LEAVESYNC

" EFFORTLESS EMPLOYEE LEAVE MANAGEMENT, SEAMLESSLY SYNCED WITH SUCCESS "

A PROJECT SUBMITTED TO

Atmiya University

Department of Computer Science & Information Technology

RAJKOT

Submitted in partial fulfillment of the requirements for the degree of

"B.Sc. Information Technology"

Sem-6

(Year 2021-2024)

Submitted By: -

Jay Jayeshbhai Sanghani

Guided By: -

Mr. Malay Solanki

Mr. Kshitij Vachhani

Project ID: BIT6F2Y008 **Date:** 06-03-2024

CERTIFICATE

This is to certify that,

Mr. Jay Jayeshbhai Sanghani

Of B.Sc. Information Technology

Semester VI

Has satisfactorily completed the project on

LeaveSync

For, Department of Computer Science & Information Technology,

ATMIYA University, Rajkot.

Signature	**Signature**
Mr. Malay Solanki	**Dr. Divyesh Gohel**
(Project Guide)	(Program Coordinator)

STUDENT PROFILE

Name	Sanghani Jay Jayeshbhai
Program	B.Sc. I.T.
Enrolment No	210802100
Subject Code	21BITCC604
Class	F2
Batch	Y
Roll No	52
Project Title	LeaveSync
Address	Bl no.- 62, Madhuvan Park – 2, Bapa Sitaram Chock, Mavdi, Rajkot -360004.
Mobile	6353123580
Email Id	sanghanijay6353@gmail.com 15612121055@atmiyauni.edu.in

PROJECT PROFILE

Project Title	**“LeaveSync”**
Organization	**Atmiya University-Rajkot**
Front-End Tools	**Microsoft edge, Google chrome, VS Code**
Back-End Tools	**Xampp**
Language	**JavaScript, HTML, PHP**
Platform Used	**Git:** For tracking changes in source code. **Visual Studio Code:** For Code writing. **MS Word:** For Documentation **MS Visio:** For Diagrams
Developed By	**1) Jay Sanghani**
Project Guide	**1) Mr. Malay Solanki** **2) Mr. Kshitij Vachhani**

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to **Mr. Malay Solanki**, **Mr. Kshitij Vachhani** and for giving permission to execute this project. His constant care about me has provided a new direction to work, and his support has been invaluable throughout this project. I will always remain grateful to him for his guidance, encouragement, and motivation.

I must also express my gratitude to the B.Sc. I.T. & BCA department of Atmiya University for granting permission to do this project on campus. I consider myself privileged to have found an opportunity to express my heartily thanks to the head of the computer department. The success of any project is never limited to the individual undertaking the project. However, there are some key personalities whose role is vital.

I'm highly obliged to express my appreciation to all the officials of the Computer Department of Atmiya University for their invaluable assistance, guidance, and inspiration in this project. Without their contributions and support, this project would not have been possible.

DECLARATION

I, hereby declare that the project work entitled “**LeaveSync**” is the original work done by me, and I further declare that it is never submitted anywhere else in part or in full.

Signature

Jay Sanghani

210802100

ABSTRACT

“LeaveSync" is a Leave Management System developed in PHP, designed to streamline the process of managing employee leaves within an organization. The system grants all administrative permissions to the admin user, allowing them to add employees, departments, and leave types. In case an employee forgets their password, they can reset it through the system.

When an employee applies for leave, the admin is notified through the admin panel. The admin can then choose to approve or deny the leave request. Additionally, the system provides features to view employee salaries on a monthly basis, including incentives and PF details. Employees can also download their salary slips through the system.

Overall, "LeaveSync" simplifies the leave management process, enhances communication between employees and admin, and provides transparency in salary-related matters.

INDEX

Chapter 1 :

Introduction

1.1 PROBLEM STATEMENT

Project Title : **LeaveSync**

The current process of handling paperwork for employee information and leave management poses various challenges. Firstly, relying on physical paperwork increases the risk of human errors, misplacement of documents, and potential unauthorized access. Additionally, the manual nature of these processes is time-consuming and may result in delays and inefficiencies.

Many existing systems lack employee self-service features, meaning that employees cannot directly access and manage their personal information. This often leads to a dependency on HR departments or managers, causing unnecessary delays in obtaining essential details. For multinational companies, storing all employee information at a central headquarters makes it challenging to access this data remotely and quickly when needed.

The proposed project aims to address these issues by establishing an Employee Information System with password-protected access. This system will include details about an employee's status, educational background, and work experience, facilitating efficient monitoring of performance and achievements.

The current challenges also extend to leave management. Generating individual leave reports, searching, editing, and updating data is complicated and time-consuming. The absence of employee visibility into the leave status of their colleagues makes it difficult to manage and track various types of employee leave. The existing e-leave management system only allows employees to view their own leave applications online without providing a comprehensive leave report.

1.2 PROJECT SCOPE

1. Employee Portal:

- **Profile Management**: Employees will have the ability to view and update their personal information securely.
- **Leave Application**: A user-friendly interface for employees to submit leave requests with details such as leave type, duration, and reason.
- **Salary Slip Download**: Convenient access for employees to download their salary slips through the portal.
- **Password Management**: Secure functionality allowing employees to change and manage their passwords.

2. Admin Portal:

- **Employee Management**: Administrators can add, update, or delete employee records, ensuring an up-to-date and accurate employee database.
- **Leave Type Management**: The flexibility for administrators to modify leave types based on organizational needs.
- **Department Management**: Efficient management of departmental information for better organizational structuring.
- **Action on Leave Requests**: Administrators can promptly process leave requests, improving response time and employee satisfaction.
- **Dashboard**: A centralized dashboard providing real-time insights into leave statuses, pending approvals, and other key metrics.

3. Employee Information System:

- **Status Tracking**: The system will maintain records of employee status, educational background, and work experience for performance monitoring.
- **Password-Protected Access**: To ensure the security of sensitive employee information, access will be restricted through password protection.

4. Leave Reporting and Visibility:

- **Individual Leave Reports**: The system will allow for the generation of individual leave reports for each employee, facilitating easy tracking and management.
- **Employee Leave Summary**: An integrated summary of an employee's current leave status will be available for quick reference.
- **Inter-Employee Visibility**: Employees will have the ability to view the leave status of their colleagues, promoting transparency and coordination.

5. Notification System:

- **HR Notifications**: The system will enable HR personnel to send notifications regarding approved leave applications, ensuring timely communication.

6. Remote Accessibility:

- **Multinational Support**: The system will be designed to support remote access, facilitating the retrieval of employee information from different locations, especially for multinational companies.

1.3 PROJECT PURPOSE

The primary purpose of the Employee Leave Management System project is to develop a robust web-based tool specifically designed for efficiently managing leaves within an organization or educational institution. The project addresses several key objectives based on the identified needs and challenges:

1. Database Design and Management:

- **Objective**: Develop a well-designed database to store comprehensive employee information.
- **Purpose**: To create a centralized and organized repository for employee data, ensuring easy accessibility and data integrity.

2. Efficient Information Retrieval:

- **Objective**: Enable easy retrieval of employee information.
- **Purpose**: Facilitate quick and hassle-free access to employee data for HR personnel, managers, and employees themselves, streamlining various HR processes.

3. Leave Calculation Automation:

- **Objective**: Implement a system for the easy calculation of various types of leaves.
- **Purpose**: Automate the process of calculating leave balances, ensuring accuracy and reducing the administrative burden on HR personnel.

4. Leave Request Status Checking:

- **Objective**: Allow employees to check the status of their leave requests and those of their colleagues.
- **Purpose**: Provide transparency and real-time visibility into the status of leave applications, fostering better communication and coordination among team members.

5. Employee Leave Reporting:

- **Objective**: Generate employee leave reports for detailed analysis.
- **Purpose**: Offer a tool for HR and management to analyze leave patterns, trends, and overall leave utilization, aiding in strategic decision-making.

6. Web-Based Automation:

- **Objective**: Develop an intranet-based application accessible throughout the organization or specific departments.
- **Purpose**: Streamline the workflow of leave applications and approvals, promoting a more efficient and paperless process.

7. User-Friendly Leave Application:

- **Objective**: Provide an easy way for employees to apply for leave.
- **Purpose**: Simplify the leave application process, allowing employees to log in securely, apply for leave, and provide reasons for approval. The status of their leave requests will be easily accessible in their accounts.

Chapter 2 :

Requirements And Analysis

2.1 SYSTEM ANALYSIS

The model that is basically being followed is the WATER FALL MODEL, which states that the phases are organized in a linear order. First of all, the feasibility study is done. Once that part is over the requirement analysis and project planning begins. If system exists one and modification and addition of new module is needed, analysis of present system can be used as basic model.

The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are: -

- Requirement Analysis
- Project Planning
- System design
- Detail design
- Coding
- Unit testing
- System integration & testing

Here the linear ordering of these activities is critical. End of the phase and the output of one phase is the input of another phase. The output of each phase is to be consistent with the overall requirement of the system. Some of the qualities of spiral model are also incorporated like after the people concerned with the project review completion of each of the phase the work done.

The system analysis phase involves a detailed examination of the current system, identification of user requirements, and the design of the proposed system. In the context of the Employee Leave Management System, the following key aspects will be analyzed:

Current System Assessment:

Evaluate the existing manual leave management and employee information systems.

Identify pain points, inefficiencies, and challenges in the current system.

User Requirements:

Conduct interviews and surveys to gather requirements from employees, and administrators.

Define functional and non-functional requirements for the Employee Leave Management System.

Use Case Modelling:

Develop use case diagrams to illustrate interactions between users and the system.

Identify scenarios for leave application, approval, reporting, and system administration.

Data Modelling:

Design entity-relationship diagrams (ERD) to model the relationships between different data entities.

Specify the database schema for storing employee information, leave details, and system configurations.

System Design:

Create system flowcharts to illustrate the flow of information and processes within the system. Define the architecture and components of the Employee Leave Management System.

2.2 Hardware And Software Requirements

Hardware Requirements:

Processor	**Dual-core processor, 2.0 GHz or above**
RAM	2 GB or above
Hard Disk Space	20 GB HDD or above

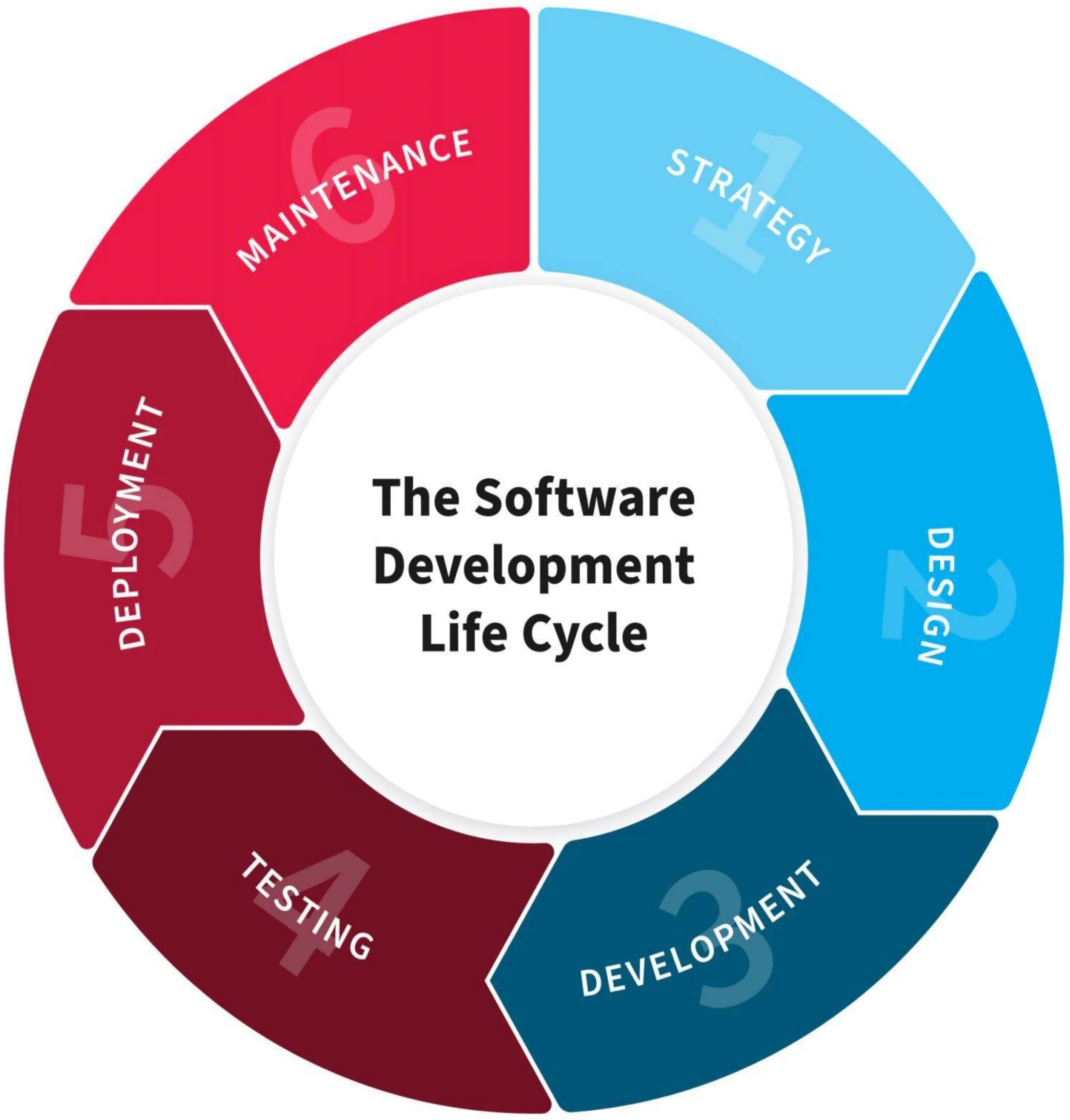
Software Requirements:

Operating System	**Windows 10 or Linux (Ubuntu, CentOS)**
Web Server	Apache or Nginx
Database Management System	MySQL 5.7 or above
Server-Side Scripting Language	PHP 7.2 or above
Frontend Technologies	HTML5, CSS3, Bootstrap 4 or 5
Text Editor or IDE	Visual Studio Code, Sublime Text, or any PHP-friendly IDE
Version Control	Git

Chapter 3:

Project Planning And Scheduling

Software Development Life Cycle (SDLC)



1. Analysis:

- **Objective**:
  - Understand and define the problem that the software is intended to solve.
- **Activities**:
  - Gather and document user requirements.
  - Conduct feasibility studies.
  - Identify constraints, risks, and potential challenges.
  - Deliverables: Requirements document, feasibility report.

2. Design:

- **Objective**:
  - Create a blueprint for the software solution based on the gathered requirements.
- **Activities:**
  - Develop high-level and low-level design specifications.
  - Define the architecture, data structures, and algorithms.
  - Consider security, scalability, and maintainability aspects.
  - Deliverables: Design documents, prototypes, and architectural diagrams.

3. Implementation:

- **Objective:**
  - Write code based on the design specifications.
- **Activities:**
  - Translate design documents into actual code.
  - Conduct unit testing to ensure individual components work correctly.
  - Follow coding standards and best practices.
  - Deliverables: Source code, unit test results.

4. Testing:

- **Objective**:
  - Ensure the software functions as intended and meets quality standards.
- **Activities**:
  - Perform various testing levels, including unit, integration, system, and

user acceptance testing.
- Identify and fix defects.
- Validate that the software meets requirements.
- Deliverables: Test plans, test cases, defect reports, and a tested software product.

5. Deployment:

- **Objective**:
  - Release the software for use by end-users or customers.
- **Activities**:
  - Develop deployment plans and installation procedures.
  - Train end-users or provide user documentation.
  - Ensure a smooth transition from development to production.
  - Deliverables: Deployed and operational software.

6. Maintenance:

- **Objective**:
  - Sustain and enhance the software to address issues and meet evolving needs.
- **Activities**:
  - Address and fix bugs and issues reported by users.
  - Implement updates and enhancements as required.
  - Provide ongoing support and troubleshooting.
  - Deliverables: Patch releases, updated documentation, user support.

Creating a Timeline:

I created a timeline that outlined the start and end dates for each task and milestone. This helped me allocate time effectively and stay on top of deadlines.

Phase 1	**:**	Definition Verification or Submission
Duration	**:**	3 weeks
Activities	**:**	

- ✓ Clearly define the project's objectives and goals.
- ✓ Identify the problem statement or need the project aims to address.
- ✓ Establish the project's scope, including what will be included and excluded from the project.
- ✓ Identify all stakeholders

Phase 2 : Requirements Gathering, Analysis and System Design and Branding

Duration : 5 weeks

Activities :

- ✓ I researched and collected design references, such as websites, apps, or projects with similar features or aesthetics to inspire my project's design.
- ✓ I analyzed any existing systems or solutions that my project might replace or integrate with.
- ✓ I identified the strengths, weaknesses, and limitations of the current system
- ✓ I identified and listed the specific features, functionalities, or components that my project needed to include to meet user requirements and project objectives.
- ✓ I prioritized these features based on their importance and feasibility.
- ✓ Create a rough system diagram or architecture that illustrates the components and their interactions within your project.
- ✓ Prepare and submit the page designs for your project's user interface (UI).
- ✓ Ensure that the designs align with the project's requirements and design references gathered in the previous phase.
- ✓ Define a project logo that represents your project's identity and purpose.
- ✓ Create a slogan or tagline that succinctly conveys the essence of your project.

Phase 3 : Implementation, Maintenance and Support

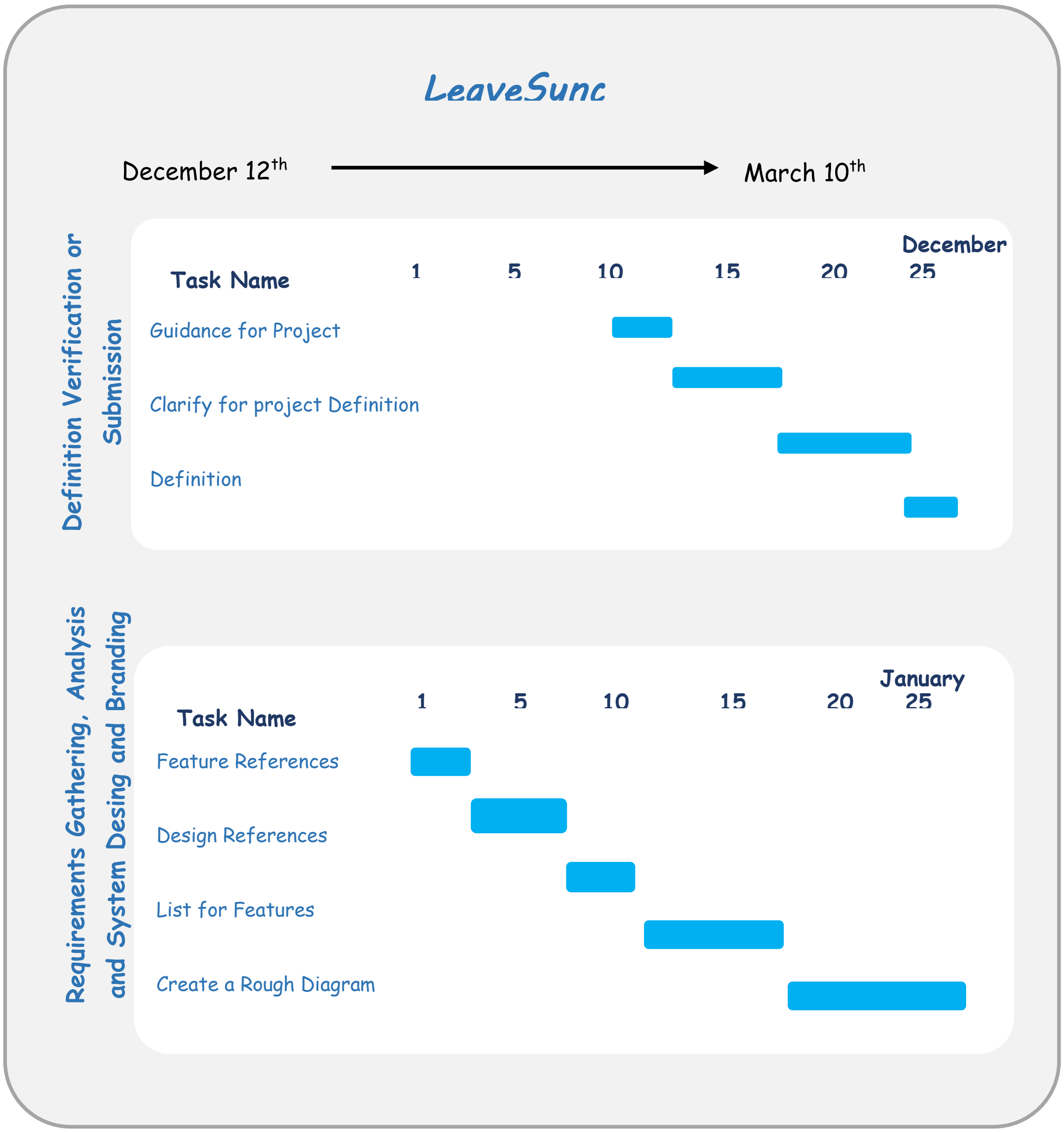
Duration : 3 weeks

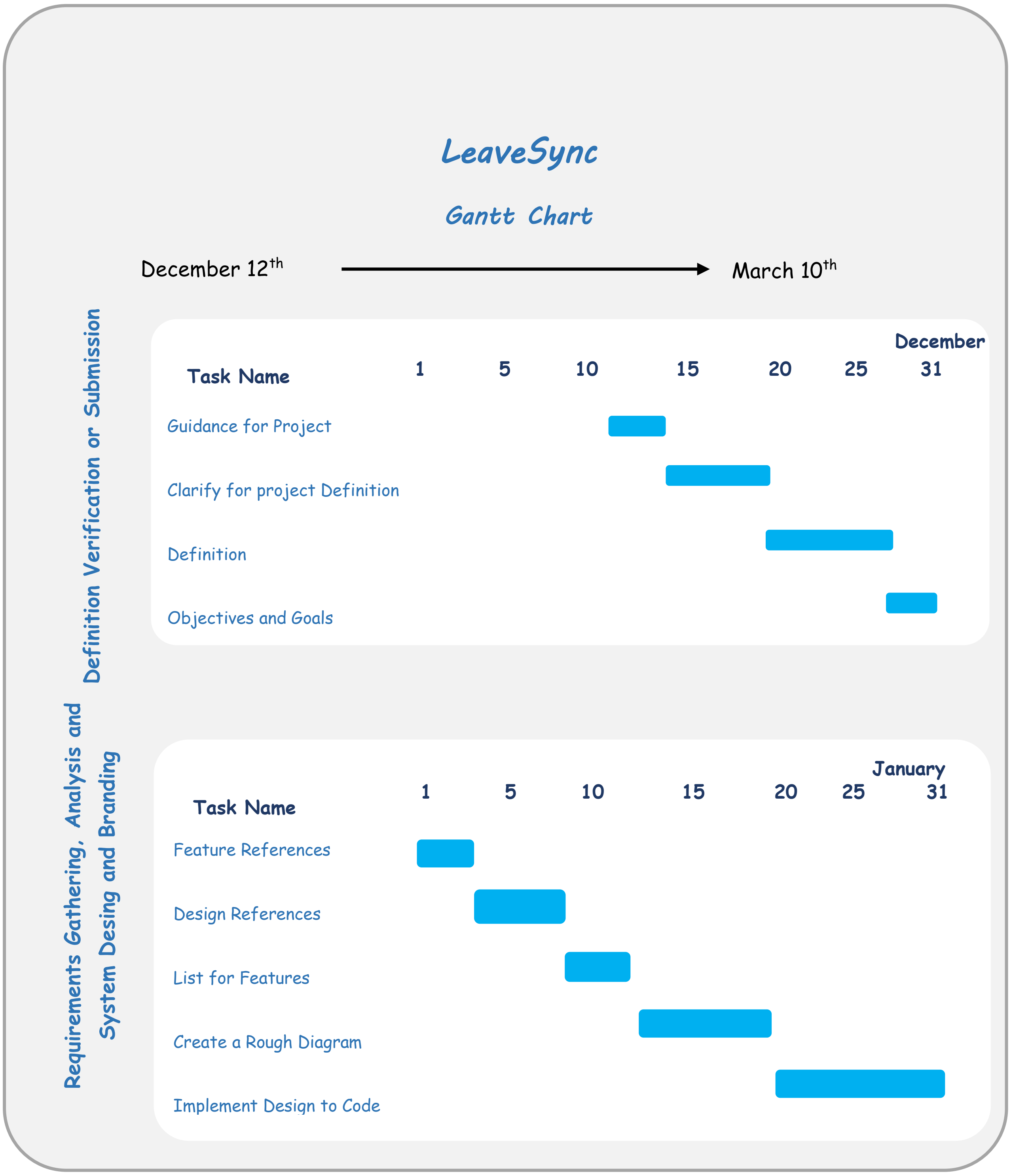
Activities :

- ✓ Conduct system testing and user acceptance testing.

- ✓ Provide user training and support.
- ✓ Monitor system performance and resolve issues.
- ✓ Provide ongoing user support and training.
- ✓ Update system features and functionalities as needed.

Phase 4	:	Documentation
Duration	:	1.5 weeks
Activities	:	Complete report of LeaveSync




LeaveSync
Gantt Chart
December 12th
March 10th
Definition Verification or Submission
Task Name
1
5
10
15
20
25
December
31
Guidance for Project
Clarify for project Definition
Definition
Objectives and Goals
Requirements Gathering, Analysis and
System Desing and Branding
Task Name
1
5
10
15
20
25
January
31
Feature References
Design References
List for Features
Create a Rough Diagram
Implement Design to Code

Chapter 4:
System Design

UML Use Case Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

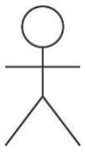
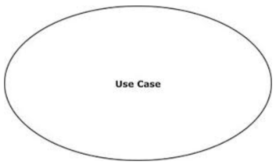
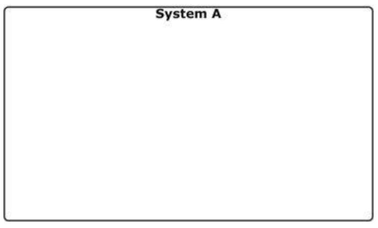
A Use Case Diagram is a visual representation that helps depict the interactions between actors (users or external systems) and a system (involving Customers, Vendors, Invoices, Admin, Employees, Items, Purchase Bills, and Employee Registration). Here's how we can create a theoretical framework for a Use Case Diagram for your project

Purpose of Use Case Diagrams

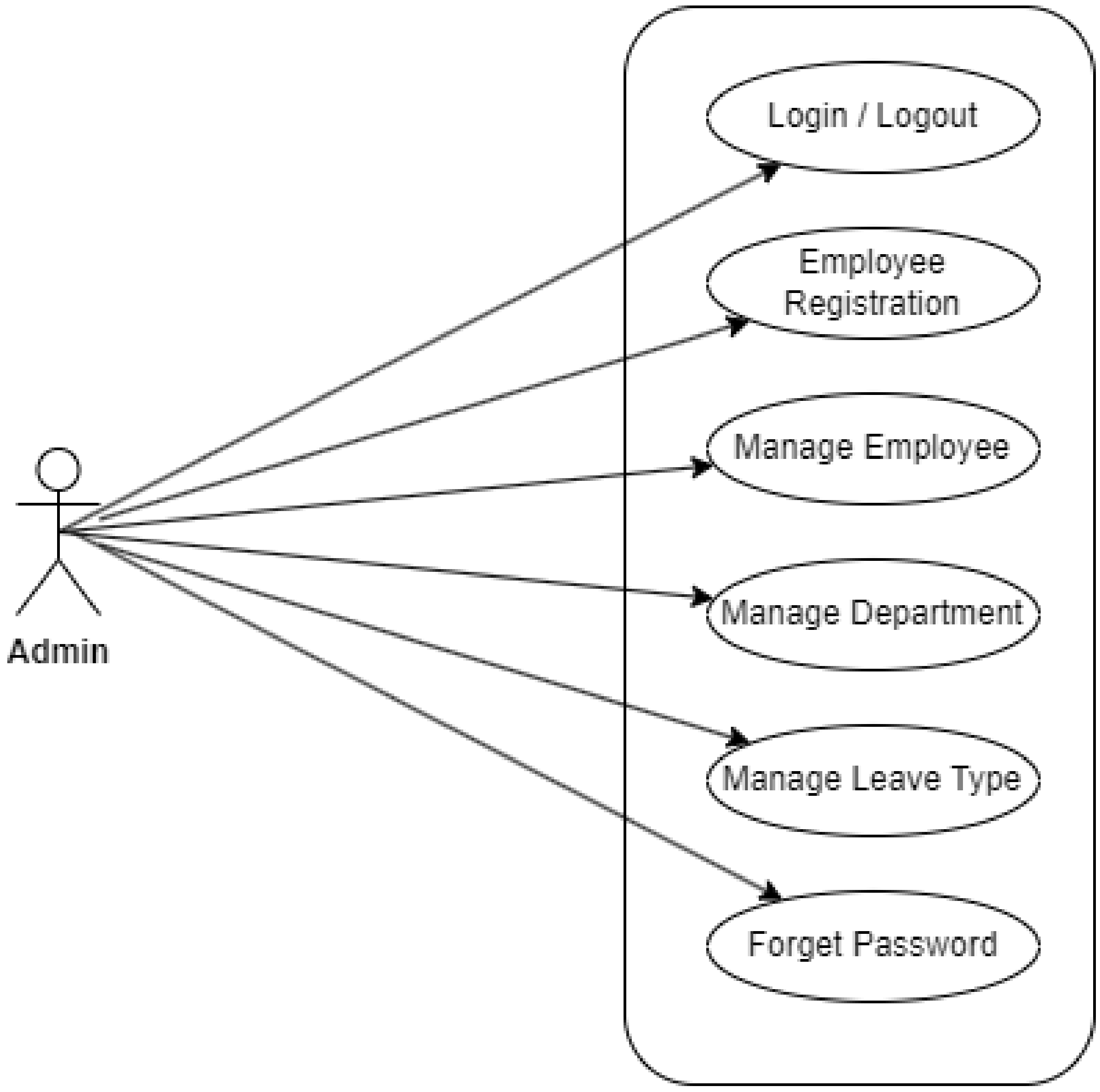
The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

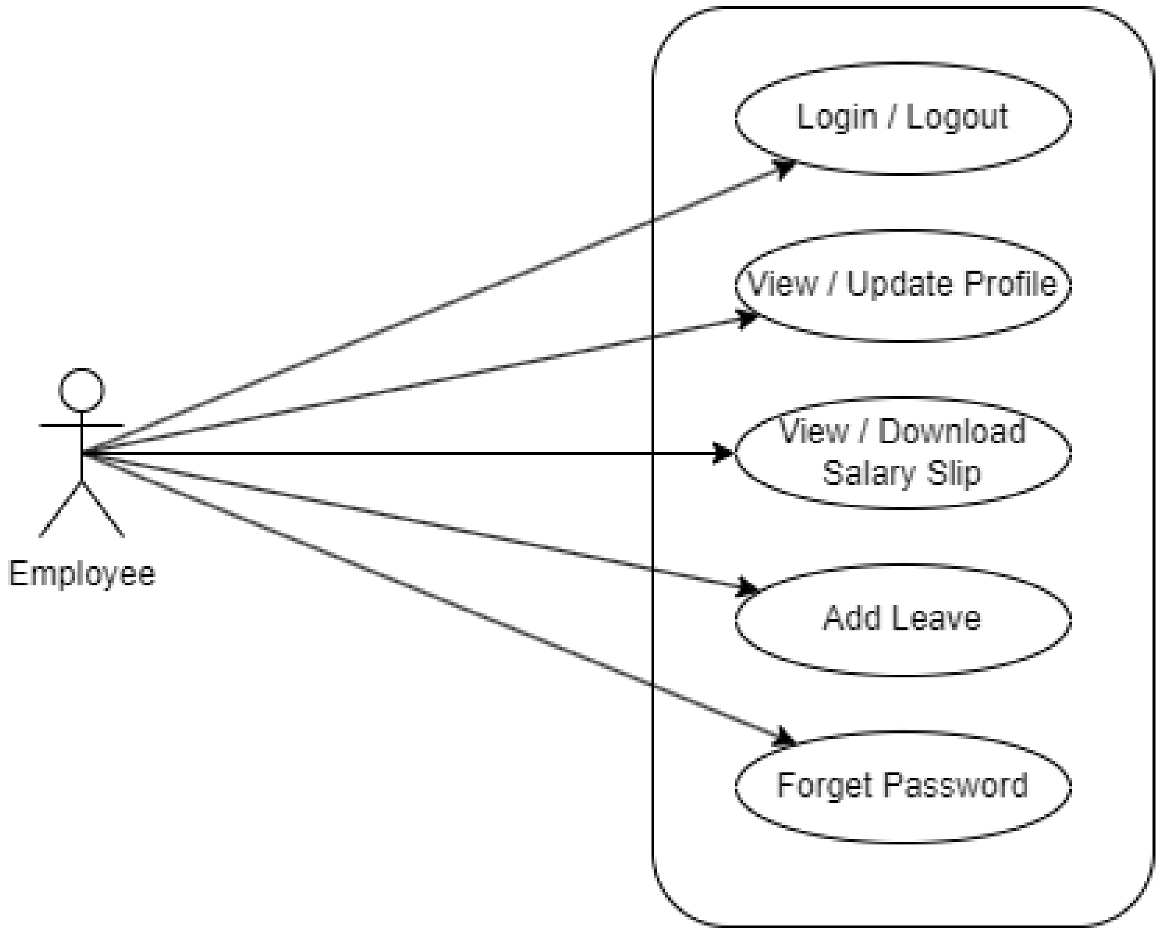
- ✓ It gathers the system's needs.
- ✓ It depicts the external view of the system.
- ✓ It recognizes the internal as well as external factors that influence the system.
- ✓ It represents the interaction between the actors.

Symbol	Explanation
 Actor	When an actor interacts with the system, it triggers a use case. Actors should be placed outside the system.
 System Use Case	An oval shape represents a use case. Use cases represent the functionality of the system, as well as the end-goal of the actor. Use cases should be placed inside the system.
Relationship	Arrows are used to indicate a relationship between an actor and a use case, or between two use cases. An extension indicates that one use case may include the behaviour of another use case. An inclusion represents one use case using the functionality of another use case.
	The rectangular boundary is the system. Use cases fall inside it, and actors will be placed outside it.

Use case of Admin



Use case of Employee



Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

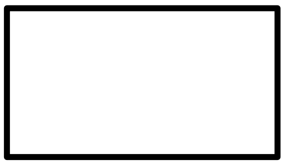
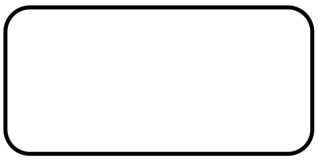
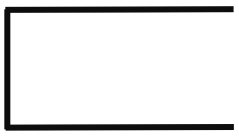
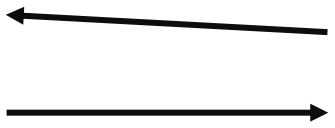
It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system.

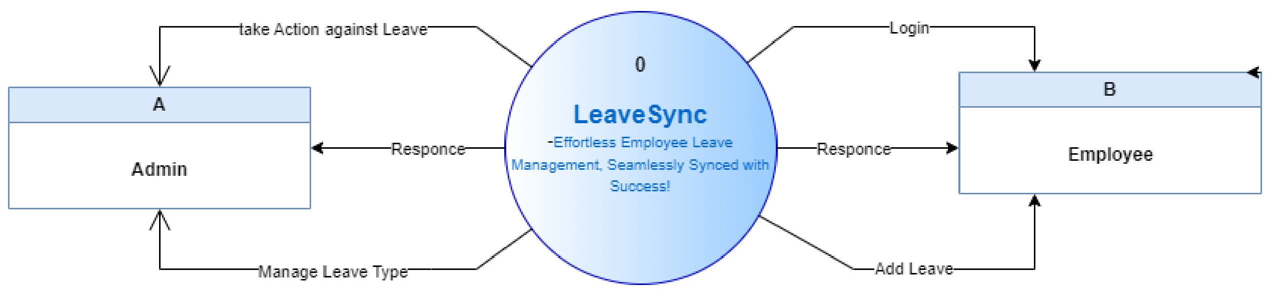
DFDs are defined in levels with every level decreasing the level of abstraction as well as defining a greater detail of the functional organs of the system. A zero level DFD also known as context or fundamental system model represents the entire software elements as a single bubble with input and output data entities which are indicated as incoming and outgoing arrows. Data Flow Diagram help understanding the basic flow of data from one process to another process. This 0 level DFD represents fundamental overview of the LeaveSync.

- To provide the logical flow of information in a system.
- To depict details about the physical connectivity and streaming of data.
- It is also used to represent the network connectivity of data in a system.
- It helps us understand the scalability of the system so that we can improve its functionality.
- Other benefits include a stepwise refinement of the system and learning the behavior of various components.

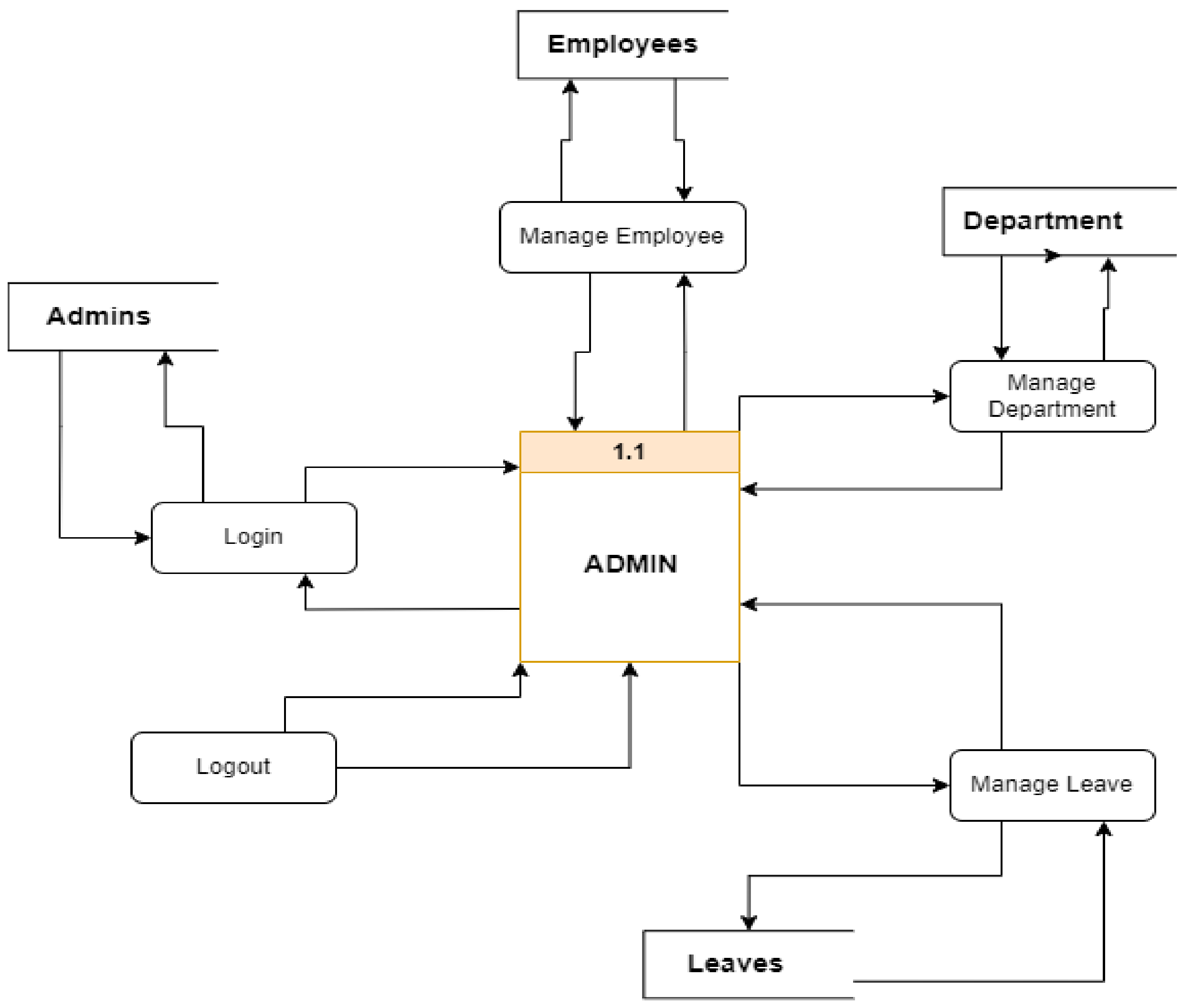
Common Flow Diagram Symbols and Meanings

Symbols	Explanation
	**External Entity** A square box defines source or destination of the system. It is also called entity. It is represented by rectangle
	**Process** It represents as a process that gives us information. It is also called processing box.
	**Data Store** An open rectangle is a data store. In this data is store either temporary or permanently.
	**Data flow** An arrow identifies the data flow i.e. it gives information to the data that is in motion

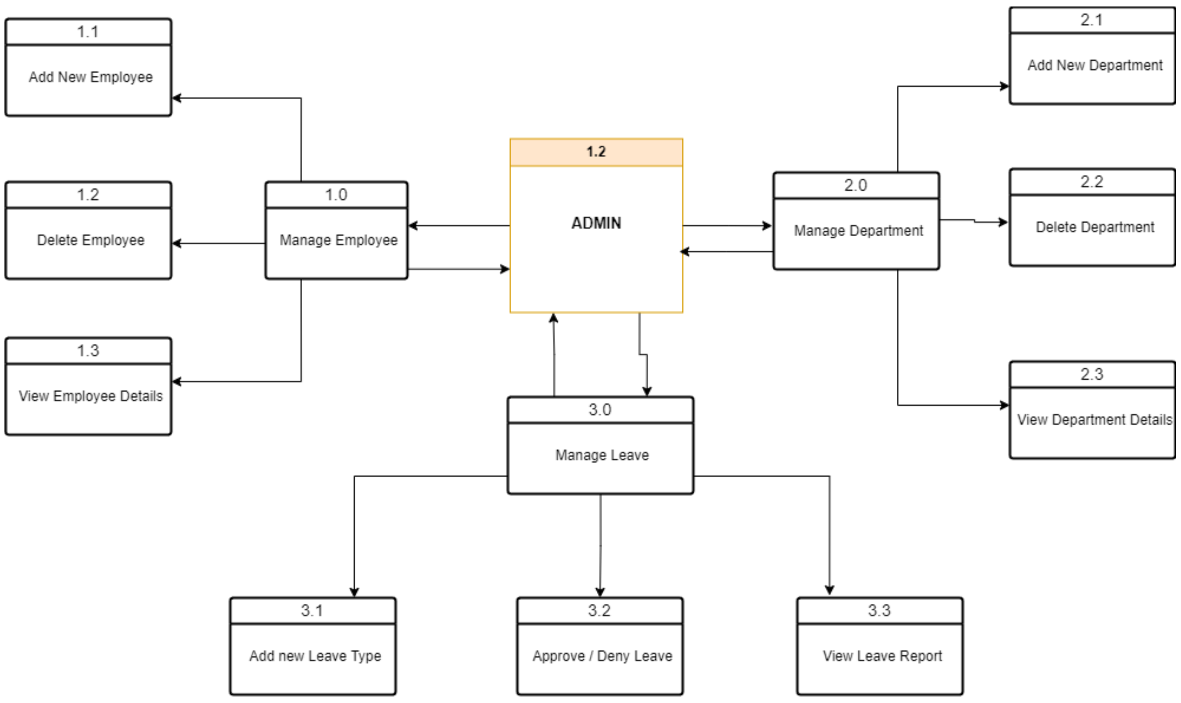
Context Diagram (0 – level dfd)


take Action against Leave
A
Admin
Responce
Manage Leave Type
0
LeaveSync
-Effortless Employee Leave Management, Seamlessly Synced with Success!
Login
B
Employee
Responce
Add Leave

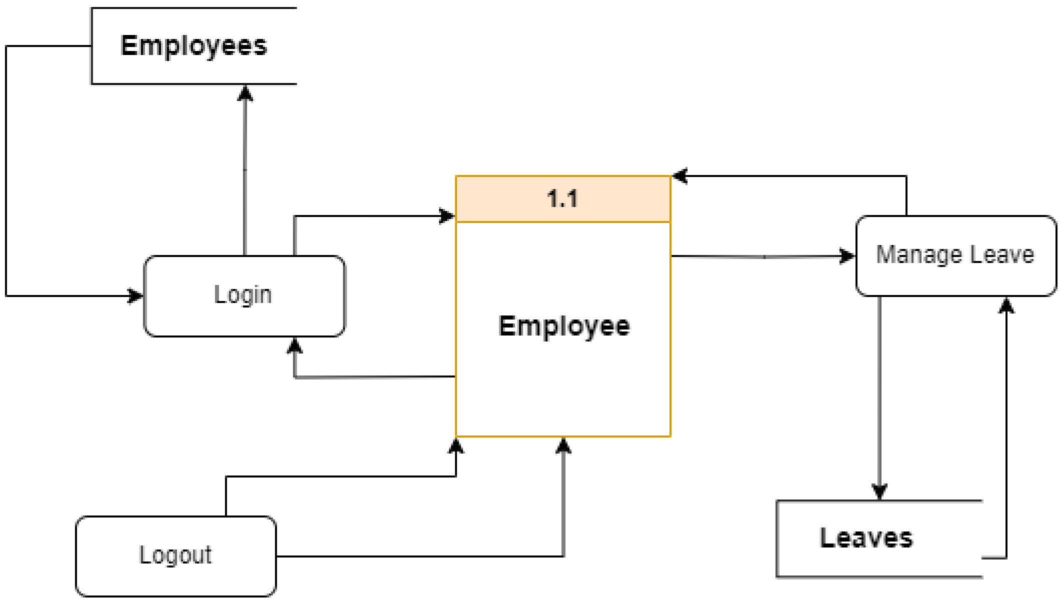
1 - Level DFD of Admin



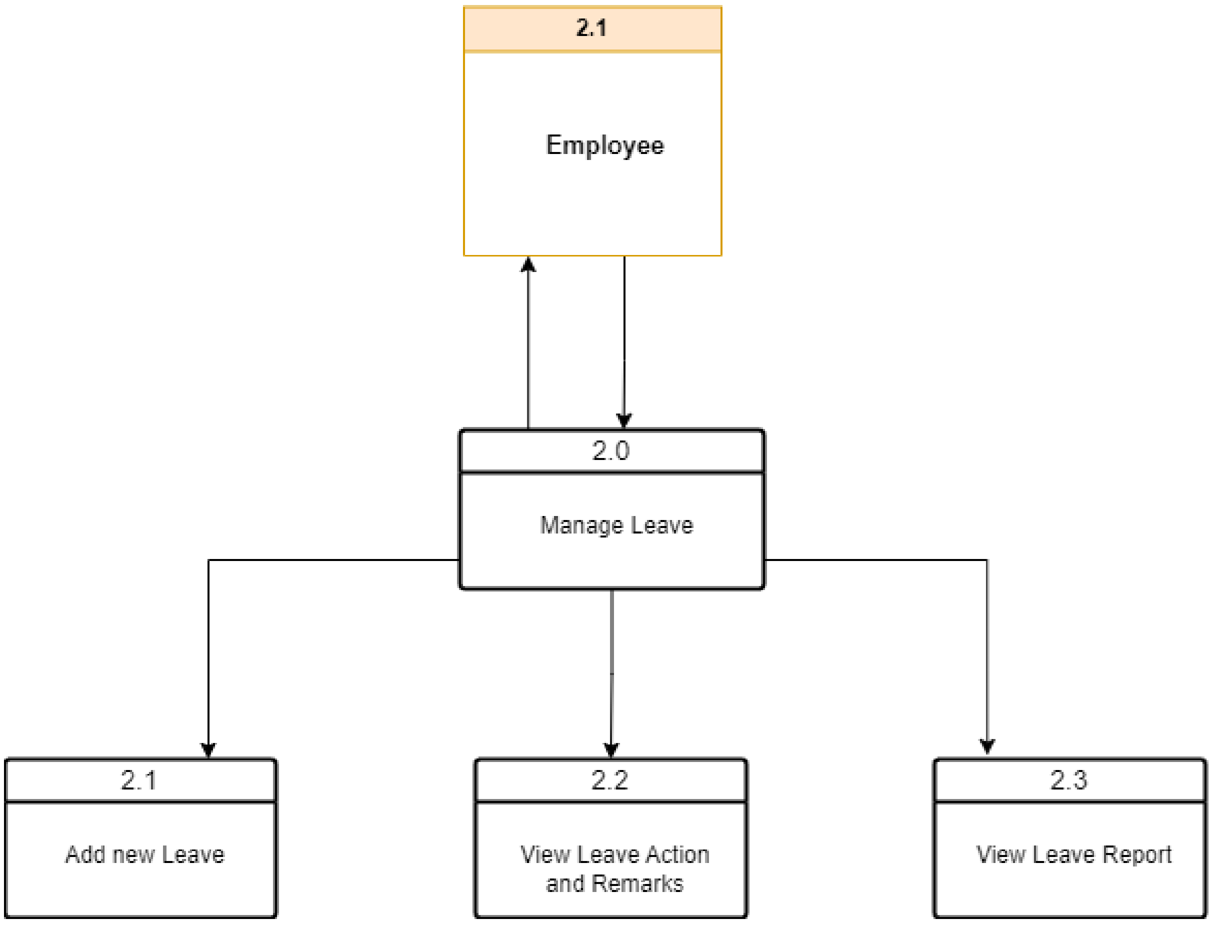
2 - Level DFD of Admin



1 – Level DFD of Employee



2-Level DFD of Employee



Entity Relationship Diagram

ER modeling is a fundamental technique used in database design. It involves defining the entities, attributes, and relationships between entities within your system. ER diagrams, as mentioned earlier, visually represent these aspects and serve as the foundation for your database schema.

1. **Employees**:

Represent individuals working within the organization.

Attributes may include employee ID, name, contact information, department, and position.

2. Leave Requests:

Represent the requests submitted by employees for taking leave.

Attributes may include request ID, leave start date, leave end date, reason, and status (pending, approved, rejected).

3. Leave Balances:

Represent the leave balances for each employee.

Attributes may include employee ID, vacation balance, sick leave balance, and other types of leave balances.

4. Supervisors:

Represent individuals responsible for approving or rejecting leave requests.

Attributes may include supervisor ID, name, and contact information.

Relationships:

In an Employee Leave Management System, relationships between entities play a crucial role:

1. **Employees and Leave Requests:**

One-to-Many Relationship: An employee can have multiple leave requests over time.

2. **Employees and Leave Balances:**

One-to-One Relationship: Each employee has a unique set of leave balances associated with their profile.

3. **Employees and Supervisors:**

Many-to-One Relationship: Many employees can have the same supervisor.

4. **Leave Requests and Supervisors:**

Many-to-One Relationship: Many leave requests can be handled by the same supervisor.

Database Schema:

The database schema for an Employee Leave Management System outlines the structure of the database:

Employee Table:

Columns: EmployeeID (Primary Key), Name, ContactInformation, Department, Position.

LeaveRequest Table:

Columns: RequestID (Primary Key), EmployeeID (Foreign Key), StartDate, EndDate, Reason, Status..

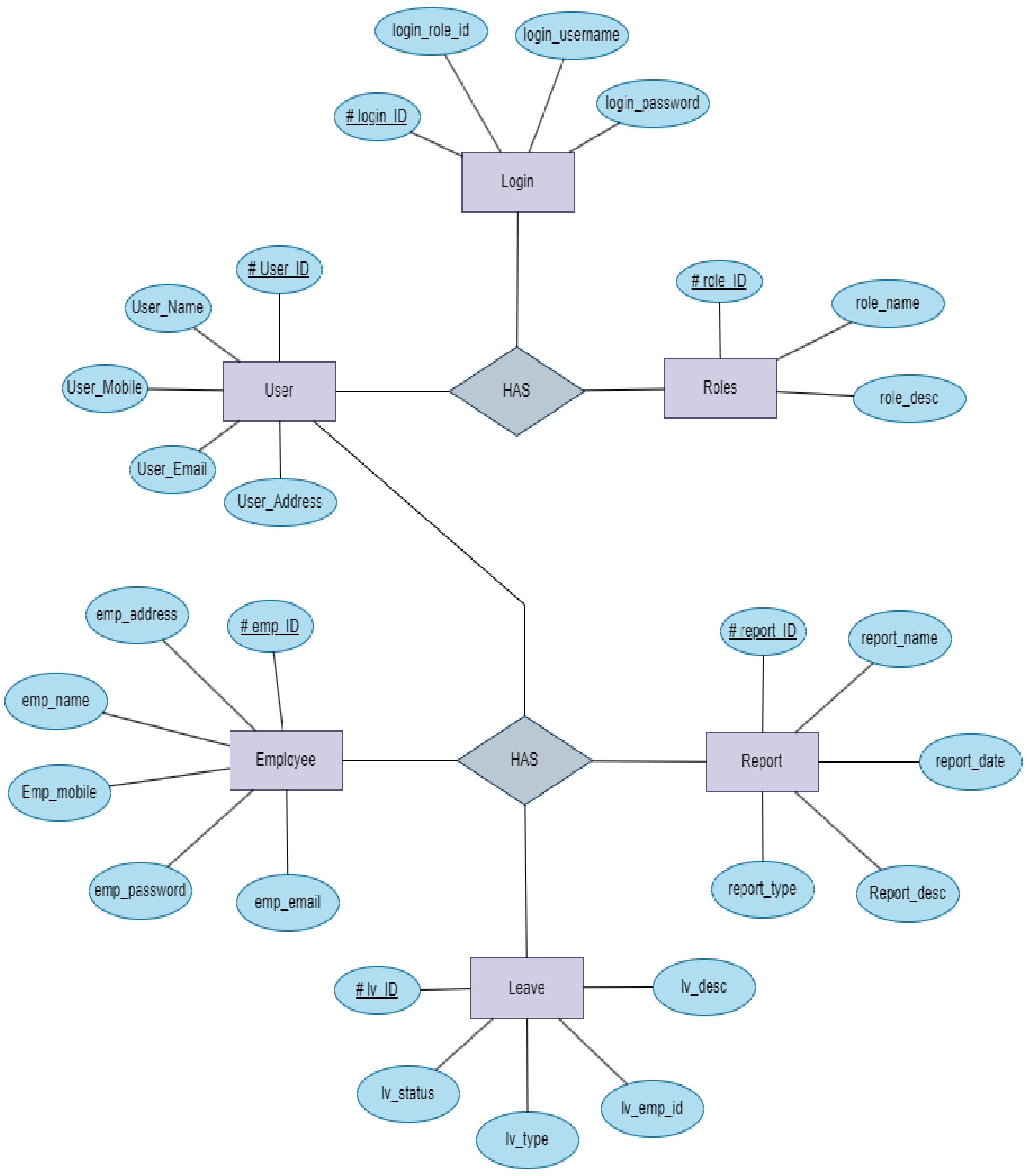
LeaveBalance Table:

Columns: EmployeeID (Primary Key, Foreign Key), VacationBalance, SickLeaveBalance, OtherLeaveBalances.

Supervisor Table:

Columns: SupervisorID (Primary Key), Name, ContactInformation.

The relationships between these tables are established through foreign key constraints, ensuring data integrity and consistency.


login_role_id
login_username
# login_ID
login_password
Login
# User_ID
User_Name
User_Mobile
User
User_Email
User_Address
HAS
# role_ID
role_name
Roles
role_desc
emp_address
# emp_ID
emp_name
Employee
Emp_mobile
emp_password
emp_email
HAS
# report_ID
report_name
Report
report_date
report_type
Report_desc
# lv_ID
Leave
lv_desc
lv_status
lv_type
lv_emp_id

Modules Design (Client and Admin)

Employee /Admin side before Login

- Admin Login
- Employee Login
- Forgot Password

Employee Side after Login

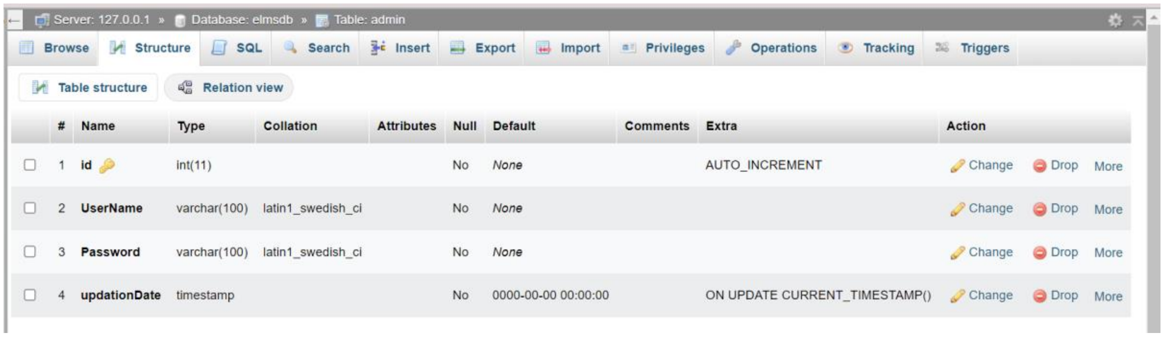
- Dashboard
- My Profile
- Leave
  - Apply Leave
  - Leave Histroy
- Salary Slip
- Change Password
- Sign out

Admin Side after Login

- Dashboard
- Department
  - Add Department
  - Manage Department
- Leave Type
  - Add Leave Type
  - Manage Leave Type
- Employee
  - Add Employee
  - Manage Employee
- Leave Management
  - All Leave
  - Pending Leave
  - Approved Leave
  - Not Approved Leave
- Change Password
- Sign out

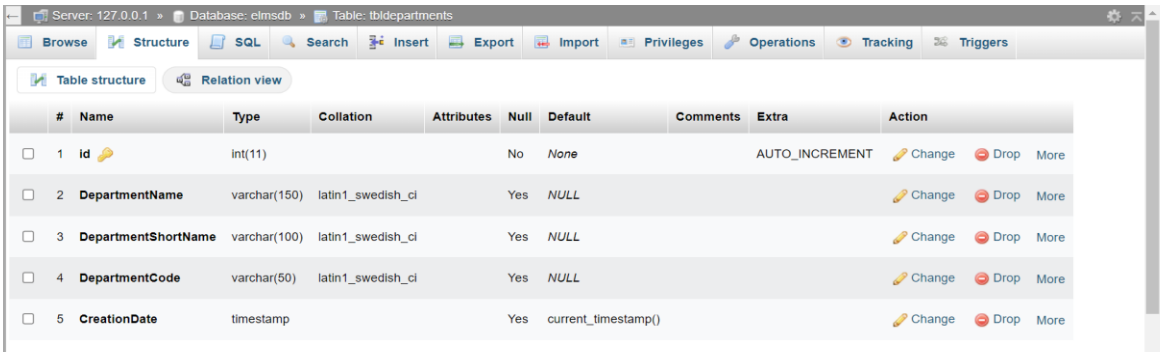
4.1.5 Data Dictionary

- **Admin**



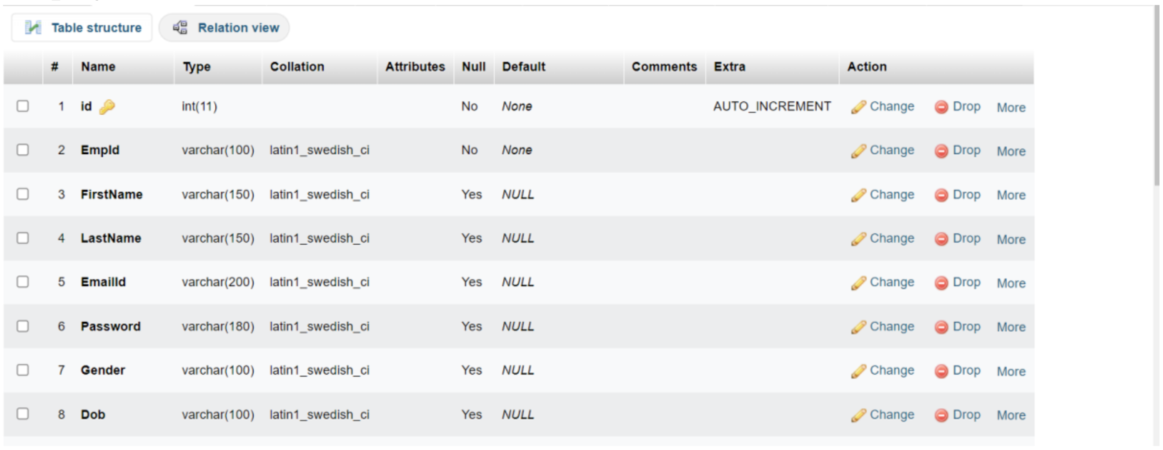
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change	Drop	More
☐	2	UserName	varchar(100)	latin1_swedish_ci		No	None			Change	Drop	More
☐	3	Password	varchar(100)	latin1_swedish_ci		No	None			Change	Drop	More
☐	4	updationDate	timestamp			No	0000-00-00 00:00:00		ON UPDATE CURRENT_TIMESTAMP()	Change	Drop	More

- **Department**



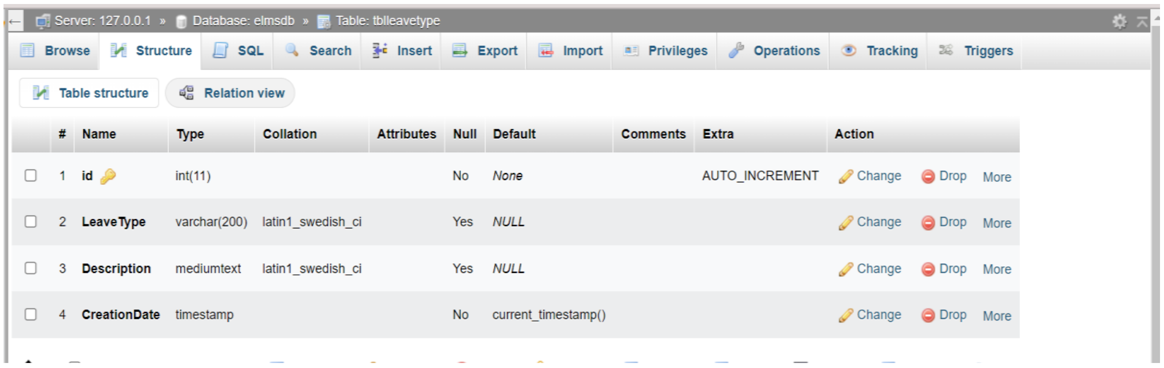
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change	Drop	More
☐	2	DepartmentName	varchar(150)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	3	DepartmentShortName	varchar(100)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	4	DepartmentCode	varchar(50)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	5	CreationDate	timestamp			Yes	current_timestamp()			Change	Drop	More

- **Employee**



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change	Drop	More
☐	2	EmpId	varchar(100)	latin1_swedish_ci		No	None			Change	Drop	More
☐	3	FirstName	varchar(150)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	4	LastName	varchar(150)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	5	EmailId	varchar(200)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	6	Password	varchar(180)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	7	Gender	varchar(100)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	8	Dob	varchar(100)	latin1_swedish_ci		Yes	NULL			Change	Drop	More

- **Leavetype**



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change	Drop	More
☐	2	LeaveType	varchar(200)	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	3	Description	mediumtext	latin1_swedish_ci		Yes	NULL			Change	Drop	More
☐	4	CreationDate	timestamp			No	current_timestamp()			Change	Drop	More

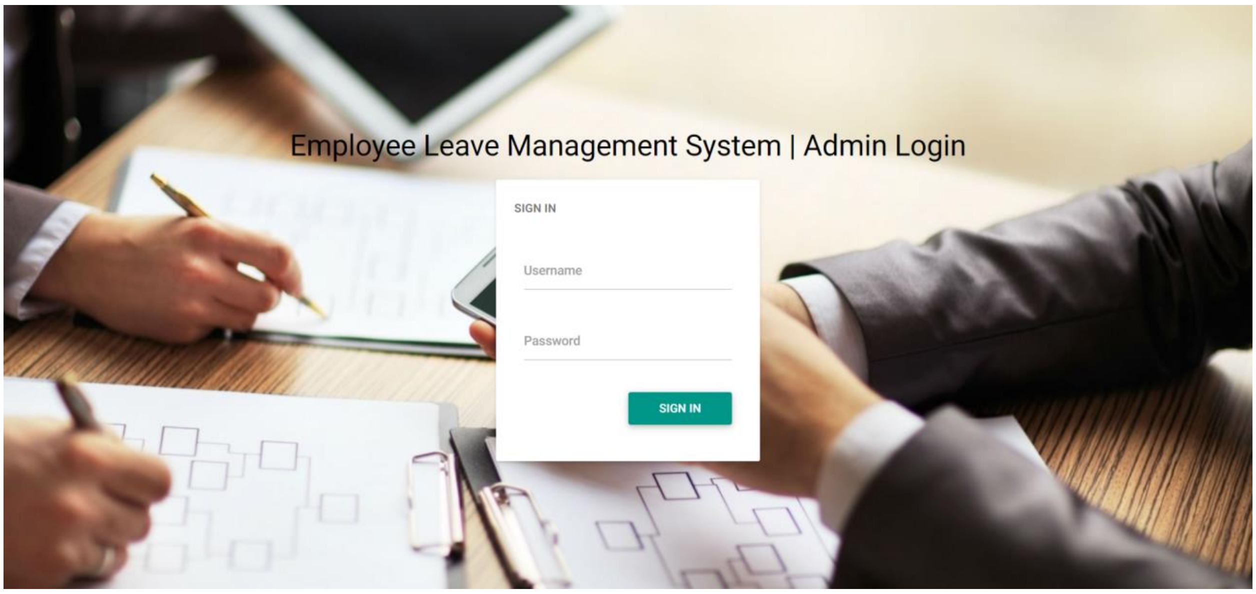
Chapter 5:

Screen layout And Testing

5.1 Screen Layout

5.1.1 Admin Side

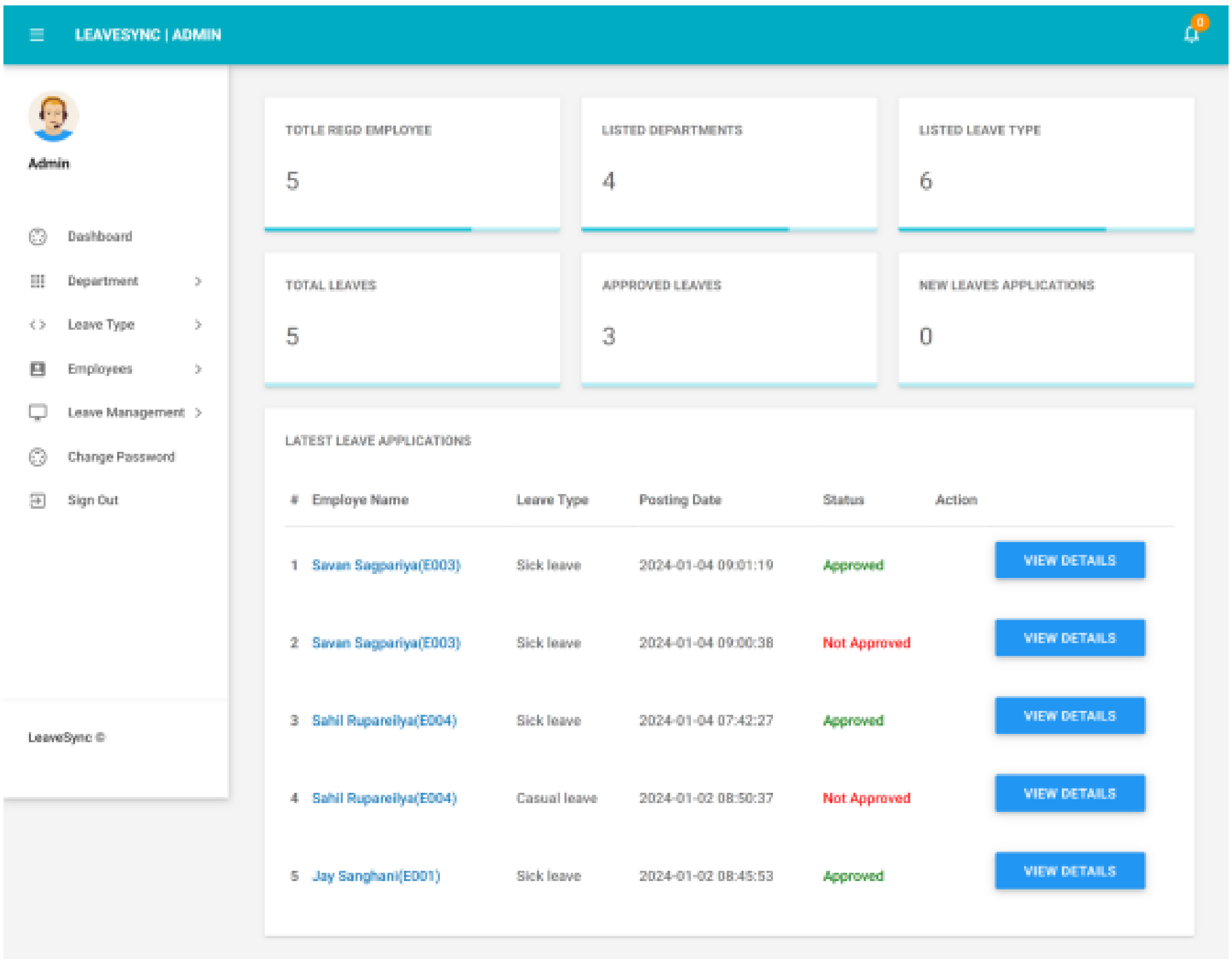
- **Before Login**



```
<?php
session_start();
include('includes/config.php');
if(isset($_POST['signin']))
{
$uname=$_POST['username'];
$password=md5($_POST['password']);
$sql ="SELECT UserName,Password FROM admin WHERE UserName=:uname and
Password=:password";
$query= $dbh -> prepare($sql);
$query-> bindParam(':uname', $uname, PDO::PARAM_STR);
$query-> bindParam(':password', $password, PDO::PARAM_STR);
$query-> execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
if($query->rowCount() > 0){
$_SESSION['alogin']=$_POST['username'];
echo "<script type='text/javascript'> document.location = 'dashboard.php'; </script>";
} else{
  echo "<script>alert('Invalid Details');</script>";} }
?>
```

AfterLogin

- **Dashboard.php**



TOTLE REGD EMPLOYEE
5

LISTED DEPARTMENTS
4

LISTED LEAVE TYPE
6

TOTAL LEAVES
5

APPROVED LEAVES
3

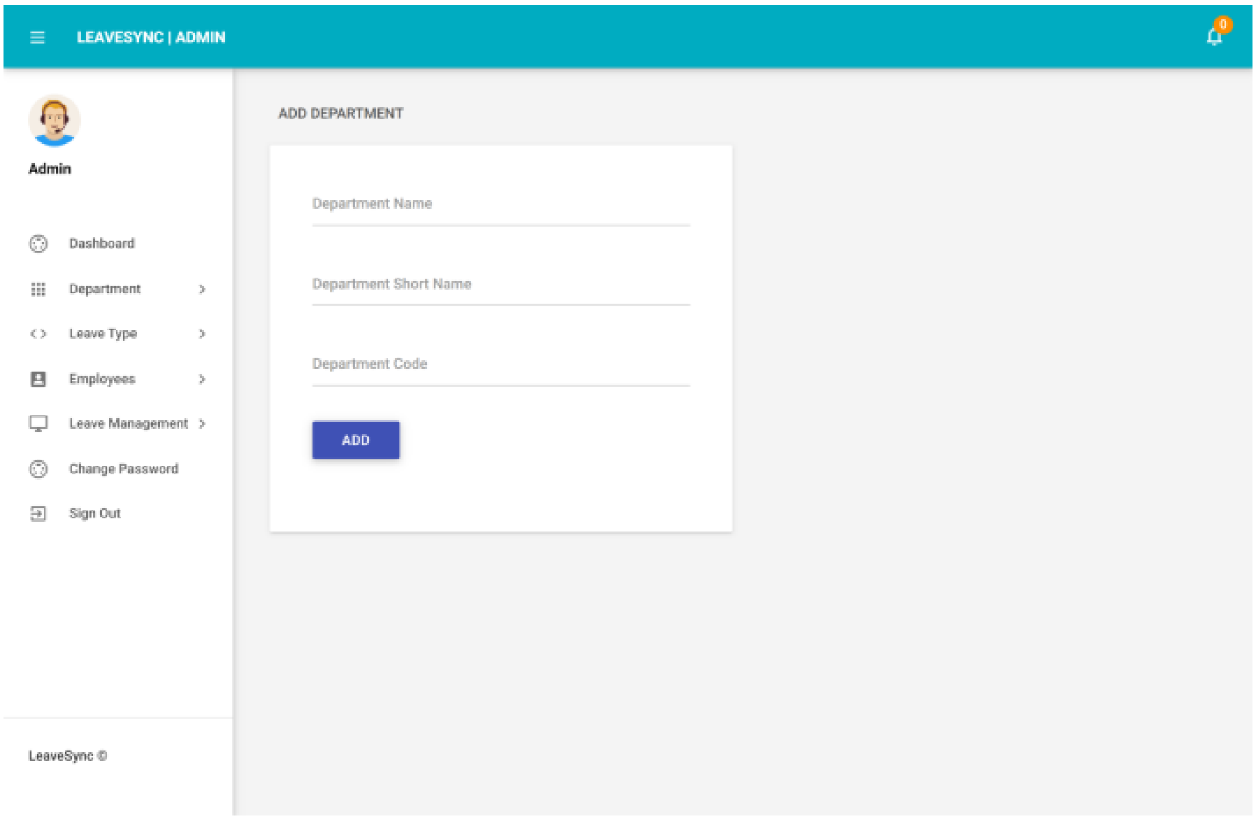
NEW LEAVES APPLICATIONS
0

LATEST LEAVE APPLICATIONS

#	Employe Name	Leave Type	Posting Date	Status	Action
1	Savan Sagpariya(E003)	Sick leave	2024-01-04 09:01:19	Approved	VIEW DETAILS
2	Savan Sagpariya(E003)	Sick leave	2024-01-04 09:00:38	Not Approved	VIEW DETAILS
3	Sahil Rupareilya(E004)	Sick leave	2024-01-04 07:42:27	Approved	VIEW DETAILS
4	Sahil Rupareilya(E004)	Casual leave	2024-01-02 08:50:37	Not Approved	VIEW DETAILS
5	Jay Sanghani(E001)	Sick leave	2024-01-02 08:45:53	Approved	VIEW DETAILS

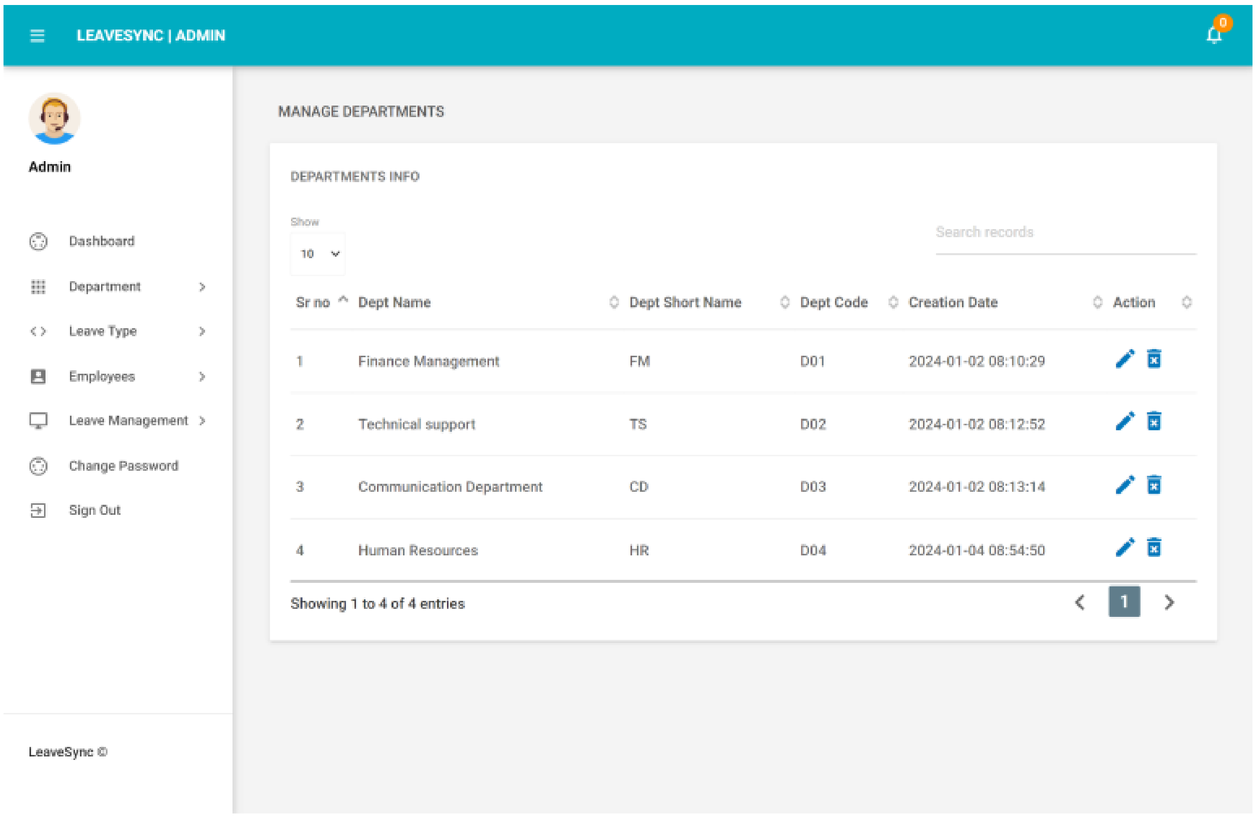
```
 <?php include('includes/header.php');?
      <?php include('includes
<?php
$sql = "SELECT id from tblemployees";
$query = $dbh -> prepare($sql);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$empcount=$query->rowCount();
?>
<?php
$sql = "SELECT id from  tblleaves where Status=0";
$query = $dbh -> prepare($sql);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$approvedleaves=$query->rowCount();
?>
```

- **Adddepartment.php**



```
<?php
session_start();
error_reporting(0);
include('includes/config.php');
if(strlen($_SESSION['alogin'])==0){
header('location:index.php');}
else{
if(isset($_POST['add'])){
$deptname=$_POST['departmentname'];
$deptshortname=$_POST['departmentshortname'];
$deptcode=$_POST['deptcode'];
$sql="INSERT INTO
tbldepartments(DepartmentName,DepartmentCode,DepartmentShortName)
VALUES(:deptname,:deptcode,:deptshortname)";
$query = $dbh->prepare($sql);
$query->bindParam(':deptname',$deptname,PDO::PARAM_STR);
$query->bindParam(':deptcode',$deptcode,PDO::PARAM_STR);
$query->bindParam(':deptshortname',$deptshortname,PDO::PARAM_STR);
$query->execute();
$lastInsertId = $dbh->lastInsertId();
if($lastInsertId){
$msg="Department Created Successfully";}
else {
$error="Something went wrong. Please try again";} } ?>
```

- **Managedepartment.php**



MANAGE DEPARTMENTS

DEPARTMENTS INFO

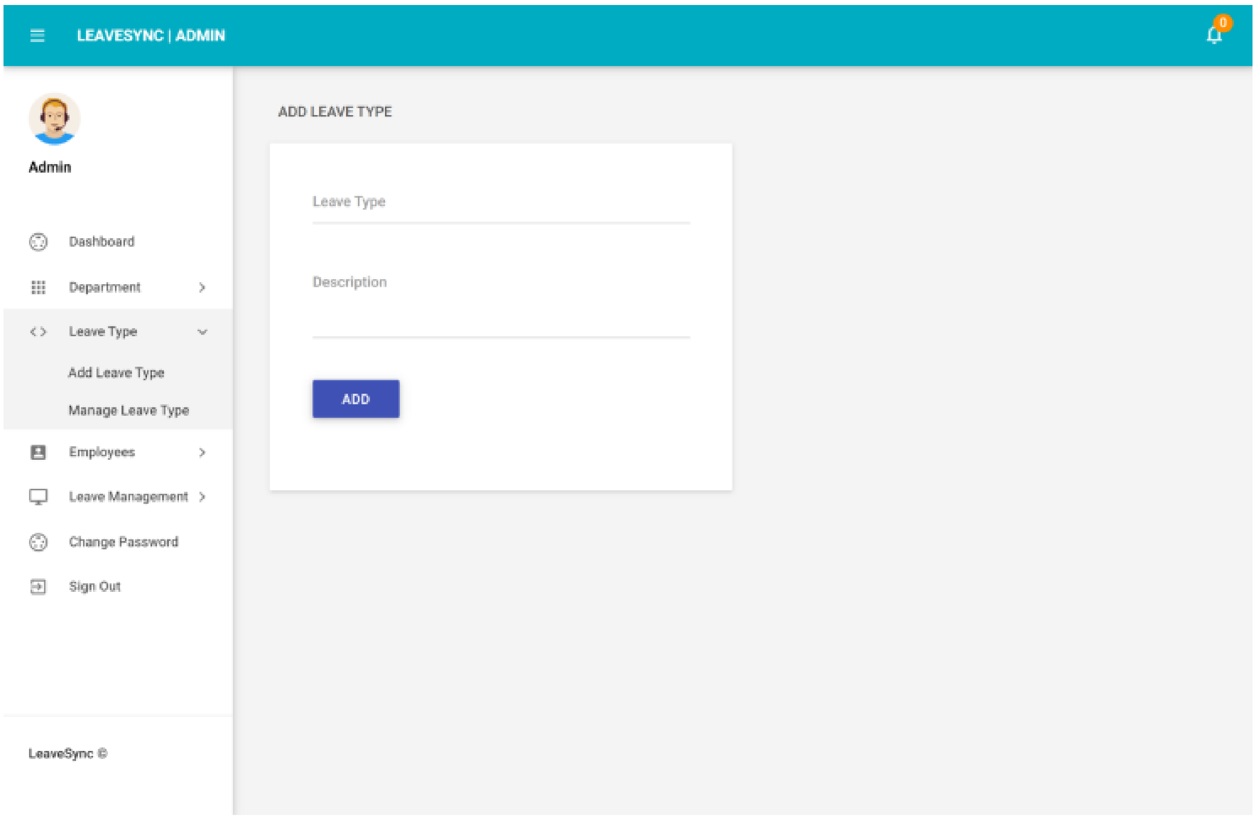
Show 10 | Search records

Sr no	Dept Name	Dept Short Name	Dept Code	Creation Date	Action
1	Finance Management	FM	D01	2024-01-02 08:10:29	
2	Technical support	TS	D02	2024-01-02 08:12:52	
3	Communication Department	CD	D03	2024-01-02 08:13:14	
4	Human Resources	HR	D04	2024-01-04 08:54:50	

Showing 1 to 4 of 4 entries

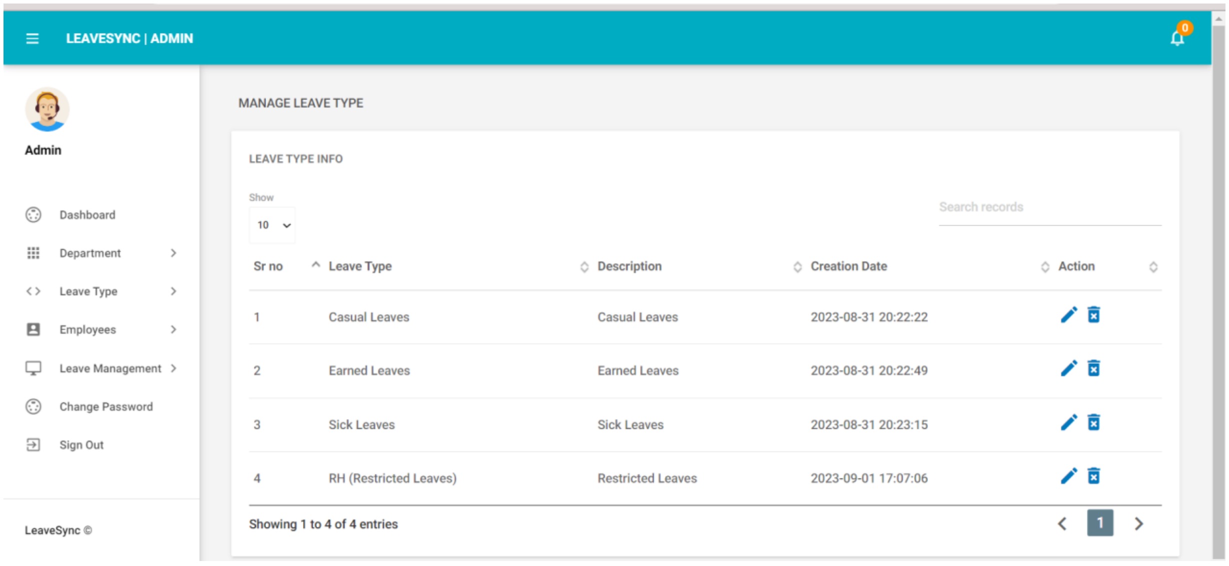
```
<?php $sql = "SELECT * from tbldepartments";
$query = $dbh -> prepare($sql);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$cnt=1;
if($query->rowCount() > 0)
{
foreach($results as $result)
{               ?>   <tr>
  <td> <?php echo htmlentities($cnt);?></td>
  <td><?php echo htmlentities($result->DepartmentName);?></td>
  <td><?php echo htmlentities($result->DepartmentShortName);?></td>
  <td><?php echo htmlentities($result->DepartmentCode);?></td>
  <td><?php echo htmlentities($result->CreationDate);?></td>
<td><a href="editdepartment.php?deptid=<?php echo htmlentities($result->id);?>"><i
class="material-icons">mode_edit</i></a><a href="managedepartments.php?del=<?php
echo htmlentities($result->id);?>" onclick="return confirm('Do you want to delete');"> <i
class="material-icons">delete_forever</i></a></td>  </tr>
  <?php $cnt++;} }?>
```

Addleavetype.php



```
if(isset($_POST['add']))
{
$leavetype=$_POST['leavetype'];
$description=$_POST['description'];
$sql="INSERT INTO tblleavetype(LeaveType,Description)
VALUES(:leavetype,:description)";
$query = $dbh->prepare($sql);
$query->bindParam(':leavetype',$leavetype,PDO::PARAM_STR);
$query->bindParam(':description',$description,PDO::PARAM_STR);
$query->execute();
$lastInsertId = $dbh->lastInsertId();
if($lastInsertId)
{
$msg="Leave type added Successfully";
}
else
{
$error="Something went wrong. Please try again";
}
}?>
```

- **Manageleavetype.php**



MANAGE LEAVE TYPE

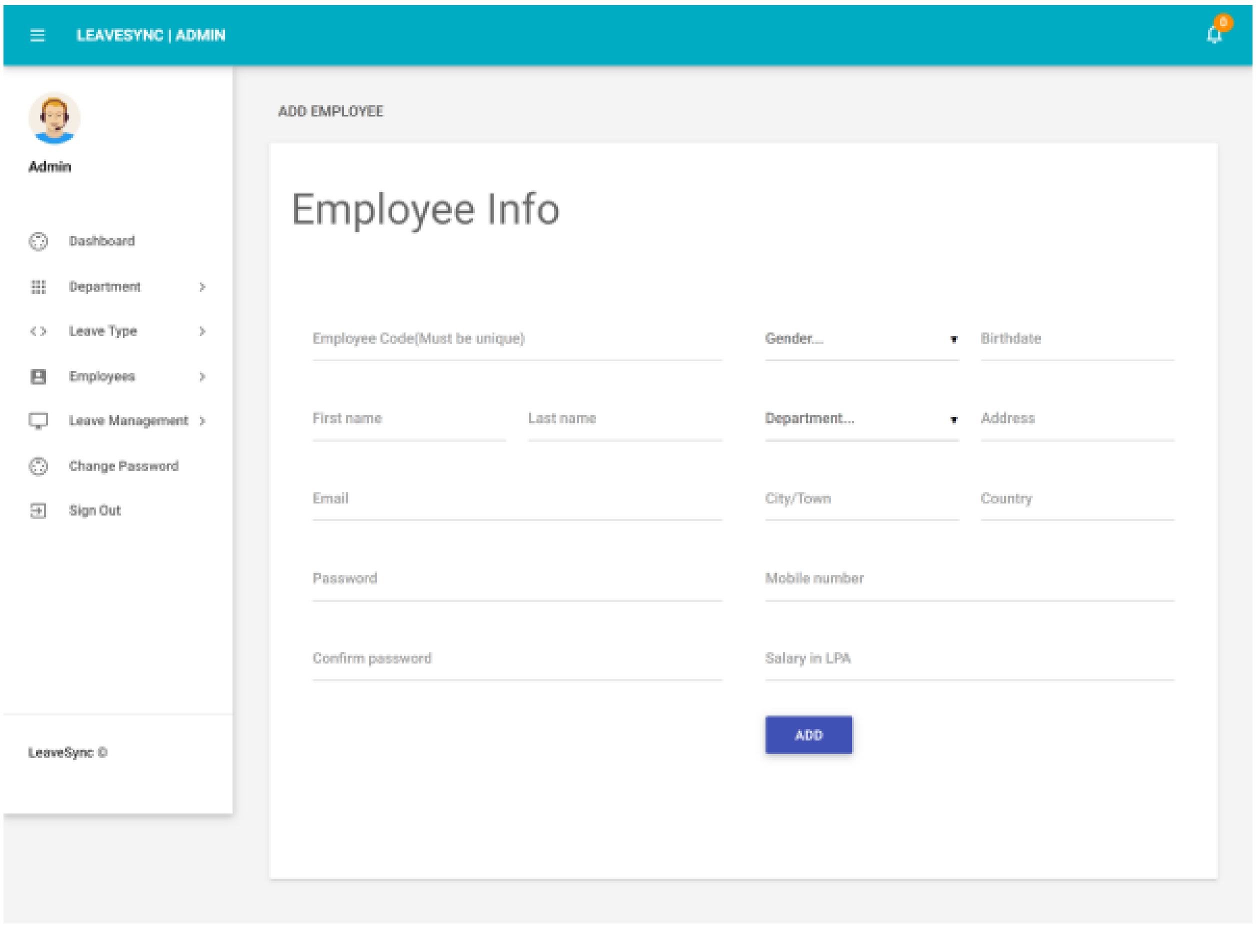
LEAVE TYPE INFO

Sr no	Leave Type	Description	Creation Date	Action
1	Casual Leaves	Casual Leaves	2023-08-31 20:22:22	
2	Earned Leaves	Earned Leaves	2023-08-31 20:22:49	
3	Sick Leaves	Sick Leaves	2023-08-31 20:23:15	
4	RH (Restricted Leaves)	Restricted Leaves	2023-09-01 17:07:06	

Showing 1 to 4 of 4 entries

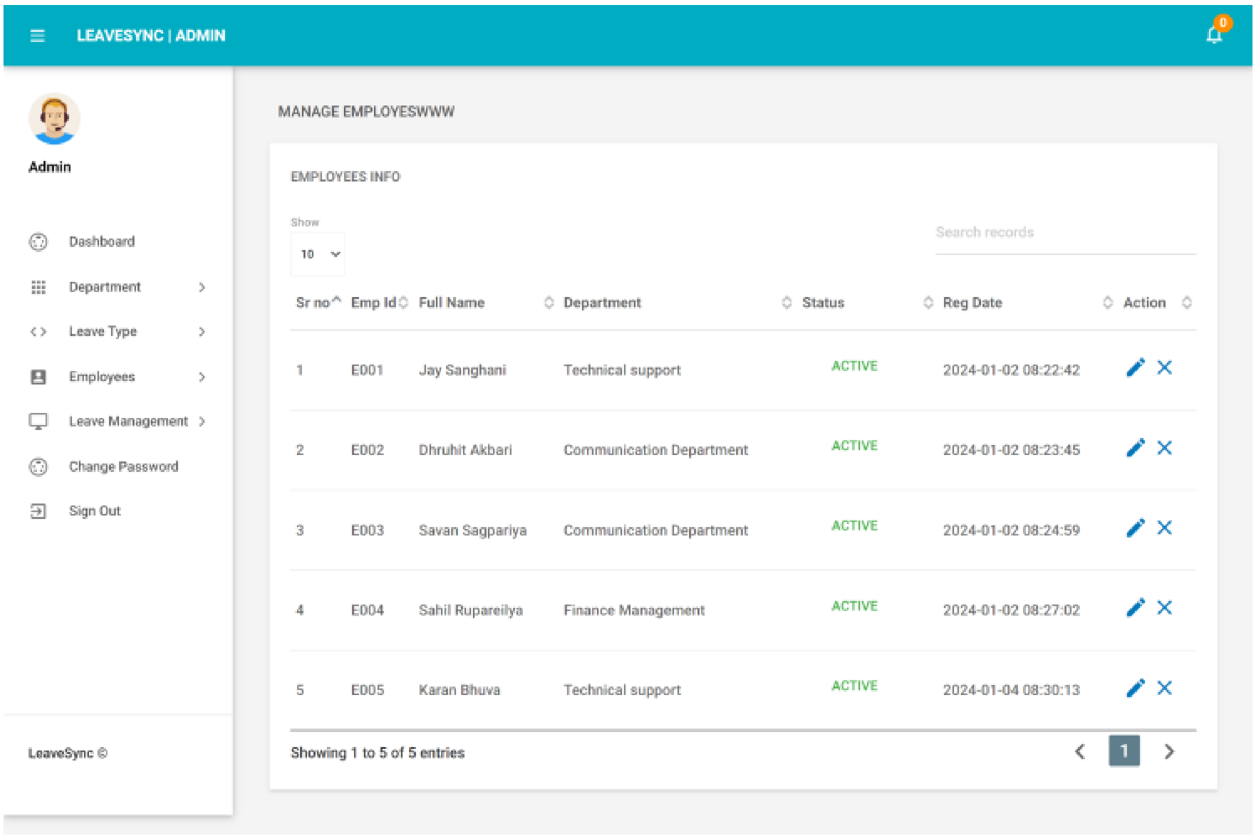
```
if(isset($_GET['del'])){
$id=$_GET['del'];
$sql = "delete from  tblleavetype  WHERE id=:id";
$query = $dbh->prepare($sql);
$query -> bindParam(':id',$id, PDO::PARAM_STR);
$query -> execute();
$msg="Leave type record deleted";}
<?php $sql = "SELECT * from tblleavetype";
$query = $dbh -> prepare($sql);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$cnt=1;
if($query->rowCount() > 0){
foreach($results as $result){
?> <tr>
<td> <?php echo htmlentities($cnt);?></td>
<td><?php echo htmlentities($result->LeaveType);?></td>
<td><?php echo htmlentities($result->Description);?></td>
<td><?php echo htmlentities($result->CreationDate);?></td>
<td><a href="editleavetype.php?lid=<?php echo htmlentities($result->id);?>"><i
class="material-icons">mode_edit</i></a>
<a href="manageleavetype.php?del=<?php echo htmlentities($result->id);?>"
onclick="return confirm('Do you want to delete');"> <i class="material-
icons">delete_forever</i></a> </td>  </tr>
 <?php $cnt++;} }?>
```

- **Addemployee.php**



```
if (isset($_POST['add'])) {
        $empid = $_POST['empcode'];
        $fname = $_POST['firstName'];
        $email = $_POST['email'];
        $password = md5($_POST['password']);
        $gender = $_POST['gender'];
        $salary = $_POST['salary'];
        $status = 1;
 $sql = "INSERT INTO
tblemployees(EmpId,FirstName,LastName,EmailId,Password,Gender,Dob,Department,A
ddress,City,Country,Phonenumber,salary,Status)
VALUES(:empid,:fname,:lname,:email,:password,:gender,:dob,:department,:address,:city
,:country,:mobileno,:salary,:status)";
        $query = $dbh->prepare($sql);
        $query->bindParam(':empid', $empid, PDO::PARAM_STR);
        $query->bindParam(':fname', $fname, PDO::PARAM_STR);
        $query->bindParam(':lname', $lname, PDO::PARAM_STR);
        $query->bindParam(':email', $email, PDO::PARAM_STR);
        $query->bindParam(':password', $password, PDO::PARAM_STR);
        $query->execute();
        $lastInsertId = $dbh->lastInsertId();
```

- **Manageemployee.php**



MANAGE EMPLOYESWWW

EMPLOYEES INFO

Show 10

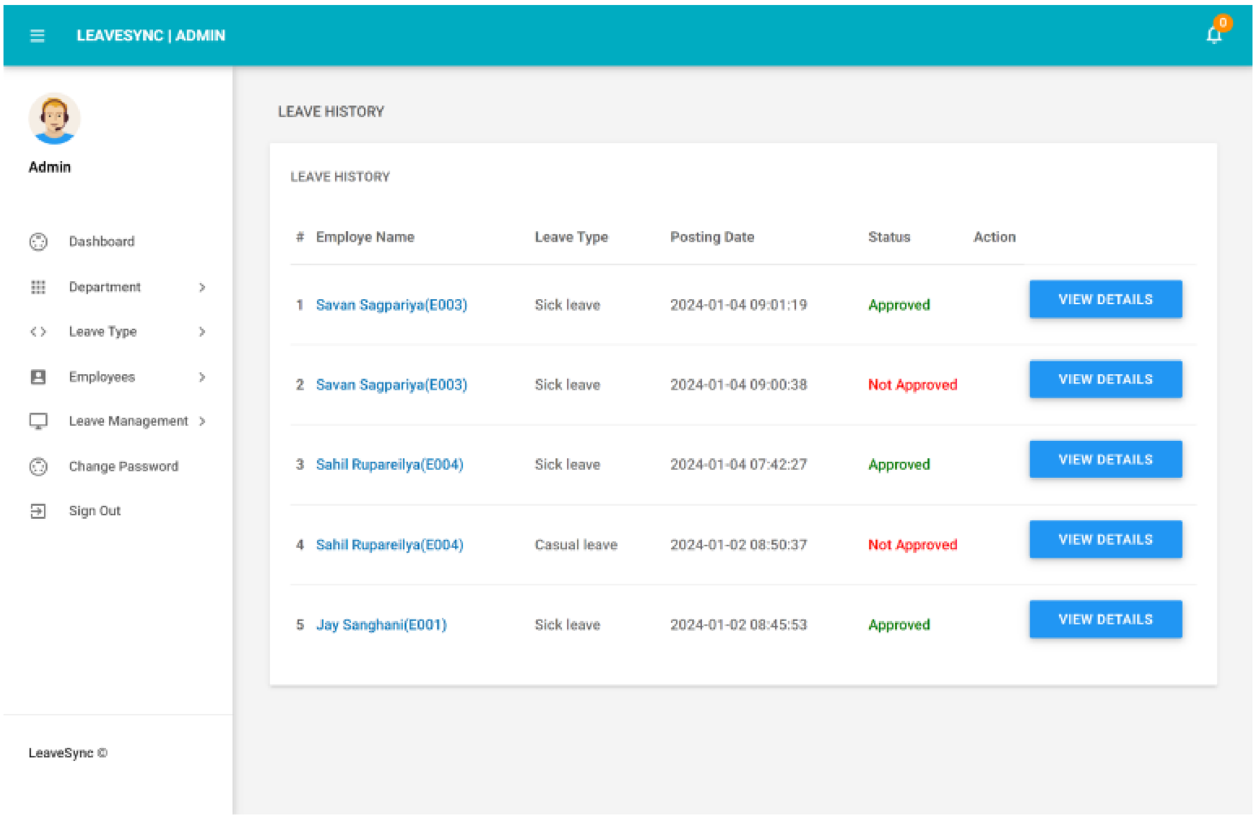
Search records

Sr no	Emp Id	Full Name	Department	Status	Reg Date	Action
1	E001	Jay Sanghani	Technical support	ACTIVE	2024-01-02 08:22:42	
2	E002	Dhruhit Akbari	Communication Department	ACTIVE	2024-01-02 08:23:45	
3	E003	Savan Sagpariya	Communication Department	ACTIVE	2024-01-02 08:24:59	
4	E004	Sahil Rupareilya	Finance Management	ACTIVE	2024-01-02 08:27:02	
5	E005	Karan Bhuva	Technical support	ACTIVE	2024-01-04 08:30:13	

Showing 1 to 5 of 5 entries

```
<?php $sql = "SELECT EmpId,FirstName,LastName,Department,Status,RegDate,id
from  tblemployees";
$query = $dbh -> prepare($sql);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$cnt=1;
if($query->rowCount() > 0){
foreach($results as $result){       ?>
  <tr><td> <?php echo htmlentities($cnt);?></td>
<td><?php echo htmlentities($result->EmpId);?></td>
 <td><?php echo htmlentities($result->FirstName);?>&nbsp;<?php echo
htmlentities($result->LastName);?></td
<td><?php echo htmlentities($result->Department);?></td>
<td><?php $stats=$result->Status;
if($stats){?>
<a class="waves-effect waves-green btn-flat m-b-xs">Active</a>
<?php } else { ?>
 <a class="waves-effect waves-red btn-flat m-b-xs">Inactive</a>
<?php } ?>
```

- **Leaves.php**



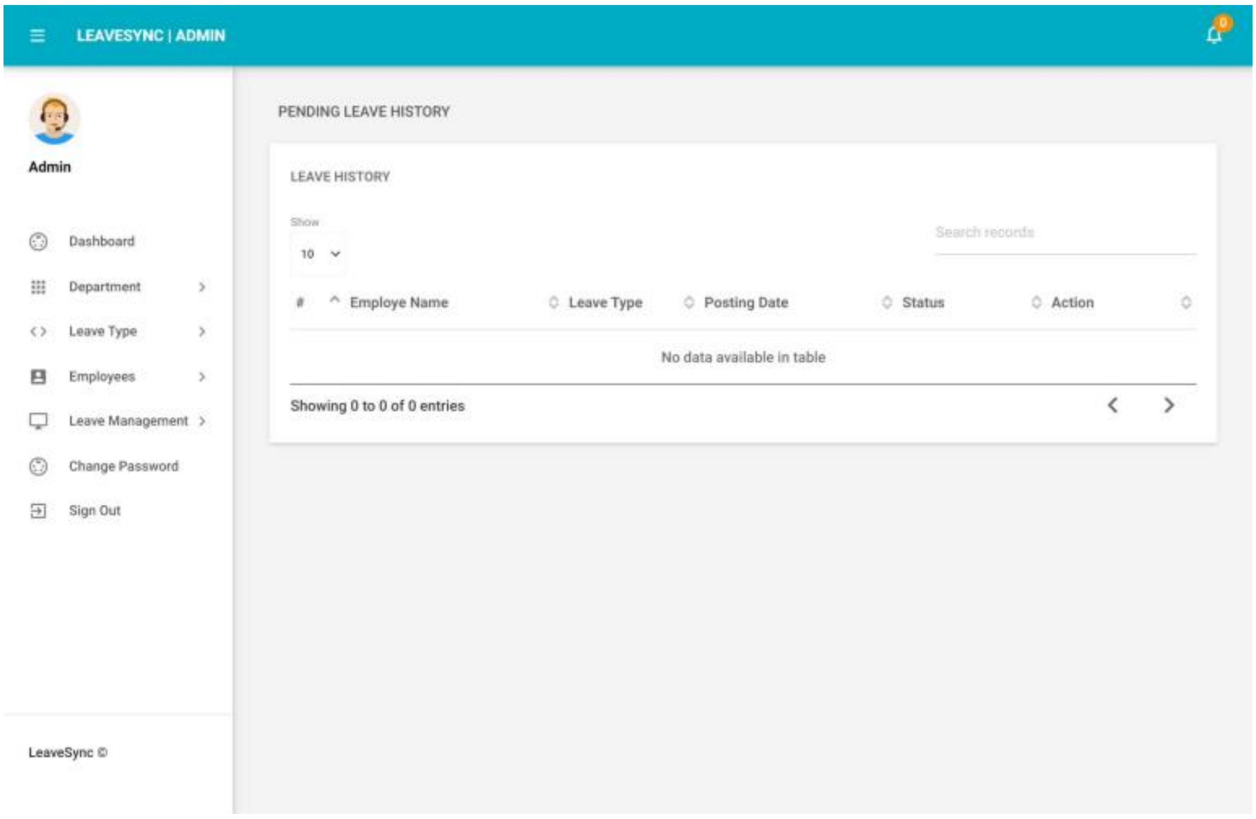
LEAVE HISTORY

LEAVE HISTORY

#	Employe Name	Leave Type	Posting Date	Status	Action
1	Savan Sagpariya(E003)	Sick leave	2024-01-04 09:01:19	Approved	VIEW DETAILS
2	Savan Sagpariya(E003)	Sick leave	2024-01-04 09:00:38	Not Approved	VIEW DETAILS
3	Sahil Rupareilya(E004)	Sick leave	2024-01-04 07:42:27	Approved	VIEW DETAILS
4	Sahil Rupareilya(E004)	Casual leave	2024-01-02 08:50:37	Not Approved	VIEW DETAILS
5	Jay Sanghani(E001)	Sick leave	2024-01-02 08:45:53	Approved	VIEW DETAILS

```
<?php $sql = "SELECT tblleaves.id as
lid,tblemployees.FirstName,tblemployees.LastName,tblemployees.EmpId,tblemployees.i
d,tblleaves.LeaveType,tblleaves.PostingDate,tblleaves.Status from tblleaves join
tblemployees on tblleaves.empid=tblemployees.id order by lid desc";
$query = $dbh -> prepare($sql);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$cnt=1;
if($query->rowCount() > 0){
foreach($results as $result){     >?
<tr><td> <b><?php echo htmlentities($cnt);?></b></td>
<td><a href="editemployee.php?empid=<?php echo htmlentities($result->id);?>"
target="_blank"><?php echo htmlentities($result->FirstName." ".$result-
>LastName);?>(<?php echo htmlentities($result->EmpId);?>)</a></td>
<td><?php echo htmlentities($result->LeaveType);?></td>
<td><?php echo htmlentities($result->PostingDate);?></td>
<td><?php $stats=$result->Status;if($stats==1){ ?>
<span style="color: green">Approved</span>
<?php } if($stats==2)  { ?>
<span style="color: red">Not Approved</span>
 <?php } if($stats==0)  { ?>
 <span style="color: blue">waiting for approval</span> <?php } ?>
```

Pending-leave.php



PENDING LEAVE HISTORY

LEAVE HISTORY

Show 10

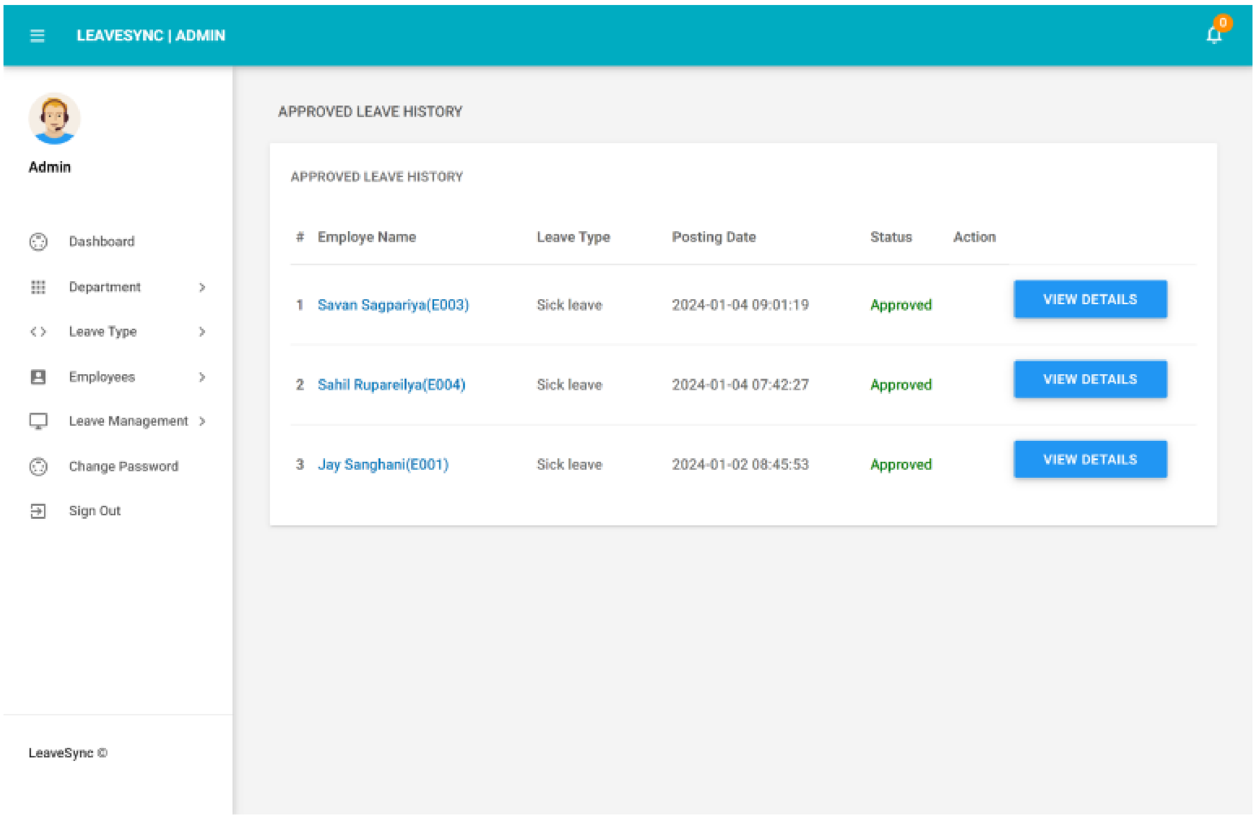
Search records

#	Employe Name	Leave Type	Posting Date	Status	Action
No data available in table					

Showing 0 to 0 of 0 entries

```
<?php
$status=0;
$sql = "SELECT tblleaves.id as
lid,tblemployees.FirstName,tblemployees.LastName,tblemployees.EmpId,tblemployees.i
d,tblleaves.LeaveType,tblleaves.PostingDate,tblleaves.Status from tblleaves join
tblemployees on tblleaves.empid=tblemployees.id where tblleaves.Status=:status order by
lid desc";
$query = $dbh -> prepare($sql);
$query->bindParam(':status',$status,PDO::PARAM_STR);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$cnt=1;
if($query->rowCount() > 0){foreach($results as $result){      ?>
<tr><td> <b><?php echo htmlentities($cnt);?></b></td>
  <td><a href="editemployee.php?empid=<?php echo htmlentities($result->id);?>"
target="_blank"><?php echo htmlentities($result->FirstName." ".$result-
>LastName);?>(<?php echo htmlentities($result->EmpId);?>)</a></td>
<td><?php echo htmlentities($result->LeaveType);?></td>
<td><?php echo htmlentities($result->PostingDate);?></td>
<td><?php $stats=$result->Status;
if($stats==1){?><span style="color: green">Approved</span>
<?php } if($stats==2)  { ?><span style="color: red">Not Approved</span>
<?php } if($stats==0)  { ?>
 <span style="color: blue">waiting for approval</span> <?php } ?
```

- **Approvedleave.php**



LEAVESYNC | ADMIN

Admin

- Dashboard
- Department
- Leave Type
- Employees
- Leave Management
- Change Password
- Sign Out

LeaveSync ©

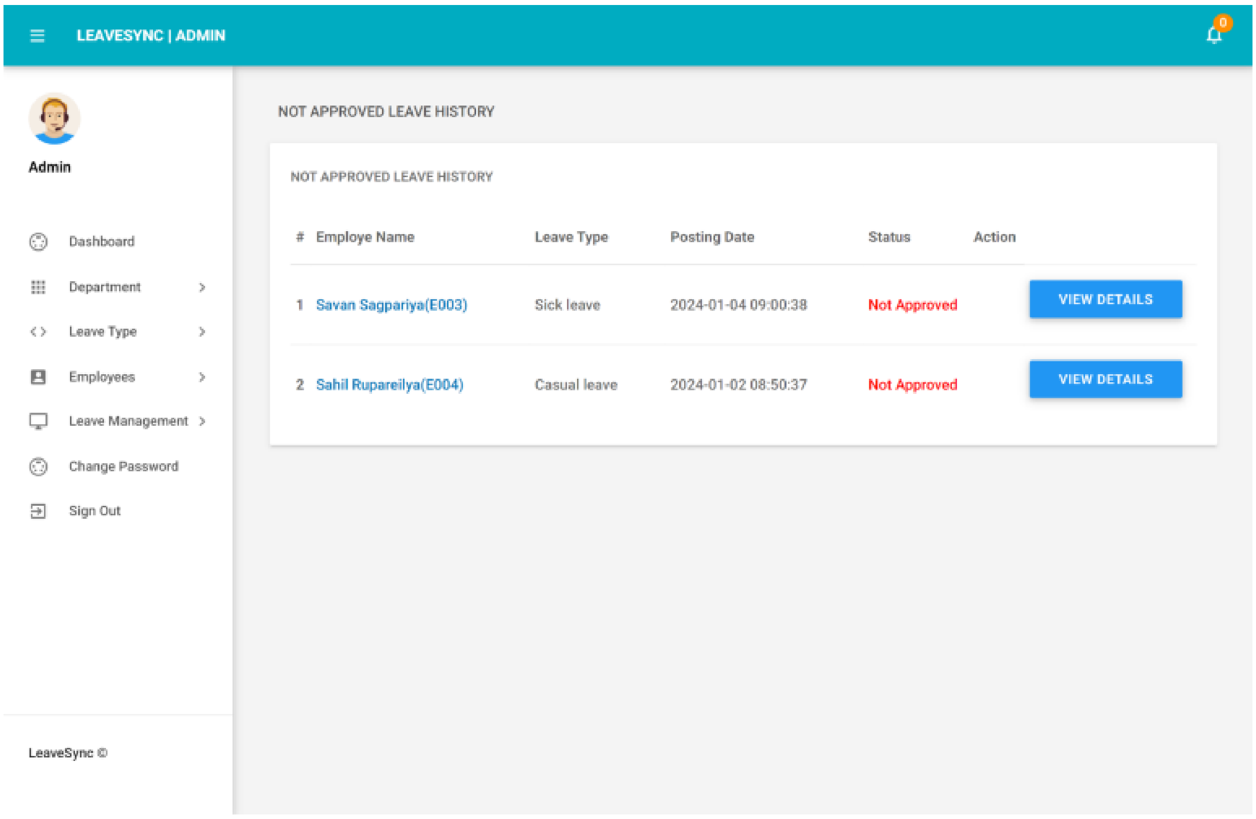
APPROVED LEAVE HISTORY

APPROVED LEAVE HISTORY

#	Employe Name	Leave Type	Posting Date	Status	Action
1	Savan Sagpariya(E003)	Sick leave	2024-01-04 09:01:19	Approved	VIEW DETAILS
2	Sahil Rupareilya(E004)	Sick leave	2024-01-04 07:42:27	Approved	VIEW DETAILS
3	Jay Sanghani(E001)	Sick leave	2024-01-02 08:45:53	Approved	VIEW DETAILS

```
<?php
$status=1;
$sql = "SELECT tblleaves.id as
lid,tblemployees.FirstName,tblemployees.LastName,tblemployees.EmpId,tblemployees.i
d,tblleaves.LeaveType,tblleaves.PostingDate,tblleaves.Status from tblleaves join
tblemployees on tblleaves.empid=tblemployees.id where tblleaves.Status=:status order by
lid desc";
$query = $dbh -> prepare($sql);
$query->bindParam(':status',$status,PDO::PARAM_STR);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ); $cnt=1;
if($query->rowCount() > 0){
foreach($results as $result){ ?>
<tr> <td> <b><?php echo htmlentities($cnt);?></b></td>
 <td><a href="editemployee.php?empid=<?php echo htmlentities($result->id);?>"
target="_blank"><?php echo htmlentities($result->FirstName." ".$result-
>LastName);?>(<?php echo htmlentities($result->EmpId);?>)</a></td>
 <td><?php echo htmlentities($result->LeaveType);?></td>
 <td><?php $stats=$result->Status;
if($stats==1){?><span style="color: green">Approved</span>
<?php } if($stats==2)  { ?>
<span style="color: red">Not Approved</span>
<?php } if($stats==0)  { ?>
 <span style="color: blue">waiting for approval</span> <?php } ?>
```

- **Notapproved.php**



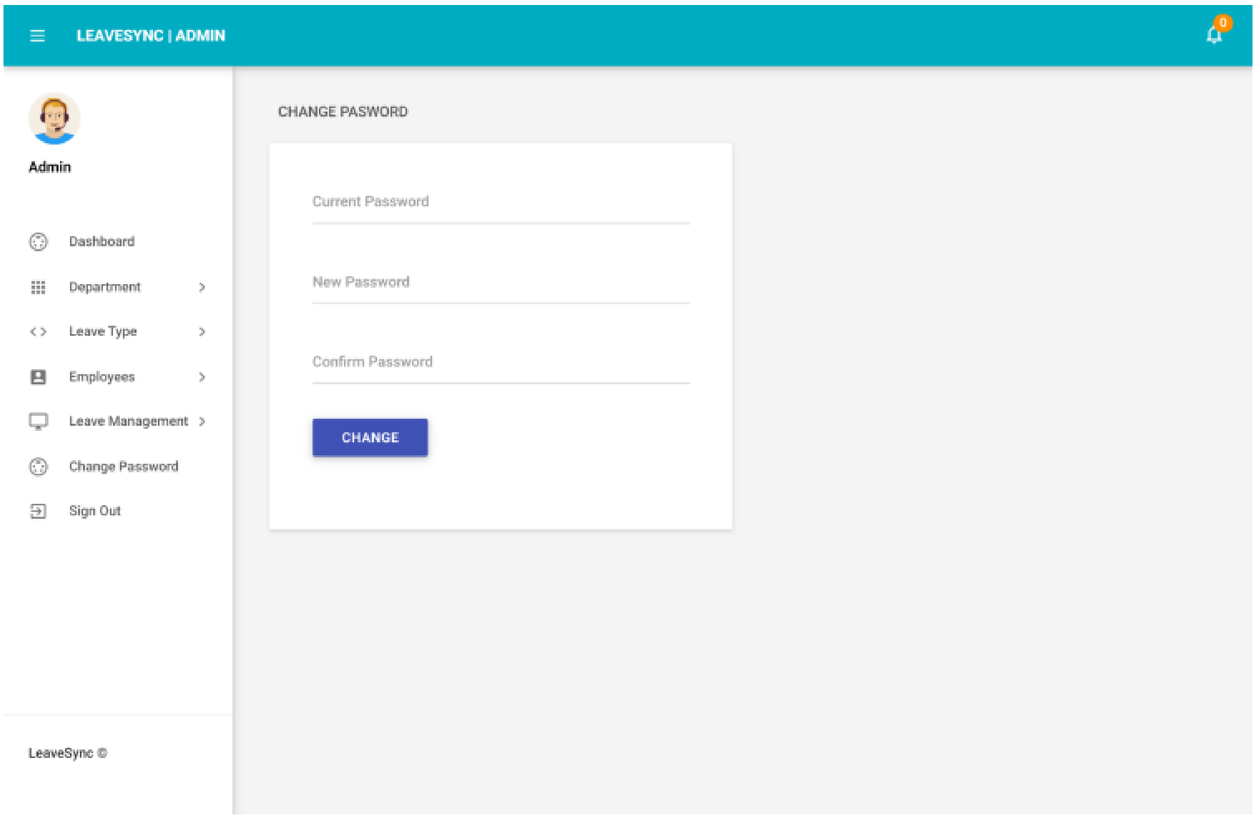
NOT APPROVED LEAVE HISTORY

NOT APPROVED LEAVE HISTORY

#	Employe Name	Leave Type	Posting Date	Status	Action
1	Savan Sagpariya(E003)	Sick leave	2024-01-04 09:00:38	Not Approved	VIEW DETAILS
2	Sahil Rupareilya(E004)	Casual leave	2024-01-02 08:50:37	Not Approved	VIEW DETAILS

```
<?php
$status=2;
$sql = "SELECT tblleaves.id as
lid,tblemployees.FirstName,tblemployees.LastName,tblemployees.EmpId,tblemployees.i
d,tblleaves.LeaveType,tblleaves.PostingDate,tblleaves.Status from tblleaves join
tblemployees on tblleaves.empid=tblemployees.id where tblleaves.Status=:status order by
lid desc";
$query = $dbh -> prepare($sql);
$query->bindParam(':status',$status,PDO::PARAM_STR);
$query->execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
$cnt=1;
if($query->rowCount() > 0){
foreach($results as $result)}?>
<tr><td> <b><?php echo htmlentities($cnt);?></b></td>
<td><a href="editemployee.php?empid=<?php echo htmlentities($result->id);?>"
target="_blank"><?php echo htmlentities($result->FirstName." ".$result-
>LastName);?>(<?php echo htmlentities($result->EmpId);?>)</a></td>
<td><?php echo htmlentities($result->LeaveType);?></td>
<td><?php $stats=$result->Status;
if($stats==1){?>
<span style="color: green">Approved</span>
<?php } if($stats==2)  { ?>
<span style="color: red">Not Approved</span>
```

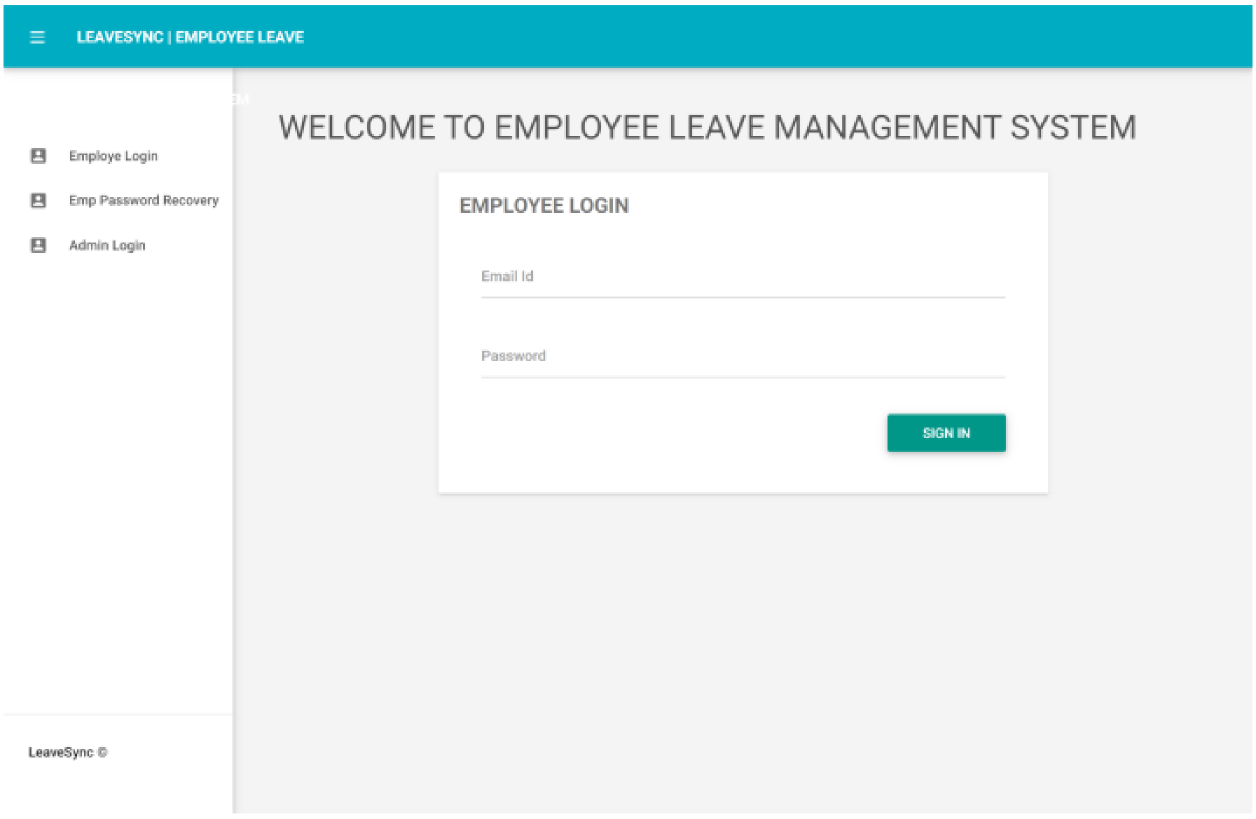
- **Changepasswoard.php**



```
if(isset($_POST['change'])){
$password=md5($_POST['password']);
$newpassword=md5($_POST['newpassword']);
$username=$_SESSION['alogin'];  $sql ="SELECT Password FROM admin WHERE
UserName=:username and Password=:password";
$query= $dbh -> prepare($sql);
$query-> bindParam(':username', $username, PDO::PARAM_STR);
$query-> bindParam(':password', $password, PDO::PARAM_STR);
$query-> execute();
$results = $query -> fetchAll(PDO::FETCH_OBJ);
if($query -> rowCount() > 0){
$con="update admin set Password=:newpassword where UserName=:username";
$chngpwd1 = $dbh->prepare($con);
$chngpwd1-> bindParam(':username', $username, PDO::PARAM_STR);
$chngpwd1-> bindParam(':newpassword', $newpassword, PDO::PARAM_STR);
$chngpwd1->execute();
$msg="Your Password succesfully changed";}else{
$error="Your current password is wrong";    }}?>
```

Employee Side

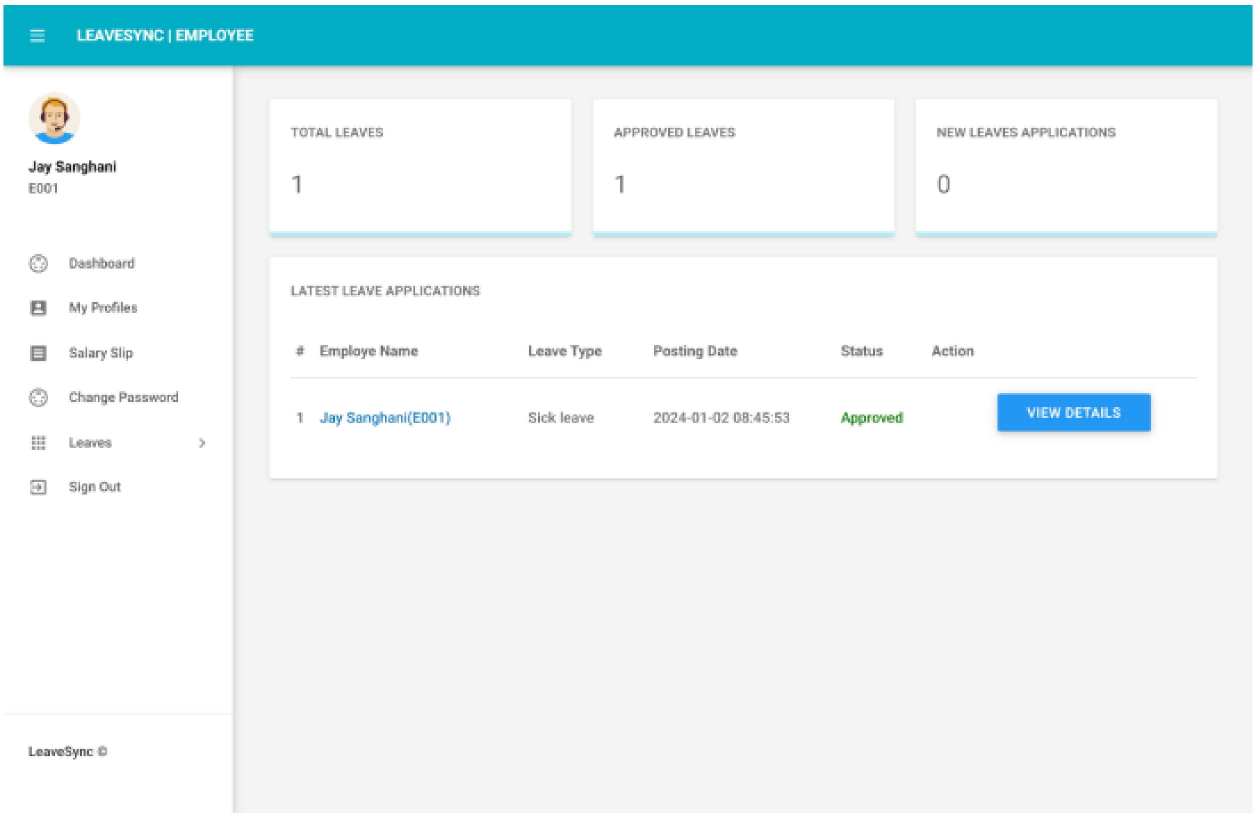
- **Before Login**



```
if(isset($_POST['signin'])){
$uname=$_POST['username'];
$password=md5($_POST['password']);
$sql ="SELECT EmailId,Password,Status,id FROM tblemployees WHERE
EmailId=:uname and Password=:password";
$query= $dbh -> prepare($sql);
$query-> bindParam(':uname', $uname, PDO::PARAM_STR);
$query-> bindParam(':password', $password, PDO::PARAM_STR);
$query-> execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
if($query->rowCount() > 0){
 foreach ($results as $result) {
    $status=$result->Status;
    $_SESSION['eid']=$result->id;}
if($status==0){
$msg="Your account is Inactive. Please contact admin";} else{
$_SESSION['emplogin']=$_POST['username'];
echo "<script type='text/javascript'> document.location = 'dashboard.php'; </script>";}}
else{  echo "<script>alert('Invalid Details');</script>";}
```

AfterLogin

- **Dashboard.php**



LEAVESYNC | EMPLOYEE

Jay Sanghani
E001

Dashboard
My Profiles
Salary Slip
Change Password
Leaves
Sign Out

LeaveSync ©

TOTAL LEAVES
1

APPROVED LEAVES
1

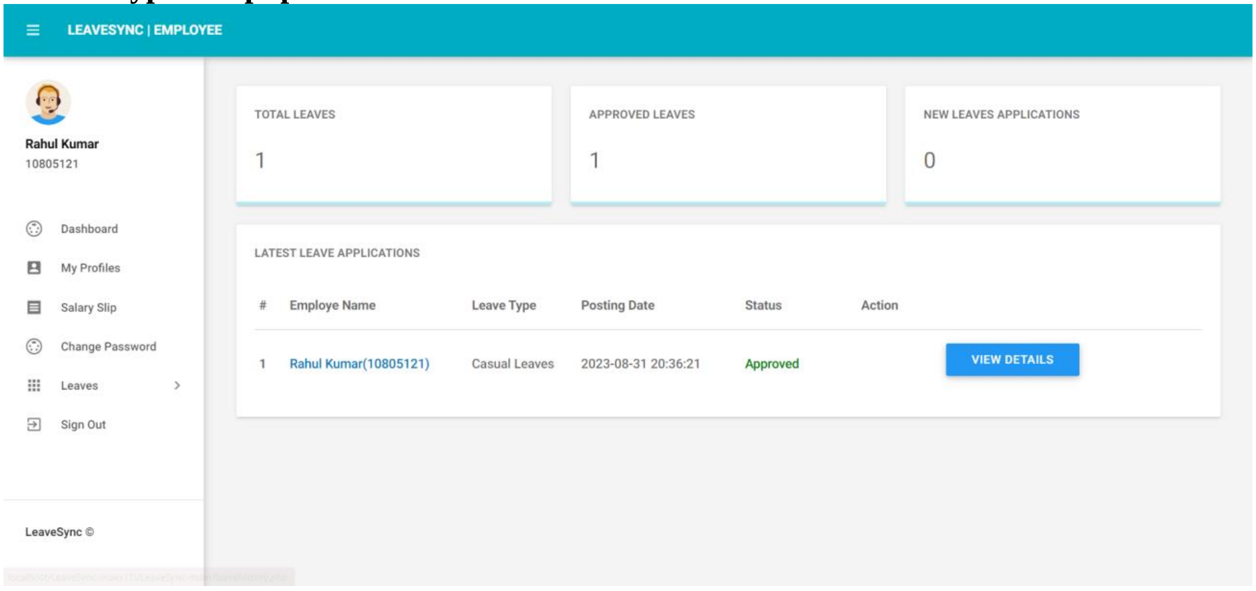
NEW LEAVES APPLICATIONS
0

LATEST LEAVE APPLICATIONS

#	Employe Name	Leave Type	Posting Date	Status	Action
1	Jay Sanghani(E001)	Sick leave	2024-01-02 08:45:53	Approved	VIEW DETAILS

```
<span class="card-title">Total Leaves</span>
<?php $eid = $_SESSION['eid'];
$sql = "SELECT id from  tblleaves where empid ='$eid'";
$query = $dbh->prepare($sql);
$query->execute();
$results = $query->fetchAll(PDO::FETCH_OBJ);
$totalleaves = $query->rowCount();?>
<span class="stats-counter"><span class="counter"><?php echo
htmlentities($totalleaves); ?></span></span>
<span class="card-title">Approved Leaves</span>
<?php $sql = "SELECT id from  tblleaves where Status=1 and empid ='$eid'";
$query = $dbh->prepare($sql);
$query->execute();
$results = $query->fetchAll(PDO::FETCH_OBJ);
$approvedleaves = $query->rowCount();?>
<span class="stats-counter"><span class="counter"><?php echo
htmlentities($approvedleaves); ?></span></span>
<span class="card-title">New Leaves Applications</span><?php
$sql = "SELECT id from  tblleaves where Status=0 and empid ='$eid'";
$query = $dbh->prepare($sql); $query->execute();
$results = $query->fetchAll(PDO::FETCH_OBJ);
$approvedleaves = $query->rowCount();?>
```

- **Myprofile.php**


LEAVESYNC | EMPLOYEE

Rahul Kumar
10805121

Dashboard
My Profiles
Salary Slip
Change Password
Leaves
Sign Out

LeaveSync ©

TOTAL LEAVES
1

APPROVED LEAVES
1

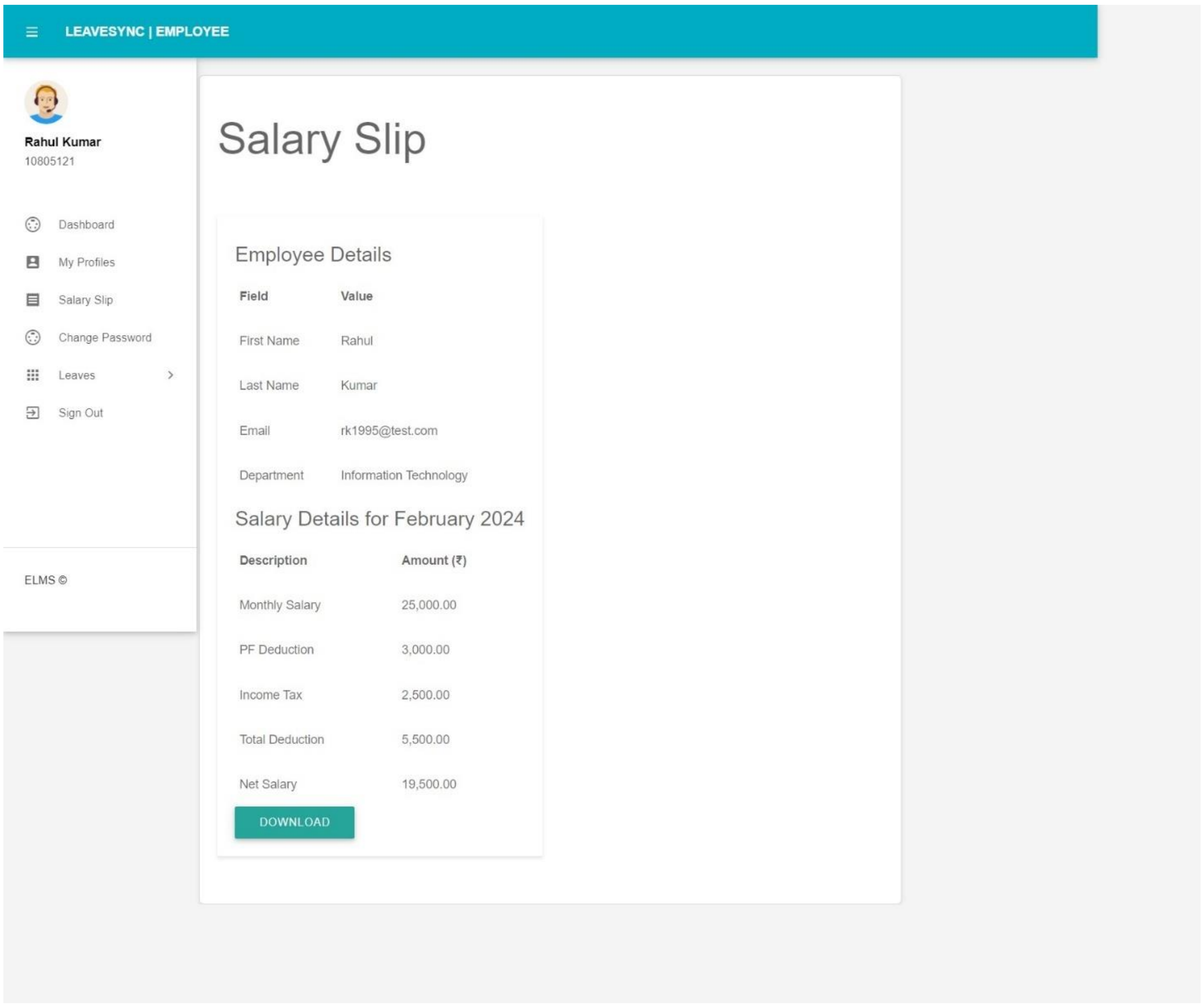
NEW LEAVES APPLICATIONS
0

LATEST LEAVE APPLICATIONS

#	Employe Name	Leave Type	Posting Date	Status	Action
1	Rahul Kumar(10805121)	Casual Leaves	2023-08-31 20:36:21	Approved	VIEW DETAILS

```
 $eid = $_SESSION['emplogin'];
   if (isset($_POST['update'])) {
       $fname = $_POST['firstName'];
       $lname = $_POST['lastName'];
       $gender = $_POST['gender'];
       $dob = $_POST['dob'];
       $department = $_POST['department'];
       $address = $_POST['address'];
       $city = $_POST['city'];
       $country = $_POST['country'];
       $mobileno = $_POST['mobileno'];
       $sql = "update tblemployees set
FirstName=:fname,LastName=:lname,Gender=:gender,Dob=:dob,Department=:departme
nt,Address=:address,City=:city,Country=:country,Phonenumber=:mobileno where
EmailId=:eid";
       $query = $dbh->prepare($sql);
       $query->bindParam(':fname', $fname, PDO::PARAM_STR);
       $query->bindParam(':lname', $lname, PDO::PARAM_STR);
       $query->bindParam(':gender', $gender, PDO::PARAM_STR);
       $query->bindParam(':dob', $dob, PDO::PARAM_STR);
       $query->bindParam(':department', $department, PDO::PARAM_STR);
       $query->bindParam(':address', $address, PDO::PARAM_STR);
       $query->bindParam(':city', $city, PDO::PARAM_STR);
       $query->bindParam(':country', $country, PDO::PARAM_STR);
       $query->bindParam(':mobileno', $mobileno, PDO::PARAM_STR);
       $query->bindParam(':eid', $eid, PDO::PARAM_STR);
       $query->execute();
       $msg = "Employee record updated Successfully";}?>
```

- **Salaryslip.php**



Salary Slip

Employee Details

Field	Value
First Name	Rahul
Last Name	Kumar
Email	rk1995@test.com
Department	Information Technology

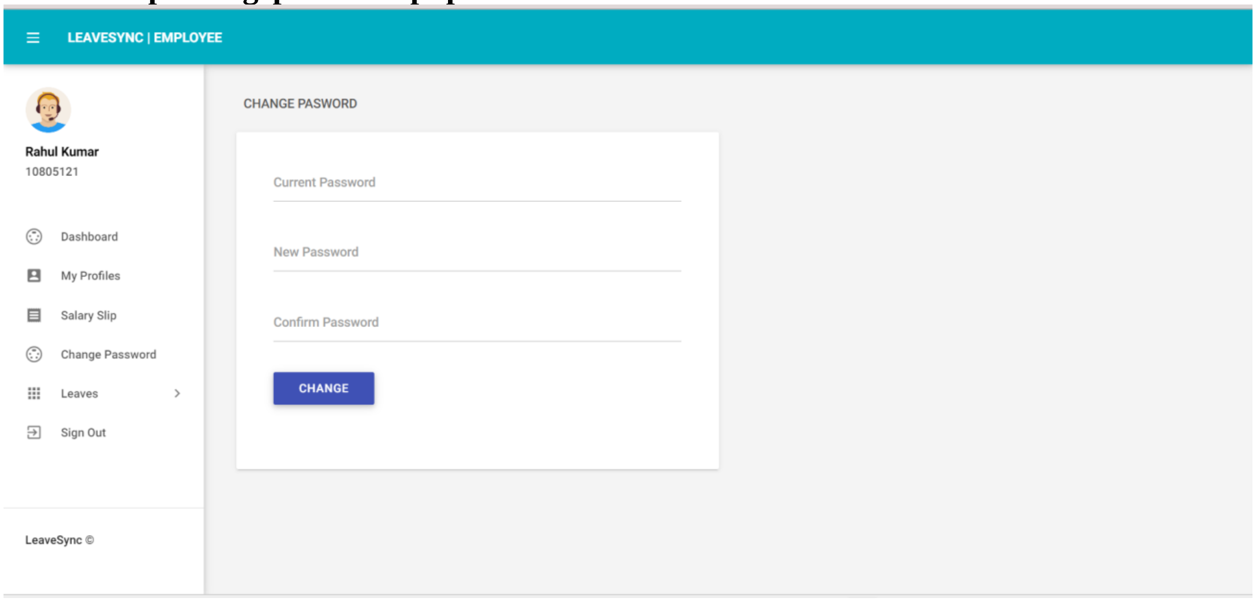
Salary Details for February 2024

Description	Amount (₹)
Monthly Salary	25,000.00
PF Deduction	3,000.00
Income Tax	2,500.00
Total Deduction	5,500.00
Net Salary	19,500.00

```
if (isset($_GET['id'])) {
$id = $_GET['id'];
$sql = "SELECT * FROM tblemployees WHERE id = :id";
$query = $dbh->prepare($sql);
$query->bindParam(':id', $id, PDO::PARAM_STR);
$query->execute();
$result = $query->fetch(PDO::FETCH_ASSOC);
  <?php  // Calculate salary, PF, and incentive
$lpa = $result['salary'];
$salary = $lpa / 12;
$pf = $salary * 0.12; // 12% of the salary
$incomeTax = $salary * 0.10; // 10% of the salary
$currentMonth = date('F');
$currentYear = date('Y');?>
 <table class="table table-bordered border border-2 border-dark">
<tbody><tr><th class="border border-2 border-dark ">Description</th>
<th class="border border-2 border-dark">Amount (₹)</th>
```

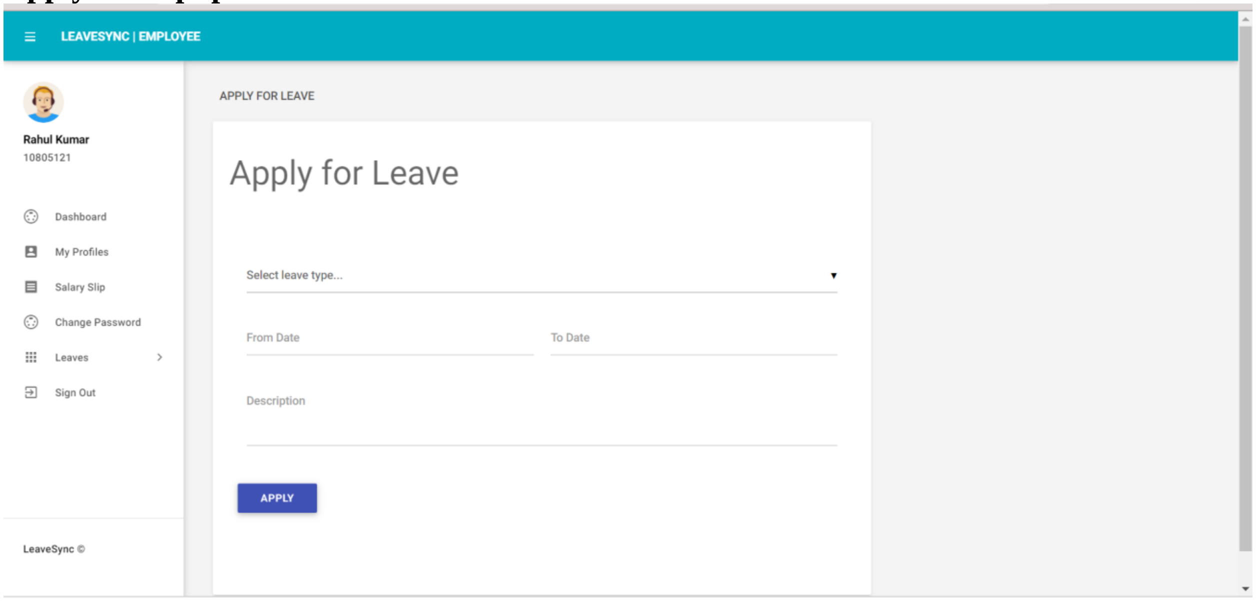
```
</tr><tr><td class="border border-2 border-dark fw-bold">Monthly Salary</td>
<td class="border border-2 border-dark"><?php echo number_format($salary, 2);
?></td></tr><tr>  <td class="border border-2 border-dark fw-bold">PF
Deduction</td><td class="border border-2 border-dark"><?php echo
number_format($pf, 2); ?></td></tr><tr>
  <td class="border border-2 border-dark fw-bold">Income Tax</td>
  <td class="border border-2 border-dark"><?php echo number_format($incomeTax, 2);
?></td></tr><tr>
<td class="border border-2 border-dark fw-bold">Total Deduction</td>
<td class="border border-2 border-dark"><?php echo number_format($pf + $incomeTax,
2); ?></td></tr><tr>
<td class="border border-2 border-dark fw-bold">Net Salary</td>
 <td class="border border-2 border-dark"><?php echo number_format($salary - $pf -
$incomeTax, 2); ?></td> </tr>  </tbody></table>
```

- **Emp-changepassword.php**



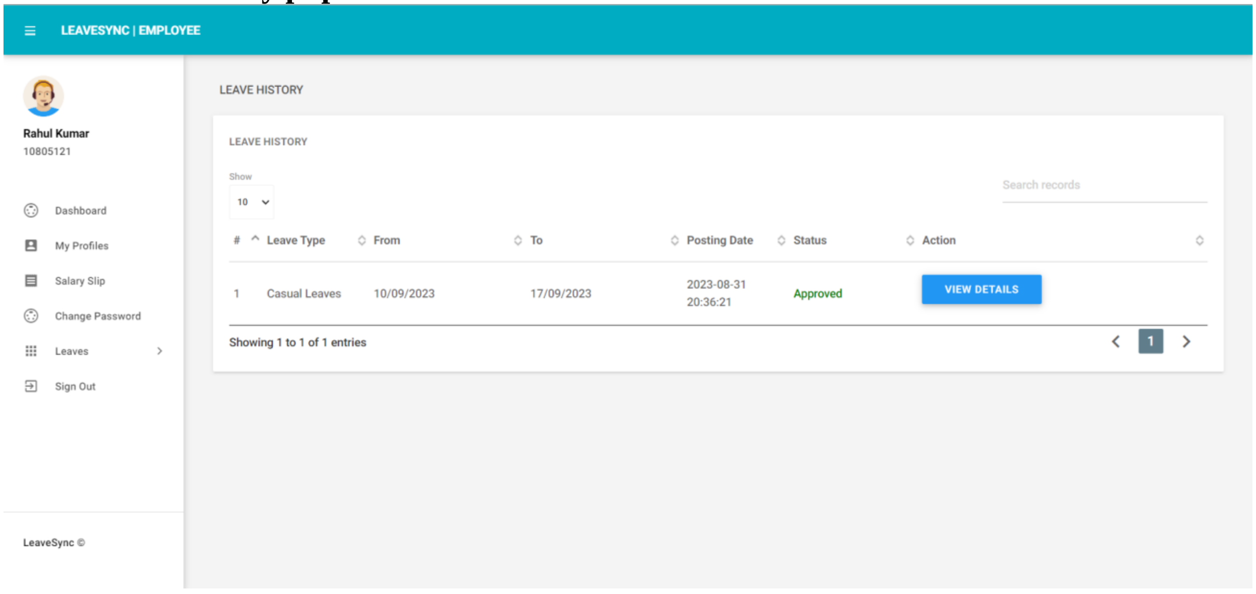
```
if(isset($_POST['change'])){
$password=md5($_POST['password']);
$newpassword=md5($_POST['newpassword']);
$username=$_SESSION['emplogin'];
   $sql ="SELECT Password FROM tblemployees WHERE EmailId=:username and
Password=:password";
$query= $dbh -> prepare($sql);
$query-> bindParam(':username', $username, PDO::PARAM_STR);
$query-> bindParam(':password', $password, PDO::PARAM_STR);
$query-> execute();
$results = $query -> fetchAll(PDO::FETCH_OBJ);
if($query -> rowCount() > 0){
$con="update tblemployees set Password=:newpassword where EmailId=:username";
$chngpwd1 = $dbh->prepare($con);
$chngpwd1-> bindParam(':username', $username, PDO::PARAM_STR);
$chngpwd1-> bindParam(':newpassword', $newpassword, PDO::PARAM_STR);
$chngpwd1->execute();
$msg="Your Password succesfully changed";}
else {
$error="Your current password is wrong";    } } ?>
```

- **Apply-leave.php**



```
if(isset($_POST['apply'])){
$empid=$_SESSION['eid'];
 $leavetype=$_POST['leavetype'];
$fromdate=$_POST['fromdate'];
$todate=$_POST['todate'];
$description=$_POST['description'];
$status=0;$isread=0;
if($fromdate > $todate){
 $error=" ToDate should be greater than FromDate ";}
$sql="INSERTINTO
tblleaves(LeaveType,ToDate,FromDate,Description,Status,IsRead,empid)
VALUES(:leavetype,:todate,:fromdate,:description,:status,:isread,:empid)";
$query = $dbh->prepare($sql);
$query->bindParam(':leavetype',$leavetype,PDO::PARAM_STR);
$query->bindParam(':fromdate',$fromdate,PDO::PARAM_STR);
$query->bindParam(':todate',$todate,PDO::PARAM_STR);
$query->bindParam(':empid',$empid,PDO::PARAM_STR);
$query->execute();
$lastInsertId = $dbh->lastInsertId();
if($lastInsertId){
$msg="Leave applied successfully";}
else {$error="Something went wrong. Please try again";}}?>
```

- **Leavehistory.php**



LEAVE HISTORY

#	Leave Type	From	To	Posting Date	Status	Action
1	Casual Leaves	10/09/2023	17/09/2023	2023-08-31 20:36:21	Approved	VIEW DETAILS

Showing 1 to 1 of 1 entries

```
<?php
$eid = $_SESSION['eid'];
$sql = "SELECT tblleaves.id as lid
,LeaveType,ToDate,FromDate,Description,PostingDate,AdminRemarkDate,AdminRema
rk,Status from tblleaves where empid=:eid";
$query = $dbh->prepare($sql);
$query->bindParam(':eid', $eid, PDO::PARAM_STR);
$query->execute();
$results = $query->fetchAll(PDO::FETCH_OBJ);
$cnt = 1;
if ($query->rowCount() > 0) {
foreach ($results as $result) {             ?>
<tr>
<td> <?php echo htmlentities($cnt); ?></td>
<td><?php echo htmlentities($result->LeaveType); ?></td>
<td><?php echo htmlentities($result->FromDate); ?></td>
<td><?php echo htmlentities($result->ToDate); ?></td>
<td><?php echo htmlentities($result->PostingDate); ?></td>
<td><?php $stats = $result->Status;
 if ($stats == 1) {?>
<span style="color: green">Approved</span>
<?php }if ($stats == 2) { ?>
<span style="color: red">Not Approved</span>
<?php }
if ($stats == 0) { ?>
<span style="color: blue">waiting for approval</span>
<?php } ?>
```

5.2 Testing Approach

Testing Approach

1. Black Box Testing :

Black box testing is a software testing technique that focuses on examining the functionality of a software application without considering its internal code structure, logic, or implementation details. In other words, testers conduct black box testing without any knowledge of the software's internal workings; they treat the software as a "black box" where inputs are provided, and outputs are observed and analyzed.

➢ **Common types of black box testing include:**

Functional Testing: Ensures that the software functions according to its specifications, including testing individual functions, features, and user interactions.

- Integration Testing: Validates the interactions between different components or modules of the software to ensure they work together as expected.
- System Testing: Examines the complete software system to ensure it meets the specified requirements and performs well as a whole.
- Acceptance Testing: Performed by users or stakeholders to determine whether the software meets their acceptance criteria and is ready for deployment.

Black box testing is valuable for uncovering defects, verifying that the software meets user requirements, and ensuring that it functions correctly from an end-user perspective. It complements other testing techniques like white box testing, which examines the internal code structure and logic of the software. Together, these testing methods provide comprehensive coverage of software quality assurance.

2. White Box Testing :

White box testing techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

White box testing is also known as structural testing or code-based testing, and it is used to test the software's internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

- **White box testing techniques :**
  - Statement Coverage - This technique is aimed at exercising all programming statements with minimal tests.
  - Branch Coverage - This technique is running a series of tests to ensure that all branches are tested at least once.
  - Path Coverage - This technique corresponds to testing all possible paths which means that each statement and branch is covered.

- **Advantages:**
  1. White box testing is thorough as the entire code and structures are tested.
  2. It results in the optimization of code removing errors and helps in removing extra lines of code.
  3. It can start at an earlier stage as it doesn't require any interface as in the case of black box testing.
  4. Easy to automate, Easy Code Optimization.
  5. White box testing can be easily started in Software Development Life Cycle.

3. Grey Box Testing :

Greybox testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a combination of black box and white box testing because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

GreyBox testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

- **Objective of Gray Box Testing :**
  - To provide combined advantages of both black box testing and white box testing.
  - To combine the input of developers as well as testers.
  - To improve overall product quality.
  - To reduce the overhead of long process of functional and non-functional testings.
  - To provide enough free time to developers to fix defects.
  - To test from the user point of view rather than a designer point of view.

- **Gray Box Testing Techniques:**
  - Matrix Testing:

    In matrix testing technique, business and technical risks which are defined by the developers in software programs are examined. Developers define all the variables that exist in the program. Each of the variables has an inherent technical and business risk and can be used with varied frequencies during its life cycle.

  - Pattern Testing:

    To perform the testing, previous defects are analyzed. It determines the cause of the failure by looking into the code. Analysis template includes reasons for the defect. This helps test cases designed as they are proactive in finding other failures before hitting production.

  - Orthogonal Array Testing:

    It is mainly a black box testing technique. In orthogonal array testing, test data have n numbers of permutations and combinations. Orthogonal array testing is preferred when maximum coverage is required when there are very few test cases and test data is large. This is very helpful in testing complex applications.

Sr. No.	Test Case	Expected Result	Passed / Failed
1	Attempt to register with a blank username and password.	System should display an error message indicating required fields.	Passed
2	Attempting to Register Again.	An appropriate error message should be displayed indicating that the user is already registered.	Passed
3	Attempt to log in with an invalid username and password.	Website should display properly and function smoothly across various devices	Passed
4	Attempt to access a restricted page without proper authorization.	System should deny access and redirect user to a page indicating insufficient permissions	Passed
5	Update profile information with valid data.	System should successfully update the user's profile information and display a confirmation message.	Passed
6	Test "Forgot Password" feature.	Password reset should be change successfully .	Passed
7	Test the behavior of session data	Ensure that session data is encrypted and secure during transmission over the network	Passed
8	Attempt to submit an empty contact form.	System should display an error message indicating required fields.	Passed
9	Test website responsiveness on different devices	Website should display properly and function smoothly across various devices	Failed
10	Check the font and size of font in different pages	The font and size of font should be same	Passed

11	Click on ‘Back’ button on the browser (other than Home Page)	It should go back to the previous page	Passed
12	Login to the site and click on ‘Refresh’ button on the browser	The page should get refreshed	Passed
13	Test the application's responsiveness to user interactions (clicks, inputs, etc.)	Application should respond promptly to user interactions without noticeable delays	Passed
14	Test the logout functionality	User should be successfully logged out of the system and redirected to the home page.	Passed
15	Check the functionalities in all the browsers like different versions of IE.	In all the versions of the browsers the functionalities, fonts and images should be same.	Failed
16	Admin notifies an employee regarding their approved or rejected leave request..	Employee receives a notification about the status of their leave request.	Passed
17	Employee attempts to download their salary slip from the system.	The system should generate and provide the correct salary slip file in a readable format (e.g., PDF).	Passed
18	Employee requests leave on a public holiday.	System should deny the leave request and prompt the employee to choose an alternative date.	Failed
19	Check session is destroyed after logout, Try direct URL	It should not allow you to move directly without login to application. And login page should be displayed	Passed
20	Deduct salary for the days an employee is absent without approved leave.	The system deducts the equivalent salary for the two days of absence from the employee's monthly salary.	Passed

Chapter 6:

Security & Measures

SECURITY & MEASURES

In our Leave Management System made with PHP, keeping data safe is super important. Only authorized admins with the right username and password can log in. We make passwords really secure by turning them into secret codes that even bad guys can't understand. Plus, we add extra protection with special codes called salts.

Admins are like bosses. They can do important stuff like adding new employees, setting up leave types, and making departments. But we're careful who gets these powers. Only admins who log in can do these things, so we keep the system safe from unauthorized access.

When employees want time off, admins decide if they can have it or not. This happens in a safe place where the info is kept private. We also use a special code called HTTPS to make sure nobody can listen in on our conversations between the admin panel and employees.

And just to be extra sure, we send admins a quick message whenever an employee asks for time off. These messages are sent in a secret way, so only admins can read them. Overall, our system puts safety first by using strong passwords, controlling who has special powers, and keeping everything private and secure.

For Session generation & login functionality I used following code :

```
if($query->rowCount() > 0){
 foreach ($results as $result) {
    $status=$result->Status;
    $_SESSION['eid']=$result->id;}
if($status==0){
$msg="Your account is Inactive. Please contact admin";
} else{
$_SESSION['emplogin']=$_POST['username'];
echo "<script type='text/javascript'> document.location = 'dashboard.php'; </script>";
} }
else{
  echo "<script>alert('Invalid Details');</script>";
}}
```

1. **Authentication and Authorization:**
    - Ensure that only authenticated users can access sensitive functionalities like adding leave, approving leave, adding departments, and managing employees.
    - Implement role-based access control (RBAC) to ensure that users have appropriate permissions based on their roles (e.g., admin, employee).
2. **Data Validation and Sanitization:**
    - Validate and sanitize all user inputs to prevent common vulnerabilities like SQL injection,
    - Implement server-side validation for form submissions to ensure data integrity.
3. **Session Management:**
    - Secure session management by using HTTPS to encrypt data transmitted over the network.
    - Set proper session timeouts to invalidate sessions after a period of inactivity.
    - Avoid storing sensitive information in session variables, especially if sessions are stored in cookies.
4. **Password Security:**
    - Enforce strong password policies, including minimum length, complexity requirements, and expiration periods.
    - Hash and salt passwords before storing them in the database using industry-standard algorithms like bcrypt.
    - Implement password reset functionality with appropriate security measures, such as email verification and temporary tokens.

Chapter 7:

Future Scope And Enhancement

FUTURE SCOPE AND ENHANCEMENT:

1. **Flexible Leave Policies**:
   Let us change the rules for how people can take time off to fit our company better. For example, we might have different rules for different kinds of leave.
2. **Integration with HR Systems:**
   Connect our Leave Management System with our other HR tools so they can share information easily. This will help us manage our employees better.
3. **Advanced Reporting:** Make reports that show us things like when people take leave, how often they're absent, and other important facts about our workforce.
4. **Mobile Application:** Make an app version of our Leave Management System so people can ask for time off and get it approved using their phones.
5. **Calendar Integration:**
   Connect our leave system with popular calendars so people can see their time off in their personal calendars.
6. **Automated Reminders:**
   Set up the system to send reminders to people and their managers when they have leave requests pending or when they have time off coming up.
7. **Employee Self-Service Portal**: Make a place where employees can see how much time off they have left, ask for leave, and see if their requests are approved without Needing to ask a manager.
8. **Multi-level Approval Workflow:**
   Have a system where more than one person needs to approve a leave request, especially if it's for a long time or if it's really important.
9. **Comprehensive Salary Management**:
   Make sure people's paychecks show all the money they've earned and everything that's been taken out, like taxes and insurance.
10. **Attendance Tracking Integration**:
    Connect our leave system with how we track when people come to work so we can pay them the right amount and know if they're absent.
11. **Employee Loan Management**:
    Help people get loans from the company and make sure they pay them back by taking the money out of their paychecks until it's all paid back.

Chapter 8:
Conclusion And Limitations

CONCLUSION AND LIMITATIONS

Conclusion:

project is a Leave Management System made in PHP. It helps handle employee leave requests. The admin can add employees to the system. Employees can log in and ask for different types of leave, like sick leave or paid leave. They can also reset their password if they forget it.

When an employee requests leave, the admin can approve or deny it. The admin gets notified when an employee asks for leave. They can also create different departments in the system.

One important feature is that the system generates monthly salary slips for employees. Employees can easily download these slips to check their salary details. Overall, your system makes it easy to manage leave requests, passwords, departments, and salary information.

Limitation:

- **Multicurrency Support:**

The system can't handle different currencies, impacting multinational companies.

- **Multilanguage Support:**

No support for multiple languages affects usability.

- **Salary Deduction for Leaves:**

Leaves don't trigger automatic salary deductions, causing payroll discrepancies.

- **Local Machine Limitation:**

Restricted to local machines, limiting accessibility and scalability.

- **Limited Customization:**

Customization options such as leave policies or notification templates are limited, impacting the system's adaptability to unique organizational requirements or regulatory needs.

Chapter 9:

Bibliography

BIBLIOGRAPHY

We learned from online sources like YouTube and have utilized specific technologies like Bootstrap5 ,CSS in our project,

I. YouTube Videos :

- Introduction to Bootstrap Tutorial In Hindi:

  Bootstrap Tutorial #1 …

CodeWithHarry

II. Websites:

We had seen the demo or free trial for some websites and analyze for Leave Management System

- https://www.greythr.com/leave-management-software
- https://www.zoho.com/people/hrknowledgehive/What-is-a-leave-management-system.html
- https://www.leavedates.com/leave-management-system
- https://leaveboard.com/

III. Reference:

- https://phpgurukul.com/

IV. Template Link:

- https://www.free-css.com/free-css-templates/page210/adminlte