

|| Jai Sri Gurudev ||

ADICHUNCHANAGIRI UNIVERSITY
B G NAGARA, KARNATAKA



A Mini Project Report On

“EMPLOYEE LEAVE MANAGEMENT SYSTEM”

Submitted in partial fulfilment for the academic year 2023-24

Bachelor of Engineering
In
Information Science and Engineering

Submitted by,

BENAKARAJ S U [21ISE006]
ARUN M B [21ISE005]

Under the guidance of:

Ms. SMITHA K
Asst. Professor,
Dept. of AI&ML
BGSIT, BG Nagara



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
B G S INSTITUTE OF TECHNOLOGY

B G NAGARA- 571448

2023-2024

||Jai Sri Gurudev||

ADICHUNCHANAGIRI UNIVERSITY
B G S INSTITUTE OF TECHNOLOGY
Department of Information Science and Engineering
BG Nagara-571448, MANDYA



CERTIFICATE

This is to certify that the mini project entitled “**EMPLOYEE LEAVE MANAGEMENT SYSTEM**” carried out by **Mr. BENAKARAJ S U**, bearing **USN: 21ISE006** and **Mr. ARUN M B**, bearing **USN:21ISE005** of **BGS Institute OF Technology**, B.G Nagara in partial fulfilment for the award of Bachelor of Engineering in **Information Science and Engineering** of **Adhichunchanagiri University** during the year 2023-24. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library.

Signature of Guide

Signature of HOD

.....
Ms. SMITHA K
Assistant Professor,
Dept. of AI&ML
BGSIT, BG Nagar

.....
Dr. SIDDHARTHA B K
Associate Professor &
HOD, Dept. of IS&E
BGSIT, BG Nagar

External Viva

Name of the Examiners

Signature with date

1. _____
2. _____

ACKNOWLEDGEMENT

We sincerely convey our regards and thanks to **Dr. B N SHOBHA, Principal, BGSIT, BG Nagar, Mandya**, for giving us a chance to carry out and present our mini project work.

Our sincere thanks to **Dr. SIDDARTHA B K, Prof. and Head of Department, IS&E, BGSIT, B G Nagar, Mandya**, for giving us a chance to carry out and present our project work with all the support and facilities.

We would like to thank **Ms. SMITHA K, Assistant Professor, Department of AI&ML, BGSIT, BG Nagar** our honourable guides who stood as an excellent guide to carry out our work has been always available as an expressive evaluator for the creation and correction of the report towards our work. They have taken pain and time to go through our work when needed.

Our heartfelt gratitude to all the teaching and non-teaching faculties of **Information Science of Engineering Department, BGSIT, BG Nagar, Mandya**, for their support and guidance towards the completion of our mini project work.

Finally, we would also extend our heartfelt thanks to our family members, classmates, friends and well-wishers for their support and encouragement throughout this effort.

BENAKARAJ S U (21ISE006)

ARUN M B (21ISE005)

ABSTRACT

Managing employee leaves efficiently is crucial for organizations to maintain productivity and employee satisfaction. Traditional methods of leave management often involve cumbersome paperwork and are prone to errors. To address these challenges, an Employee Leave Management System (ELMS) offers a digital solution to streamline the leave request, approval, and tracking processes. The ELMS provides a user-friendly interface accessible to both employees and managers. Employees can submit leave requests online, specifying the type of leave, dates, and reason, while managers can review and approve requests digitally. This system facilitates transparency and reduces processing time by automating routine tasks such as leave balance calculations and notifications. Key features of the ELMS include real-time leave status updates, integration with organizational calendars, and customizable reporting functionalities. By centralizing leave data in a secure database, the system ensures compliance with organizational policies and regulations. Overall, the Employee Leave Management System optimizes operational efficiency, enhances communication between employees and management, and contributes to a more organized and productive work environment. Employee Leave Management Systems (ELMS) are software solutions designed to automate and streamline the process of managing employee leave requests, approvals, and tracking within organizations. Traditionally, leave management has been a manual and paper-based process, prone to errors, delays, and inefficiencies. ELMS aims to address these challenges by offering a digital platform that enhances transparency, efficiency, and compliance.

CONTENTS

LIST OF CONTENTS

	Pg. No
CHAPTER -1 INTRODUCTION	1-2
1.1 Overview of the project	2
1.2 Problem statement	2
1.3 Aim and Objective	2
1.4 Importance of leave management	2
CHAPTER - 2 PROPOSED SYSTEM	3
2.1 Advantages & Disadvantages	3
CHAPTER - 3 SYSTEM REQUIREMENTS	4
3.1 Hardware requirements	4
3.2 Software(Front end & Backend) requirements	4
CHAPTER - 4 SYSTEM DESIGN	5-6
4.1 System architecture	5
4.2 Database design	6
4.3 User interface design	6
CHAPTER-5 TESTING	7-11
5.1 Introduction	7
5.2 Levels of testing	8-11
CHAPTER-6 SOURCE CODE	12-24
CHAPTER-7 SNAPSHOTS	25-28
CHAPTER-8 CONCLUSION	29
CHAPTER-9 REFERENCES	30

LIST OF FIGURES

FIG NO	CAPTION/TITLE	PAGE NO
7.1	HOME SCREEN	25
7.2	ADDITION OF USER	26
7.3	APPLY LEAVE	26
7.4	SEARCH SCREEN	27
7.5	MODIFICATION OF LEAVE	27
7.6	DISPLAY SCREEN	28

CHAPTER 1

INTRODUCTION

The Employee Leave Management System is a web-based application designed to automate and simplify the process of requesting, approving, and tracking employee leave. The system provides a centralized platform where employees can submit leave requests, managers can review and approve them, and HR can monitor leave balances and generate reports. By transitioning from manual processes to an automated system, the ELMS aims to enhance operational efficiency, improve transparency, and ensure compliance with organizational policies and legal requirements.

The primary objectives of the ELMS project are to automate leave processes, enhance transparency, improve efficiency, ensure compliance and consistency, and develop a user-friendly interface. Automation of leave processes aims to streamline the leave request and approval workflow, reducing administrative burden and eliminating manual errors. Enhanced transparency will provide employees and managers with real-time visibility into leave balances, request statuses, and historical leave data. Improved efficiency will accelerate the leave request and approval process, ensuring timely responses and reducing delays. Compliance and consistency will ensure that leave policies are applied consistently and in accordance with company rules and labor regulations. Finally, a user-friendly interface will enhance the overall user experience for both employees and managers.

Effective leave management is crucial for maintaining workforce productivity and ensuring organizational compliance with labor laws. Traditional leave management methods, often reliant on paper forms and manual tracking, can be inefficient and error-prone. In response to these challenges, the Employee Leave Management System (ELMS) project aims to develop a comprehensive, digital solution that streamlines the entire leave management.

1.1 Overview of the project

The Employee Leave Management System is a web-based application designed to automate and simplify the process of requesting, approving, and tracking employee leave. The system provides a centralized platform where employees can submit leave requests, managers can review and approve them, and HR can monitor leave balances and generate reports.

1.2 Problem statement

In many organizations, the process of managing employee leave is often fraught with inefficiencies and complexities due to reliance on traditional methods. These methods typically involve paper-based forms, manual tracking, and cumbersome approval processes. Such practices not only lead to administrative burdens but also introduce various challenges that can hinder organizational productivity and employee satisfaction.

1.3 Aim and Objective

The primary aim of the Employee Leave Management System (ELMS) is to create a streamlined, efficient, and user-friendly platform that automates the entire process of managing employee leave. By implementing this system, organizations seek to achieve several key objectives that address the shortcomings of traditional leave management methods and improve overall operational efficiency and employee satisfaction.

1.4 Importance of leave management

An effective Employee Leave Management System (ELMS) is crucial for maintaining a productive and compliant workplace. By automating the leave application, approval, and tracking processes, ELMS significantly reduces the administrative workload for HR and managers, ensuring quick turnaround times and minimizing workflow disruptions. It provides real-time access to leave balances, pending requests, and approval statuses for employees and managers, fostering transparency and clear communication. The system ensures consistent application of leave policies and helps organizations comply with labor laws, reducing the risk of legal issues and penalties. Accurate leave tracking and comprehensive reporting offer valuable insights for workforce planning and decision-making.

CHAPTER 2

PROPOSED SYSTEMS

The proposed Employee Leave Management System (ELMS) aims to automate and streamline the entire process of requesting, approving, and tracking employee leave. It features an Employee Self-Service Portal where employees can submit leave requests, view real-time leave balances, and track the status of their requests. Managers benefit from an Approval Portal, enabling them to review, approve, or reject leave requests and manage team leave schedules effectively. The HR Dashboard allows for comprehensive policy management, organization-wide leave tracking, and detailed reporting, ensuring compliance with company policies and labor laws.

2.1 Advantages and Disadvantages

Advantages:

1. Increased Operational Efficiency
2. Improved Compliance and Consistency
3. Enhanced Data Management and Reporting
4. Cost Savings
5. Scalability
6. Enhanced Security

Disadvantages:

1. Initial Cost and Resource Investment
2. Technical Challenges
3. Dependence on Technology
4. Security and Privacy Concerns
5. Over-Reliance on Automation

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Hardware requirements

For Server:

- Processor: Minimum 2.0 GHz dual-core processor
- RAM: Minimum 4 GB (8 GB or higher recommended)
- Storage: Minimum 100 GB of free disk space
- Network: Reliable internet connection with adequate bandwidth

For Client Machines:

- Processor: Minimum 1.0 GHz processor
- RAM: Minimum 2 GB
- Storage: Minimum 10 GB of free disk space
- Network: Reliable internet connection

3.2 Software requirements

Server Side:

- Operating System: Windows Server 2016 or later / Ubuntu 18.04 LTS or later /
- Web Server: Apache 2.4 / Nginx 1.14 or later
- Programming Language: PHP 7.2 or later / Python 3.6 or later / Java 8 or later

Client Side:

- Operating System: Windows 10 / macOS 10.13 or later / Ubuntu 18.04 LTS or later
- Web Browser: Google Chrome 70 or later / Mozilla Firefox 60 or later / Microsoft Edge 42 or later /
- Other Software: Adobe Acrobat Reader, Office Suite

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

The Employee Leave Management System (ELMS) follows a three-tier architecture: presentation layer, application layer, and data layer.

1. Presentation Layer:

- Components: Web and mobile interfaces.
- Technologies: HTML5, CSS3, JavaScript, Swift (iOS), Kotlin/Java (Android).
- Responsibilities: User interface rendering, input handling, API communication.

2. Application Layer:

- Components: Web server, application server, authentication, leave management, notification, reporting modules.
- Technologies: Node.js (Express), Java (Spring Boot), Python (Django), OAuth 2.0, JWT, Docker, Kubernetes.
- Responsibilities: Business logic processing, request validation, security enforcement, data interaction.

3. Data Layer:

- Components: Relational and NoSQL databases, cache, backup and recovery systems.
- Technologies: MySQL, PostgreSQL, MongoDB, Redis, automated backup tools.
- Responsibilities: Data storage and management, ensuring data integrity and security, providing data access.

4.2 Database Design

The database design for the Employee Leave Management System (ELMS) is structured to effectively manage employee data, leave requests, leave policies, and notifications through a well-organized relational database. The primary entities include Employee, Leave Request, Manager, Leave Policy, and Notification tables. The Employee table stores essential details such as Employee ID, Name, Department, Position, and Leave Balance. Each employee's leave request is recorded in the Leave Request table, containing Request ID, Employee ID, Leave Type, Start Date, End Date, and Status. Managers are documented in the Manager table with Manager ID, Name, and Department, indicating the department they manage. The Leave Policy table defines the rules for different types of leaves with Policy ID, Leave Type, Accrual Rate, and Max Balance. Lastly, the Notification table keeps track of messages sent to employees, including Notification ID, Employee ID, Message, and Date. The relationships between these tables, defined by foreign keys, ensure data integrity and consistency. For this design facilitates efficient data management and retrieval, ensuring smooth operation of the leave management process.

4.3 User interface design

The user interface design for the Employee Leave Management System (ELMS) aims to provide an intuitive and efficient experience for all users, including employees, managers, and HR administrators. The employee portal features a dashboard displaying the current leave balance, quick links to submit leave requests, view leave history, and recent notifications. The leave request form allows employees to easily select leave type, start and end dates, and submit requests for approval. The leave history section provides a comprehensive view of all past leave requests and their statuses. The manager portal includes a dashboard showing pending leave requests, a team leave calendar, and notifications, enabling managers to efficiently manage their team's leave. Managers can review leave request details, approve or reject requests, and add comments in the leave approval section. The HR administrator portal offers tools for managing leave policies, viewing all leave requests, and generating reports.

CHAPTER 5

TESTING

5.1 INTRODUCTION

The testing phase is an important part of software development. It is the processes of finding errors and missing operations and also complete verifications to determine whether the objectives are requirements are satisfied. Software testing is carried out in three steps.

The first step includes unit testing where in each module is tested to provide its correctness, to determine any missing operations and to verify whether the objectives have been met. Errors are noted down and corrected immediately. Unit testing is the important and major part of the project. So, errors are rectified easily in particular modules and program quality is increased. In this project, entire system is divided into several modules.

Second step include integration testing. If we need not be the case that software whose modules when run individually and showing perfect result with also show perfect result as whole. The individual modules are clipped under this major module and tested again and verified the results. A module can have inadvertent, adverse effect on any other on the global data structure causing serious problems.

Levels in testing:

5.1.1 Unit testing

5.1.2 Integration testing

5.1.3 Validation testing

5.2 LEVELS OF TESTING

5.2.1 UNIT TESTING

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In object-oriented programming a unit is often an entire interface, such as a class, but could be an individual method. For unit testing first we adopted the code testing strategy, which examined the logic of program. During the development process itself all the syntax errors etc. got rooted out. For this developed test case that result in executing every instruction in the program or module i.e., every path through program was tested.

User Input

User will be inputting all the data from using a web browser.

Error Handling

In this system, we have tried to handle all the errors that occurred while running the application. the common errors we saw were reading a tuple with an attribute set to null and database connection getting lost. For Testing we used Top-Down design a decomposition process which focuses as the flow of control, at latter strategies concern itself with code production. The first step is to study the overall aspects of the tasks at hand and break it into a number of independent modules. The second step is to break one of these modules further into independent sub modules.

INTEGRATION TESTING

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined may not produce the desired functions. Integrated testing is the systematic testing to uncover the errors with an interface. This testing is done with simple data and developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance.

Steps to perform integration testing:

Step 1: Create a Test Plan

Step 2: Create Test Cases and Test Data

Step 3: Once the components have been integrated execute the test cases

Step 4: Fix the bugs if any and re test the code

Step 5: Repeat the test cycle until the components have been successfully integrated.

Name of the Test	Integration testing
Test plan	To check whether the system works properly when all the modules are integrated.
Test Data	Sample text data

Table 5.2.2 Test cases for Integration Testin

5.3 SYSTEM TESTING

Ultimately, software is included with other system components and the set of system validation and integration tests are performed. System testing is a series of different tests whose main aim is to fully exercise the computer-based system. Although each test has a different role all work should verify that all system elements are properly integrated and formed allocated functions.

Name of the Test	System Testing
Item being tested	Over all functioning of File structure with all functions properly linked.
Sample Input	Sample data files
Expected Output	All the modules like adding donor details, searching donor details, etc; working as expected
Actual Output	Application reacts to user inputs in expected manner.
Remarks	Successful

Table 5.2.1 Test cases for Input-Output

5.2.2 VALIDATION TESTING

At the culmination of black box testing, software is completely assembled as a package. Interfacing errors have been uncovered and the correct and final series of tests, i.e., validation tests begin. Validation test is defined with a simple definition that validation succeeds when the software function in a manner that can be reasonably accepted by the customer.

5.2.3 OUTPUT TESTING

After performing validation testing, the next step is output testing of the proposed system. Since the system cannot be useful if it does not produce the required output. Asking the user about the format in which the system is required tests the output displayed or generated by the system is required tests the output displayed or generated by the system under consideration. The output format on the screen is found to be corrected as the format was designated in the system has according to the user needs. As for the hard copy the output comes according to the specification requested by the user. The output testing does not result in any correction in the system.

5.2.4 TEST DATA AND OUTPUT

Taking various kind soft data plays a vital role in system testing. After preparing the test data system under study is tested using the test data. While testing, errors are again uncovered and corrected by using the above steps and corrections are also noted for future use.

5.2.5 USER ACCEPTANCE TESTING

User acceptance testing of the system is the key factor for the success of the system. A system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system at the time of development and making change whenever required. This is done with regard to the input screen design and output screen design.

CHAPTER 6

SOURCE CODE

```
#include <iostream>
#include <cstdio>
#include <string.h>
#include <fstream>
#include <stdlib.h>
#include <sstream>
#define s6 "\t\t\t\t\t\t"
#define s5 "\t\t\t\t\t"
#define s4 "\t\t\t\t"
#define s3 "\t\t\t"
#define s2 "\t\t"
#define s1 "\t"
#define sp2 " "
#define nl endl
#define cyp "Choose your option : "
using namespace std;
void recover(char* , char*);
float getSal(string );
int hashed(char , char, int);
class Employee
{
    public :
    string name, ssn, sal, age, exp, casual, sick, study, parental, sdate, edate, days;
    Employee(){
        casual="20",sick="15",study="10",parental="40",sdate="-",edate="-",days="0";
    }
    void deleteLeave();
    void searchLeave();
    void modLeave();
    void dispLeave();
    void addUser();
    void modUser();
};

void addUser()
{
    Employee e;
    cout<<s4<<"Enter the ID : ";
    cin>>e.ssn;
    e.ssn.resize(8);
    cout<<s4<<"Enter the Name : ";
    cin>>e.name;
    e.name.resize(8);
```

```

        cout<<s4<<"Enter the Age : ";
        cin>>e.age;
        e.age.resize(8);
        cout<<s4<<"Enter the Salary : ";
        cin>>e.sal;
        e.sal.resize(8);
        cout<<s4<<"Enter the Experience : ";
        cin>>e.exp;
        e.exp.resize(8);
        fstream file1,file2,file3;
        file1.open("emp.txt",ios::binary|ios::app);
        file2.open("leave.txt",ios::binary|ios::app);
        file3.open("salary.txt",ios::binary|ios::app);
        if(!file1||!file2){
            cout<<s4<<"Error Occured!"<<nl;
            return;
        }
        e.casual=e.casual+"/20-0",e.sick=e.sick+"/15-0",e.study=e.study+"/10-
0",e.parental=e.parental+"/40-0";
        e.casual.resize(10),e.sick.resize(10),e.study.resize(10),e.parental.resize(10),e.sdate.resi
ze(10),e.edate.resize(10),e.days.resize(5);
        file1<<s4<<"|"<<e.ssn<<"|"<<e.name<<"|"<<e.age<<"|"<<e.sal<<"|"<<e.exp<<"|"<<n
l;
        file2<<e.ssn<<"|"<<e.name<<"|"<<e.casual<<"|"<<e.sick<<"|"<<e.study<<"|"<<e.par
ental<<"|"<<e.sal<<"|"<<e.sdate<<"|"<<e.edate<<"|"<<e.days<<"|"<<nl;
        file3<<e.ssn<<"|"<<e.sal<<"|"<<nl;
        file1.close();
        file2.close();
        file3.close();
        int index = hashed(e.ssn[0],e.ssn[1],1);

        cout<<s4<<"User added successfully!"<<nl<<nl<<nl;
    }
    int hashed(char ch1, char ch2, int n)
    {
        fstream file1;
        cout<<s4;
        for(int i=0;i<50;i++){
            cout<<".";
        }
        if(n==1)
            cout<<"Storing data at hash index "<<(ch1 + ch2)% 100;
        else
            cout<<"Retrieving data from hash index "<<(ch1 + ch2)% 100;

        for(int i=0;i<50;i++){
            cout<<".";
        }
    }

```

```
        cout<<endl;
        return (ch1+ch2)% 100;
    }
    void modUser()
    {
        char ch;
        string id,str;
        int found=0,count=0;
        fstream file1,file2;
        file1.open("emp.txt",ios::binary|ios::in);
        cout<<s4<<"Enter the Employee ID to modify : ";
        cin>>id;
        id.resize(8);
        if(!file1){
            cout<<s4<<"Error Occured"<<endl;
            return;
        }
        int index=5;
        while(file1){
            file1.seekg(index, ios::beg);
            getline(file1,str,"");

            if(str==id){
                found=1;
                break;
            }
            else{
                index+=51;
                str.clear();
                count++;
            }
        }
        file1.close();
        if(found==1)
        {
            start:
            int ch,beg,end;
            string newstr,str;
            cout<<s4<<"1. EDIT AGE"<<endl;
            cout<<s4<<"2. EDIT SALARY"<<endl;
            cout<<s4<<"3. EDIT EXPERIENCE"<<endl<<endl;
            cout<<s4<<cyp;
            cin>>ch;
            cout<<endl;
            if(ch==1 || ch==2 || ch==3){
                cout<<s4<<"Enter the new value : ";
                cin>>newstr;
            }
        }
    }
}
```

```
        else
            goto start;
        newstr.resize(8);

        file1.open("emp.txt",ios::binary|ios::in);
        file2.open("temp.txt",ios::binary|ios::app);

        while(count--){
            getline(file1,str);
            file2<<str<<endl;
        }
        if(ch==1)
            beg=3,end=2;
            else if(ch==2)
                beg=4,end=1;
            else if(ch==3)
                beg=5,end=0;
        for(int i=0;i<beg;i++){
            getline(file1,str,');
            file2<<str<<"|";
        }

        file2<<newstr<<"|";
        getline(file1,str,');
        for(int i=0;i<end;i++){
            getline(file1,str,');
            file2<<str<<"|";
        }

        while(file1){
            getline(file1,str);
            file2<<str;
            if(file1)
                file2<<endl;
        }
        file1.close();
        file2.close();
        cout<<s4<<"Employee has been updated!"<<nl<<nl;
        char fn1[]="emp.txt",fn2[]="temp.txt";
        recover(fn1,fn2);

    }
    else
        cout<<s4<<"Employee not found!"<<nl<<nl;

}

void recover(char *fn1,char *fn2)
```

```
{
    remove(fn1);
    rename(fn2,fn1);
}

void displayUsers()
{
    string str;
    fstream file;
    file.open("emp.txt",ios::binary|ios::in);
    if(!file){
        cout<<s4<<"Error Occured"<<nl;
        return;
    }
    cout<<nl<<s4<<"Employee
ID"<<s2<<"Name"<<s2<<"Age"<<s2<<"Salary"<<s1<<sp2<<"Experience"<<nl<<nl;
    while(file){
        getline(file,str,'|');
        cout<<str<<s1;
    }
    file.close();
    int t=110;
    cout<<nl<<s3;
    while(t--)
        cout<<"-";
    cout<<nl<<nl;
    file.close();
}

void modLeave()
{
    string id,str;
    int found=0,count=0;
    fstream file1,file2;
    file1.open("leave.txt",ios::binary|ios::in);
    cout<<s4<<"Enter the Employee ID : ";
    cin>>id;
    id.resize(8);
    if(!file1){
        cout<<s4<<"Error Occured"<<nl;
        return;
    }
    int index=0;
    while(file1){
        file1.seekg(index, ios::beg);
        getline(file1,str,'|');
        if(str==id){
            found=1;
        }
    }
}
```

```
        break;
    }
    else{
        index+=100;
        str.clear();
        count++;
    }
}
file1.close();
if(found==1)
{
    start:
    string sdate,edate;
    int ch,com=7,beg,end,nol,ex=0,lea,tot,days=0,leave,nod;
    float sal,per;
    cout<<s4<<"1. CASUAL LEAVE"<<endl;
    cout<<s4<<"2. SICK LEAVE"<<endl;
    cout<<s4<<"3. STUDY LEAVE"<<endl;
    cout<<s4<<"4. PARENTAL LEAVE"<<endl<<endl;
    cout<<s4<<cyp;
    cin>>ch;
    if(ch==0 || ch>4)
        goto start;
    cout<<endl;
    if(ch==1){ beg=0,end=3,tot=20,per=0.025; }
    else if(ch==2){ beg=1,end=2,tot=15,per=0.025; }
    else if(ch==3){ beg=2,end=1,tot=10,per=0.025; }
    else if(ch==4){ beg=3,end=0,tot=40,per=0.025; }
    file1.open("leave.txt",ios::binary|ios::in);
    file2.open("temp1.txt",ios::binary|ios::app);
    while(count--){
        getline(file1,str);
        file2<<str<<endl;
    }
    for(int i=0;i<2;i++){
        getline(file1,str,"|");
        file2<<str<<"|";
    }
    for(int i=0;i<beg;i++){
        getline(file1,str,"|");
        file2<<str<<"|";
    }
    cout<<s4<<"Enter the number of leaves : ";
    cin>>nol;
    while(nol>tot){
        cout<<s4<<"Cannot apply more than "<<tot<<" leaves"<<endl;
        cout<<s4<<"Enter the number of leaves : ";
        cin>>nol;
    }
}
```

```
    }
    getline(file1,str,'|');
    stringstream buf1(str);
    buf1>>lea;
    leave=lea;
    if(lea-nol<0)
        days=abs(lea-nol);
    lea=lea-nol;
    if(days!=0)
        lea=0;
    fstream buf2;
    buf2.open("mod.txt",ios::binary|ios::out);
    buf2<<lea<<"/"<<tot<<"-"<<days;
    buf2.close();
    buf2.open("mod.txt",ios::binary|ios::in);
    getline(buf2,str);
    buf2.close();
    remove("mod.txt");
    str.resize(10);
    file2<<str<<"|";
    for(int i=0;i<end;i++){
        getline(file1,str,'|');
        file2<<str<<"|";
    }
    getline(file1,str,'|');
    sal=getSal(id);

    if(leave-nol<0)
        sal=(float)sal*per*abs(leave-nol);

    ostringstream buf4;
    buf4<<sal;
    str=buf4.str();
    str.resize(8);
    file2<<str<<"|";

    cout<<s4<<"Enter the Start Date(dd/mm/yyyy) : ";
    cin>>sdate;
    cout<<s4<<"Enter the End Date(dd/mm/yyyy) : ";
    cin>>edate;
    sdate.resize(10);
    edate.resize(10);
    file2<<sdate<<"|";
    file2<<edate<<"|";
    for(int i=0;i<3;i++)
        getline(file1,str,'|');

    stringstream buf5(str);
```

```

        buf5>>nod;
        nod=nod+nol;

        ostreamstream buf6;
        buf6<<nod;
        str=buf6.str();
        str.resize(5);
        file2<<str<<"|";

        while(file1){
            getline(file1,str);
            file2<<str;
            if(file1)
                file2<<endl;
        }
        file1.close();
        file2.close();
        char fn1[]="leave.txt",fn2[]="temp1.txt";
        recover(fn1,fn2);
        cout<<s4<<"Leave updated!"<<nl<<nl;
    }
    else
        cout<<s4<<"User not found!"<<nl<<nl;
}

void dispLeave()
{
    string str;
    fstream file;
    file.open("leave.txt",ios::binary|ios::in);
    if(!file){
        cout<<s4<<"Error Occured"<<nl;
        return;
    }
    cout<<nl<<"Employee
ID\t"<<"Name\t"<<"CL(20)\t"<<"SL(15)\t"<<"STL(10)\t"<<"PL(40)\t"<<"Salary\t"<<"
Start Date"<<sp2<<sp2<<"End Date"<<sp2<<sp2<<"NoofLeaves"<<nl<<nl;
    while(file){
        getline(file,str,"|");
        cout<<str<<"\t";
    }
    cout<<nl<<nl;
    cout<<"Note : This table shows remaining leaves."<<nl<<sp2<<"CL : CASUAL
LEAVE"<<nl<<sp2<<"SL : SICK LEAVESEAVE"<<nl<<sp2<<"STL : STUDY
LEAVE"<<nl<<sp2<<"PL : PARENTAL LEAVE"<<nl<<sp2<<"Number after '-' sign indicates
extra leaves."<<nl<<nl;
    file.close();
}

```

```
void deleteLeave()
{
    string id,str;
    int found=0,count=0;
    fstream file1,file2;
    file1.open("leave.txt",ios::binary|ios::in);
    cout<<s4<<"Enter the Employee ID to delete : ";
    cin>>id;
    id.resize(8);
    if(!file1){
        cout<<s4<<"Error Occured"<<endl;
        return;
    }
    int index=0;
    while(file1){
        file1.seekg(index,ios::beg);
        getline(file1,str,"|");
        if(str==id){
            found=1;
            break;
        }
        else{
            index+=100;
            str.clear();
            count++;
        }
    }
    file1.close();
    if(found==1)
    {
        file1.open("leave.txt",ios::binary|ios::in);
        file2.open("temp1.txt",ios::binary|ios::app);
        while(count--){
            getline(file1,str);
            file2<<str<<endl;
        }
        for(int i=0;i<2;i++){
            getline(file1,str,"|");
            file2<<str<<"|";
        }
        str="20/20-0";str.resize(10);
        file2<<str<<"|";
        str="15/15-0";str.resize(10);
        file2<<str<<"|";
        str="10/10-0";str.resize(10);
        file2<<str<<"|";
        str="40/40-0";str.resize(10);
```

```

        file2<<str<<"|";
        float sal=getSal(id);
        ostringstream buf;
        buf<<sal;
        str=buf.str();
        str.resize(8);
        file2<<str<<"|";
        str="- ";str.resize(10);
        file2<<str<<"|"<<str<<"|";
        str="0";str.resize(5);
        file2<<str<<"|";
        for(int i=0;i<8;i++)
            getline(file1,str,"|");
        while(file1){
            getline(file1,str);
            file2<<str;
            if(file1)
                file2<<endl;
        }
        file1.close();
        file2.close();
        char fn1[]="leave.txt",fn2[]="temp1.txt";
        recover(fn1,fn2);
    }
    else
        cout<<s4<<"User not found!"<<endl;
}

```

```

float getSal(string id)
{
    fstream file;
    float sal;
    string str;
    file.open("salary.txt",ios::binary|ios::in);
    if(!file){
        cout<<s4<<"Error Occured"<<endl;
        return 0;
    }
    int index=0;
    while(file){
        file.seekg(index,ios::beg);
        getline(file,str,"|");
        str.resize(8);
        if(str==id){
            break;
        }
        else{
            index+=19;
        }
    }
}

```

```

        str.clear();
    }
}
str.clear();
getline(file, str, '|');
stringstream buf(str);
buf >> sal;
file.close();
return sal;
}
void searchLeave()
{
    string id, str;
    cout << s4 << "Enter the Employee ID : ";
    cin >> id;
    id.resize(8);
    fstream file;
    file.open("leave.txt", ios::binary | ios::in);
    if (!file) {
        cout << s4 << "Error Occured" << endl;
        return;
    }
    int index = 0;
    while (file) {
        file.seekg(index, ios::beg);
        getline(file, str, '|');
        str.resize(8);
        if (str == id) {
            break;
        }
        else {
            index += 100;
            str.clear();
        }
    }
    int hash = hashed(id[0], id[1], 0);
    cout << nl << "Employee
ID\t" << "Name\t" << "CL(20)\t" << "SL(15)\t" << "STL(10)\t" << "PL(40)\t" << "Salary\t" << "
Start Date" << sp2 << sp2 << "End Date" << sp2 << sp2 << "NoofLeaves" << nl << nl;
    cout << str << "\t";
    for (int i = 0; i < 9; i++) {
        getline(file, str, '|');
        cout << str << "\t";
    }
    file.close();
    cout << nl << nl;
}
int main()

```

```

{
    while(true){
        start:
        int ch;
        cout<<s3<<"----- EMPLOYEE LEAVE MANAGEMENT SYSTEM
M ---- "<<nl<<nl;
        cout<<s6<<"--- MAIN MENU--- "<<nl<<nl;
        cout<<s4<<"1. APPLY LEAVE"<<nl;
        cout<<s4<<"2. DELETE LEAVE"<<nl;
        cout<<s4<<"3. SEARCH LEAVE"<<nl;
        cout<<s4<<"4. MODIFY LEAVE"<<nl;
        cout<<s4<<"5. DISPLAY ALL LEAVES"<<nl;
        cout<<s4<<"6. USER MANAGEMENT"<<nl;
        cout<<s4<<"7. EXIT"<<nl;
        cout<<nl<<s4<<cyp;
        cin>>ch;
        cout<<nl;
        switch(ch)
        {
            case 1:
                modLeave();
                break;
            case 2:
                deleteLeave();
                break;
            case 3:
                searchLeave();
                break;
            case 4:
                modLeave();
                break;
            case 5:
                dispLeave();
                break;
            case 6:
                int ch1;
                cout<<s4<<"1. ADD USER"<<nl<<s4<<"2. MODIFY USER"<<nl;
                cout<<s4<<"3. DISPLAY ALL USERS INFORMATION"<<nl;
                cout<<nl<<s4<<cyp;
                cin>>ch1;
                cout<<nl;
                if(ch1==1)
                    addUser();
                else if(ch1==2)
                    modUser();
                else if(ch1==3)
                    displayUsers();
                break;
        }
    }
}

```

```
        case 7:
            exit(0);

        default : goto start;
    }
}
return 0;
}
```

CHAPTER 7

SNAPSHOTS AND RESULTS

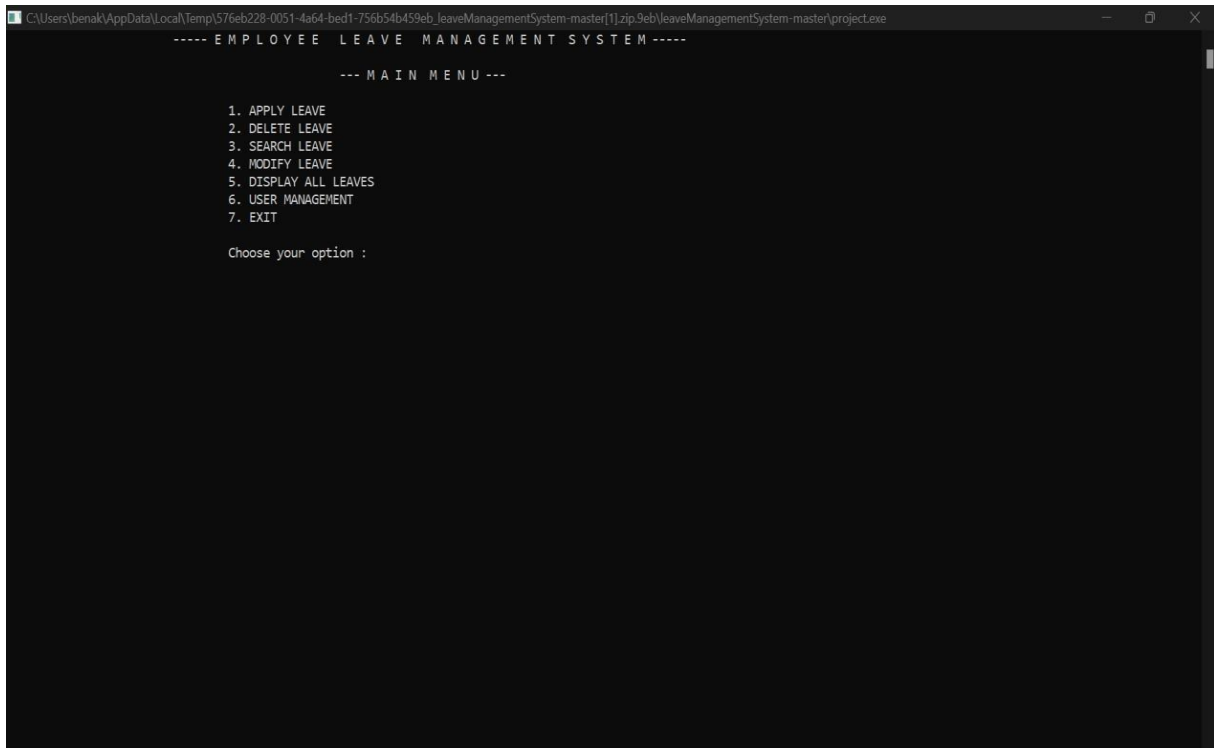


Fig 7.1 Home Screen

The image depicts a terminal-based interface for an Employee Leave Management System, presenting a main menu with several options. Users can manage their leave by applying for leave, deleting leave, searching for specific leave records, and modifying existing leave entries. Additionally, there is an option to display all leave records and manage user accounts within the system. The menu also includes an exit option to close the application. To navigate through the system, users are prompted to choose an option by entering the corresponding number.

```
1. ADD USER
2. MODIFY USER
3. DISPLAY ALL USERS INFORMATION

Choose your option : 1

Enter the ID : prajwal 123
Enter the Name :                Enter the Age : 25
Enter the Salary : 25000
Enter the Experience : 1
.....Storing data at hash index 26.....
User added successfully!
```

Fig. 7.2 Addition of user

The above figure displays a terminal interface where a user interacts with the system to manage user information. The user selects the option to add a new user by entering '1'. The system prompts for various details including ID, name, age, salary, and experience.

```
1. APPLY LEAVE
2. DELETE LEAVE
3. SEARCH LEAVE
4. MODIFY LEAVE
5. DISPLAY ALL LEAVES
6. USER MANAGEMENT
7. EXIT

Choose your option : 1

Enter the Employee ID : prajwal 123
1. CASUAL LEAVE
2. SICK LEAVE
3. STUDY LEAVE
4. PARENTAL LEAVE

Choose your option :                1. CASUAL LEAVE
2. SICK LEAVE
3. STUDY LEAVE
4. PARENTAL LEAVE

Choose your option : 1

Enter the number of leaves : 3
Enter the Start Date(dd/mm/yyyy) : 1/7/2024
Enter the End Date(dd/mm/yyyy) : 3/7/2024
Leave updated!
```

Fig. 7.3 Apply leave

In the above figure the employee is applying for a leave. The employee is applying a casual leave for three days from 1st July 2024 to 3rd July 2024. Then the leave is updated.

```

----- EMPLOYEE LEAVE MANAGEMENT SYSTEM -----

--- MAIN MENU ---

1. APPLY LEAVE
2. DELETE LEAVE
3. SEARCH LEAVE
4. MODIFY LEAVE
5. DISPLAY ALL LEAVES
6. USER MANAGEMENT
7. EXIT

Choose your option : 3

Enter the Employee ID : prajwal 123
.....Retrieving data from hash index 26.....

Employee ID   Name      CL(20)    SL(15)    STL(10)   PL(40)    Salary    Start Date  End Date  NoofLeaves
prajwal      123      17/20-0   15/15-0   10/10-0   40/40-0   25000     1/7/2024    3/7/2024    3

```

Fig. 7.4 Search screen

The above figure the leave of the employee which contains employee ID, Name, Salary, starting date of leave and last date of the leave and shows the number of leaves.

```

----- EMPLOYEE LEAVE MANAGEMENT SYSTEM -----

--- MAIN MENU ---

1. APPLY LEAVE
2. DELETE LEAVE
3. SEARCH LEAVE
4. MODIFY LEAVE
5. DISPLAY ALL LEAVES
6. USER MANAGEMENT
7. EXIT

Choose your option : 4

Enter the Employee ID : prajwal 123
1. CASUAL LEAVE
2. SICK LEAVE
3. STUDY LEAVE
4. PARENTAL LEAVE

Choose your option : 1. CASUAL LEAVE

Choose your option : 2

Enter the number of leaves : 5
Enter the Start Date(dd/mm/yyyy) : 1/7/2024
Enter the End Date(dd/mm/yyyy) : 5/7/2024
Leave updated!

```

Fig. 7.5 Modification of leave

The above figure displays the modification of the leave. Employees can use this system to manage their leave requests. Here, the user appears to be modifying a leave request. After entering their employee ID (prajwal 123), the system is prompting them to choose the specific type of leave they want to modify.

```
----- EMPLOYEE LEAVE MANAGEMENT SYSTEM -----  
  
--- MAIN MENU ---  
  
1. APPLY LEAVE  
2. DELETE LEAVE  
3. SEARCH LEAVE  
4. MODIFY LEAVE  
5. DISPLAY ALL LEAVES  
6. USER MANAGEMENT  
7. EXIT  
  
Choose your option : 5  
  
Employee ID   Name      CL(20)    SL(15)    STL(10)   PL(40)    Salary    Start Date  End Date   NoofLeaves  
prajwal      123       17/20-0   10/15-0   10/10-0   40/40-0   25000     1/7/2024    5/7/2024    8  
  
Note : This table shows remaining leaves.  
CL : CASUAL LEAVE  
SL : SICK LEAVESEAVE  
STL : STUDY LEAVE  
PL : PARENTAL LEAVE  
Number after '-' sign indicates extra leaves.
```

Fig. 7.6 Display screen

The above figure shows the display screen. The display screen displays all the leaves applied by the employee which contains his Name, Employee ID, Salary, number of leaves from starting date to last date of a leave.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

Implementing an effective employee leave management system is crucial for maintaining organizational efficiency and enhancing employee satisfaction. Such a system streamlines the process of managing various types of leave, ensuring compliance with company policies and legal regulations. By automating leave requests and approvals, administrative workloads are reduced, errors are minimized, and time is saved for both employees and HR personnel. A centralized system provides clear visibility into leave balances, policies, and the status of leave requests, promoting transparency and better communication within the organization.

Moreover, an integrated leave management system ensures adherence to labor laws and company policies, mitigating the risk of non-compliance and potential legal issues. Real-time data on employee availability facilitates more effective resource planning and allocation, helping to maintain productivity even during peak leave periods. Simplified leave processes, easy access to leave information, and timely approvals contribute to higher employee satisfaction and morale. Employees appreciate the ability to plan their time off without unnecessary delays and complications, which in turn can lead to higher motivation and loyalty to the company.

For HR professionals, an automated leave management system provides valuable insights through comprehensive reporting and analytics. This data can be used to identify patterns, predict staffing needs, and develop strategies to address absenteeism. The system also allows for seamless integration with payroll and other HR systems, ensuring accurate calculations of leave entitlements and preventing discrepancies.

REFERENCES

➤ Reference Books:

- Dessler, Gary. *Human Resource Management*. Pearson Education, 2017. This book covers various aspects of HR management, including leave management.
- Armstrong, Michael. *Armstrong's Handbook of Human Resource Management Practice*. Kogan Page, 2016. It includes comprehensive coverage of leave policies and systems.

➤ Academic Journals and Articles:

- Nawab, S., & Bhatti, K. K. (2011). "Influence of Employee Compensation on Organizational Commitment and Job Satisfaction: A Case Study of Educational Sector of Pakistan". *International Journal of Business and Social Science*, 2(8), 25-32. This article discusses the broader impacts of HR practices, including leave management.
- Lee, T. W., & Mitchell, T. R. (1994). "An Alternative Approach: The Unfolding Model of Voluntary Employee Turnover". *Academy of Management Review*, 19(1), 51-89. This article examines factors influencing employee turnover, including leave policies.

➤ Websites and Online Resources:

- Society for Human Resource Management (SHRM): www.shrm.org. SHRM provides numerous resources, articles, and best practices related to employee leave management.
- Kronos: www.kronos.com. Kronos offers information on their leave management solutions and best practices.
- BambooHR: www.bamboohr.com. BambooHR provides various resources on HR management, including leave management systems.