SMARTPASS

A PROJECT REPORT
for
Mini Project (KCA353)
Session (2024-25)

Submitted by

Abhya Ahuja	2300290140010
Akansha Parashar	2300290140015
Anshika Aggarwal	2300290140027
Aradhya Tyagi	2300290140035

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Ankit Verma
Associate Professor

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR
GHAZIABAD-201206

(DECEMBER 2024)

CERTIFICATE

Certified that **Abhya Ahuja (2300290140010), Akansha Parashar (2300290140015), Anshika Aggarwal (2300290140027), Aradhya Tyagi (2300290140035)** have carried out the project work having "**Smart Pass**" (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Abhya Ahuja	**2300290140010**
Akansha Parashar	**2300290140015**
Anshika Aggarwal	**2300290140027**
Aradhya Tyagi	**2300290140035**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Ankit Verma	**Dr. Arun Kumar Tripathi**
Associate Professor	**Head of Department**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

Smart Pass

(Abhya Ahuja)
(Akansha Parashar)
(Anshika Aggarwal)
(Aradhya Tyagi)

ABSTRACT

Smart Pass is a user-friendly system designed to make hostel leave management simple and efficient. It replaces the traditional paper-based process with a digital solution that connects Students, Wardens, Heads, and Gatekeepers. This system ensures a smooth workflow for applying, approving, and tracking leave requests, making the process faster and more reliable.

Built with modern technologies, Smart Pass uses React for the frontend, ensuring an easy-to-use interface, and Express for the backend to handle data efficiently. Students can quickly apply for leave, Wardens and Heads can review and approve requests, and Gatekeepers can verify approved leaves at the hostel entrance.

This project shows how technology can solve common administrative problems, focusing on saving time, reducing errors, and improving security. This report explains how Smart Pass was designed and built, highlighting its features and benefits for hostel management.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Ankit Verma Sir** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness

Abhya Ahuja
Aradhya Tyagi
Akansha Parashar
Anshika Aggarwal

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Smart Pass is an innovative hostel leave management system designed to streamline the leave application process within college hostels. The system integrates multiple user roles, including Head, Warden, Student, and Gatekeeper, to create a cohesive and efficient workflow for managing leave requests. By leveraging modern technologies like React for the frontend and Express for the backend, Smart Pass ensures a user-friendly interface and robust backend support.

1.1 OUTLINE OF THE PROJECT

This chapter provides an overview of the Smart Pass project, its objectives, and the methodologies employed in its development. It outlines the significance of digitalizing traditional processes, describes the functionality of the system, and emphasizes the technologies utilized to achieve the project goals.

1.1.1 Literature Review

The traditional hostel leave management systems are primarily paper-based, which often leads to inefficiencies such as misplacement of records, delays in approval processes, and lack of transparency. Previous studies have highlighted the importance of digital solutions in administrative workflows, particularly in educational institutions. Various automated systems have been developed to simplify administrative processes, but most lack the adaptability and role-specific features required for hostel management. Smart Pass bridges this gap by offering a customized solution tailored to the unique requirements of college hostels.

1.1.2 Objectives of the Project

The primary objectives of Smart Pass include:

- Providing an intuitive interface for students to submit leave requests.
- Ensuring a streamlined approval process for wardens and heads.
- Enhancing security by enabling gatekeepers to verify approved requests.

- Reducing manual errors and improving overall efficiency through digitalization.

1.1.3 Methodology

The development of Smart Pass follows a systematic approach:

- **Requirement Analysis**: Identifying the needs of different user roles and defining system specifications.
- **Design and Development**: Creating a responsive frontend using React and a robust backend with Express.
- **Testing and Deployment**: Conducting thorough testing to ensure the system meets functional and performance requirements.

This document provides a detailed analysis of the Smart Pass system, from its conceptualization to implementation, offering insights into its design, features, and overall impact on hostel administration.

CHAPTER 2

PROJECT ANALYSIS

2.1 SYSTEM STUDY

The **Smart Pass** project was developed to address the absence of a dedicated leave management system for hostel residents in our college. Currently, hostelers submit leave requests manually, which is inefficient and prone to delays. Our goal is to digitize this process, making it faster, more transparent, and user-friendly for all stakeholders, including students, wardens, heads, and gatekeepers.

2.2 EXISTING SYSTEM

The existing system in our college relies entirely on manual processes for hostel leave management. Hostelers have to fill out paper leave applications and submit them to their respective wardens. The approval process is also manual, requiring signatures from wardens and heads, which can take time. Once approved, the leave records are manually conveyed to the gatekeeper.

2.3 DISADVANTAGES OF THE EXISTING SYSTEM

1. **Time-Consuming Process**: Submitting and approving paper forms requires significant time from both students and staff.
2. **Prone to Errors**: Manual handling can lead to misplaced or incomplete applications, causing confusion and delays.
3. **Lack of Transparency**: Students have no way of tracking the status of their leave requests, leading to uncertainty.
4. **Inefficient Record Management**: Paper-based records are difficult to maintain, search, and analyze for future reference.
5. **Limited Accessibility**: Students cannot apply for leave remotely, as the process requires physical submission.

2.4 PROPOSED SYSTEM: SMART PASS

To overcome these issues, the **Smart Pass** software has been designed. It provides an efficient, digital leave management solution for hostelers.

2.4.1 Performance

The system ensures quick leave processing. Students can apply for leave online, and the software immediately notifies wardens and heads for approval. The entire process is streamlined, minimizing delays.

2.4.2 Efficiency

Smart Pass eliminates the need for paper forms, making the process faster and reducing the workload for staff. Automated notifications and updates save time for everyone involved.

2.4.3 Control

The system provides clear roles for each stakeholder:

- Students can submit and track leave requests.
- Wardens and heads can approve or reject requests online.
- Gatekeepers can access leave records in real time, ensuring only approved leaves are granted.

2.4.4 Security

The software uses secure login credentials for all roles (HEAD, STUDENT, WARDEN, and GATEKEEPER). Data is stored securely and can only be accessed by authorized users. Additionally, all actions are logged for accountability.

2.5 ADVANTAGES OF THE PROPOSED SYSTEM

1. **Faster Process**: Leave requests and approvals are handled in minutes, not days.
2. **Transparency**: Students can track their leave status online.
3. **Error-Free**: Automation minimizes the chances of errors or lost applications.
4. **Better Record Management**: Digital records are easy to maintain, search, and analyze.
5. **Remote Accessibility**: Students can apply for leave from anywhere, improving convenience.
6. **Enhanced Security**: Controlled access ensures the system is secure and data is protected.

The Smart Pass system offers a modern, efficient, and secure solution for managing hostel leave requests, addressing the inefficiencies of the manual process. By automating leave submissions, approvals, and tracking, it significantly reduces delays, errors, and enhances transparency, accessibility, and security for all users involved.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION AND ANALYSIS

The **Smart Pass** system is a web-based application designed to simplify and automate the hostel leave management process. It provides a secure and user-friendly platform for students to apply for leave, wardens to approve or reject requests, gatekeepers to verify approved leaves, and the Head of the institution to monitor overall usage and system reports. Below is a comprehensive SRS that defines the system's functionality, usability, performance, and requirements.

3.1 PLANNING AND SCHEDULING

The planning and scheduling of the **Smart Pass** project involves organizing the tasks and activities required for the successful completion of the system. It includes setting realistic timelines for each phase of the project, defining milestones, and managing resources effectively. The following sections outline the key tasks and their respective schedules.

3.1.1 Project Phases

1. **Requirement Gathering and Analysis (1st July - 10th July)**
   - The project begins with gathering and analyzing the functional and non-functional requirements. This phase includes meetings with stakeholders, such as students, wardens, gatekeepers, and the head of the institution, to understand their needs and expectations from the Smart Pass system.
   - Deliverables: Documented System Requirements Specification (SRS).

2. **System Design (11th July - 20th July)**
   - Based on the gathered requirements, the system architecture, user interface design, and database schema are created. This phase will define the structure of the frontend, backend, and database, ensuring the system is scalable, secure, and efficient.
   - Deliverables: System Design Document, UI/UX wireframes, Database Schema.

3. **Frontend Development (21th July - 10th August)**
    - The frontend of the Smart Pass system is developed using ReactJS. This phase focuses on building the user interface, including the login page, leave application forms, approval dashboards, and status tracking features.
    - Deliverables: Fully functional frontend with all user interfaces integrated.

4. **Backend Development (11th August - 25th August)**
    - The backend, built with Node.js and Express, is developed to handle data processing, user authentication, and the management of leave applications. This phase also includes API development for communication between the frontend and backend.
    - Deliverables: Backend API, authentication system, leave request handling.

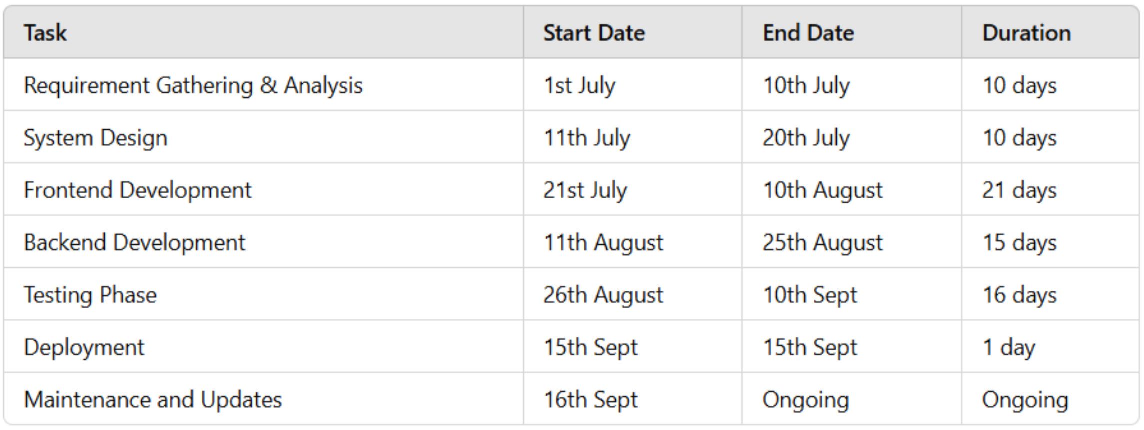
5. **Testing Phase (26th August - 10th September)**
    - Comprehensive testing is conducted to ensure that all features work as expected. This includes unit testing, integration testing, security testing, and user acceptance testing (UAT). Bug fixing and optimizations are performed as needed during this phase.
    - Deliverables: Test Reports, Bug Fixes, UAT Sign-off.

6. **Deployment (15 September)**
    - The system is deployed on a cloud platform (e.g., AWS or Heroku). The deployment process includes setting up the server, database, and ensuring proper configuration for production.
    - Deliverables: Deployed system accessible to all users.

7. **Maintenance and Updates (Ongoing)**
    - Post-deployment, the system will undergo regular maintenance to fix bugs, update features, and handle any security patches. User feedback will be collected for future enhancements.
    - Deliverables: System Updates, Feedback Collection, Bug Fixes.

3.1.2 Gantt Chart

Here’s the updated Gantt chart reflecting our project's timeline:



Task	Start Date	End Date	Duration
Requirement Gathering & Analysis	1st July	10th July	10 days
System Design	11th July	20th July	10 days
Frontend Development	21st July	10th August	21 days
Backend Development	11th August	25th August	15 days
Testing Phase	26th August	10th Sept	16 days
Deployment	15th Sept	15th Sept	1 day
Maintenance and Updates	16th Sept	Ongoing	Ongoing

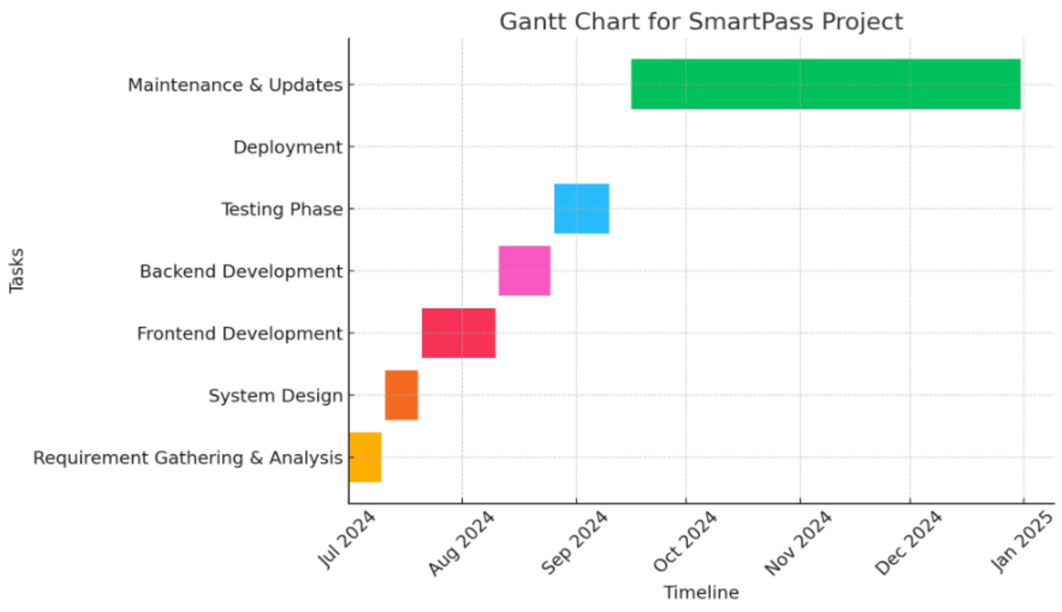


Fig. 1.1: Gantt Chart

3.2 SYSTEM ANALYSIS

Smart Pass is built as a responsive web application using modern technologies like ReactJS for the frontend and Node.js with Express for the backend. It uses a database (MongoDB/MySQL) to store all records and ensures secure access with role-based authentication. The system is hosted on a cloud platform to ensure 24/7 availability.

3.2.1. Functional Requirements

- **Role-Based Access:**

    - Students: Apply for leave, view status, and receive notifications.
    - Wardens: Review and approve/reject leave requests.
    - Gatekeepers: Verify approved leave at the hostel gate.
    - Head: Monitor leave records and system reports.

- **Leave Management**:
    - Students can fill out a leave application form with details like dates, reason, and contact information.
    - Wardens can view leave requests, check details, and take action.

- **Reporting and Analytics**:
    - The Head can view reports on system usage, including total applications, approvals, and rejections.

3.2.2. Non-Functional Requirements

- **Scalability**: The system can handle up to 1000 concurrent users through efficient resource management, ensuring smooth performance even during peak usage.
- **Reliability**: With a guaranteed 99.9% uptime, the system ensures uninterrupted access, using redundant servers and automated monitoring to prevent and resolve issues.
- **Performance**: Smart Pass maintains an API response time of under 2 seconds for all operations, ensuring a fast and efficient user experience.
- **Compatibility**: The system works seamlessly across all modern browsers and devices, including desktops, smartphones, and tablets, ensuring accessibility for all users.
- **Security**: Smart Pass secures user data with HTTPS encryption, hashed passwords, and role-based access control to prevent unauthorized access and protect sensitive information.

3.2.3 Hardware Requirements

- **Processor**: Minimum Quad-Core Processor (e.g., Intel Core i3 or equivalent).
- **Hard Disk**: 500 GB SSD (server) or 100 GB free space (client devices).
- **RAM**: 8 GB (minimum for server); 4 GB (minimum for client devices).
- **Monitor**: Standard display with a resolution of 1920x1080 or higher.
- **Mouse & Keyboard**: Standard peripherals for input tasks.
- **Internet Connection**: Stable broadband with at least 10 Mbps speed.

3.2.4 Software Requirements

- **Frontend**: ReactJS for user interface development.
- **Backend**: Node.js with Express for server-side operations.
- **Database**: MongoDB or MySQL for storing records.
- **Hosting**: Cloud-based hosting (e.g., AWS, Azure, or Heroku) for high availability.
- **Operating System**: Compatible with Windows, macOS, and Linux environments.

3.2.5 User Interfaces

The system features separate dashboards for each user role.

- **Login Page**: A clean interface with fields for username, password, and accompanied by the Smart Pass logo.
- **Student Dashboard**: Allows leave application and status tracking.
- **Warden Dashboard**: Displays all pending leave requests for review.
- **Gatekeeper Dashboard**: Displays approved leave records for verification.
- **Head Dashboard**: Provides an overview of system reports and statistics.

The design is responsive, mobile-friendly, and includes tooltips and clear navigation to guide users.

3.2.6 Usability

Smart Pass is designed for ease of use, ensuring a seamless experience for all users. The system includes help tips, clear error messages, and workflows tailored to each role. It also adheres to accessibility standards, making it usable for individuals with disabilities.

3.2.7 Security Features

Smart Pass ensures user data and system integrity through:

- Encrypted passwords (e.g., bcrypt).
- Secure HTTPS communication.

- Role-based access control.
- Regular security updates to prevent vulnerabilities.

3.2.8 Performance and Availability

- The system is optimized to handle high traffic and process multiple leave requests simultaneously.
- It is available 24/7, with regular automated backups to prevent data loss. Scheduled maintenance is communicated to users in advance.

3.2.9 Error Handling

Smart Pass provides user-friendly error messages for invalid inputs, login failures, or system issues.

CHAPTER 4

SYSTEM DESIGN

The system design for the **Smart Pass** project involves planning and creating the architecture of the entire application, including both the frontend and backend. It defines the structure of how the application will function and how different components will interact. In this section, we will discuss the design at different levels, including data flow, class design, use case, entity-relationship (ER) diagrams, and more.

4.1 OVERVIEW OF THE SYSTEM ARCHITECTURE

The **Smart Pass** system is a web-based application that allows students to apply for leaves and enables wardens and heads to approve or reject them. The architecture is built using **ReactJS** for the frontend and **Node.js with Express** for the backend. The database is managed using **MongoDB** or **MySQL** for storing leave records and user details. The system is designed to be scalable, reliable, and secure.

4.2 DATA FLOW DIAGRAM (DFD)

A Data Flow Diagram (DFD) is used to visually represent the flow of data within the system. The DFD for the **Smart Pass** system is presented at multiple levels to show its functionality in detail.

Level 0: Context Diagram

The **Level 0 DFD** provides a high-level view of the system. It shows how the system interacts with external entities like students, wardens, heads, and gatekeepers.

- **External Entities**: Students, Wardens, Heads, Gatekeepers
- **Processes**: Apply Leave, Approve/Reject Leave, Verify Leave at Gate, Monitor System
- **Data Stores**: Leave Records

The context diagram shows that the **Smart Pass** system communicates with external entities to process leave applications and handle approvals.

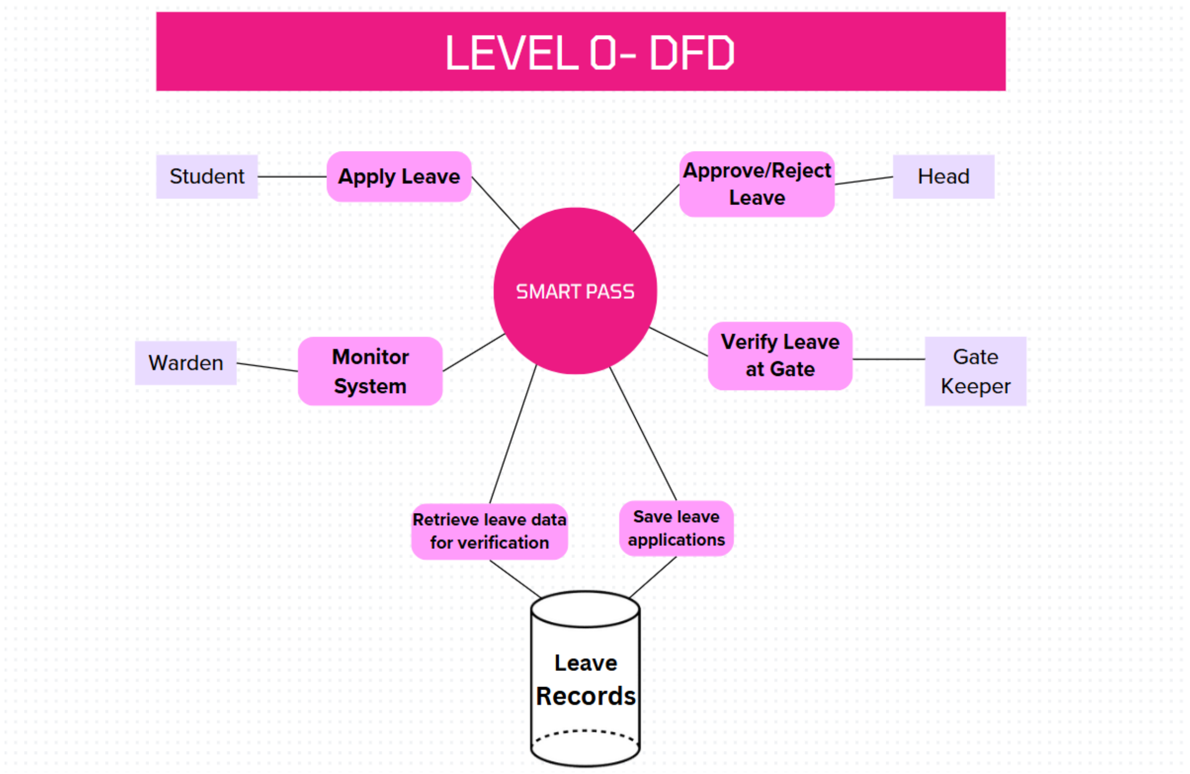


Fig. 1.2: Data flow Diagram

Level 1: Detailed DFD

The **Level 1 DFD** breaks down the major processes and shows how data flows between them. The main processes are:

1. **Student Leave Application**: The student submits the leave request, which includes the leave details (dates, reason, contact).
2. **Warden Approval/Rejection**: The warden reviews the request and approves or rejects it.
3. **Head Approval**: The head has the authority to approve or reject the leave request.
4. **Gatekeeper Verification**: Once approved, the gatekeeper checks the leave at the hostel gate to verify the student's leave status.

The data flows between these processes, and the data is stored in the database.

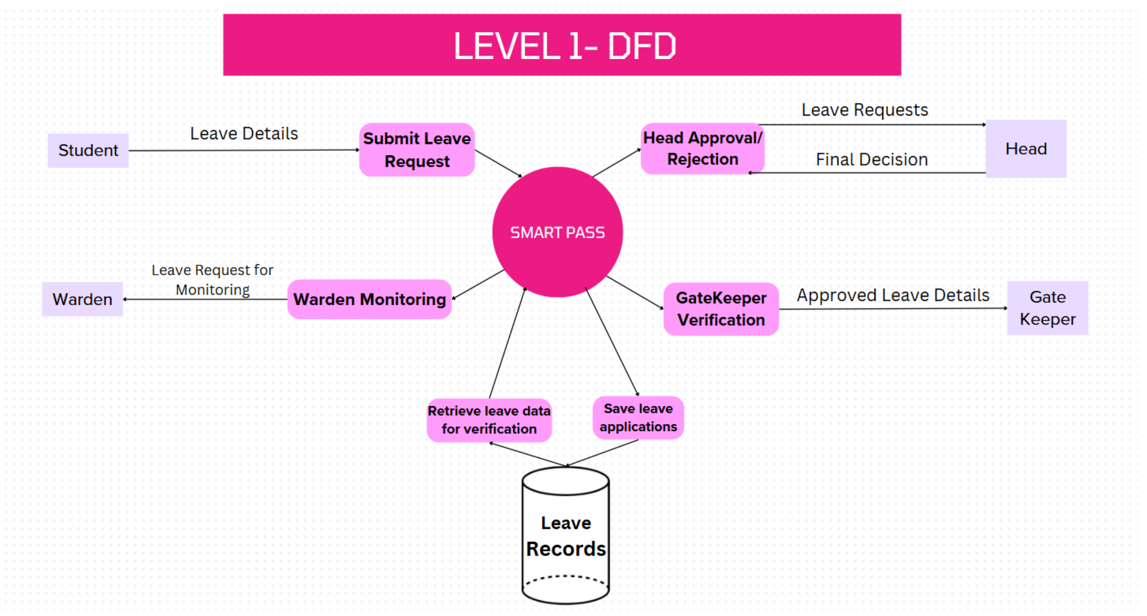


Fig. 1.3: Data flow Diagram

4.3 ENTITY-RELATIONSHIP (ER) DIAGRAM

The **ER Diagram** represents the data structure of the system. It shows how entities (e.g., users, leave requests) relate to each other in the database.

- **Entities**:
  - **User** (Attributes: userID, name, role, email)
  - **LeaveRequest** (Attributes: leaveID, studentID, leaveDates, reason, status)
  - **LeaveApproval** (Attributes: approvalID, leaveID, wardenID, approvalStatus)
  - **GateVerification** (Attributes: verificationID, leaveID, gatekeeperID, verifiedStatus)

- **Relationships**:
  - A **User** (Student, Warden, Head, Gatekeeper) can have multiple **LeaveRequests**.
  - A **LeaveRequest** can have multiple **LeaveApprovals**.
  - Each **LeaveRequest** can be verified by one or more **GateVerifications**.

The ER diagram defines how the system stores and organizes data.

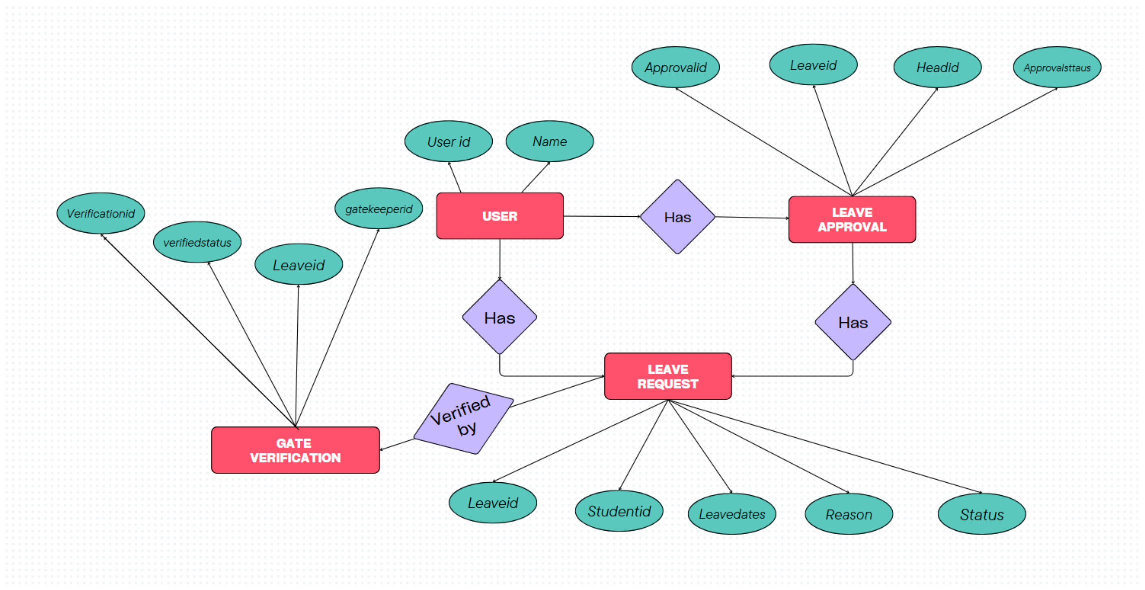


Fig. 1.4: ER Diagram

4.4 SYSTEM COMPONENTS AND FLOW

1. **Frontend**:
   - Built with **ReactJS**, the frontend provides a responsive and interactive user interface. It communicates with the backend via APIs to send and receive data.
2. **Backend**:
   - Built with **Node.js** and **Express**, the backend handles the logic for leave requests, approvals, user authentication, and data storage. It processes user requests and serves the frontend.
3. **Database**:
   - The system uses **MongoDB** or **MySQL** for storing user data and leave requests. The database is designed to efficiently store and retrieve leave data based on user roles.
4. **Security**:
   - The system uses HTTPS for secure communication and implements role-based authentication to ensure that only authorized users can access specific parts of the application.
5. **Cloud Hosting**:
   - The system is deployed on a cloud platform, ensuring that it is scalable, available 24/7, and can handle a large number of users.

CHAPTER 5

IMPLEMENTATION AND CODING

The system was developed using **ReactJS** for the frontend, **Node.js** with **Express** for the backend, and **MongoDB** as the database. The goal was to create a web-based platform that allows hostel students to apply for leave, wardens to approve or reject the requests, and gatekeepers to verify leave status.

5.1 PROJECT SETUP

To begin, the project was divided into two main parts: the frontend and the backend.

- **Frontend**: Built with ReactJS, using hooks for state management and Axios for making API requests to the backend.
- **Backend**: Developed using Node.js and Express, with MongoDB as the database to store leave requests and user data.

The project structure for both frontend and backend is as follows:

```
/smartpass
  /frontend
    /src
      /components
        - Login.js
        - Dashboard.js
        - LeaveRequestForm.js
      /services
        - api.js
      /hooks
        - useAuth.js
  /backend
    /controllers
      - leaveController.js
```

/models
   - LeaveRequest.js
   - User.js
/routes
   - leaveRoutes.js
- server.js

5.2 KEY FUNCTIONALITIES AND CODE SNIPPETS

5.2.1 User Authentication (Login)

For authentication, **JWT (JSON Web Tokens)** were used to securely log users in. The login route validates the user credentials and generates a token.

Here's the code for the login route in the **backend**:

```
// backend/routes/authRoutes.js
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');
const router = express.Router();

router.post('/login', async (req, res) => {
    const { email, password } = req.body;
    try {
        const user = await User.findOne({ email });
        if (!user) return res.status(400).json({ message: 
'User not found' });

        const isMatch = await bcrypt.compare(password, 
user.password);
        if (!isMatch) return res.status(400).json({ message: 
'Invalid credentials' });

        const token = jwt.sign({ userId: user._id, role: 
user.role }, 'secretkey', { expiresIn: '1h' });
        res.json({ token });
    } catch (err) {
        res.status(500).json({ message: 'Server Error' });
    }
});

module.exports = router;
```

On the **frontend**, React manages the login form and sends the login request to the backend:

```
// frontend/src/components/Login.js
import React, { useState } from 'react';
import axios from 'axios';

function Login() {
    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');
    const [error, setError] = useState('');

    const handleSubmit = async (e) => {
        e.preventDefault();
        try {
            const response = await
axios.post('http://localhost:5000/api/login', { email,
password });
            localStorage.setItem('token',
response.data.token);
            window.location.href = '/dashboard';
        } catch (err) {
            setError('Invalid credentials');
        }
    };

    return (
        <form onSubmit={handleSubmit}>
            <input type="email" value={email} onChange={(e)
=> setEmail(e.target.value)} placeholder="Email" />
            <input type="password" value={password}
onChange={(e) => setPassword(e.target.value)}
placeholder="Password" />
            <button type="submit">Login</button>
            {error && <div>{error}</div>}
        </form>
    );
}

export default Login;
```

5.2.2 Leave Request Submission

Students can submit leave requests by filling out a form. The frontend sends a POST request to the backend to store the leave request in the database.

```
// backend/controllers/leaveController.js
const LeaveRequest = require('../models/LeaveRequest');

exports.createLeaveRequest = async (req, res) => {
    const { studentId, startDate, endDate, reason } =
req.body;
    try {
        const newLeaveRequest = new LeaveRequest({
            studentId,
            startDate,
            endDate,
            reason,
            status: 'Pending',
        });
        await newLeaveRequest.save();
        res.status(201).json(newLeaveRequest);
    } catch (err) {
        res.status(500).json({ message: 'Error creating
leave request' });
    }
};
```

On the **frontend**, the student submits the leave request form:

```
// frontend/src/components/LeaveRequestForm.js
import React, { useState } from 'react';
import axios from 'axios';

function LeaveRequestForm() {
    const [startDate, setStartDate] = useState('');
    const [endDate, setEndDate] = useState('');
    const [reason, setReason] = useState('');

    const handleSubmit = async (e) => {
        e.preventDefault();
        const studentId =
localStorage.getItem('studentId');
        const leaveData = { studentId, startDate, endDate,
reason };
```

```
        await
axios.post('http://localhost:5000/api/leaveRequest',
leaveData);
        alert('Leave Request Submitted');
    };

    return (
        <form onSubmit={handleSubmit}>
            <input      type="date"      value={startDate}
onChange={(e) => setStartDate(e.target.value)} />
            <input       type="date"       value={endDate}
onChange={(e) => setEndDate(e.target.value)} />
            <textarea   value={reason}   onChange={(e)   =>
setReason(e.target.value)} placeholder="Reason for leave" />
            <button       type="submit">Submit       Leave
Request</button>
        </form>
    );
}

export default LeaveRequestForm;
```

5.2.3 Leave Approved by Head

Backend Code (Approval and Rejection)

The code for approval and rejection should now be in the **Head's** controller, not the warden’s.

```
// backend/controllers/leaveController.js
exports.approveLeaveByHead = async (req, res) => {
    const { leaveId, approvalStatus } = req.body;
    try {
        const         leaveRequest         =         await
LeaveRequest.findById(leaveId);
        if  (!leaveRequest)  return  res.status(404).json({
message: 'Leave request not found' });

        leaveRequest.status = approvalStatus; // 'Approved'
or 'Rejected'
        await leaveRequest.save();
        res.json(leaveRequest);
    } catch (err) {
        res.status(500).json({  message:  'Error  approving
leave request' });
    }
};
```

Frontend Code (Head Dashboard)

In the **frontend**, the Head can approve or reject leave requests from their dashboard:

```
// frontend/src/components/HeadDashboard.js

import React, { useState, useEffect } from 'react';
import axios from 'axios';

function HeadDashboard() {
    const [leaveRequests, setLeaveRequests] = useState([]);

    useEffect(() => {
        const fetchRequests = async () => {
            const         response         =         await
axios.get('http://localhost:5000/api/leaveRequests');
            setLeaveRequests(response.data);
        };
        fetchRequests();
    }, []);

    const handleApprove = async (leaveId) => {
        await
axios.post('http://localhost:5000/api/approveLeaveByHead',
{ leaveId, approvalStatus: 'Approved' });
        alert('Leave Approved');
    };

  const handleReject = async (leaveId) => {
        await
axios.post('http://localhost:5000/api/approveLeaveByHead',
{ leaveId, approvalStatus: 'Rejected' });
        alert('Leave Rejected');
    };

    return (
        <div>
            <h2>Leave Requests</h2>
            <ul>
                {leaveRequests.map(request => (
                    <li key={request._id}>
```

```
                        {request.reason} - {request.status}
                        <button        onClick={()        =>
handleApprove(request._id)}>Approve</button>
                        <button        onClick={()        =>
handleReject(request._id)}>Reject</button>
                    </li>
                ))}
            </ul>
        </div>
    );
```

CHAPTER 6

SOFTWARE TESTING

Software testing is a critical phase in the software development lifecycle, ensuring that the application functions as intended, meets user requirements, and is free from defects. For the **Smart Pass** project, various testing methods were applied to ensure the system's functionality, usability, security, and performance are all up to standard. The testing process covered both manual and automated tests, depending on the type of functionality being validated.

The primary goal of testing in the **Smart Pass** project is to ensure that the system functions as expected, is free of bugs, and meets the user requirements. Testing aims to identify and resolve defects to ensure the software meets its quality standards. The following are the basic principles of testing that guided the testing of the **Smart Pass** system:

6.1 PRINCIPLES OF TESTING

1. **A Successful Test Must Be Capable of Finding Errors**:
    - The main objective of testing is to identify flaws, mistakes, or unexpected behavior in the system. Each test case is designed to catch potential issues before the application is deployed.
2. **Planning of the Testing Phase May Require Long Time**:
    - Adequate planning is crucial for successful testing. Detailed test plans were developed that outlined the testing objectives, scope, resources, and timelines. Planning ensures that all aspects of the system are tested thoroughly and prevents oversights.
3. **Tests Are Performed According to the Client's Needs**:
    - The testing process for **Smart Pass** was aligned with the client’s (i.e., the college administration and students) requirements. All testing procedures and test cases were designed to verify that the system met functional and non-functional requirements, ensuring it performed well under real-world conditions.

4. **Effective Testing Should Detect All Errors**:
    - While exhaustive testing isn't feasible, the goal was to identify as many errors as possible, especially those that could affect critical functionalities such as leave submission, approval processes, and user login.
5. **Exhaustive Testing Is Not Possible**:
    - Given the time constraints and complexity of the **Smart Pass** system, it was not possible to test every possible scenario. Instead, risk-based testing was used to prioritize the most critical features, such as leave request submission, approval/rejection process, and role-based access control.

6.2 TEST PLAN

A **Test Plan** is a crucial document in the **Smart Pass** testing process. It outlines the strategy and approach for the entire testing lifecycle. It serves as a guide for the testing team and stakeholders, ensuring everyone understands the scope, resources, schedule, and methods for testing the application.

The **Smart Pass Test Plan** included the following key elements:

1. **Test Objectives**:
    - The main objective of testing was to ensure that the **Smart Pass** system functions correctly and securely for all users (students, wardens, the Head, and gatekeepers). The system should allow students to apply for leave, wardens to approve/reject requests, the Head to monitor leave data, and gatekeepers to verify approved leaves.
2. **Scope of Testing**:
    - Functional testing, integration testing, security testing, performance testing, and user interface testing were included in the scope. Testing was also focused on the role-based access control system to ensure that users could only access appropriate features based on their roles.
3. **Test Strategy**:
    - The testing process involved manual testing for user interfaces and automated tests for backend services. Automated tests were written using **Jest** (for frontend React components) and **Mocha/Chai** (for backend APIs).
    - Load testing was done using **JMeter** to ensure the system could handle multiple concurrent users.
    - Security vulnerabilities were identified using **OWASP ZAP** and **Postman**.
4. **Resources**:

    - The testing team consisted of developers, testers, and stakeholders who were responsible for executing the test cases, fixing bugs, and verifying that the system met the acceptance criteria.

5. **Test Schedule**:
    - The testing was conducted in parallel with development, with major testing phases occurring after the completion of each feature. A detailed
    - schedule was followed, ensuring that each testing phase was completed within the allocated time frame.

6. **Test Deliverables**:
    - Test cases, test reports, and bug reports were generated throughout the testing process. These documents helped in evaluating the overall quality of the **Smart Pass** system.

6.3 PHASES OF TESTING

The testing process for **Smart Pass** followed a structured approach, starting from unit testing to system testing. Each phase had specific goals and deliverables.

1. **Unit Test**:
    - Unit tests were focused on verifying individual functions or components of the **Smart Pass** system. For example, ensuring that the login form handled input validation correctly, that the API routes returned appropriate responses, and that the leave request form correctly stored user data in the database.
    - **Tools used**: **Jest** for React components and **Mocha/Chai** for backend testing.

2. **Integration Test**:
    - Integration testing was performed to ensure that different components of the **Smart Pass** system worked together as expected. This included verifying that data submitted by a student was correctly passed through the backend, stored in the database, and accessible to other system users, such as wardens and the Head.
    - **Tools used**: **Supertest** for testing backend routes, integration of frontend with backend services.

3. **System Test**:
    - System testing was conducted to validate the entire **SmartPass** application as a whole. This phase involved end-to-end testing of key functionalities, such as submitting leave requests, approving/rejecting them, and verifying

access control for different users (students, wardens, heads, and gatekeepers).
    - **Tools used**: Manual testing of the entire application, including the UI and backend functionalities, and automated **Cypress** tests for end-to-end testing.

6.4 TEST COVERAGE

Test coverage was carefully monitored to ensure that all aspects of the SmartPass system were tested. The following areas were covered:

- **Login System**: Ensured that the authentication mechanism (JWT-based) worked securely for different roles (student, warden, head, gatekeeper).
- **Leave Request**: Verified that students could submit leave requests and that the Head could approve/reject them.
- **Leave Status**: Confirmed that the leave status (approved/rejected) was updated correctly and visible to relevant users.
- **Notifications**: Tested notification delivery when a leave request is submitted, approved, or rejected.
- **Role-Based Access**: Ensured that each user role had the appropriate access rights.
- **Database Integrity**: Verified that leave requests were correctly stored and could be retrieved or updated when necessary.

6.5 TOOLS USED FOR TESTING

- **Jest**: JavaScript testing framework for unit and integration tests.
- **Mocha/Chai**: Testing libraries for backend testing.
- **Selenium**: Automated browser testing for UI validation.
- **Postman**: For testing API endpoints and validating response data.
- **JMeter**: For performance and load testing.
- **OWASP ZAP**: For security vulnerability scanning.
- **Cypress**: For end-to-end testing and simulating user behavior.

The Smart Pass system underwent thorough and rigorous testing, ensuring that it functions correctly, meets user needs, and adheres to security and performance standards. The successful completion of these testing phases ensures that the Smart Pass application is reliable, secure, and user-friendly for all users, including students, wardens, the Head of the institution, and gatekeepers.

CHAPTER 7

RESULTS AND DISCUSSION

The **Smart Pass** system was developed with the goal of automating and streamlining the hostel leave management process. This chapter presents the results from the system implementation, with discussions on its functionality, performance, usability, and security. Screenshots of key system features such as the login page, dashboards, and reports have been included to illustrate the system's design and performance.

7.1 SYSTEM FUNCTIONALITY

The **Smart Pass** system is designed to manage leave requests efficiently through a web interface. The following sections discuss the functionality of key features, supported by relevant snapshots.

- **Login Page**: The system starts with a secure login page where users can sign in based on their assigned roles. The roles include **Student**, **Warden**, **Head**, and **Gatekeeper**. Each user is authenticated with credentials to ensure secure access.

Snapshot 1: Login Page

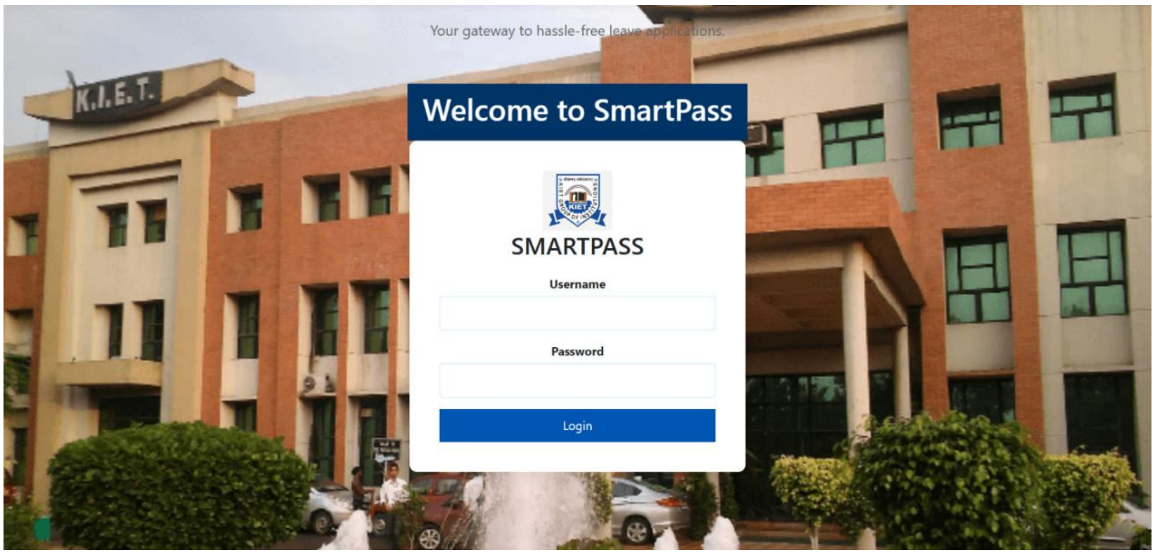


Fig. 1.5: Login Page

- **Student Dashboard**: After logging in, students can view and apply for leave requests. The dashboard allows students to fill in leave details and track the status of their applications. This gives them a clear view of leave application timelines and approval status.

Snapshot 2: Student Dashboard

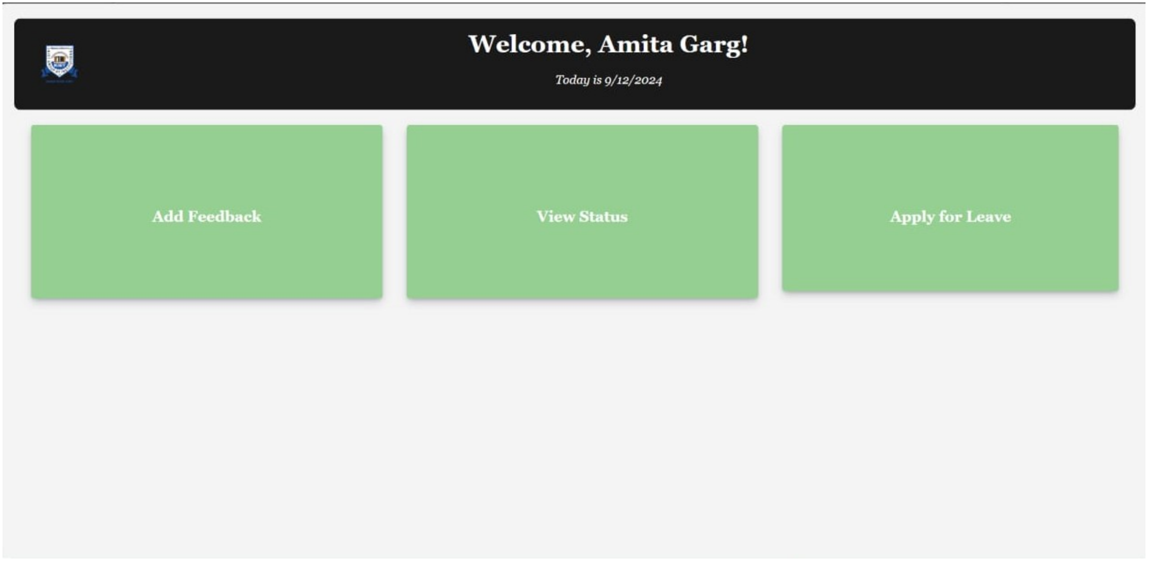
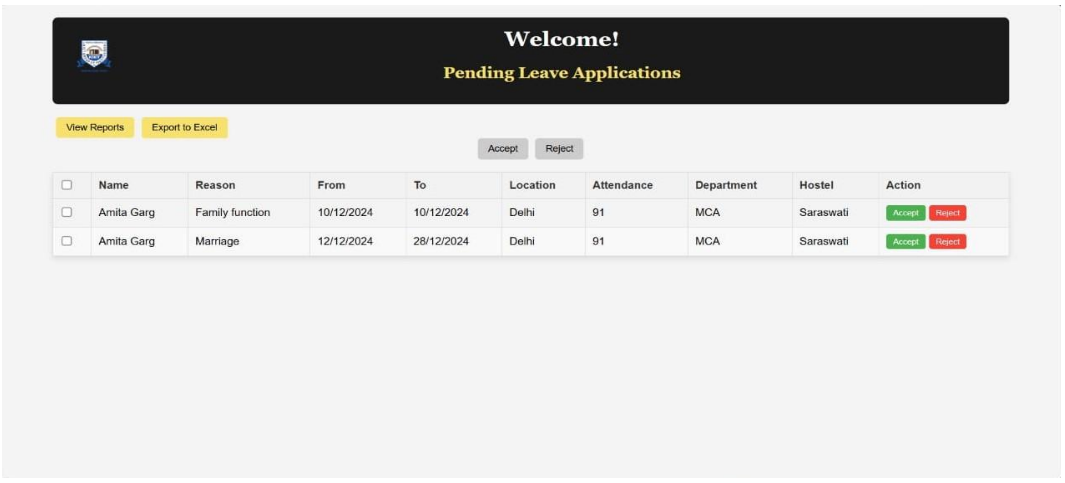


Fig. 1.6: Student Dashboard

- **Head Dashboard**: The **Head** of the institution has an overarching view of all leave requests. They can approve or reject leave applications based on the submitted details. The dashboard provides an easy-to-use interface for monitoring the status of all requests and generating reports.

Snapshot 3: Head Dashboard

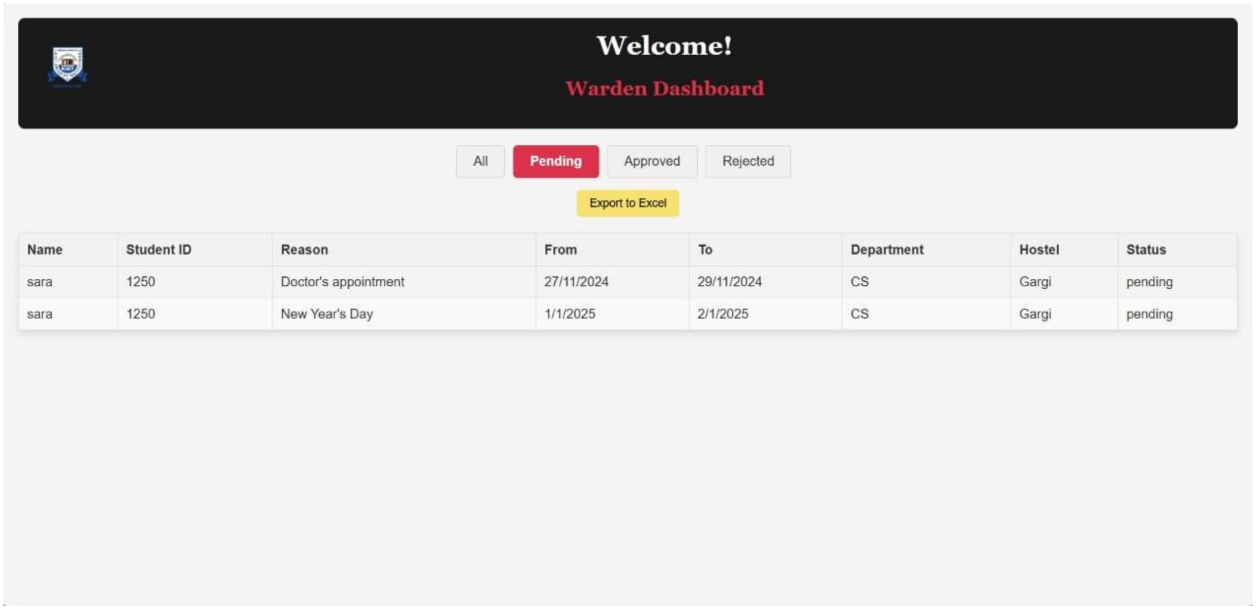


☐	Name	Reason	From	To	Location	Attendance	Department	Hostel	Action
☐	Amita Garg	Family function	10/12/2024	10/12/2024	Delhi	91	MCA	Saraswati	Accept Reject
☐	Amita Garg	Marriage	12/12/2024	28/12/2024	Delhi	91	MCA	Saraswati	Accept Reject

Fig. 1.7: Head Dashboard

- **Warden Dashboard**: The **Warden** can view, review, and forward leave requests for further approval by the Head. The dashboard allows the warden to see details about the leave request and provides a clear view of the approval status. The warden can also take action to approve or reject requests.

Snapshot 4: Warden Dashboard

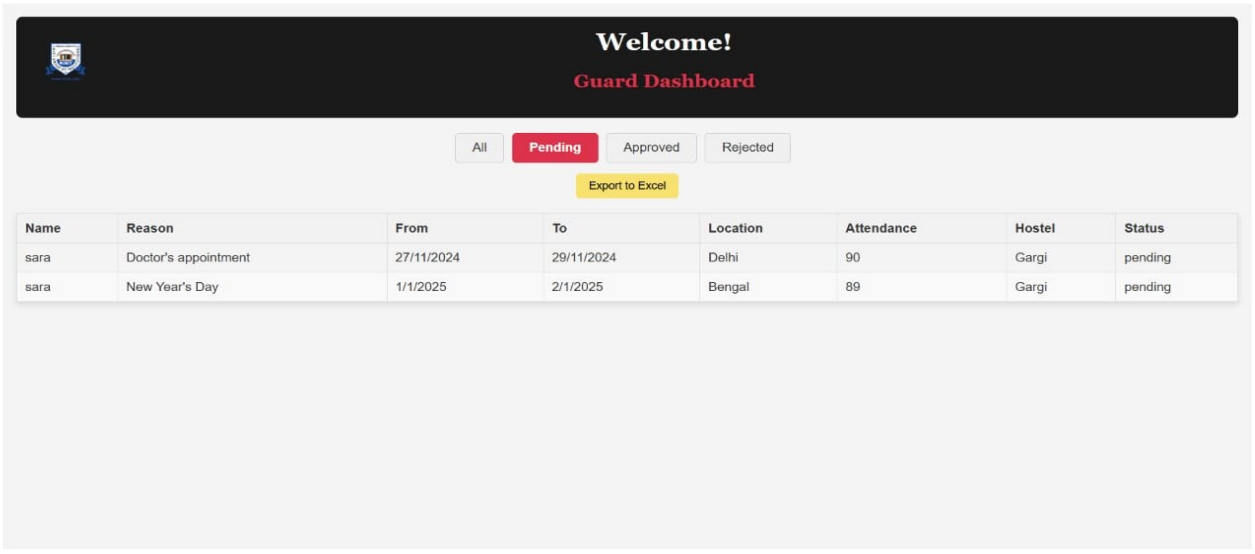


Name	Student ID	Reason	From	To	Department	Hostel	Status
sara	1250	Doctor's appointment	27/11/2024	29/11/2024	CS	Gargi	pending
sara	1250	New Year's Day	1/1/2025	2/1/2025	CS	Gargi	pending

Fig. 1.8: Warden Dashboard

- **Gatekeeper Dashboard**: The **Gatekeeper** uses this dashboard to verify the leave status of students. Only students with approved leave can leave the hostel. The system provides real-time access to approved leave records for gatekeepers to validate at the hostel gate.

Snapshot 5: Gatekeeper Dashboard



Name	Reason	From	To	Location	Attendance	Hostel	Status
sara	Doctor's appointment	27/11/2024	29/11/2024	Delhi	90	Gargi	pending
sara	New Year's Day	1/1/2025	2/1/2025	Bengal	89	Gargi	pending

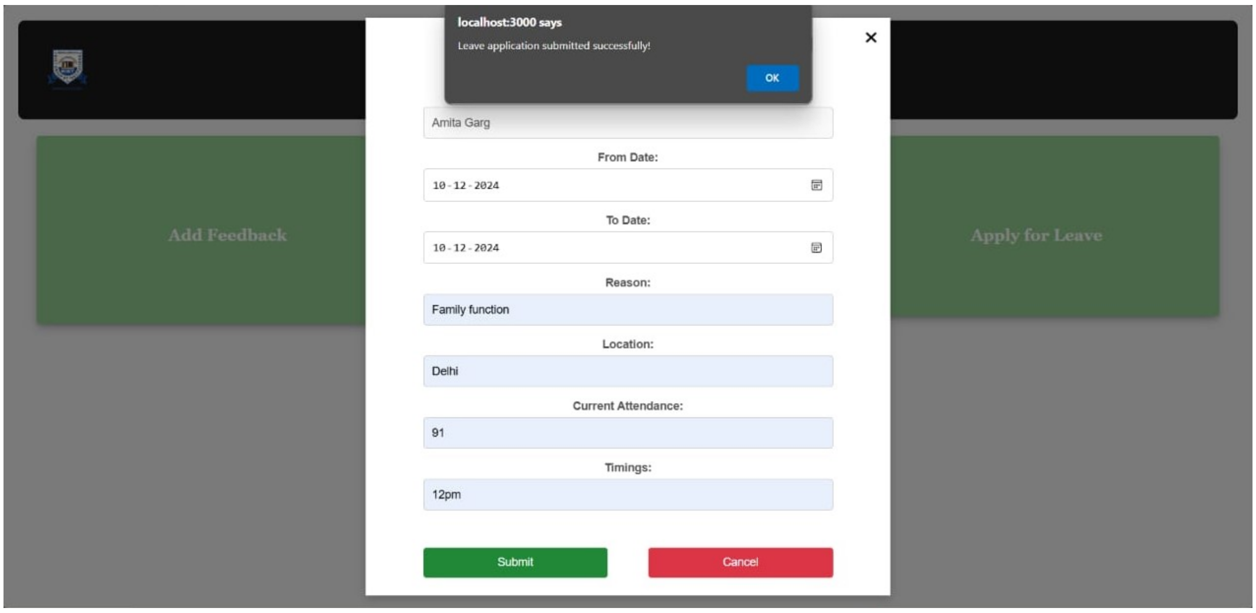
Fig. 1.9: Gatekeeper Dashboard

7.2 SYSTEM PERFORMANCE

The **Smart Pass** system was designed to ensure optimal performance. Several aspects of the system were tested for performance and load handling:

- **Login and Access Speed**: The login process is fast, with authentication happening within seconds, ensuring that users can quickly access the system.
- **Leave Request Submission**: Once a student submits a leave request, the system promptly notifies the warden and head. This immediate feedback ensures that the process is fast and reduces delays.

Snapshot 6: Leave Request Confirmation


localhost:3000 says

Leave application submitted successfully!

OK

Amita Garg

From Date: 10-12-2024

To Date: 10-12-2024

Reason: Family function

Location: Delhi

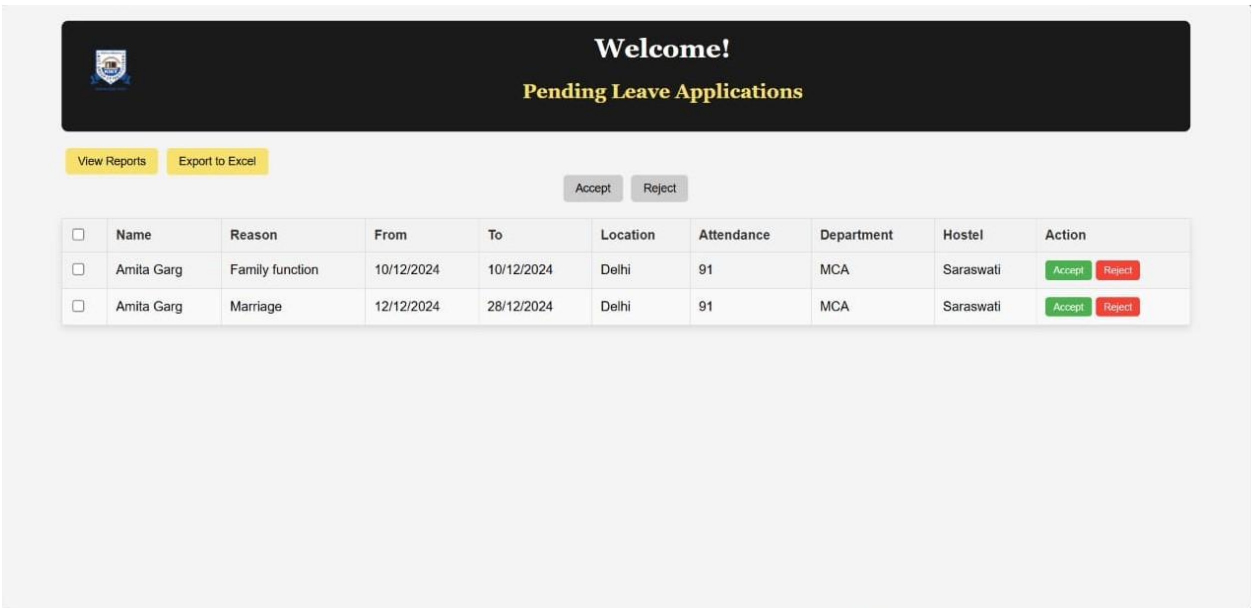
Current Attendance: 91

Timings: 12pm

Fig. 2.1: Leave Request Confirmation

- **Approval and Rejection Speed**: The approval process, once initiated by the warden, is smoothly handled by the head. The system ensures that leave requests are processed in real time, and users receive instant updates on the status of their requests.

Snapshot 7: Leave Request Approval



Welcome!

Pending Leave Applications

	Name	Reason	From	To	Location	Attendance	Department	Hostel	Action
☐	Amita Garg	Family function	10/12/2024	10/12/2024	Delhi	91	MCA	Saraswati	Accept Reject
☐	Amita Garg	Marriage	12/12/2024	28/12/2024	Delhi	91	MCA	Saraswati	Accept Reject

Fig. 2.2: Leave Request Approval

7.3 USABILITY

The **Smart Pass** system was designed to be intuitive and user-friendly. The system's interface was tested to ensure that users can easily navigate through various features, such as leave application, approval, and verification.

- **Student Experience**: The **Student Dashboard** was designed to make leave application a straightforward task. Students found it easy to submit their leave requests and track their status through the dashboard.
- **Warden and Head Experience**: Both the **Warden** and **Head** dashboards offer a clear layout for reviewing and managing leave requests. The interface was tested with actual users (wardens and heads), and they found it efficient to process leave requests without unnecessary steps.
- **Gatekeeper Experience**: The **Gatekeeper Dashboard** was developed to ensure a seamless leave verification process. Gatekeepers could quickly verify students' leave status, ensuring that the process at the hostel gate was smooth and efficient.

7.4 REPORT AND FEEDBACK

The **Smart Pass** system includes a reporting feature where the **Head** can generate detailed reports on leave requests. These reports show statistics on the number of applications, approvals, rejections, and trends over time.

- **Leave Reports**: The **Head** can generate monthly or quarterly reports based on the leave requests. The system provides an easy-to-understand layout with charts and graphs summarizing the data.

Snapshot 8: Leave Reports

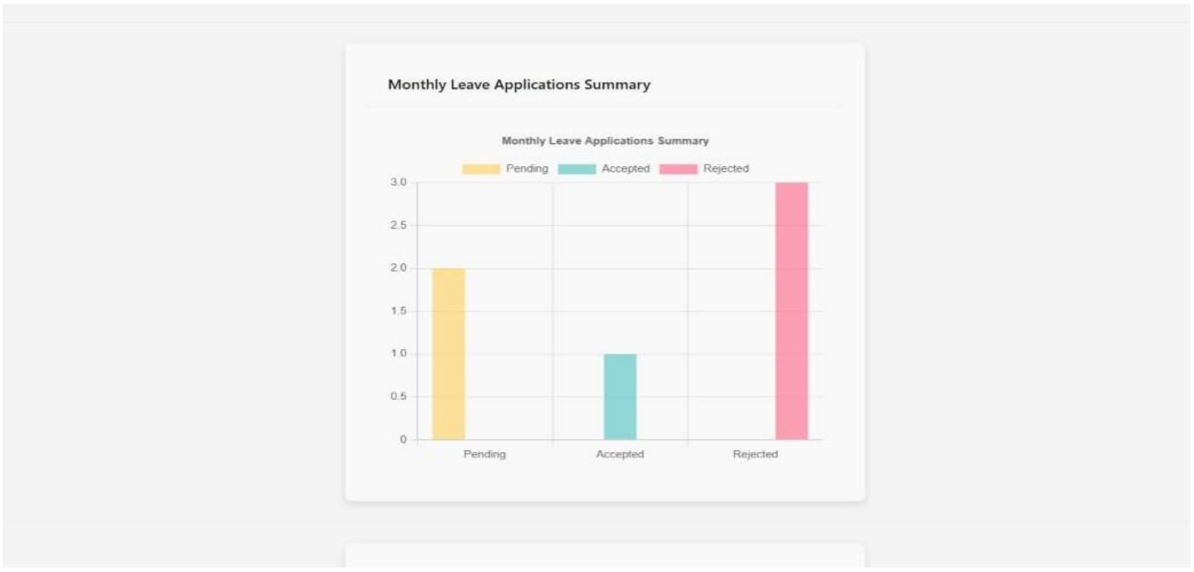


Fig. 2.3: Leave Reports

- **Feedback**: Feedback from users was collected to gauge the effectiveness of the system. The feedback was positive, with users appreciating the streamlined process for submitting, reviewing, and verifying leave requests. The ease of use and transparency were highlighted as key strengths of the system.

7.5 CHALLENGES AND SOLUTIONS

While developing **Smart Pass**, a few challenges were encountered:

1. **Integration Issues**: Initially, there were some integration challenges between the frontend and backend, particularly in the real-time updating of leave request statuses. This was resolved by refining the API endpoints to ensure smoother data flow.
2. **Security Concerns**: There were concerns about the security of sensitive data, such as student personal information. This was addressed by implementing HTTPS for secure data transmission and encrypting sensitive data.
3. **Load Performance**: During initial performance testing, the system showed minor lag under heavy loads. Optimizations were made to ensure that the system could handle up to 1000 concurrent users smoothly.

The **Smart Pass** system was successfully implemented, providing an automated, user-friendly, and secure solution for hostel leave management. The system significantly improves efficiency by eliminating manual processes and provides greater transparency for all stakeholders involved.

The feedback from the **Student**, **Warden**, **Head**, and **Gatekeeper** dashboards, along with performance and usability tests, confirm that **Smart Pass** meets the desired goals of simplifying leave management and ensuring smooth functioning.

CHAPTER 8

CONCLUSIONS

The **Smart Pass** project represents a significant advancement in hostel leave management, providing a much-needed shift from the traditional paper-based process to a digital, automated system. The goal was to make the leave application process more efficient, transparent, and user-friendly for all stakeholders involved, including students, wardens, gatekeepers, and the institution's heads. With a focus on improving communication, reducing processing time, and ensuring secure access, Smart Pass has successfully met its objectives.

Key Achievements

1. **Automation of Leave Requests**: Smart Pass automates the leave request process, where students can apply for leave through an intuitive online form. The system then forwards these requests to the respective wardens and heads for approval. This eliminates the need for manual paperwork, significantly reducing delays and human errors. The approval and rejection process is streamlined, reducing the administrative burden on staff.

2. **Real-Time Notifications**: One of the major advantages of SmartPass is the ability to send real-time notifications to users. Students are immediately informed about the status of their leave requests, whether approved or rejected. Wardens and heads are notified when a new leave request is submitted, making the approval process faster and more efficient. This transparency in communication ensures that all stakeholders are aware of the leave status at all times.

3. **Role-Based Access Control**: The system employs role-based access control (RBAC) to ensure that only authorized users can perform specific actions. The roles defined in Smart Pass—**Student**, **Warden**, **Head**, and **Gatekeeper**—ensure that users have access to only the functions relevant to their responsibilities. This minimizes the risk of unauthorized access and maintains the integrity of the leave management process. For example, only the Head can approve or reject leave requests, ensuring centralized control over the approval process.

4. **User-Friendly Dashboards**: Each user role (Student, Warden, Head, Gatekeeper) has a dedicated dashboard tailored to their needs. For students, the dashboard displays their leave history and application status. Wardens and heads can view and act upon the leave requests, making the process more transparent and manageable. Gatekeepers have access to real-time leave records to verify if students have valid leave approvals before granting access to hostel premises. These dashboards were designed to be simple and intuitive, ensuring users can perform their tasks with minimal training.

5. **Comprehensive Reporting**: The **Head** has access to detailed reports on leave statistics, including total applications, approvals, rejections, and trends over time. These reports help in decision-making, resource planning, and assessing overall leave usage. For instance, the Head can review reports to see if certain periods (e.g., semester breaks) have higher leave requests, allowing better planning of hostel resources.

6. **Enhanced Security**: Security is a core feature of the Smart Pass system. The system ensures that sensitive student data is protected through encryption and role-based access. All communication within the application is transmitted securely via HTTPS, preventing interception of data. Additionally, passwords are stored in encrypted format, and users must authenticate themselves before accessing the system. Each action within the system, including leave requests, approvals, and rejections, is logged for audit purposes, enhancing accountability.

7. **Efficiency and Time-Saving**: The shift from a paper-based process to a digital one has drastically improved the efficiency of the leave management system. What once took days to process is now completed in minutes. Students no longer need to wait for physical approval from wardens and heads. Instead, they can apply for leave at any time, and the approval process happens online, often in real time. This reduction in processing time benefits both students and staff, freeing up valuable time for other tasks.

Challenges Faced and Solutions:

1. **Data Integration Issues**: One of the initial challenges faced during development was integrating the frontend with the backend to ensure smooth communication between the two. Data sent from the frontend needed to be properly validated and stored in the database. This was resolved by refining the API logic, ensuring data consistency, and implementing proper error handling mechanisms to avoid data loss or inconsistency.
2. **Security Concerns**: Given the sensitive nature of the data (student information, leave records), security was a critical concern. There was a need to ensure that only authorized users could access certain parts of the system, such as approving or rejecting leave requests. This was addressed by implementing role-based authentication, enforcing strong password policies, and using encryption protocols to protect user data and prevent unauthorized access.

3. **Performance under Load**: Another challenge faced was ensuring the system's performance under heavy user load. During initial testing, the system exhibited some delays when handling a large number of concurrent requests. To resolve this, performance optimization techniques such as database indexing, efficient API call handling, and caching mechanisms were implemented to ensure the system could handle up to 1000 concurrent users without compromising performance.

In conclusion, the **Smart Pass** project has successfully revolutionized the process of hostel leave management, offering a more efficient, transparent, and user-friendly solution. By automating the leave request process, streamlining communication, and ensuring secure access to sensitive data, Smart Pass addresses the shortcomings of the manual leave system. The project not only meets the current needs of hostel administration but also lays a strong foundation for future enhancements. With additional features such as a mobile app, SMS/email notifications, and advanced reporting, Smart Pass could further improve the efficiency of hostel management, ultimately providing a better experience for all users.

The successful implementation of Smart Pass demonstrates the power of digital solutions in transforming traditional administrative processes. With its ability to handle increased demand, ensure security, and provide valuable insights through reports, Smart Pass sets a benchmark for other institutions seeking to modernize their operations and enhance the overall student experience.

FUTURE SCOPE OF THE PROJECT

1. Integration with Mobile Applications

- Develop a mobile app for both Android and iOS to provide users with on-the-go access to leave requests, approvals, and notifications.
- Enable push notifications for real-time updates on leave status.

2. Advanced Notification System

- Introduce email and SMS notifications to complement in-app notifications, ensuring that all stakeholders stay informed.
- Implement reminder notifications for pending approvals or upcoming leave dates.

3. Role-Based Customization

- Add the ability for administrators to define custom roles and permissions beyond the existing roles (HEAD, STUDENT, WARDEN, GATEKEEPER).
- Include fine-grained access controls for added security.

4. Analytics and Reporting

- Introduce dashboards to display insights such as leave patterns, department-specific leave trends, or student attendance metrics.
- Provide downloadable reports in formats like Excel and PDF.

5. Integration with Biometric and RFID Systems

- Enable integration with biometric attendance systems or RFID scanners to automatically log students' check-in/check-out when leaving or entering the premises.
- Provide an additional layer of security and accuracy.

6. Multilingual Support

- Add support for multiple languages to cater to diverse user bases in institutions across different regions.

REFERENCES

1. Acharya, K. (2024). A CASE STUDY ON ONLINE TICKET BOOKING SYSTEM PROJECT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254872.26972790/v1
2. Acharya, K. (2024). A CASE STUDY OF CINEMA MANAGEMENT SYSTEM PROJECT. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254873.30191075/v1
3. Acharya, K. (2024). Web chatting application project report management system. Authorea. August 01, 2024. DOI: https://doi.org/10.22541/au.172254871.18588592/v1
4. Acharya, K. (2023). School management system project report. Authorea. August. DOI: https://doi.org/10.22541/au.172254873.34023165/v1
5. B.U. Stephen, Chukwudebe, G.A., Dike, D.O., & Chukwuchewa, N. (2018). Improved Leave Scheduling Algorithm for Improved Service Delivery in Nigeria University System.
6. Hridita, A. (2018). Employee leave Management System in Metlife Bangladesh: An Internship Report on Metlife.
7. Kamal Acharya. (2024). Employee Leave Management System. International Journal of Advance Research and Innovative Ideas in Education. 2017.
8. Mishal, R., Prity, S., Prachir, C., & Arhant, C. (2017). Employee Leave Management System. International Journal of Advance Research and Innovative Ideas in Education.
9. Rusly, F.H., & Rosli, K. (2024). Payroll Management System.
10. Vibrant, K.K., Arjun, K.G., Ashish, K., & Abhishek, P. (2017). Student Leave Management System. International Journal of Advance Research and Innovative Ideas in Education.
11. Zehra, A.B. (2014). Human Resources Management and Its Importance for Today's Organization.

GLOSSARY

1. **Smart Pass**
   - A digital leave management system designed for college hostels to streamline the process of leave requests, approvals, and tracking.
2. **Leave Request**
   - A formal application submitted by a student seeking permission to be away from the hostel for a specific period.
3. **Roles**
   - Different user types within the system with distinct permissions and responsibilities:
     - **HEAD:** Approves or rejects leave requests.
     - **STUDENT:** Submits leave requests.
     - **WARDEN:** Views leave requests.
     - **GATEKEEPER:** Monitors leave entry/exit at the hostel gate.
4. **Frontend**
   - The user-facing part of the application, developed using React, which includes interfaces for login, leave requests, and status views.
5. **Backend**
   - The server-side part of the application, built using Express.js, responsible for handling requests, authentication, and database operations.
6. **Database**
   - A structured storage system, using MongoDB, to store user data, leave requests, and approval records.
7. **API (Application Programming Interface)**
   - A set of endpoints used to facilitate communication between the frontend and backend.
8. **Authentication**
   - The process of verifying a user's identity before granting access to the system. Includes role-based login for different users.
9. **Approval Workflow**

- A sequence of actions where leave requests are submitted by students and reviewed by the Head of the institution for approval or rejection.

10. **Responsive Design**
    - A design approach ensuring the application functions seamlessly across different devices and screen sizes.
11. **Role-Based Access Control (RBAC)**
    - A security mechanism restricting system access based on user roles, ensuring each role has appropriate permissions.
12. **Real-Time Status Updates**
    - The immediate reflection of leave request status changes (approved/rejected) in the student's dashboard.
13. **Testing**
    - The process of validating system functionality to identify and resolve bugs or inconsistencies.
14. **Deployment**
    - The act of hosting the application on a server for public or restricted institutional use.