VII. SYSTEM IMPLEMENTATION

System Implementation is the process of deploying the developed Internal Mark Allocation System into a real-time environment for actual use. It involves installing the system, configuring necessary settings, and making it operational for administrators and faculty.

During this phase, the system is integrated with required hardware and software, and users are provided with access to perform their tasks such as entering marks, managing student data, and generating reports.

Step 1 -The first step in system implementation is installing the Internal Mark Allocation System in the required environment. This involves setting up the software on the server or local machines so that it can be accessed by administrators and faculty.

Step 2 After installation, the next step is configuring the system according to the institution's requirements. This step ensures that the Internal Mark Allocation System is customized to match academic rules and user roles.

Step 3 - Data Migration is the process of transferring existing data into the new Internal Mark Allocation System. This ensures that all necessary student and academic information is available in the system before it is used.

Step 4 - User Training is the process of teaching administrators and faculty how to effectively use the Internal Mark Allocation System. This step ensures that users understand the system features and can perform their tasks without errors.

Step 5 - System Deployment is the final step where the Internal Mark Allocation System is made live and available for actual use by administrators and faculty. After successful installation, configuration, data migration, and training, the system is officially launched.

Step 6- System Maintenance is the final phase where the Internal Mark Allocation System is continuously monitored, updated, and improved after deployment. This ensures that the system remains efficient, secure, and error-free during its usage.

VIII. SYSTEM MAINTENANCE

System Maintenance is the process of regularly updating, monitoring, and improving the Internal Mark Allocation System after it has been deployed. This phase ensures that the system continues to function efficiently, securely, and without errors over time.

It involves fixing bugs, updating features, and adapting the system to new requirements or changes in academic processes.

For the purpose of convenience, maintenance may be categorized into three types:

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance

CORRECTIVE MAINTENANCE

Corrective maintenance involves identifying and removing errors that have crept into the system due to faulty design, wrong assumptions, or unforeseen edge cases in production usage. Planned corrective maintenance activities for this system include:

- Fixing database query errors arising from unexpected data inconsistencies or NULL values in student records.
- Resolving file upload errors due to storage permission issues or disk space limitations on the server.
- correcting frontend JavaScript errors affecting form submission on older browser versions.
- Fixing session expiry issues in the admin dashboard that may cause unexpected logouts during extended sessions.
- Addressing PDF generation formatting issues that may arise with unusually long student names or department names.

ADAPTIVE MAINTENANCE

Adaptive maintenance involves modifying the system to accommodate changes in the technical environment, platform updates, or evolving institutional requirements.

Planned adaptive maintenance activities include:

- Upgrading Node.js runtime to newer LTS versions (v20.x, v22.x) as they become available and are widely adopted.
- Migrating to newer MySQL versions to benefit from performance improvements and security patches.
- Updating all npm dependencies to address known security vulnerabilities identified by npm audit.
- Adapting the system for full cloud deployment on AWS EC2 with Amazon RDS for MySQL and Amazon S3 for file storage.
- Modifying the database schema to support additional events as the institution's academic calendar evolves.

PERFECTIVE MAINTENANCE

Perfective maintenance involves adding new features or enhancing existing functionality to improve system performance, usability, or utility.

Planned perfective maintenance activities include:

- Adding QR code generation to invitation PDFs using the or code npm package for digital attendance verification at event entry points.
- automated email notifications using Node mailer to notify students and parents/guardians upon successful registration.
- Enhancing the admin dashboard with export-to-CSV and export-to-Excel functionality using the json2csv and xlsx npm packages.
- Implementing role-based access control to support multiple admin levels (Super Admin, Department Admin, Read-Only Staff).
- Adding pagination to the admin dashboard to handle large numbers of registrations efficiently.

IX. CONCLUSION

The Internal Mark Allocation System is an efficient and reliable solution for managing and calculating students' internal marks in an academic environment. It automates the process of mark entry, calculation, and report generation, reducing manual effort and minimizing errors.

The system ensures accuracy, transparency, and consistency in evaluating student performance. With modules like admin dashboard, mark calculation, and report generation, it provides a structured approach to handling academic data.

By implementing this system, institutions can save time, improve data management, and enhance overall productivity. It also offers flexibility to adapt to different evaluation criteria and supports better decision-making through clear and organized reports.

Overall, the Internal Mark Allocation System is a valuable tool that simplifies the internal assessment process and improves the efficiency of academic management.

X. FUTURE ENHANCEMENT

The Internal Mark Allocation System can be further improved by adding new features and advanced technologies to enhance its functionality and usability.

Some possible future enhancements include:

- Mobile Application Support: Develop a mobile app for easy access by faculty and students
- Student Portal: Allow students to view their internal marks, attendance, and performance online
- Automated Notifications: Send alerts via email or SMS for mark updates and announcements
- Advanced Analytics: Provide graphical reports and performance analysis for better decision-making
- Integration with Other Systems: Connect with college ERP or attendance systems for seamless data flow
- Cloud-Based Storage: Enable secure online data storage and remote access
- Role-Based Access Enhancement: Improve security with advanced user permissions
- AI-Based Insights: Predict student performance and identify weak areas

APPENDICES

APPENDIX 1 SCREEN SHOTS

SIGN PAGE

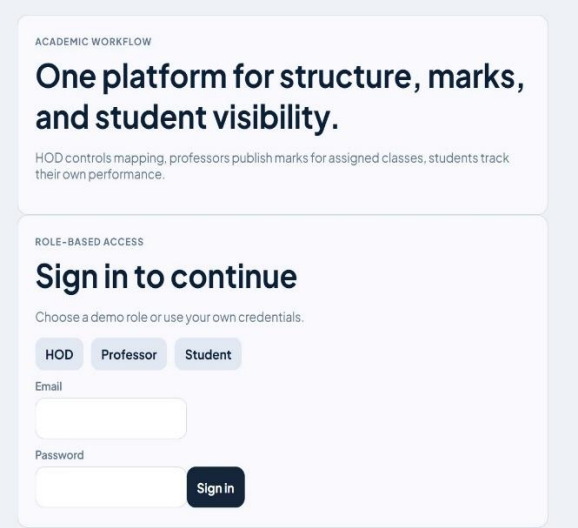

Fig 1: The Sign In page is the entry point of the Internal Mark Allocation system.

It allows authorized users such as administrators, faculty, and staff to securely access the system. Users must enter their valid credentials, including username and password, to log in. This ensures that only permitted users can view or modify internal marks, maintaining data security and integrity

HOD DASHBOARD

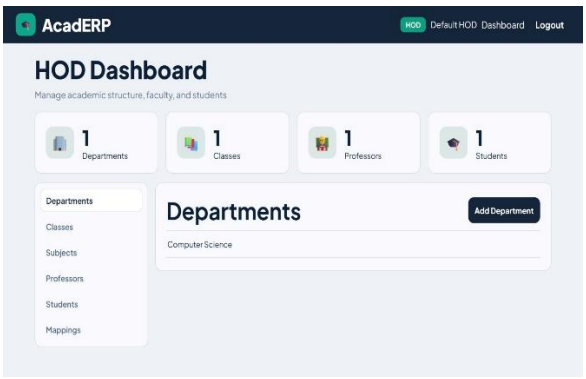

Fig 2: The Head of Department (HOD) Dashboard is an important module in the Internal Mark Allocation System.

It provides a centralized interface where the HOD can monitor, manage, and control all department-related academic activities. This dashboard helps in supervising faculty performance, verifying internal marks, and ensuring accuracy in student assessments.

CLASSES

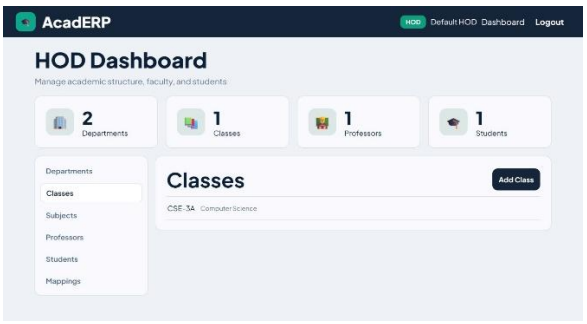

Fig 3: The Classes Page is an essential module in the Internal Mark Allocation System that manages all class-related information within the department.

It provides a structured view of different classes, sections, and their assigned subjects and faculty members. This page helps users like HOD and faculty to easily access and organize class details.

SUBJECTS

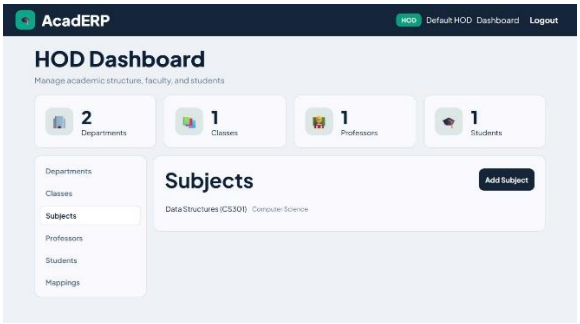

Fig 4: The Subject Page is an important module in the Internal Mark Allocation System that manages all subject-related information within a department.

It allows administrators and HOD to add, update, and organize subjects offered for different classes and semester.

PROFESSORS

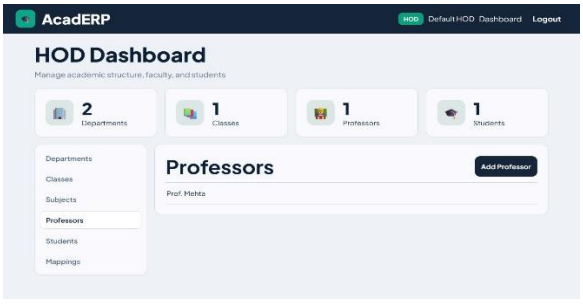

Fig 4: The Professor Page is a key module in the Internal Mark Allocation System that manages all faculty-related information.

It allows the administrator or HOD to add, update, and view details of professors handling different subjects and classes

STUDENTS PAGE

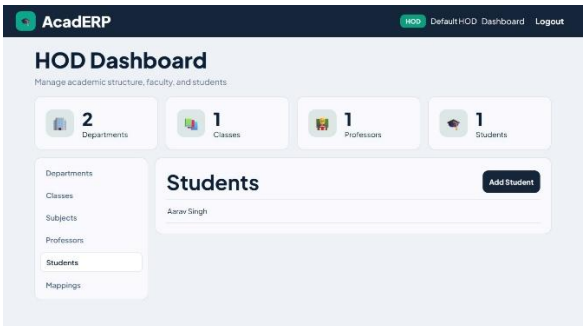

Fig 5: The Students Page is an essential module in the Internal Mark Allocation System that manages all student-related information.

MAPPING PAGE

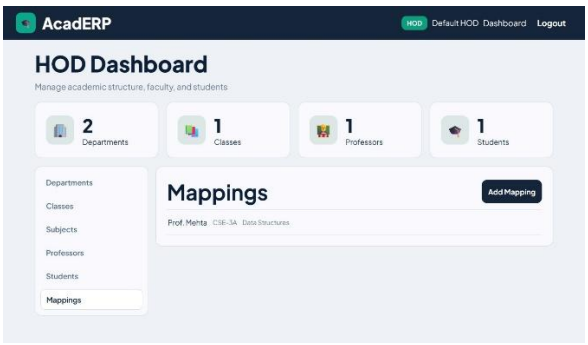

Fig 6: The Mapping Page is a crucial module in the Internal Mark Allocation System that connects and organizes relationships between different academic entities such as students, classes, subjects, and professors.

It ensures that all components are properly linked, enabling smooth internal mark entry and management.

APPENDIX 2
SAMPLE CODING
React.js

```
import React, {useEffect, useState } from "react";
import axios from "axios";

function App () {

// ---------- STATE ----------
const [students, setStudents] = useState([]);
const [form, setForm] = useState({
reg_no: "",
name: "",
subject: "",
assignment_marks: "",
test_marks: "",
attendance_marks: ""
});

const [editMode, setEditMode] = useState(false);
const [selectedReg, setSelectedReg] = useState("");

// ---------- FETCH DATA ----------
useEffect(() => {
fetchStudents();
}, []);

const fetchStudents = async () => {
try {
const res = await
axios.get("http://127.0.0.1:8000/students/");
setStudents(res.data);
} catch (error) {
console.error("Error fetching students:", error);
}
};

// ---------- INPUT CHANGE ----------
const handleChange = (e) => {
setForm({
...form,
[e. target.name]: e.target.value
});
};

// ---------- VALIDATION ----------
const validateForm = () => {
if (!form.reg_no || !form.name || !form.subject) {
alert ("Please fill all required fields");
return false;
}

if (
form.assignment_marks > 100 ||
form.test_marks > 100 ||
form.attendance_marks > 100
) {
alert ("Marks should not exceed 100");
return false;
}
```

```
return true;
};

// ---------------- ADD STUDENT ----------------
const addStudent = async () => {
if (!validateForm()) return;

try {
await axios.post("http://127.0.0.1:8000/add_student/",
form);
alert("Student added successfully");

resetForm();
fetchStudents();
} catch (error) {
console.error("Error adding student:", error);
}
};

// ---------------- UPDATE STUDENT ----------------
const updateStudent = async () => {
try {
await axios.put(
`http://127.0.0.1:8000/update/${selectedReg}`,
form
);

alert("Updated successfully");
setEditMode(false);
resetForm();
fetchStudents();
} catch (error) {
console.error("Error updating:", error);
}
};

// ---------------- DELETE STUDENT ----------------
const deleteStudent = async (reg_no) => {
if (!window.confirm("Are you sure?")) return;

try {
await
axios.delete(`http://127.0.0.1:8000/delete/${reg_no}`
);
alert("Deleted successfully");
fetchStudents();
} catch (error) {
console.error("Delete error:", error);
}
};

// ---------------- EDIT LOAD ----------------
const editStudent = (student) => {
setForm({
reg_no: student.reg_no,
name: student.name,
subject: student.subject,
assignment_marks: student.assignment_marks,
test_marks: student.test_marks,
attendance_marks: student.attendance_marks
});

setSelectedReg(student.reg_no);
setEditMode(true);
};

// ---------------- RESET ----------------
const resetForm = () => {
setForm({
reg_no: "",
name: "",
subject: "",
assignment_marks: "",
test_marks: "",
attendance_marks: ""
});
};

// ---------------- CALCULATE TOTAL ----------------
const calculateTotal = () => {
const a = Number(form.assignment_marks) || 0;
const t = Number(form.test_marks) || 0;
const att = Number(form.attendance_marks) || 0;

return (a * 0.3 + t * 0.5 + att * 0.2).toFixed(2);
};

// ---------------- UI ----------------
return (
<div style={{ padding: "20px" }}>
<h1>Internal Mark Allocation System</h1>

{/* FORM */}
<div style={{ marginBottom: "20px" }}>
<input name="reg_no" placeholder="Reg No"
value={form.reg_no} onChange={handleChange} />
<input name="name" placeholder="Name"
value={form.name} onChange={handleChange} />
```

```
<input name="subject" placeholder="Subject" value={form.subject} onChange={handleChange} />

<input name="assignment_marks" placeholder="Assignment Marks" value={form.assignment_marks} onChange={handleChange} />
<input name="test_marks" placeholder="Test Marks" value={form.test_marks} onChange={handleChange} />
<input name="attendance_marks" placeholder="Attendance Marks" value={form.attendance_marks} onChange={handleChange} />

<p><strong>Total:</strong> {calculateTotal()}</p>

{!editMode ? (
<button onClick={addStudent}>Add Student</button>
) : (
<>
<button onClick={updateStudent}>Update</button>
<button onClick={() => { setEditMode(false); resetForm(); }}>Cancel</button>
</>
)}
</div>

{/* TABLE */}
<table border="1" cellPadding="10">
<thead>
<tr>
<th>Reg No</th>
<th>Name</th>
<th>Subject</th>
<th>Assignment</th>
<th>Test</th>
<th>Attendance</th>
<th>Total</th>
<th>Actions</th>
</tr>
</thead>

<tbody>
{students.map((s, index) => (
<tr key={index}>
<td>{s.reg_no}</td>
<td>{s.name}</td>
<td>{s.subject}</td>
<td>{s.assignment_marks}</td>
<td>{s.test_marks}</td>
<td>{s.attendance_marks}</td>
<td>{s.total}</td>

<td>
<button onClick={() => editStudent(s)}>Edit</button>
<button onClick={() => deleteStudent(s.reg_no)}>Delete</button>
</td>
</tr>
))}
</tbody>
</table>
</div>
);
}

export default App;
```

form.js

```
import React from "react";

function StudentForm({ form, handleChange, handleSubmit, editMode, resetForm }) {

return (
<div className="card p-3 mb-4">
<h3>{editMode ? "Update Student Marks": "Add Student Marks"}</h3>

<div className="row">
<div className="col-md-4">
<input
type="text"
name="reg_no"
className="form-control"
placeholder="Register Number"
value={form.reg_no}
onChange={handleChange}
/>
</div>

<div className="col-md-4">
```

```
<input
type="text"
name="name"
className="form-control"
placeholder="Student Name"
value={form.name}
onChange={handleChange}
/>
</div>

<div className="col-md-4">
<input
type="text"
name="subject"
className="form-control"
placeholder="Subject"
value={form.subject}
onChange={handleChange}
/>
</div>
</div>

<br />

<div className="row">
<div className="col-md-4">
<input
type="number"
name="assignment_marks"
className="form-control"
placeholder="Assignment Marks"
value={form.assignment_marks}
onChange={handleChange}
/>
</div>

<div className="col-md-4">
<input
type="number"
name="test_marks"
className="form-control"
placeholder="Test Marks"
value={form.test_marks}
onChange={handleChange}
/>
</div>

<div className="col-md-4">
<input
type="number"
name="attendance_marks"
className="form-control"
placeholder="Attendance Marks"
value={form.attendance_marks}
onChange={handleChange}
/>
</div>
</div>

<br />

<button className="btn btn-primary"
onClick={handleSubmit}>
{editMode ? "Update" : "Add"}
</button>

{editMode && (
<button className="btn btn-secondary ms-2"
onClick={resetForm}>
Cancel
</button>
)}
</div>
);
}

export default StudentForm;
```

table.js

```
import React, { useState } from "react";

function StudentTable({ students, onEdit, onDelete })
{

const [search, setSearch] = useState("");

const filteredStudents = students.filter((s) =>
s.name.toLowerCase().includes(search.toLowerCase(
)) ||
s.reg_no.toLowerCase().includes(search.toLowerCas
e())
);

return (
<div className="card p-3">
<h3>Student Records</h3>
```

```
{/* Search */}
<input
type="text"
className="form-control mb-3"
placeholder="Search by Name or Reg No"
value={search}
onChange={(e) => setSearch(e.target.value)}
/>

<table className="table table-bordered">
<thead>
<tr>
<th>Reg No</th>
<th>Name</th>
<th>Subject</th>
<th>Assignment</th>
<th>Test</th>
<th>Attendance</th>
<th>Total</th>
<th>Actions</th>
</tr>
</thead>

<tbody>
{filteredStudents.map((s, index) => (
<tr key={index}>
<td>{s.reg_no}</td>
<td>{s.name}</td>
<td>{s.subject}</td>
<td>{s.assignment_marks}</td>
<td>{s.test_marks}</td>
<td>{s.attendance_marks}</td>
<td>{s.total}</td>

<td>
<button
className="btn btn-warning btn-sm"
onClick={() => onEdit(s)}
>
Edit
</button>

<button
className="btn btn-danger btn-sm ms-2"
onClick={() => onDelete(s.reg_no)}
>
Delete
</button>
</td>
</tr>
))}
</tbody>
</table>
</div>
);
}

export default StudentTable;


================================
# INTERNAL MARK ALLOCATION SYSTEM
# FASTAPI BACKEND
# ================================

from fastapi import FastAPI, HTTPException, Depends
from pydantic import BaseModel
import sqlite3
from typing import List

app = FastAPI()

# ================================
# DATABASE CONNECTION
# ================================

conn = sqlite3.connect("internal_marks.db", check_same_thread=False)
cursor = conn.cursor()

# Create tables
cursor.execute("""
CREATE TABLE IF NOT EXISTS users (
id INTEGER PRIMARY KEY AUTOINCREMENT,
username TEXT,
password TEXT,
role TEXT
)
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS students (
id INTEGER PRIMARY KEY AUTOINCREMENT,
reg_no TEXT,
name TEXT,
department TEXT,
subject TEXT,
```

```
assignment_marks INTEGER,
test_marks INTEGER,
attendance_marks INTEGER,
total_marks REAL
)
""")

conn.commit()

# ================================
# SCHEMAS
# ================================

class LoginSchema(BaseModel):
username: str
password: str


class StudentSchema(BaseModel):
reg_no: str
name: str
department: str
subject: str
assignment_marks: int
test_marks: int
attendance_marks: int

# ================================
# UTILITY FUNCTIONS
# ================================

def calculate_total(a, t, att):
return round ((a * 0.3) + (t * 0.5) + (att * 0.2), 2)


def validate_marks(a, t, att):
if a > 100 or t > 100 or att > 100:
raise HTTPException(status_code=400,
detail="Marks cannot exceed 100")

# ================================
# AUTHENTICATION
# ================================

@app.post("/login")
def login(data: LoginSchema):
cursor.execute("SELECT * FROM users WHERE
username=? AND password=?",
(data.username, data.password))

user = cursor.fetchone()

if not user:
raise HTTPException(status_code=401,
detail="Invalid credentials")

return {"message": "Login successful", "role": user
[3]}

# ================================
# ADD STUDENT
# ================================

@app.post("/students")
def add_student(student: StudentSchema):

validate_marks(
student.assignment_marks,
student.test_marks,
student.attendance_marks
)

total = calculate_total(
student.assignment_marks,
student.test_marks,
student.attendance_marks
)

cursor.execute("""
INSERT INTO students
(reg_no, name, department, subject,
assignment_marks, test_marks, attendance_marks,
total_marks)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
""", (
student.reg_no,
student.name,
student.department,
student.subject,
student.assignment_marks,
student.test_marks,
student.attendance_marks,
total
))

conn.commit()
return {"message": "Student added", "total": total}
```

```
# ================================
# GET ALL STUDENTS
# ================================

@app.get("/students")
def get_students():

cursor.execute("SELECT * FROM students")
rows = cursor.fetchall()

students = []
for row in rows:
students.append({
"id": row[0],
"reg_no": row[1],
"name": row[2],
"department": row[3],
"subject": row[4],
"assignment_marks": row[5],
"test_marks": row[6],
"attendance_marks": row[7],
"total_marks": row[8]
})

return students

# ================================
# GET STUDENT BY REG NO
# ================================

@app.get("/students/{reg_no}")
def get_student(reg_no: str):

cursor.execute("SELECT * FROM students WHERE
reg_no=?", (reg_no,))
row = cursor.fetchone()

if not row:
raise HTTPException(status_code=404,
detail="Student not found")

return {
"reg_no": row[1],
"name": row[2],
"department": row[3],
"subject": row[4],
"total": row[8]
}

# ================================
# UPDATE STUDENT
# ================================

@app.put("/students/{reg_no}")
def update_student(reg_no: str, student:
StudentSchema):

validate_marks(
student.assignment_marks,
student.test_marks,
student.attendance_marks
)

total = calculate_total(
student.assignment_marks,
student.test_marks,
student.attendance_marks
)

cursor.execute("""
UPDATE students
SET assignment_marks=?, test_marks=?,
attendance_marks=?, total_marks=?
WHERE reg_no=?
""", (
student.assignment_marks,
student.test_marks,
student.attendance_marks,
total,
reg_no
))

conn.commit()

return {"message": "Updated successfully", "total":
total}

# ================================
# DELETE STUDENT
# ================================

@app.delete("/students/{reg_no}")
def delete_student(reg_no: str):

cursor.execute("DELETE FROM students WHERE
reg_no=?", (reg_no,))
conn.commit()
```

```
return {"message": "Deleted successfully"}

# ==============================
# REPORT GENERATION
# ==============================

@app.get("/report")
def generate_report():

cursor.execute("SELECT name, total_marks FROM students")
rows = cursor.fetchall()

report = []
for row in rows:
report.append({
"name": row[0],
"total_marks": row[1],
"status": "Pass" if row[1] >= 50 else "Fail"
})

return report

# ==============================
# TOP STUDENTS
# ==============================

@app.get("/top-students")
def top_students():

cursor.execute("SELECT * FROM students ORDER BY total_marks DESC LIMIT 5")
rows = cursor.fetchall()

result = []
for r in rows:
result.append({
"name": r[2],
"total": r[8]
})

return result

# ==============================
# RUN SERVER
# ==============================

# uvicorn main:app --reload
```

```
import sqlite3

class Database:

def __init__(self):
self.conn = sqlite3.connect("internal_marks.db", check_same_thread=False)
self.cursor = self.conn.cursor()

def create_tables(self):

self.cursor.execute("""
CREATE TABLE IF NOT EXISTS subjects (
id INTEGER PRIMARY KEY AUTOINCREMENT,
subject_name TEXT,
max_marks INTEGER
)
""")

self.cursor.execute("""
CREATE TABLE IF NOT EXISTS attendance (
id INTEGER PRIMARY KEY AUTOINCREMENT,
reg_no TEXT,
percentage INTEGER
)
""")

self.conn.commit()

def insert_subject(self, subject_name, max_marks):
self.cursor.execute(
"INSERT INTO subjects (subject_name, max_marks) VALUES (?, ?)",
(subject_name, max_marks)
)
self.conn.commit()

def get_subjects(self):
self.cursor.execute("SELECT * FROM subjects")
return self.cursor.fetchall()

def close(self):
self.conn.close()


db = Database()
db.create_tables()
```

```
from fastapi import HTTPException

# Dummy admin credentials
ADMIN_USERNAME = "admin"
ADMIN_PASSWORD = "admin123"

def authenticate(username: str, password: str):
if username == ADMIN_USERNAME and password
== ADMIN_PASSWORD:
return True
else:
raise HTTPException(status_code=401,
detail="Invalid Login")

from fastapi import APIRouter
import sqlite3

router = APIRouter()

conn = sqlite3.connect("internal_marks.db",
check_same_thread=False)
cursor = conn.cursor()

# ==============================
# SUBJECT-WISE REPORT
# ==============================
@router.get("/subject-report/{subject}")
def subject_report(subject: str):

cursor.execute("SELECT * FROM students WHERE
subject=?", (subject,))
rows = cursor.fetchall()

return [
{
"reg_no": r[1],
"name": r[2],
"total": r[8]
} for r in rows
]

# ==============================
# CLASS AVERAGE
# ==============================
@router.get("/class-average")
def class_average():

cursor.execute("SELECT AVG(total_marks) FROM
students")
avg = cursor.fetchone()[0]

return {"class_average": round(avg, 2) if avg else 0}

# ==============================
# PASS / FAIL COUNT
# ==============================
@router.get("/result-summary")
def result_summary():

cursor.execute("SELECT total_marks FROM
students")
rows = cursor.fetchall()

pass_count = 0
fail_count = 0

for r in rows:
if r[0] >= 50:
pass_count += 1
else:
fail_count += 1

return {
"pass": pass_count, "fail": fail_count }
```

ACKNOWLEDGEMENT

I am very much grateful to the ALMIGHTY and to my PARENTS who have helped me all the way in my career. I thank them for their support that they have showed me throughout the project for its successful completion.

I (SOUNDARYA S) would like to extend my heartfelt thanks to our Chairman Dr. K. RAJENDRAN, MS., FICS, FAIS, who has given us an excellent opportunity to exhibit the mode of creativity by this project.

I wish to express my gratitude and deepest thanks to our dynamic and beloved Principal Dr. G. ELANGO, M.E., Ph.D., who has provided all the help needed in executing the project successfully in our college.

I convey my genuine thanks to our Vice-Principal Dr. K. RAGHU, M.Sc., M.Phil., Ph.D., who has supported in completing the project successfully.

I convey my grateful thanks to Dr. R. VIJAYALAKSHMI, M.C.A., M.Phil., Ph.D., Associate Professor & HOD of MCA department and my project advisor, for the valuable suggestion and comments that she provided apart from sharing project

expertise when it was needed from the beginning.
Finally, I would like to use this opportunity to impart my sincere thanks to all the Teaching and non-teaching staff members of MCA department and my friends for all the help provided by them, for the successful completion of this project.

REFERENCES

[1] The Times of India, "LU Revises LLB Evaluation, 75:25 Internal Ratio," *The Times of India*, 2025. Available: https://timesofindia.indiatimes.com/city/lucknow/lu-revises-llb-evaluation-sys-raises-internal-assessment-weightage/articleshow/122276132.cms
[2] University of Delhi, *Internal Assessment Scheme*. Available: https://www.scribd.com/document/418678802/University-of-Delhi-Internal-Assessment-Scheme
[3] Noyal M. J., "KTU Internal Mark Calculator," GitHub Repository, Python application based on KTU 25+15+10 marks rule. Available: https://github.com/NoyalMJ22/KTU-Internal-Mark-Calculator
[4] *Academic Evaluation Guidelines: 30 Marks Internal Breakdown*. Available: https://www.scribd.com/document/528000013/Academic-Evaluation-Guidelines
[5] *Internal Assessment Criteria for BBA: 30 Marks Split*. Available: https://www.scribd.com/document/502188381/Internal-Assessment-Criteria-For-BBA
[6] *Guidelines for Research Project Report & Viva Voce: Internal 50 + External 100 Marks Pattern*. Available: https://www.scribd.com/document/577304369/Guidelines-For-Research-Project-Report-Viva-Voce
[7] Pearson, "Coursework Moderation of Internal Components and Mark Adjustments." Available: https://support.pearson.com/en/support/solutions/articles/43000541351-coursework-moderation-of-internal-components-and-mark-adjustments
[8] *Internal Mark Assessment System – Project Paper*. Available: https://www.scribd.com/document/416103019/Internal-Mark-Assessment-System