DUTY LEAVE MANAGEMENT SYSTEM

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THEAWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

Submitted By:

Astha Sharma (2104079)

Gurleen kaur (2104101)

Gurarpit Singh (2104101)

Submitted To.:

Dr. Hardeep Singh Kang

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

November,2024

Abstract

.

The Duty Leave Management System is a web-based application designed to streamline and simplify the process of duty leave requests, approvals, and documentation for educational institutions. Built using the MERN stack (MongoDB, Express.js, React, and Node.js), the system allows students to easily submit leave requests for participation in technical, co-curricular, and sports events. Faculty members can efficiently review, approve, and manage these requests, with access to a well-organized dashboard that displays all pending and approved leaves. This organized structure helps institutions maintain clear records of student participation in various events, minimizing administrative workload and reducing the potential for manual errors.A standout feature of the system is its automated notifications, which remind students to upload participation certificates as proof of attendance, ensuring accountability and maintaining a verified record. The platform's centralized records can also be accessed for insights into student involvement, supporting analysis of trends in extra-curricular participation. With MongoDB's flexible data handling and React's responsive interface, the Duty Leave Management System provides a user-friendly experience that enhances administrative efficiency, reduces paperwork, and improves transparency. This comprehensive system supports educational institutions in fostering a supportive environment for student growth and engagement beyond academics.

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to Dr. Sehejpal Singh, Principal of Guru Nanak DevEngineering College (GNDEC), for providing us with the opportunity to undertake this major project.

The unwavering guidance and support from Dr. Kiran-Jyoti, Head of the Department of Computer Science and Engineering at GNDEC, have been invaluable in the successful execution of this project. We express our sincere thanks and appreciation for her constant encouragement.

A special mention of gratitude goes to Prof. Hardeep Singh. His profound insights andinvaluable guidance have played a pivotal role in steering this project towards successful completion.

We would also like to acknowledge the support and assistance extended by the facultymembers of the Computer Science and Engineering department at GNDEC. Their intellectual contributions have enriched our project significantly.

Lastly, we are deeply indebted to all individuals who have contributed to this project in any capacity. Their efforts have been instrumental in shaping the outcomes we haveachieved.

Thank you to everyone involved in making this project a reality.

ASTHA SHARMA

GURARPIT

GURLEEN

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Introduction to Project

The Student Duty Leave Management System is a system meant to create efficiency in the process by which a student applies for leave and manages a leave associated with technical, co-curricular, or sporting events. Students in educational institutions usually have extracurricular activities outside the scheduled time of academics; thus, some system must be in place to request and record such absences. This system is able to overcome the barriers of handling such leave requests efficiently and then ensures that the entire process is very smooth and effective for the students and administrators.

This system will be of use mainly to make it easy for the application for duty leave. By having an easy-to-use interface, the student can apply for leaving their duties, the nature of the event for which they are moving out, and the duration for which they would be absent. The process altogether eliminates the need for manual paperwork and enables the quick processing of requests. Moreover, the system ensures that all applications get tracked and reviewed systematically, resulting in negligible chances of errors or delays. Moreover, Duty Leave Management System has capable record-keeping capabilities and allows the maintenance of a comprehensive record of every student's participation in different events. Such data can measure the involvement of students and keep track of the achievements properly. The system tracks the submission of required documents, like certificates of participation, in addition to its core integration. Automated reminders for the students are sent to upload these required documents before any deadline set by the institution so as not to delay verification.

While the whole Duty Leave Management System merely decreases the administrative tangles associated with the issue of managing leave requests, it does increase the transparency and accountability involved. So, through the use of technology, these processes have been made

possible to get an even more organized and efficient management of student participation in extracurricular activities, which eventually leads to a more efficient academic environment.

1.2 Project Category

The Duty Leave Management System is an application for automating and making streamlined the leave requests of students who attend technical, co-curricular or sports events. The system is meant for educational institutions in the form of allowing each student to send a leave request efficiently through the option allowed to faculty members for approval or rejection of leave applications through an easy online workflow. This reduces human interactive processes, reduces paperwork, and helps track the status of leave clearly.

Student interface In this system, the student interface provides provision to the participant for submitting event details and uploading participation certificates, along with the current status update regarding application and whether they are being approved or rejected. It not only benefits the student in terms of clear communication but also assists faculty as it provides a structured process for managing requests, approvals, and records.

Built using the MERN stack—MongoDB, Express.js, React.js, and Node.js—the application draws on robust, scalable technologies while being well-tuned for a user-friendly experience for students and faculty. This tech stack offers a responsive interface, secure management of data, and smooth user experience, making it a valuable solution to education institutions. Outside of education, the flexibility of the application allows it to be adopted by organizations that maintain similar leave management protocols, thus providing a very efficient, scalable alternative to the traditional ways of doing it.

In all, the Duty Leave Management System saves administrative jobs, increases transparency, and fosters effective communication. This innovation may possibly find wider adoption by institutions requiring systematic leave management solutions.

1.3 Problem Formulation

The inefficiency and complexity of traditional approaches in manually managing the duty leaves of students to participate in technical, co-curricular, and sports events motivated the development of Duty Leave Management System. In the traditional approach, paper-based applications or e-mails used to be submitted by students, and the whole process consumed lots of time while tracking, reviewing, and finally making approvals from faculty and administrative staff during processing. This manual process leads to delays, lost records, and, most importantly, lack of centralized oversight. This process brings inconvenience to students as well as to staff.

An unstructured way to upload participation certificates or other documents required through the current system does not exist. Incomplete records result in administrative follow-ups. In the absence of an organized system, educational institutions find it challenging to analyze and report student engagement in extracurricular activities, which would otherwise provide the valuable insight into participation trends and departmental needs.

The intended Duty Leave Management System would therefore automate and centralize this process, thus allowing quick submission, real-time tracking, automated notifications, and easy access to historical data. In respect of the foregoing, the system addresses these inefficiencies and saves time, reduces administrative workload, and enhances the experience for students and faculty alike.

1.4 Identification/Recognition of Need

Student participation in extracurricular activities, which includes technical, co-curricular, and sports events, is vital for all-round development at educational institutions. The system faced problems in handling duty leaves of students while there has been a rise in the participation of students. There is an urgent need for an easy management system so that

students can participate in such events with ease and not with unnecessary delays or administrative hurdles in procuring their duty leaves.

As it is, institutions are handling the leave application manually; this leads to inefficiency in terms of too much time spent processing approvals, lost applications, and also wrong records. The students thus risk their applications getting lost or taking ages before they get approved while faculty members use this as a multi-tasking avenue for academics and other issues about leaves that also take up their time. Students' participation levels in such subjects are affected when not streamlined, and their actual personal and professional developments are thus affected.

This would improve efficiency in operations, record-keeping, and create a fair and timely process in the approval of duty leaves to create an environment that supports students' development. This need would be met by the Duty Leave Management System, which would ensure that the duty leave requests are processed digitally in an organized and streamlined manner.

1.5 Existing System

In most educational institutions, the exiting system dealing with student duty leaves is usually a manual paper-based process or informal communication approach. Students seeking to participate in extracurricular activities such as technical events, co-curricular programs, or sports often have to write leave applications to their departments or faculty advisors. These requests go through the review authority, which might be faculty members or administrative staff who require a higher level of approvals. The processes usually involve checking one's participation, cross-checking academic calendars, and is thus time-consuming and error-prone.

This manual system is prone to several inefficiencies like misplaced or lost applications for

leave, delayed approval because one is busy or cannot access the information pertinent to the request, and no centralized records for monitoring student participation in events. Faculty members are continuously bombarded with trying to reconcile academic obligations with the administrative paperwork of leave requests.

Further, students are not clearly informed about the status of their applications as there is no central system that could inform them about the status at all times until the last minute. This is therefore not easy for the students in preparing or committing in event participation. Yet, this impact will extend to the tracking of participation metrics by the institution and the evaluation of which students are effectively engaged in extracurricular activities.

Essentially, the current manual system is not suitable, time-consuming, and cannot be integrated with the automation to effectively support students and faculty. A digital solution is in high demand to simplify the process, so institutions can operate efficiently while managing duty leave requests with transparent record-keeping and ensuring everything is documented.

1.6 OBJECTIVES

1. To enable students to easily apply for duty leave when participating in technical, co-curricular, or sports events, ensuring a smooth and efficient process
2. To maintain a detailed and organized record of student participation across various events, facilitating easy access to historical data and participation analytics
3. To provide automated alerts to students, reminding them to upload participation certificates

1.7 Proposed System

The proposed Duty Leave Management System, DLMS, is a web application intended to automatically and efficiently process requests for duty leave by students in co-curricular, technical, and sports activities. It has been developed using the MERN stack: MongoDB, Express, React, Node.js, so that its implementation overcomes any shortcomings of manually done processes and thus puts forward in view an efficient, transparent, and accessible solution for all stakeholders.

The system gives an intuitive interface which assists the students to send simple requests for the duty leave under the event details along with supporting documents for verification purposes. After that, faculty and administrative staff can view and approve or reject requests based on a set of criteria. The student will receive automated notifications as regards the status of their applications in clear communication upon approval or rejection.

DLMS also boasts a comprehensive record-keeping feature, where it collects all extensive details of every leave request: date, event type, and approval status. Access is quite easy for faculty and administration to follow the history regarding engagement and participation by a student. It further holds an automatic alert reminding students to upload certificates of participation after the end of the event in ensuring that the necessary documentation is all there and accessible.

The tools that enable analysis of data allow the institution to evaluate trends in students' participation, thus providing valuable insights into activities and assisting the institution in making informed choices based on data, always with a view to student development. The DLMS reduces the time-consuming effort of manually performing most tasks and eliminates unnecessary paperwork, creating a streamlined and efficient process that heightens the overall educational experience.

1.8 Unique Features of the Proposed System

Unique features of the proposed DLMS include an automated request and approval workflow.

In this way, the request to submit and review leave does not rely on the nature of individual employees or on lengthy paper records but through the system, it is hastened and made prompt for better and efficient service. Students can upload relevant documents such as certificates directly obtained from the events; this also makes it easier for faculty to validate leave requests. Further, the system provides real-time notifications and alerts which keep parties-in-study and faculty-informed at all stages, including submission and approval of leave requests and reminders for uploading certificates post-event participation.

What is sought to be represented above is the tracking of participation in co-curricular, technical, and sports events by each student. This helps faculty keep track of student participation, thus allowing them to create an insight for students. The admin dashboard of the system equips administrators with robust tracking features of leave requests with analyzing participation trends as well as generating reports which assist them in making better decisions and oversight. The student self-service portal helps the students independently track the status of their leave requests and also to check participation history, thus bringing in more transparency and user autonomy.

The system also allows customizable leave categories to include any type of variety of leave available; thus, it has options for technical, co-curricular, and sports leaves, among others, tailored to the needs of different institutions. Role-based access controlensures that only authorized users will manage or approve leave requests, providing data security and privacy. There is also mobile responsiveness so that users may access and manage requests from anywhere with greater convenience and flexibility. Last, but not least, the system includes automated certificate upload reminders so that students process the essential documents after events are held, making the entire leave management process complete. These unique features together culminate in a more efficient, user-friendly, and comprehensive system for the management of duty leave requests.

Chapter 2: Requirement Analysis and System Specification

2.1Feasibility Study

Technical Feasibility

The technical feasibility of the proposed Duty Leave Management System (DLMS) is high. The system will be developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js). This is one of the most adopted and robust technology stacks for developing scalable and efficient web applications. The database would be implemented using MongoDB so that there is flexibility in the storage of leave request records and other dynamic data. Node.js and Express.js could be used to author the scalable and performance-oriented backend for serving API requests. React.js would be quite excellent for building a responsive and interactive frontend for the system. File upload functionality will be included to enable students to upload event participation certificates by using scalability in cloud storage solutions. The system design should also be mobile responsive, meaning that it can work well on a smartphone, tablet, or desktop, ensuring that faculty and students can interact with the system from any device.

This will enable the development team to leverage the immense availability of resources and libraries based on modern, open-source technologies used, which will speed up the development process and significantly reduce the chances of hitting major technical roadblocks. The project will also include real-time notifications and alerts, which can be achieved using well-established tools, such as Socket.io or even email integration, to ensure smooth communication among students and faculty.

Economical Feasibility

The proposed system is also economically feasible, and open-source technologies of MongoDB, Express.js, React.js, and Node.js are used; there are no licensing fees to incur. It is very cheap in terms of development costs because these tools are free and quite popular, with a large number of developers continuously working on the support and improvement activities. In this project, one

does not need proprietary software or hardware along with its licenses that would demand a substantial expense.

Further, it can be hosted on inexpensive cloud platforms such as Heroku, AWS, or DigitalOcean and also provides scalable solutions at a fraction of the cost of traditional server configurations. MongoDB Atlas comes with free tier that would support initial database needs; then one can scale up when there is an increase in users. Thus the development and the maintenance costs can be kept within a modest budget so it would be economically feasible for institutions of education.

Operational Feasibility

The operational feasibility of the DLMS is also high. It will greatly enhance the efficiency in managing the requests for duty leave by the system. This will be achieved by automating the workflow, reducing the administrative overhead, and minimizing human errors. The automated workflow on leave request submission, approval, and document upload enhances the overall operation workflow of the institution.

In terms of user adoption, it is user-friendly to both faculty and students, whose very intuitive interfaces require minimal training. Having a mobile-responsive design allows users to access the system anywhere and at any time, making it convenient for students and faculty to interact with it in real-time-whether they are at school or otherwise. Such is one of the most crucial aspects of its accessibility in attaining operational success.

The admin dashboard also provides all tools to check requests for leaves, analyze trends, and generate reports, which are very important factors for smooth operation and minimal training requirement and operational overhead.

The system also integrates with existing institutional infrastructures, such as a student database or authentication systems, so that data easily flows in with no operational friction.

The proposed Duty Leave Management System is quite feasible from the technical, economic, as well as the operational points of view; it will also give a scalable, cost-effective, and user-friendly facility for improving leave request management at educational institutions.

2.2 Software Requirement Specification Document

The Duty Leave Management System (DLMS) shall effectively manage faculty and student leave requests with smooth operation, strong security, and performance.

Data Requirements

- User Information: This information will be given for each user such as the student, faculty, and administrator. It will include personal details like name, email, department, and contact number for identification and verification purposes. The faculty members will add further information regarding the departments to manage leave requests.
- Leave Data: The system will keep records of the following details regarding the leave requests: event name, leave dates, request status (approved, pending, rejected), any accompanying documents and comments if approved or rejected.
- Documentation Data: Leave requests should be accompanied by attaching supporting documents such as participation certificates, event details and others that are upload by students for validation purposes.
- Role and Access Control: Admins should have the control in managing the leave requests, user roles, and access levels where they can approve/reject a request made

Functional Requirements

- User Authentication: The system should be able to support user registration, login, and profile management. Using role-based access control (RBAC), only the students should be able to submit and track the requests, while faculty can approve/reject, and admins will have full control of the system.
- Leave Request Management: Users could submit, view, and track their leave requests, while admins and faculty can approve/reject the requests, view reports, and send notifications to the users.
- Upload and Approve Documentation with Leave Requests: The system would include

upload and validation of related documents with the leave requests. Faculty and admin users would validate uploaded documents before approving the requests.

- Reports and Analytics: The system would include leave trends, attendance reports, and history of requests. The reports should be customized for admins.
- Notifications: Status updates via e-mail/SMS for requests, reminders for submission of documents, and other alerts that may be connected to an event.

Performance Requirements

- Speed: The system pages open and submit leave requests within less than 2 seconds. The retrieval and processing of data, both related to leave request and reports, will be optimized for response times.
- Scalability: The system will support up to 1,000 concurrent users scalable as high as 5,000 in anticipation of the growth of the institution.
- Availability: High availability in terms of as high as 99.9% uptime is guaranteed especially during peak times such as a new academic semester when leave requests are highest.

Dependability Requirements

- Reliability: The system should store and retrieve the data of leave request reliably without losing data, even when high usage takes place.
- Fault Tolerance: The system should recover from server-side errors, database failure errors, and wrong input from the user end by displaying proper error messages and fallback behaviors if that's the case.

Maintainability Requirements

- Modular Codebase: The MERN stack MongoDB, Express.js, React, Node.js) will ensure that updates, debugging, and expansion of the system are modular.
- Documentation: Each component of the system-front-end, back-end, database-will be well documented. It will use version control via Git and GitHub, which will allow the team to cooperate easily and track code changes.

Security Requirements

- User Authentication and Authorization: All passwords will be hashed using secure algorithms like bcrypt, and the authentication process will be token-based JWT to ensure safe authentication.
- Data Encryption: Any personal information as well as other leave-related documents will be encrypted at both their transit via HTTPS and rest using some database encryption techniques.
- Access Control: Role-based access control would ensure that some of the functionalities are restricted to certain users. All the admins will have full-access functionalities; faculty members would be able to accept/reject the leave requests submitted; whereas students could only submit and track their own leave requests.
- Audit Logs: All the Leave submissions, approvals, and rejections made by each and every user will be logged into the system. This would ensure accountability and create trail audits in case of any discrepancies.

Look and Feel Requirements

- User Interface (UI): It ought to be user-friendly, intuitive, and responsive to all devices-desktop, tablet, and mobile. A clean layout with intuitive navigation is the way forward. The UI will focus on experience with minimal training required..
- Brand Consistency: Overall design for the system will hold in line with the branding rules specified by the institution, thus professional yet consistent throughout all pages.

Software Requirements

1. Operating System
   - Development Environment:
     - Windows 10/11 or macOS or Linux
     - Purpose: Provides a platform for development tools and libraries.

2. Frontend Development

- React.js
    - A JavaScript library for building user interfaces.
    - Purpose: To create dynamic and responsive web applications.
- JavaScript
    - The progmming language used for client-side scripting.
    - Purpose: To enable interactivity and dynamic content on web pages.
- HTML
    - The standard markup language for creating web pages.
    - Purpose: To structure the content of the web application.
- CSS3
    - A style sheet language for describing the presentation of the document.
    - Purpose: To style and layout the application, ensuring it is visually appealing.
- CSS Frameworks (Bootstrap or Material-UI)
    - Frameworks for responsive design.
    - Purpose: To speed up the UI development process with pre-designed components.

3. Backend Development

- Node.js
    - A JavaScript runtime built on Chrome's V8 JavaScript engine.
    - Purpose: To execute JavaScript on the server side, handle requests, and serve APIs.
- Express.js
    - A web application framework for Node.js.
    - Purpose: To simplify the process of building robust web applications and APIs.
- MongoDB
    - A NoSQL database for storing application data.
    - Purpose: To store and manage leave requests and user information.
- Mongoose
    - An ODM (Object Data Modeling) library for MongoDB and Node.js.
    - Purpose: To provide a schema-based solution to model application data.

4. Authentication and Security

- JWT (JSON Web Token)
    - A compact URL-safe means of representing claims to be transferred between two parties.
    - Purpose: To implement secure authentication and authorization.
- bcrypt.js

    - A library to hash passwords.
    - Purpose: To securely store user passwords in the database.

5. Development Tools

- Visual Studio Code
    - A source-code editor with support for debugging, syntax highlighting, and version control.
    - Purpose: To provide an IDE for writing and managing code.
- Git
    - A version control system for tracking changes in code.
    - Purpose: To collaborate with other developers and maintain version history.
- Postman
    - A tool for testing APIs.
    - Purpose: To test and debug RESTful APIs during development.

Hardware Requirements

1. Development Environment

For developers working on the application, the following hardware specifications are recommended:

- Processor (CPU):
    - Minimum: Dual-core processor (e.g., Intel i3 or AMD Ryzen 3)
    - Recommended: Quad-core processor (e.g., Intel i5 or AMD Ryzen 5)
- RAM:
    - Minimum: 8 GB
    - Recommended: 16 GB or more
- Storage:
    - Minimum: 256 GB SSD (Solid State Drive) for faster read/write speeds
    - Recommended: 512 GB SSD or larger, especially if using large datasets or multiple projects
- Graphics Card (GPU):
    - Integrated graphics are sufficient for frontend development.
    - Recommended: Dedicated graphics card if doing extensive graphical work (optional).
- Display:
    - Minimum: 15-inch display with a resolution of 1920x1080 (Full HD)

    - Recommended: Dual monitor setup for improved productivity (optional).
- Network:
    - Stable internet connection for accessing online resources, cloud services, and collaboration tools.

2. Deployment Environment

For the server that will host the Duty Leave Management System, the following specifications are recommended:

- Processor (CPU):
    - Minimum: Quad-core processor
    - Recommended: Hexa-core processor or higher
- RAM:
    - Minimum: 4 GB
    - Recommended: 8 GB or more, depending on the expected number of concurrent users
- Storage:
    - Minimum: 20 GB SSD or HDD (sufficient for small to medium applications)
    - Recommended: 50 GB SSD for faster performance and handling larger data loads
- Network:
    - Minimum: 1 Gbps network interface for handling multiple user requests.
    - Recommended: High-speed internet connection with adequate bandwidth to support multiple simultaneous users.

2.3 SDLC Model to be Used

In terms of the Duty Leave Management System (DLMS), following the Agile Software Development Life Cycle (SDLC) model is a recommendation. Agile software development concentrates more on the process of fast incrementalization by flexible planning that helps in overcoming change through continuous improvement, customer collaboration, and responding to changes in requirements. The project will be more suitable to select this approach as it is going to need updates or adjustments based on its feedback and ever-evolving requirements.

Agile Iteration: The changing needs of the DLMS, because feedback from the users of the system-who are faculty members, students, and admins-is considered, means Agile uses iterative development to allow incremental growth with every iteration.

Regular Deliveries: Agile allows for frequent releases so that the system can be developed in manageable increments. Even early versions of the system can be deployed, tested, and improved over time.

Stakeholder Involvement: Developers have close cooperation with stakeholders, so the system will be built to ensure that what is delivered actually meets the needs of users, and faculty and admins can comment throughout the development process.

Faster Time-to-Market: As the system develops incrementally in manageable steps, new features or updates can be delivered quickly; this could be considered critical where urgency arises, such as implementing urgent leave requests during the middle of an academic year.

Phases of the Agile SDLC Model for DLMS

1. Planning & Requirements Gathering:

The first iteration will include gathering core requirements of the Duty Leave Management System. This would be the faculty, students, and administrator requirements; What are the must-to-have features which include requests for leave, approval procedures, uploading of documents, and notification.

It is the process of gathering high-level requirements and defining the scope for the first release.

2. Design:

After a set of requirements has been gathered, a basic system design will be developed; it would involve architectural decisions, designing the database schema, and UI/UX design for the frontend of the system.

- The system would be iteratively prototyped so that the user interfaces and interactions would be fine-tuned to be user-friendly both for students and faculty members.

3. Implementation:

Major features of DLMS like leave request submission, approval processes, uploading of documents and sending of notifications would be there in the first sprint.

The database and server-side logic will be taken care of by backend development through the MERN Stack stack, this includes MongoDB, Express.js, React.js, and Node.js.

Front-end development will be used to ensure that the components are interactive, with a user-centric approach, through the use of React.

Integration of continuous testing into each sprint. Every increment will contain unit testing, integration testing, and UATs, it will ensure the system performs according to what is desired.

Automated testing will be used to validate the correctness of such business functionality as the submission of a leave request and verification of documents.

Any usability problems or missing features will be identified from real-user feedback, students, faculty and admins collected in the course of user acceptance testing.

5. Deployment:

Features developed will be deployed after each sprint and either set for internal testing or else to a staging environment for further user validation.

Once the core features are stabilized, the system is rolled out to production in phased deployments to minimize downtime.

6. Feedback & Iteration:

Collect feedback after deployment from students, faculty, and administrators on their experience using the system.

Features or functionalities that need improvement will be of high importance and therefore included in the following sprint to ensure that the system evolves based on user needs and expectations.

New features such as advanced reports, notifications, or analytics can be added on feedback received.

7.Maintenance

After the first deployment, the system shall be in the maintenance phase, wherein all its updates and bugs are instituted based on need.

The Agile model ensures that regular updates of the system improve its performance, add new

features, or take care of newly arising problems.

In fact, by taking up the Agile SDLC Model, the Duty Leave Management System (DLMS) would be built iteratively and flexibly, providing early releases and ongoing updates. This will favor continuous improvement while ensuring that the system is always in line with user needs and changing requirements encountered and encountered in the process of developing it.

Chapter 3: System Design

3.1 Design Approach

The Duty Leave Management System adopts an Object-Oriented Design (OOD) methodology, which is particularly well-suited for creating modular, scalable, and maintainable software systems. This design approach helps in dividing the system into distinct, self-contained objects that each represent a specific aspect of the system's functionality. The use of OOD allows for better code organization, reusability, and the ability to manage complexity by encapsulating related data and behavior within objects. For the Duty Leave Management System, the design is based on several key objects, each focusing on a critical function within the platform.

1. User Object: The User object is responsible for managing the details of the students or faculty members who are requesting leave. It stores essential attributes like the user's name, contact information, department, and role (student or faculty). This object also contains methods that allow users to submit leave requests, check the status of their requests, and update personal information. The user can interact with the system by requesting leave for participation in academic or extracurricular activities, and this object helps track all their leave data.

2. LeaveRequest Object: The LeaveRequest object encapsulates all the data related to an individual leave request. It includes attributes such as the type of leave (e.g., technical, co-curricular, sports), the dates for which leave is being requested, the reason for the leave, and the current status (pending, approved, rejected). This object is central to the core functionality of the system, as it handles the creation, modification, and status tracking of leave requests. Methods within this object allow users to submit new leave requests and administrators to approve or reject them based on the organization's guidelines.

3. Admin Object: The Admin object handles the administrative functions of the system. It allows admins to manage users (students and faculty), review leave requests, and make approval or rejection decisions. The admin object also plays a vital role in maintaining the overall integrity of the system,

ensuring that leave requests are processed according to policies and that the system is running smoothly. The admin can also generate reports on leave trends, user participation, and system activity, helping streamline operations.

4.Approval Object: The Approval object manages the approval or rejection process for leave requests. It includes the logic for validating leave requests, ensuring they meet the necessary criteria (e.g., eligibility, valid dates, etc.), and updating the status accordingly. This object also stores records of approved or rejected leave requests, ensuring that administrators can easily track leave decisions over time.

By implementing this object-oriented design, the Duty Leave Management System maintains a clean and organized structure. Each object is responsible for its own functionality, reducing the complexity of the system as a whole. This modular design approach allows for easy updates, as changes to one part of the system (e.g., user management) do not require major changes to other parts (e.g., leave requests). Additionally, it supports future expansion, as new objects and features can be added without disrupting the existing system, ensuring long-term scalability and maintainability.

3.2Detailed Design

This section provides detailed design documentation for the project using structured analysis and various diagrams. The design addresses each of the project's objectives through detailed illustrations.

3.2.1 Work flow Diagram

The Duty Leave Management System starts with the student logging in and requesting leave for an event. The student fills in the event details, including dates and necessary documents, then submits the form. The system validates the information and sends the request for approval to the designated faculty or staff. The faculty or staff reviews the request and either approves or rejects it, as long as comments if rejected. Finally, the system updates the leave status and notifies the student about

the decision, ensuring an efficient process for managing, reviewing, and tracking leave requests within the system

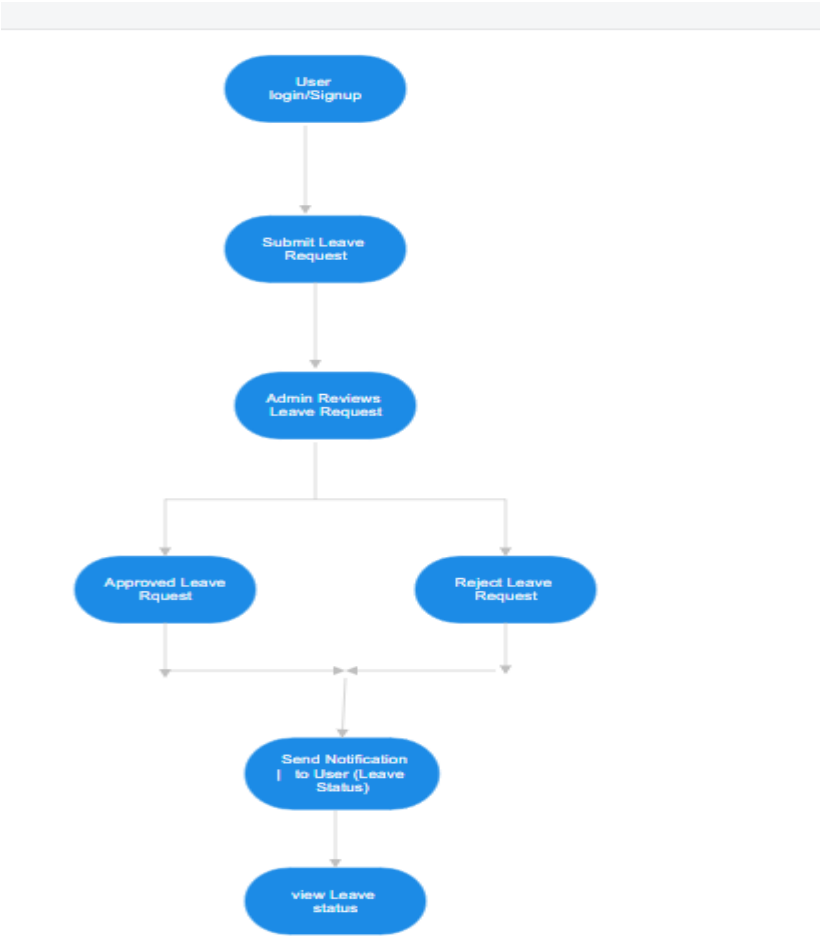

Figure3.1 Workflow diagram

3.2.2 Flow Chart

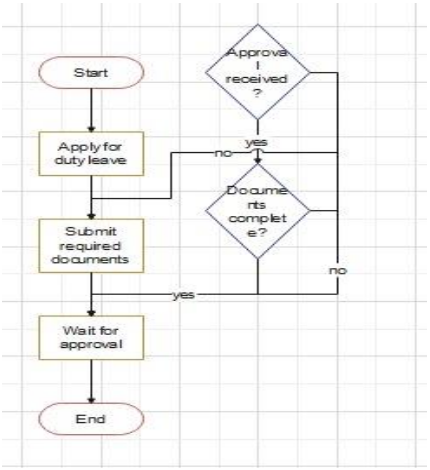

Fig3.2 Flowchart of Duty Leave Management

The Flow Chart outlines the steps involved in building a website, focusing on resume details.

Here'sa brief explanation of each step:

- Start: The process begins with the user accessing the system.
- User Login: Users (students or HOD) log into the system with their credentials.
- Is User Student?: The system checks if the logged-in user is a student or an HOD. Based on the user type, the appropriate dashboard is displayed.
- Student Dashboard:

  Students can view their leave requests and submit new leave requests.

  When submitting a new leave request, they fill out the request form.

  After submitting the request, a confirmation page is shown.
- HOD Dashboard:

  HODs can view all leave requests and approve or reject them.

  Upon approval or rejection, notifications are sent to the respective students.

  HODs can also view statistics related to leave requests.

End: The process concludes after confirmation or the HOD actions

3.2.3 Communication Diagram

Figure3.: User Communication Diagram

3.2.4Database Design: ER Diagram

ER diagram design for a Duty Leave Management This structure includes entities and relationships suitable

for a college or institution where students and faculty can request leaves for events and where these requests need approval.

Entities and Attributes

1. **Student**
    - StudentID (PK)
    - Name
    - Email
    - Department
    - Program
    - Year

- **Faculty**
    - FacultyID (PK)
    - Name
    - Email
    - Department
    - Position

- **Event**
    - EventID (PK)
    - EventName
    - EventType (Technical, Co-curricular, Sports)
    - Location
    - StartDate
    - EndDate

- **DutyLeaveRequest**
    - RequestID (PK)
    - RequestDate
    - Status (Pending, Approved, Rejected)
    - SubmissionDate

- StudentID (FK to Student)
- EventID (FK to Event)
- FacultyID (FK to Faculty who approves)

- **ParticipationCertificate**
  - CertificateID (PK)
  - UploadDate
  - FilePath
  - RequestID (FK to DutyLeaveRequest)

- **Relationships**
- Student “submits” DutyLeaveRequest
  - A Student can submit multiple leave requests, but each leave request is specific to one Student.
- DutyLeaveRequest “is approved by” Faculty
  - A Faculty member can approve multiple leave requests.
- DutyLeaveRequest “is for” Event
  - A leave request is associated with one Event.
- DutyLeaveRequest “has” ParticipationCertificate
  - Each duty leave request can have one associated certificate, which students upload as proof of participation.
- Cardinality:
  - A Student can submit multiple DutyLeaveRequest (1).
  - A Faculty can approve multiple DutyLeaveRequest (1).
  - Each DutyLeaveRequest is related to one Event (M:1).
  - Each DutyLeaveRequest can have one ParticipationCertificate (1:1).

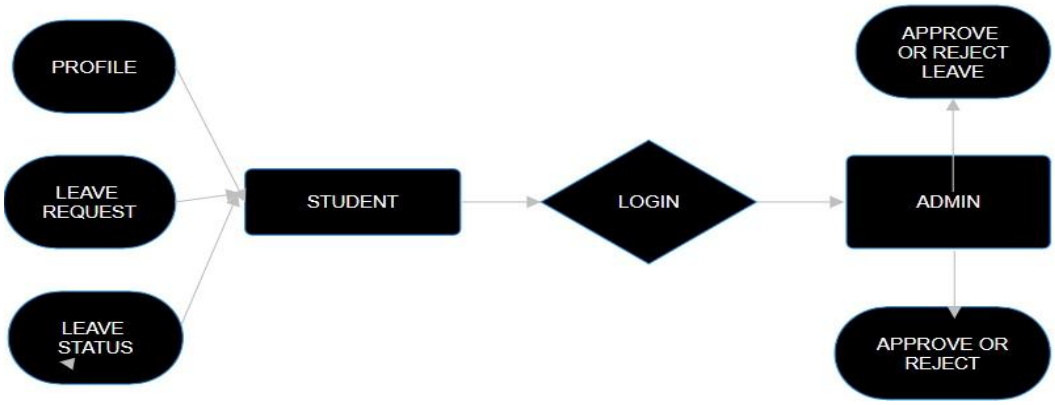

Fig 3.4 ER Diagram

3.3 User Interface Design

The User Interface (UI) Design for the Duty Leave Management System (DLMS) is crafted with a focus on simplicity, efficiency, and ease of use for both students and administrators. The design ensures a smooth, user-friendly experience for all users, prioritizing functionality and accessibility.

For students, the UI provides a clean login/signup page followed by a personalized dashboard. The dashboard includes sections like "Upcoming Leave," "Leave History," and "Request Leave," clearly labeled for easy navigation. The "Upcoming Leave" section shows all approved leave requests, while the "Leave History" section displays past requests and their statuses. A prominent "Request Leave" button allows students to submit new leave requests, with an intuitive form that guides them through selecting the leave type, dates, and reason. The system helps students by checking for any conflicts, ensuring accuracy before submission.

For administrators, the UI provides a dashboard where they can easily manage leave requests. The dashboard includes sections such as "Pending Requests," "Approved Requests," and "Rejected Requests," allowing administrators to filter and review requests. Each request contains essential details like student name, leave type, and reason, with options to approve or reject the request, along with a comment for transparency. The admin interface also allows batch processing for managing multiple requests efficiently.

The UI is designed to be responsive, ensuring consistent functionality across devices such as desktops, tablets, and mobiles. It follows modern design principles with a clean layout, intuitive navigation, and accessible features, including readable fonts and clear icons for easier interaction. The color scheme uses green for approved requests, blue for pending, and red for rejected, aiding in quick status identification.

Overall, the DLMS UI design balances functionality with ease of use, ensuring that both students and administrators can efficiently interact with the system.

3.4 Methodology

This Duty Leave Management System would be developed using an Agile methodology approach based on flexibility, collaboration, and iterative progress. It was adopted because a system based on feedback from end-users, who could engage with frequent and continuous improvement, will be capable of addressing changing needs over time in order to improve the delivery of the project. This development was split into time-boxed sprints, typically in the range of 2 to 4 weeks, with working features delivered at the end of each sprint. The team's communication was kept as transparent as

possible by using daily stand-ups and sprint retrospectives that followed the progress curve and anticipated difficulties pretty soon.

The project used a structured lifecycle for development during planning and gathering requirements wherein the user needs were understood well. The design phase of the system concentrated on system architecture, API design, and database structure. The implementation phase was iteratively conducted wherein user authentication, leave request management, and approval workflows were included. There was continuous testing throughout the process, which involved unit testing, integration testing, and user acceptance testing to ensure that the system functionality is correct and of high quality.

The deployment was done in a cloud environment that mainly ensured scalability and accessibility. The processes of continuous integration and deployment were followed to automate testing and the process of deployment, making it relatively smooth to upload any updates or bug fixes. There were regular maintenance sessions after deploying the application, incorporating updates and even introducing new features based on suggestions from users. The tools as React.js, Node.js, MongoDB, GitHub, and Postman played crucial roles during development and ensured that what was done was integrated smoothly into the whole project.

Generally, the Agile methodology would allow for constant iteration, feedback, and improvement in order to ensure that the Duty Leave Management System would be functional and adaptable to the needs of its users.

Chapter 4: Implementation and Testing

4.1 Introduction to Languages, IDEs, Tools, and Technologies Used

Modern programming languages, frameworks, tools, and technologies were used to develop the Duty Leave Management System so that it could offer high performance, scalability, and ease of use. Here, below is a detailed description of the core technologies used in the project:

1. Programming Languages and Frameworks

JavaScript: JavaScript is the core programming language that both front and back-end of the system is worked upon. It is imperative that the UI and server-side functionalities work in sync, and that is achievable with JavaScript. The possibility of handling client-side and server-side operations using Node.js makes this programming suitable for MERN stack development.

React.js: React.js was used for front-end development. As the developers were leveraging React, they could thus create UI elements that are dynamic and reusable with a component-based architecture. This made the application's front end quite interactive and efficient. Added to this was that navigation between pages did not involve page reloads with the help of the React Router.

Node.js: For the back-end, the application was completed using Node.js-a JavaScript runtime environment. Node.js is particularly suitable for real-time applications. It has a non-blocking, event-driven architecture, which makes it well-suited to multiple concurrent requests, and that was exactly the case with such a system.

Express.js: Express.js is back-end web framework used; it makes routing and handling HTTP requests very easy; the back-end framework is light in weight, and it also permits the fast creation of RESTful APIs. The front and back end are wired in such a way that it allows a leave request to submit, updating the status followed by an approval workflow.

MongoDB: The NoSQL database used to hold the data of the system; the flexible, JSON-like format or BSON is particularly effective in dealing with varied structures such as user profiles, records on leave, and the corresponding approval workflows. Its high scalability allows it to scale up as an institution's needs grow.

2. IDE (Integrated Development Environment

Visual Studio Code (VS Code): This has been the IDE where I did all my development work. It has huge extensions, and the support is given with features such as syntax highlighting, code suggestions, debugging tools, and integrating version control. As its light nature and with the wide plugins' help, I always had a prior preference in using VS Code for coding and project handling activities.

3. Version Control System

Git & GitHub: Git, in tandem with GitHub, managed to handle version control so changes in code could be traced, collaboration manipulated, and the smooth running of the development process maintained. Managing pull requests, tracking issues, and on GitHub, a shareable code base helped maintain team efficiency and ensured the integrity of the code in place.

4. Tools and Technologies for Testing

Postman: This is API testing, which will test and validate back-end APIs. With Postman, you can simulate a client request on the server and inspect the response as well as debug all the issues in API calls, thereby ensuring that all endpoints for creating, retrieving, and updating leave requests are working just fine.

MongoDB Compass: MongoDB Compass is a graphical user interface for managing MongoDB databases. The tool was used to check the performance of the database, inspect data, and run queries during development. It makes managing the MongoDB database much easier while allowing the database to store and retrieve information more effectively.

5. Hosting and Deployment

MongoDB Atlas : MongoDB's cloud version, MongoDB Atlas, was used as a database hosting platform. It provides the benefits of auto-scaling and strong security features besides providing a stable infrastructure for cloud data management.

Heroku: The Heroku hosting was used for the application during the deployment phase. It integrates very easily with GitHub, scales automatically, and also has simple configuration for a Node.js application.

4.2 Algorithm/Pseudocode Used

In the Duty Leave Management System, various algorithms and logic are used to implement core functionalities such as submitting leave requests, processing approvals, and notifying users. Below are the essential algorithms and pseudocode used in the system

1. Leave Request Submission Algorithm

Purpose: Handles the submission of a leave request by a user (faculty or staff).

Pseudocode:

plaintext

Function submit Leave Request(user, start Date, end Date, leave Type, reason):

If not is Authenticated(user):

Return "Authentication failed. Please log in."

If is Valid Leave Dates(start Date, end Date):

Leave Request = create Leave Request(user, start Date, end Date, leave Type, reason)

Save leave Request to database

Notify Admin about the new leave request

Return "Leave request submitted successfully."

Else:

Return "Invalid leave dates. Please check the dates and try again."

Is Authenticated: Checks if the user is logged in.

Is Valid Leave Dates: Verifies that the leave dates are within acceptable bounds and do not conflict with other requests.

Create Leave Request: Creates a new leave request record with the provided details.

Notify Admin: Sends a notification to the admin for further action.

2. Leave Request Approval Algorithm

Purpose: Handles the approval or rejection of a leave request by the admin.

Pseudocode:

Function approve Leave Request(admin, leave Request Id):

If not is Admin Authenticated(admin):

Return "Admin authentication failed. You do not have permission."

Leave Request = get Leave Request By Id(leave Request Id)

If leave Request is not found:

Return "Leave request not found."

If is Eligible For Approval(leave Request):

Update leave Request.status to "Approved"

Save updated leave Request to database

Notify User that leave request has been approved

Return "Leave request approved."

Else:

Update leave Request.status to "Rejected"

Save updated leave Request to database

Notify User that leave request has been rejected

Return "Leave request rejected."

Is Admin Authenticated: Verifies the admin's credentials to approve or reject requests.

Get Leave Request By Id: Fetches the leave request using its unique identifier.

Is Eligible For Approval: Validates if the leave request meets all necessary conditions for approval (e.g., no conflicts, within leave balance).

Notify User: Sends the final decision (approved/rejected) to the user.

3. Notification Management Algorithm

Purpose: Handles sending notifications to users or admins regarding status updates (e.g., approval or rejection of leave requests).

Pseudocode:

Function send Notification(recipient, message):

If recipient is not valid:

Return "Invalid recipient."

notification = create Notification(recipient, message)

Save notification to database

Send notification to recipient (via email, SMS, or in-app notification)

Return "Notification sent successfully."

Create Notification: Creates a notification record with recipient details and message.

Save notification: Stores the notification in the database for record-keeping.

Send notification: Sends the notification through the appropriate channels (e.g., email or SMS).

4. Leave Request History Retrieval Algorithm

Purpose: Allows users to view the history of their leave requests.

Pseudocode:

Function get Leave History(user):

If not is Authenticated(user):

Return "Authentication failed. Please log in."

Leave History = fetch Leave Requests By User(user)

If leave History is empty:

Return "No leave requests found."

Return leave History

is Authenticated: Ensures the user is logged in before retrieving their leave history.

fetch Leave Requests By User: Retrieves all leave requests for a specific user from the database.

5. Conflict Detection for Leave Requests

Purpose: Detects any conflicts between requested leave dates and already existing approved leaves.

Pseudocode:

Function check Leave Conflict(start Date, end Date, user Id):

Conflicting Requests = fetch Leave Request By Date Range(start Date, end Date)

For each request in conflicting Requests:

If request.user Id == user Id:

Return True // Conflict detected

Return False // No conflict

Fetch Leave Requests By Date Range: Retrieves all leave requests that overlap with the requested date range.

Conflicting Requests: Checks if any of the overlapping requests belong to the same user.

4.3 Testing Techniques

In the Duty Leave Management System, testing is essential to ensure that the system functions as expected and meets the requirements set forth by the stakeholders. Various testing techniques are employed to validate the functionality, security, and performance of the system.

The following testing techniques were used for the project:

1. Unit Testing

- Purpose: Unit testing is used to test individual components or modules of the system, ensuring that each part functions as expected in isolation.
- Tools: Jest (for JavaScript testing) was used to test backend functions and logic.
- Example: Testing the leave request submission function to ensure that it correctly checks for valid dates and handles database interaction.

2. Integration Testing

- Purpose: Integration testing ensures that multiple components of the system work

together as expected. For example, testing the interaction between the front-end user interface and the backend server.

- Tools: Postman for API testing.
- Example: Ensuring that a leave request submitted through the front-end interface triggers the correct backend API and is stored in the database.

3. Functional Testing

- Purpose: Functional testing verifies that the system behaves according to the functional requirements. Each feature is tested to ensure that it performs the desired action.
- Tools: Manual testing by the QA team.
- Example: Verifying that an admin can approve or reject leave requests from the admin panel.

4. End-to-End Testing

- Purpose: End-to-end testing simulates real user behavior to ensure that all the components of the system work together seamlessly from start to finish.
- Tools: Cypress or Selenium for automated testing.
- Example: Testing the entire process of a user submitting a leave request, getting approval from the admin, and receiving a notification.

5. Security Testing

- Purpose: Security testing ensures that the system is protected from security vulnerabilities, such as unauthorized access and data breaches.
- Tools: OWASP ZAP (Zed Attack Proxy), manual code reviews.
- Example: Testing authentication and authorization mechanisms to ensure only authorized users (admins and users) can perform actions.

6. Performance Testing

- Purpose: Performance testing measures the responsiveness, speed, and stability of the system under varying loads.
- Tools: Apache JMeter for load testing.

- Example: Testing the system's performance when multiple users submit leave requests simultaneously.

7. User Acceptance Testing (UAT)

- Purpose: User Acceptance Testing is done to verify if the system meets the expectations of end-users.
- Tools: Manual testing by actual end-users (faculty, staff).
- Example: Allowing end-users to test submitting and approving leave requests to confirm the system works as they expect.

4.4 Test Cases Designed

Test cases are designed to verify the functionality of the Duty Leave Management System. Below are the key test cases for major functionalities:

Test Case 1: User Registration

- Test Case ID: TC-001
- Description: Verify that a user can successfully register for the system.
- Preconditions: User is not registered.
- Test Steps:
  i. Navigate to the registration page.
  ii. Enter valid details (name, email, contact number, password).
  iii. Click on the "Register" button.
- Expected Result: The user should be successfully registered and redirected to the login page.
- Actual Result: [To be filled during testing]
- Pass/Fail: [To be filled during testing]

Test Case 2: Leave Request Submission

- Test Case ID: TC-002
- Description: Verify that a user can submit a leave request.

- Preconditions: User is logged in and authenticated.
- Test Steps:
    1. Log in with valid credentials.
    2. Navigate to the leave request form.
    3. Fill in the leave details (start date, end date, leave type, reason).
    4. Click on the "Submit" button.
- Expected Result: The leave request is saved in the database, and a notification is sent to the admin.
- Actual Result: [To be filled during testing]
- Pass/Fail: [To be filled during testing]

Test Case 3: Admin Leave Request Approval

- Test Case ID: TC-003
- Description: Verify that an admin can approve or reject a leave request.
- Preconditions: Admin is logged in, and a leave request exists.
- Test Steps:
    1. Log in as an admin.
    2. Navigate to the leave requests list.
    3. Select a leave request to approve or reject.
    4. Click on "Approve" or "Reject".
- Expected Result: The leave request status is updated, and the user is notified about the decision.
- Actual Result: [To be filled during testing]
- Pass/Fail: [To be filled during testing]

Test Case 4: Leave Request Conflict Detection

- Test Case ID: TC-004
- Description: Verify that the system detects leave date conflicts.

- Preconditions: User is logged in and has an existing approved leave request.
- Test Steps:
    1. Log in as a user.
    2. Submit a leave request with overlapping dates.
- Expected Result: The system should notify the user that there is a conflict with existing leave.
- Actual Result: [To be filled during testing]
- Pass/Fail: [To be filled during testing]

Test Case 5: User Authentication (Login)

- Test Case ID: TC-005
- Description: Verify that a user can log in with valid credentials.
- Preconditions: User is registered.
- Test Steps:
    1. Navigate to the login page.
    2. Enter valid credentials (email, password).
    3. Click on the "Login" button.
- Expected Result: The user should be logged in and redirected to the dashboard.
- Actual Result: [To be filled during testing]
- Pass/Fail: [To be filled during testing]

Test Case 6: Security - Invalid Access

- Test Case ID: TC-006
- Description: Verify that unauthorized users cannot access the admin dashboard.
- Preconditions: User is logged in as a non-admin.
- Test Steps:
    1. Log in with non-admin credentials.
    2. Try to access the admin panel.
- Expected Result: The user should be denied access and redirected to the homepage or

dashboard.

- Actual Result: [To be filled during testing]
- Pass/Fail: [To be filled during testing]

Chapter 5: Results and Discussions

The Duty Leave Management System offers significant benefits by enhancing accessibility for students, streamlining approval processes for faculty, and enabling centralized record-keeping for administrators. By digitizing leave requests, students gain real-time access to their application statuses and can conveniently upload participation certificates, which improves compliance and reduces paperwork. Faculty benefit from a centralized approval system with automated alerts, expediting the review process and reducing administrative overhead. Administrators gain access to a searchable database of leave records, enabling data-driven insights into student engagement across events, which informs strategic planning and resource allocation. However, successful implementation requires addressing privacy and data security concerns, ensuring system scalability, and providing adequate training to ease the transition from traditional methods. Overall, the system improves efficiency and transparency, supporting institutional goals of encouraging balanced student involvement in technical, co-curricular, and sports activities while maintaining accountability and streamlined administration.

5.1User Interface Representation

The Duty Leave Management System's user interface is designed to be intuitive and user-friendly for students, faculty, and administrators, streamlining the leave request and approval process. Upon logging in, students access a dashboard displaying their recent leave requests, statuses, and any pending certificate uploads. From here, they can easily submit new leave requests by selecting event details, providing dates, and entering a description before submission. Faculty view their own dashboard listing all pending leave requests, allowing them to quickly approve or reject applications with options to add comments. For administrators, a centralized dashboard provides an overview of all requests, event records, and analytics, along with a dedicated page for managing events and generating reports. Notifications and alerts ensure users stay informed, with students receiving

updates on request statuses and faculty receiving reminders for pending approvals. Additionally, the system allows students to upload participation certificates post-event, providing an organized, streamlined interface for maintaining compliance. This cohesive design, with clear navigation and organized workflows, simplifies the duty leave process for all users and facilitates efficient record-keeping and approval management.

5.2Snapshots of System with Brief Detail of Each and Discussion

User Login Interface

The user login interface serves as the gateway for students, faculty, or staff to securely access the Duty Leave Management System. Upon visiting the platform, users are prompted to enter their credentials, typically a username (or email) and password, to authenticate their identity. This interface ensures that only authorized users can access the system, maintaining data security. Additionally, it may offer options like "Forgot Password" for password recovery and a "Sign Up" or "Register" option for new users. Once logged in, users are directed to their respective dashboards where they can manage their leave requests and approvals.

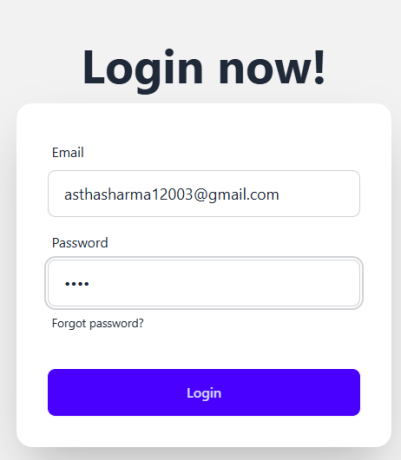

Fig5.1

Leave request form

The leave request interface allows users to submit their duty leave requests by providing essential

details. Users are required to enter the **event name**, specifying the reason for the leave. Additionally, they must select the **start and end dates** of the event to define the duration of their absence. An optional **certificate or supporting document** can be attached, serving as proof of participation in the event. Once the user fills in these fields, the leave request is submitted for review and approval. The system ensures all necessary details are provided before the request is processed.

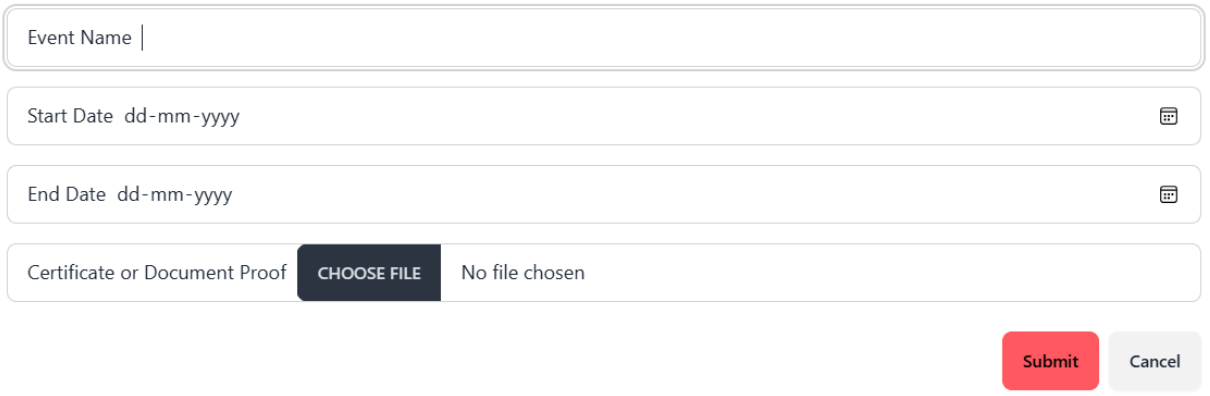

Fig5.2

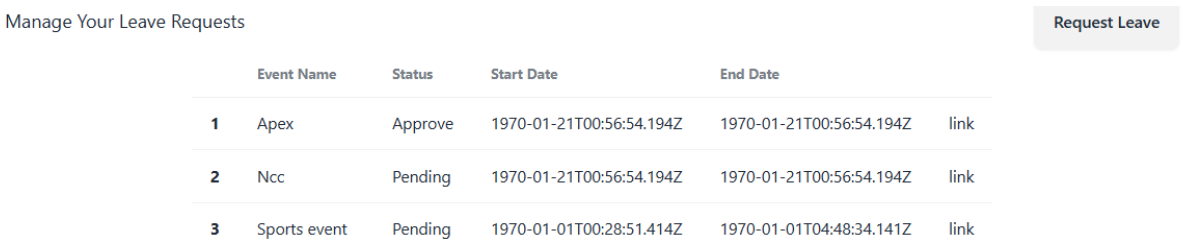

Fig5.3

Admin Dashboard

The Admin Dashboard provides a centralized interface for managing leave requests. Admins can view a list of all submitted leave requests, which includes details such as the event name, start and end dates, student name, and the status of each request. The dashboard allows admins to approve or reject each leave request with a simple click. Upon approval or rejection, the system automatically updates the leave status and notifies the respective student. Admins can also view any attached supporting documents or certificates submitted by students for verification before making a decision.

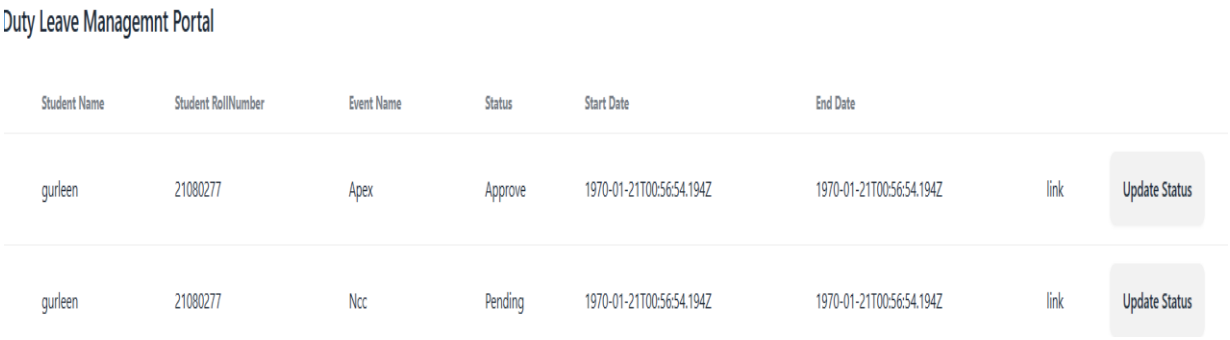

Fig5.4

Fig5.5

5.3Back-End Representation (Database)

The Back-End Representation (Database)for the College Duty Leave Management System uses MongoDB to store and manage data related to students, faculty, events, leave requests, and certificates. Collections include Students, Faculty, Events, Leave Requests, and Certificates, each with fields tailored to capture essential information. Relationships among these collections allow for efficient linking, such as connecting students to their leave requests and participation records, and faculty to their approval roles. This structure supports quick data retrieval, secure access control, and scalability, ensuring a robust foundation for managing duty leave processes within the application.

Snapshots of Database Tables with Brief Description

Below are key database collections and their structure:

User Collection

Stores all user-related data, including lo credentials and profile information..

User Collection Database

Fig 5.6

Leave Collection

The Leave Collection stores details of leave requests, including the `requestId`, `studentId` (referencing the student requesting leave), `eventId` (linking to the associated event), `status` (e.g., Pending, Approved, Rejected), `requestedDate`, and `approvedBy` (faculty ID handling the request). This collection tracks leave statuses and ensures organized leave management

Fig 5.7

Chapter 6: Conclusion and Future Scope

6.1Conclusion

In a nutshell, the Duty Leave Management System should become an inevitable tool for educational institutions to manage faculty leave requests effectively. Since it automates most of the manual processes, it reduces human error and administrative workload; besides, it is transparent and liable. This system is a streamlined process whereby a faculty member can request leave that will either be accepted or rejected based on predetermined institutional policies. Automated notification keeps the faculty and administration informed of the status of leave requests in a smooth flow of information. One of the most attractive features of the system is the accessible interface for submitting requests, as well as tracking the application status and record leaves taken by faculty, while for the administrators, it is features that involve leave approvals, setting rules, generating reports, and aids in the decision-making and management of resources. This automation will reduce paperwork and provide accurate and faster processing when it comes to leave requests.

Looking into the future, the Duty Leave Management System has a lot of scope for extension and further integration with other institutional systems like payroll, attendance, or performance tracking. This way, all faculty management is integrated into one package. Moreover, this scalable system will allow small colleges, middle-level colleges, and large-sized universities to maintain their performance and reliability regardless of size.

The future scope will bring mobile applications that provide better accessibility and AI-driven insights to optimize leave management, integrating advanced features like automatic leave suggestions for the workload and calendar. Integrating the system with cloud services will help better store, backup, and access data and provide uninterrupted services by maintaining data security.

Simply put, the Duty Leave Management System shall have a significant impact on effective functioning, increased faculty satisfaction, and proper resource planning that would make it an asset for any learning institution. The system would continue to promote improved operational

excellence, ease administrative work, and permit new features to support the educational organizations growing needs.

6.2 Future Scope

Huge potential areas of growth and development lie for this Duty Leave Management System. Future work will include the development of both an Android and iOS mobile application that the faculty and administrative staff can use to log on and check and approve leave requests remotely. AI integration will enable prediction of leave patterns, thereby optimizing resource planning and identifying peak times for leave requests. Clearly, pre-set rules would automatically enforce approval processes over leaves, minimizing intervention, and thus making the operation more efficient. Integrating this system with the external HR, payroll, and academic scheduling systems helps also smoothen the operation while maintaining up-to-date leave balances and smooth running of salary deductions as well as scheduling. Push alerts, SMS, or email would keep users abreast with changes in the status of their leaves, approvals or rejections, and upcoming dates. It would be scallable, secure, and easier to maintain the system as it would be hosted on cloud platforms. Thereby, the system could support a more significant user base. The reporting features and analytics features of the system could be further extended to offer much more detailed insights into leave trends, resources utilization, and performance metrics to enable data-based decision by the institutions. Customizable user role would have a precise approach toward leave management and incorporating a biometric system like facial recognition or fingerprint scan would ensure authenticity for attendance tracking and aversion of fraudulent leave requests. Lastly, it would ensure multilingual support, making the system available to diverse academic institutions thereby expanding its reach across regions and languages. Advances would entail improvements to the system's functionality in general while making it even more accessible, efficient, and scalable regarding education administration applications.

References

- N. Patel, S. Kumar, and A. Sharma, "Automating Duty Management in Educational Institutions: A Case Study," International Journal of Educational Management, vol. 36, no. 4, pp. 510-525

- M. Brown and T. Harris, "Enhancing Administrative Efficiency with Duty Management Systems," Journal of Higher Education Administration, vol. 29, no. 2, pp. 75-89.

- J. Smith, R. Johnson, and L. Lee, "Digital Solutions for Managing Student Activities and Leaves," Proceedings of the International Conference on Educational Technology, pp. 112-120.

- C. Chen, A. Liu, and B. Zhang, "Best Practices in Developing Duty Management Systems for Academic Institutions," Educational Technology & Society, vol. 22, no. 3, pp. 90-103.

- L. Johnson and M. Lee, "Optimizing Record-Keeping Processes with Automated Duty Management Systems," Journal of Administrative Science, vol. 20, no. 1.

Appendix A: Development Environment

1. Programming Languages and Frameworks

- JavaScript: Primary language for full-stack development (front-end and back-end).
- React.js: For dynamic user interfaces and component-based architecture.
- Node.js: JavaScript runtime for server-side operations.
- Express.js: Web framework for handling APIs and server-side routing.
- MongoDB: NoSQL database for flexible data storage and scalability.

2. IDE

Visual Studio Code: Lightweight IDE with features like syntax highlighting, Git integration, and extensions for JavaScript and Node.js development.

3. Version Control System

Git & GitHub: For code version control, collaboration, and tracking changes.

4. APIs and Integrations

- JWT: For secure user authentication.
- Node mailer: For sending email notifications

5. Development and Testing Tools

- Postman: For API testing.
- MongoDB Compass: For managing MongoDB and monitoring performance.
- Jest: For unit testing JavaScript code.

- Super test: For testing API routes.

6. Hosting and Deployment

- Heroku: For deploying Node.js applications.
- MongoDB Atlas: For cloud-hosted, secure, and scalable MongoDB database.

This environment ensures the development of a reliable, secure, and scalable Duty Leave Management System