

**SA TRACKER**  
**A PROJECT REPORT**  
for  
**Project (KCA451)**  
**Session (2024-25)**

**Submitted by**

**Kunal Gangwar**  
(2300290140093)  
**Khushi Singh**  
(2300290140089)  
**Jeetu Gangwar**  
(2300290140082)

**Submitted in partial fulfilment of the**  
**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**  
**Dr. Akash Rajak**  
**Dean**



**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**KIET Group of Institutions, Ghaziabad**  
**Uttar Pradesh-201206**

**(APRIL 2025)**

# CERTIFICATE

Certified that **Kunal Gangwar (2300290140093)**, **Khushi Singh (2300290140089)**, **Jeetu Gangwar (2300290140082)** have carried out the project work having “**SA Tracker**” (**Project-KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

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

Date:

**Dr. Akash Rajak**  
**Dean**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

**Dr. Akash Rajak**  
**Dean**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

# **SA Tracker**

## **ABSTRACT**

The Salary and Attendance Management System is designed to streamline and automate the processes related to employee attendance tracking and salary computation within an organization. This system reduces manual errors, enhances efficiency, and ensures accurate and timely payroll generation. It allows employees to mark their attendance through a secure login, while administrators can monitor attendance records, manage leaves, calculate working hours, and process monthly salaries based on predefined rules and policies.

Key features include employee profile management, real-time attendance tracking (daily/shift-wise), leave management, overtime calculation, salary slip generation, and report generation for both attendance and payroll. The system also ensures data security, role-based access, and audit trails for transparency and accountability. By implementing this system, organizations can achieve better workforce management, reduce administrative overhead, and maintain compliance with labor regulations.

In conclusion, the Salary and Attendance Management System provides an efficient, accurate, and secure solution to automate routine HR processes. It supports digital transformation in human resource operations, improves administrative efficiency, and enables organizations to focus on strategic employee development initiatives.

## ACKNOWLEDGEMENTS

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

Words are not enough to express my gratitude to **Dr. Akash Rajak, Professor** and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

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

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

**Kunal Gangwar**

**Khushi Singh**

**Jeetu Gangwar**

# TABLE OF CONTENT

Certificate .....	ii
Abstract .....	iii
Acknowledgements .....	iv
Table of Contents .....	v-vi
<b>1. Introduction .....</b>	<b>1-3</b>
1.1 Overview .....	1
1.2 Motivation .....	1
1.3 Problem Statement.....	2
1.4 Scope .....	2
1.5 Features .....	3
<b>2. Literature Survey .....</b>	<b>4-5</b>
<b>3. Requirement Analysis .....</b>	<b>6-8</b>
3.1 Introduction .....	6
3.2 Functional Requirements .....	6
3.2.1 User Module .....	6
3.2.2 Administration Module .....	7
3.2.3 Package Module.....	7
3.3 Non-Functional Requirements.....	7
3.3.1 Performance .....	7
3.3.2 Security .....	8
3.3.3 Scalability.....	8
3.3.4 Usability .....	8
<b>4. Hardware and Software Specification .....</b>	<b>9-10</b>
4.1 Hardware Specification .....	9
4.2 Software Specification.....	10
<b>5 Proposed Work .....</b>	<b>11-13</b>
5.1 Technology Description .....	11
5.2 Approach Used .....	11
5.3 Implementation Details.....	12
5.4 Challenges Faced.....	13
5.5 Future Enhancements .....	13
<b>6. Choice of Tools &amp; Technology .....</b>	<b>14-15</b>
6.1 HTML and CSS.....	14
6.2 PHP .....	14
6.3 Database :MySQL .....	15
6.4 System and Future Benefits.....	15
<b>7.System Design .....</b>	<b>16-17</b>
<b>8. Design .....</b>	<b>18-22</b>
8.1 Data Flow Diagram .....	18
8.2 E R Diagram .....	19
8.3 Use Case Diagram .....	20-21
8.4 Flow Chart.....	22

<b>9. Results</b>	<b>23-26</b>
9.1 Login Page	23
9.2 Admin Page	24
9.3 Company Page	25
9.4 Home Page	26
<b>10. Coding</b>	<b>27-53</b>
<b>11. Testing</b>	<b>54-55</b>
10.1 Introduction	54
10.2 Types of Testing	55
<b>12. Implementation of Need</b>	<b>56-57</b>
<b>13. Discussion</b>	<b>58-59</b>
11.1 Performance	58
11.2 Future Research Direction	59
<b>14. Limitations</b>	<b>60-61</b>
<b>15. Conclusion</b>	<b>62</b>
<b>16. References</b>	<b>63</b>

# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

The Salary and Attendance Management System is a software-based solution designed to automate the processes of employee attendance tracking and salary management in organizations. It replaces the traditional manual approach with a digital interface that streamlines record-keeping, enhances accuracy in payroll calculations, and reduces human errors. This system maintains a centralized database that stores employee details, attendance logs, leave records, and salary data. It facilitates real-time tracking and provides quick access to reports and salary slips, ultimately improving the efficiency of Human Resource (HR) operations.

### 1.2 Motivation

The motivation for developing this system stems from the challenges faced by organizations in managing their HR and payroll functions manually. Manual processes are:

- Time-consuming: Payroll processing takes longer when done manually, especially for large workforces.
- Error-prone: Human errors in data entry or calculations can lead to incorrect salary disbursement.
- Inefficient: Managing leave requests, attendance, and salary records across different platforms or paper files leads to confusion and inefficiency.
- Non-scalable: As organizations grow, manual systems become increasingly difficult to manage.

A centralized digital solution reduces these problems significantly by:

- Automatically calculating salaries based on attendance and company rules.
- Providing accurate and up-to-date information.
- Saving time and improving HR productivity.
- Ensuring compliance with labor laws and company policies.

The need for transparency and efficiency in managing employee-related information drives the development of a robust, automated system that handles all aspects of attendance and salary management from one platform.

### 1.3 Problem Statement

Managing attendance and salary manually is not only inefficient but also risks inaccuracies that can lead to employee dissatisfaction, payroll errors, and potential legal or compliance issues. The lack of integration between attendance data and salary processing often results in:

- Incorrect payroll calculations due to missing or wrong attendance data.
- Lack of real-time monitoring of employee attendance.
- Delayed salary processing during peak periods.
- Poor record-keeping, making audits or analysis difficult.
- Limited accessibility, with physical records hard to retrieve and manage.

This project proposes an integrated system that resolves these problems by:

- Automating attendance tracking and salary calculations.
- Managing leave approvals digitally.
- Providing easy access to records and reports.
- Reducing the dependency on human intervention.

### 1.4 Scope

The Salary and Attendance Management System is a scalable application suitable for deployment in businesses of all sizes. It is designed to handle multiple employee roles, customizable working hours, different salary structures, leave types, and user permissions. Key areas covered by the system include:

- Employee Registration and Profile Management: Admins can register new employees, assign departments, roles, and salary components.
- Attendance Management: Employees can clock in/out, and the system tracks working hours and late arrivals. Shift-based scheduling is also supported.
- Leave Management: Employees apply for leaves through the system. HR/Managers can approve or reject these requests. Leave balances are updated in real-time.
- Payroll Processing: Salaries are computed automatically based on attendance, approved leaves, overtime, deductions, and bonuses.
- Reports and Payslips: Monthly salary slips are generated for each employee. Attendance and salary reports can be exported for analysis.
- Security and Access Control: Role-based permissions restrict data access, ensuring confidentiality and data integrity.



## 1.5 Features

The system includes a range of features designed to simplify HR operations and ensure accuracy in payroll:

### 1. User Authentication

- Secure login system with role-based access (Employee, HR, Admin).
- Password reset and session management.

### 2. Employee Management

- Add, edit, or remove employee profiles.
- Assign departments, designations, and salary structures.

### 3. Attendance Tracking

- Mark in-time and out-time.
- View daily, weekly, and monthly attendance logs.
- Track late arrivals, absences, and shift changes.

### 4. Leave Management

- Apply for casual, sick, or earned leave.
- Approve or reject leave requests with comments.
- Track leave balances and leave history.

### 5. Salary Calculation

- Automatic computation of gross and net salary.
- Manage bonuses, deductions, and overtime.
- Generate and download salary slips.

### 6. Reporting

- Generate attendance and payroll reports.
- Export data to Excel/PDF formats.
- View summaries for audit and management review.

## **CHAPTER 2**

### **LITERATURE SURVEY**

Employee management is a critical aspect of any organization, with salary and attendance being two of the most important components. Managing these aspects manually often leads to inefficiencies, errors, and dissatisfaction among employees. With technological advancements, organizations are increasingly adopting automated systems to streamline salary calculations and attendance tracking.

#### **1. Salary Management Systems:**

Salary management refers to the structured process of determining and disbursing employee wages based on predefined rules, attendance, performance, and company policies. Research has shown that efficient salary systems help maintain employee motivation and retention (Kumar, 2019).

##### **- Automation and Accuracy:**

Automated salary management systems reduce human error and ensure accurate calculation of wages, bonuses, deductions, and tax compliances (Gupta, 2020). Systems like ERP (Enterprise Resource Planning) have integrated salary modules to manage complex structures.

##### **- Compliance:**

Modern systems ensure that companies comply with labor laws and tax regulations, which helps in avoiding legal penalties (Singh & Rani, 2021).

##### **- Transparency:**

Transparent salary management improves trust between employees and employers, enhancing overall organizational culture (Patel, 2020).

#### **2. Attendance Management Systems:**

Attendance tracking involves recording employees' working hours, leaves, absences, and overtime. It is closely tied to salary calculation and performance appraisal.

##### **- Biometric and RFID-Based Systems:**

Biometric (fingerprint, face recognition) and RFID systems have been widely adopted for precise tracking of attendance (Sharma, 2021). These systems minimize fraudulent practices like "buddy punching" (one employee punching in for another).

- Integration with Payroll:

Attendance records feed directly into payroll systems, automating the salary calculation based on the number of days worked and approved leaves (Chopra, 2020).

- Cloud-Based Solutions:

Recent trends show a shift towards cloud-based attendance management, offering real-time access to records and better scalability (Verma, 2022).

# CHAPTER 3

## REQUIREMENT ANALYSIS

### 3.1 Introduction

The Salary and Attendance Management System aims to automate the processes of tracking employee attendance, calculating salaries, and managing related tasks such as leave requests and overtime. This system serves to reduce human error, save time, and improve accuracy in HR management. The requirement analysis outlines the functional and non-functional requirements necessary for the effective development and deployment of this system.

### 3.2 Functional Requirements

The functional requirements describe the specific operations that the system should support to fulfill the needs of the organization. These include modules and features to be implemented in the system.

#### 3.2.1 User Module

The User Module is intended for employee use, allowing them to interact with the system. Its key features include:

**1. Login/Logout:**

- Employees should be able to log in securely using their credentials (username and password).
- Logout functionality to ensure session security.

**2. Mark Attendance:**

- Employees can mark their attendance by checking in and checking out.
- Real-time tracking of attendance data (e.g., arrival and departure times).

**3. View Attendance History:**

- Employees can view their attendance records for the month or year.
- Attendance reports include days present, absences, leave taken, and overtime.

**4. Leave Request:**

- Provide leave history for employees.

### **3.2.2 Administrator Module**

#### **1. Manage Employees:**

- Add, update, and delete employee records (personal details, salary, department, etc.).
- Assign roles, departments, and salary structure for each employee.

#### **2. Manage Attendance:**

- Approve or reject attendance data (e.g., late arrivals, early departures, missed days).
- Monitor employee attendance in real-time.

#### **3. Manage Leave Requests:**

- Approve or reject employee leave requests.
- Track leave balances (vacation, sick, etc.).

#### **4. Reports and Analytics:**

- Generate reports on employee attendance, leave history, and salary payments.
- Export reports in various formats (Excel, PDF).

### **3.2.3 Package Module**

The Package Module manages salary packages, which can include basic salary, bonuses, allowances, and other benefits. This module will be integrated with both the User and Administration modules.

## **3.3 Non-Functional Requirements**

Non-functional requirements describe the performance, security, and usability standards the system must meet. These requirements are essential for the system's usability, reliability, and future scalability.

### **3.3.1 Performance:**

The system should deliver consistent performance under all conditions. Essential actions like login, job search, and application submission should occur within a few

seconds. It must support multiple users operating simultaneously without impacting speed or functionality.

### **3.3.2 Security:**

Security is a top priority. The system must use secure authentication protocols and encrypt sensitive data such as user credentials and resumes. It should defend against common threats such as SQL injection, cross-site scripting (XSS), and unauthorized access.

### **3.3.3 Scalability:**

The platform should be capable of scaling both vertically and horizontally. As user demand grows, the system must efficiently accommodate more users, job listings, and employers without requiring significant infrastructure changes.

### **3.3.4 Usability:**

The user interface should be intuitive and easy to navigate. The system must be accessible across various devices and platforms, ensuring a smooth and responsive experience for users of all technical backgrounds.

## **CHAPTER 4**

### **HARDWARE AND SOFTWARE SPECIFICATIONS**

#### **4.1. HARDWARE SPECIFICATIONS**

- Processor: A minimum of Intel Core i3 (8th generation or later) or AMD Ryzen 3 is required. For optimal performance, an Intel Core i5 (10th generation or later) or AMD Ryzen 5 is recommended.
- Memory (RAM): At least 4 GB of RAM is needed to run the system smoothly. However, 8 GB or more is recommended for better multitasking and faster data processing.
- Storage: A minimum of 250 GB HDD is sufficient for storing basic employee and attendance data. For better speed and reliability, a 512 GB SSD or higher is recommended.
- Display: A 14-inch screen with 720p resolution is the minimum requirement. A 15.6-inch Full HD (1080p) screen is recommended for a better user interface and ease of operation.
- Network Connectivity: The system must support Ethernet or Wi-Fi to enable network-based operations, including data synchronization and access to web-based systems.
- Input Devices: A standard keyboard and mouse are essential for user interaction.

## 4.2. SOFTWARE SPECIFICATIONS

- **Operating System:** The system should operate on Windows 10 or later versions. For server deployments, Linux distributions like Ubuntu are also suitable.
- **Database Management System:** MySQL or PostgreSQL should be used to handle employee records, attendance logs, leave requests, and salary data efficiently.
- **Backend Technologies:** The backend can be developed using PHP, Python (with Django or Flask framework), or Node.js, depending on the project requirements and developer expertise.
- **Frontend Technologies:** The user interface should be built using HTML5, CSS3, and JavaScript for responsiveness and accessibility.
- **Web Server:** Apache or Nginx should be used to host the application if it is a web-based system.
- **Development Tools:** Tools like Visual Studio Code, Sublime Text, or PyCharm can be used for coding. Database management tools such as MySQL Workbench or phpMyAdmin will assist in database operations.
- **Security Software:** SSL certificates and antivirus programs should be installed to ensure secure operations, especially for systems handling sensitive employee data.



# CHAPTER 5

## PROPOSED WORK

### 5.1 Technology Description

The proposed Salary and Attendance Management System utilizes a modern, web-based architecture to streamline HR operations like attendance tracking, salary computation, and leave management. The system is developed using **HTML5**, **CSS3** for the front-end to ensure a responsive and user-friendly interface. The **backend** is developed using **PHP** for robust server-side logic. For data storage and management, a **MySQL** database is used to handle employee records, attendance logs, salary details, and leave history efficiently. Additionally, **Apache** is used as the web server in XAMPP environments to host the application locally or on the cloud. Security features such as **authentication**, **authorization**, and **data encryption** are integrated to protect sensitive data. The application is designed to be platform-independent and accessible from any device with an internet connection

### 5.2 Approach Used

The development approach used for this system is the **Agile Development Methodology**, which promotes incremental development, regular feedback, and quick adaptability to changes. The system is divided into several modules: **User Module**, **Admin Module**, **Attendance Tracking Module**, and **Payroll Module**. Each module was developed and tested in short sprints, allowing for parallel development and testing. The database schema was designed first to ensure relational integrity and normalization. Front-end pages were built with form validation, while backend APIs handled data processing, calculations (such as salary computation based on attendance), and data storage. Regular meetings with users (e.g., HR representatives) were held to gather feedback and incorporate feature enhancements early in the development lifecycle. This approach ensured that the system met the real-world needs of both employees and administrators.

#### 5.2.1 Objectives

- To develop a centralized system that automates employee attendance tracking and salary calculation.
- To reduce manual errors and increase efficiency in payroll processing.
- To provide role-based access for employees and administrators.
- To ensure data security and accurate reporting for HR decision-making.

#### 5.2.2 Technologies Used

**Frontend Technologies:**

- HTML5 & CSS3: For creating a structured and styled user interface.
- JavaScript: For client-side interactivity and form validation.
- Bootstrap: For responsive and mobile-friendly UI design.

### **Backend Technologies:**

- PHP or Python (Flask/Django): To handle server-side logic and API development.
- MySQL Database: For storing employee details, attendance logs, leaves, and salary records.
- XAMPP/LAMP Stack: Local development and testing environment.
- Apache Web Server: For hosting the application.

### **Development Tools:**

- Visual Studio Code / Sublime Text: Code editors used for development.
- phpMyAdmin: For managing and interacting with the MySQL database.

## **5.2.3 Features**

### **User Authentication & Role Management**

- Secure login system with different dashboards for employees and admins.
- Password encryption and session handling.

### **Attendance Tracking**

- Daily check-in/check-out system.
- Automatic calculation of total working hours.
- Late entry and early leave detection.

### **Leave Management**

- Leave request application with approval/rejection workflow.
- Leave balance calculation and tracking.
- Notification system for leave status.

### **Salary Calculation**

- Monthly salary computed based on working days, overtime, and leaves.
- Auto deductions for unapproved leaves or short attendance.
- Admin can set salary structure per employee.

## **Admin Panel**

- Add/edit/delete employee details.
- View department-wise reports and salary data.
- Approve or reject leaves and update attendance manually if needed.

## **Reports and Downloads**

- Attendance reports and payslips in PDF or Excel format.
- Monthly payroll summary for audit and HR analysis.

## **5.3 Implementation Details**

The system is implemented as a multi-user web application that supports both employees and administrators. When users log in, they are directed to role-specific dashboards. Employees can mark attendance, apply for leave, and view their salary slips. The system logs check-in and check-out times and calculates total working hours. Administrators can manage employee records, approve leave requests, and process monthly payroll. Salary is automatically calculated based on attendance, overtime, and deductions (e.g., for unapproved leaves). The system also generates detailed reports in PDF or Excel format. Real-time notifications are provided for leave status updates and salary issuance. All critical operations are secured with session handling and input validation to prevent unauthorized access and data breaches. The application is tested for performance and cross-browser compatibility to ensure reliability.

## **5.4 Challenges Faced**

During development, several challenges were encountered. First, integrating accurate attendance tracking and ensuring it works across different time zones and browsers was technically complex. Second, implementing the salary calculation logic, which had to consider different components like basic pay, overtime, deductions, and bonuses, required careful database design and validation. Third, data consistency and handling concurrent requests in the admin dashboard posed difficulties, especially in multi-user environments. Another major challenge was ensuring security. Implementing secure login systems and preventing SQL injection or XSS attacks took considerable time.

## **5.5 Future Enhancements**

- **Biometric or QR Code Integration** for more secure and foolproof attendance tracking.
- **Mobile App Version** to allow easier access for employees and admins on-the-go.
- **AI-Powered Analytics** for predicting employee absenteeism trends, leave patterns, and salary optimization.
- **Integration with Bank APIs** for automatic payroll transfer.
- **Multi-language Support** to serve organizations with a diverse workforce.

## CHAPTER 6

### CHOICES OF TOOLS & TECHNOLOGY

#### 6.1 HTML and CSS

The frontend of the system is developed using **HTML (HyperText Markup Language)** and **CSS (Cascading Style Sheets)**. HTML is responsible for creating the structure and layout of the web pages. It defines the content elements such as input forms, buttons, text boxes, tables, menus, and navigation bars. For example, a recruiter dashboard, job listing page, or candidate application form is all created using HTML. It is the backbone of the user interface.

CSS enhances the presentation of HTML elements by applying styles like colors, fonts, spacing, and layout design. It ensures the interface is visually appealing, consistent, and responsive across different devices such as desktops, tablets, and smartphones. A well-designed interface plays a crucial role in user experience and usability, making the system more intuitive for all users—administrators, recruiters, and applicants.

#### 6.2 PHP

**PHP (Hypertext Preprocessor)** is the server-side scripting language used to manage the backend logic of the application. It plays a vital role in processing form data, handling user login and registration, managing sessions, and connecting to the database. PHP acts as the bridge between the user interface and the data stored in the database.

When a user submits a form—such as applying for a job or posting a vacancy—PHP handles the request, validates the input, and performs actions such as inserting or retrieving data from the database. It also helps in generating dynamic content. For example, job listings shown on the website can be dynamically retrieved from the database based on filters like location, job type, or department.

PHP also plays an important role in implementing security features such as input validation, password hashing, session handling, and role-based access control. It ensures that sensitive data is protected and that only authorized users can access specific parts of the system.

#### 6.3 Database: MySQL

**MySQL** is a popular and powerful relational database management system (RDBMS) used to store and manage all the information related to the recruitment process. It stores data such as user credentials, job postings, application records, interview schedules, uploaded resumes, feedback, and reports.

Using structured query language (SQL), the system can perform operations such as inserting, updating, retrieving, and deleting records. For instance, recruiters can view the list of applicants for a specific job, filter them by qualification, and schedule interviews—all through seamless interaction with the **MySQL** database via **PHP**.

**MySQL** supports indexing, foreign key constraints, and transactions, making it suitable for managing complex data relationships and ensuring data integrity across the system. It also allows scalability and performance optimization for handling larger volumes of recruitment data.

## 6.4 System Features and Benefits

A Recruitment Management System built using HTML, CSS, PHP, and MySQL can offer a wide range of useful features:

- **Scalability:** The system can easily be expanded to accommodate more employees, additional departments, or more complex salary structures without major changes in technology.
- **Cost-Effective:** Using open-source technologies like PHP, MySQL, and Bootstrap keeps development and maintenance costs low.
- **Cross-Platform Accessibility:** As a web-based system, it is accessible on various devices and operating systems, ensuring flexible usage.
- **Easy Maintenance:** The modular code structure and wide community support for PHP, MySQL, and Bootstrap make troubleshooting and updates easier.
- **Security:** Built-in features like SQL injection prevention, session handling, and input validation ensure the security of sensitive data.

# **CHAPTER 7**

## **SYSTEM DESIGN**

### **1. Primary Design (High-Level Design)**

#### **Components:**

- **User Interface (Frontend)**
  - Employees and Admins interact with the system (dashboard, forms, reports)
- **Application Layer (Backend)**
  - Handles business logic (attendance tracking, salary calculation)
- **Database Layer**
  - Stores persistent data (employee records, attendance, salaries)

### **2. Secondary Design (Detailed Design)**

#### **A. Employee Module**

- Manage employee profiles
- Department & role assignments

#### **B. Attendance Module**

- Record daily check-in/out
- Auto status marking (Present, Late, Absent)

#### **C. Leave Module**

- Submit leave request
- Approval workflow
- Deduction handling

#### **D. Salary Module**

- Fetch attendance and leave
- Calculate salary (basic + allowances - deductions)
- Generate and download payslips

#### **E. Admin Panel**

- View analytics (monthly reports)
- Manage salary structures, leave types, users

## **User Interface Design**

#### **For Employees:**

- Dashboard (Attendance Summary, Leaves)
- Mark Attendance (Check-In/Out button)
- Apply for Leave
- View Payslips

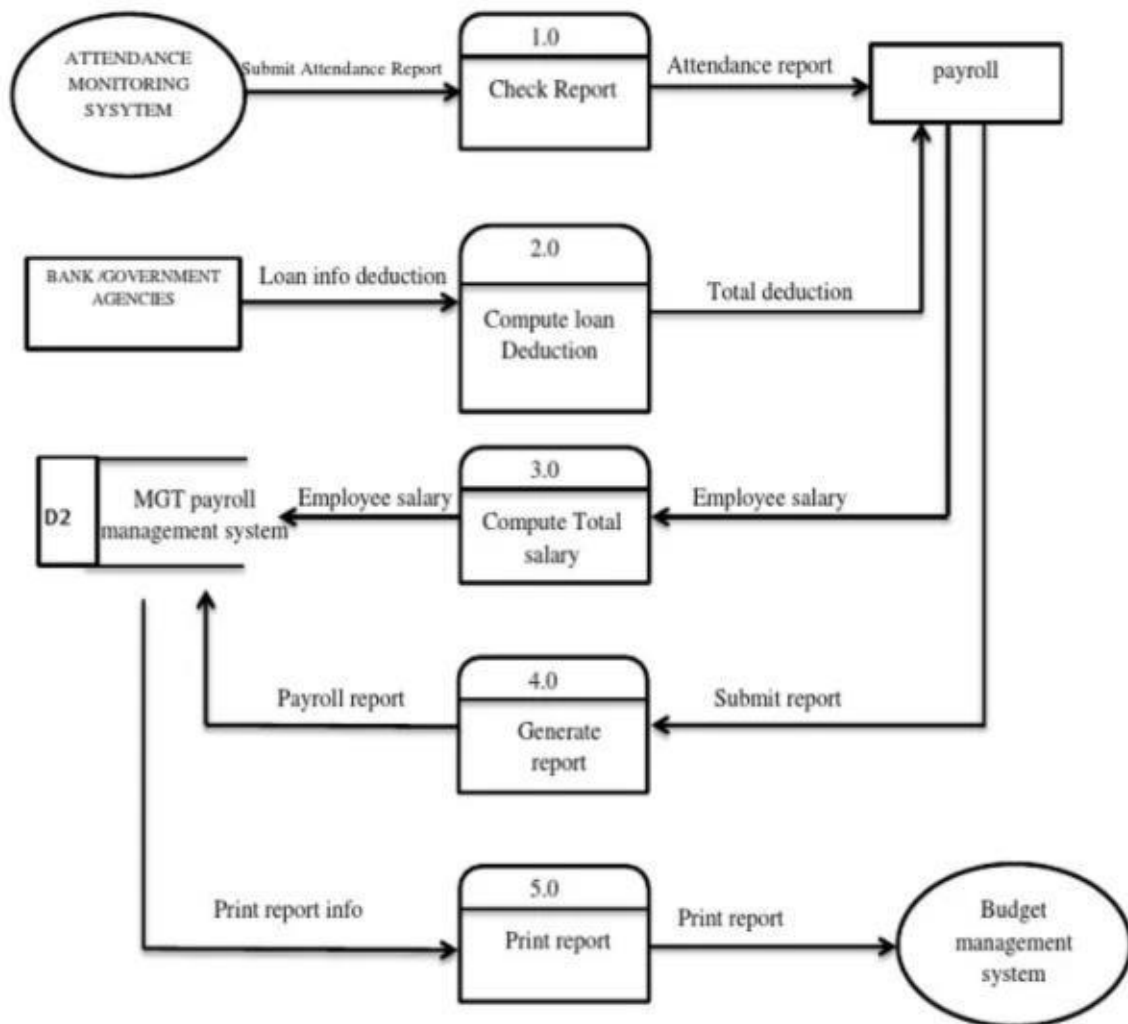
#### **For Admins:**

- Dashboard (Total Employees, Absent Today, Total Payroll)
- Manage Employees
- View Attendance Logs
- Generate Salary Reports
- Download Salary Slips

## CHAPTER 8

### DESIGN

#### 8.1 Data Flow Diagram

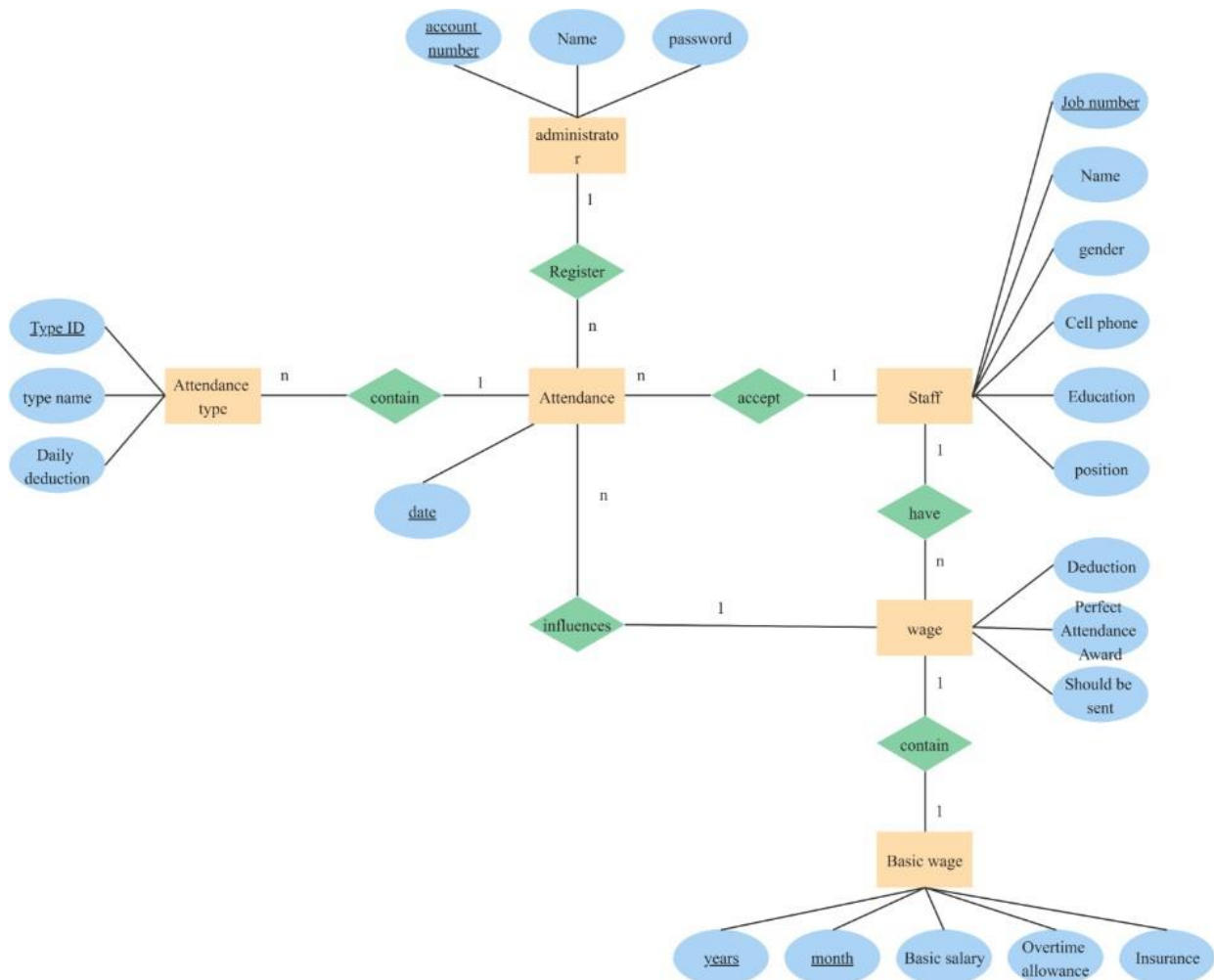


The **Salary and Attendance Management System** is a software solution designed to automate the processes of employee attendance tracking and salary computation in organizations. Traditionally, these tasks are performed manually, often leading to inefficiencies, human errors, and lack of transparency.



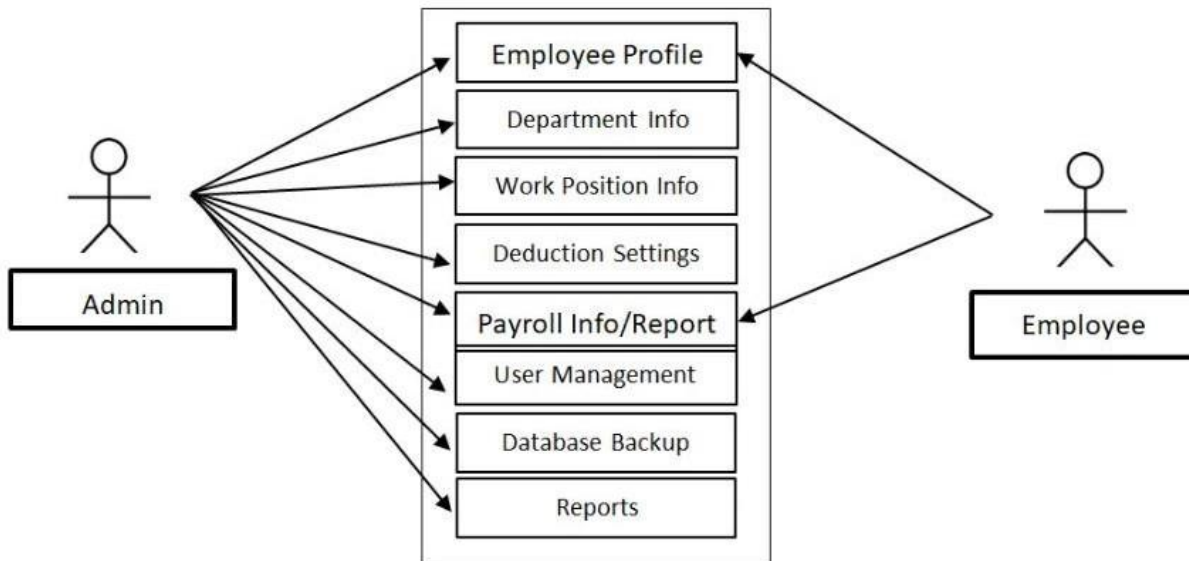
## 8.2 ER Diagram

An Entity Relationship Diagram is a diagram that represents relationships among entities in a database.

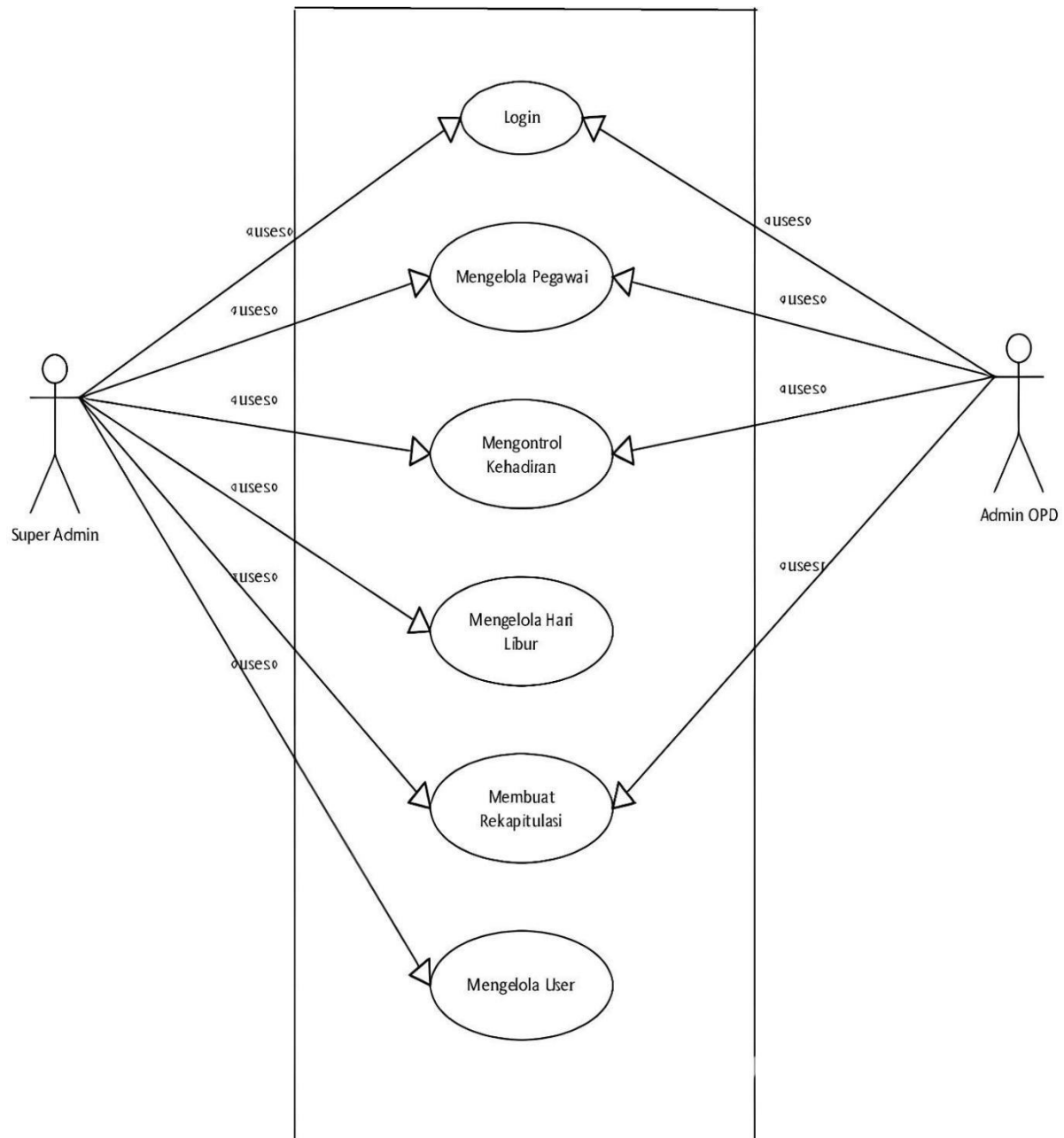


### 8.3 Use Case Diagram

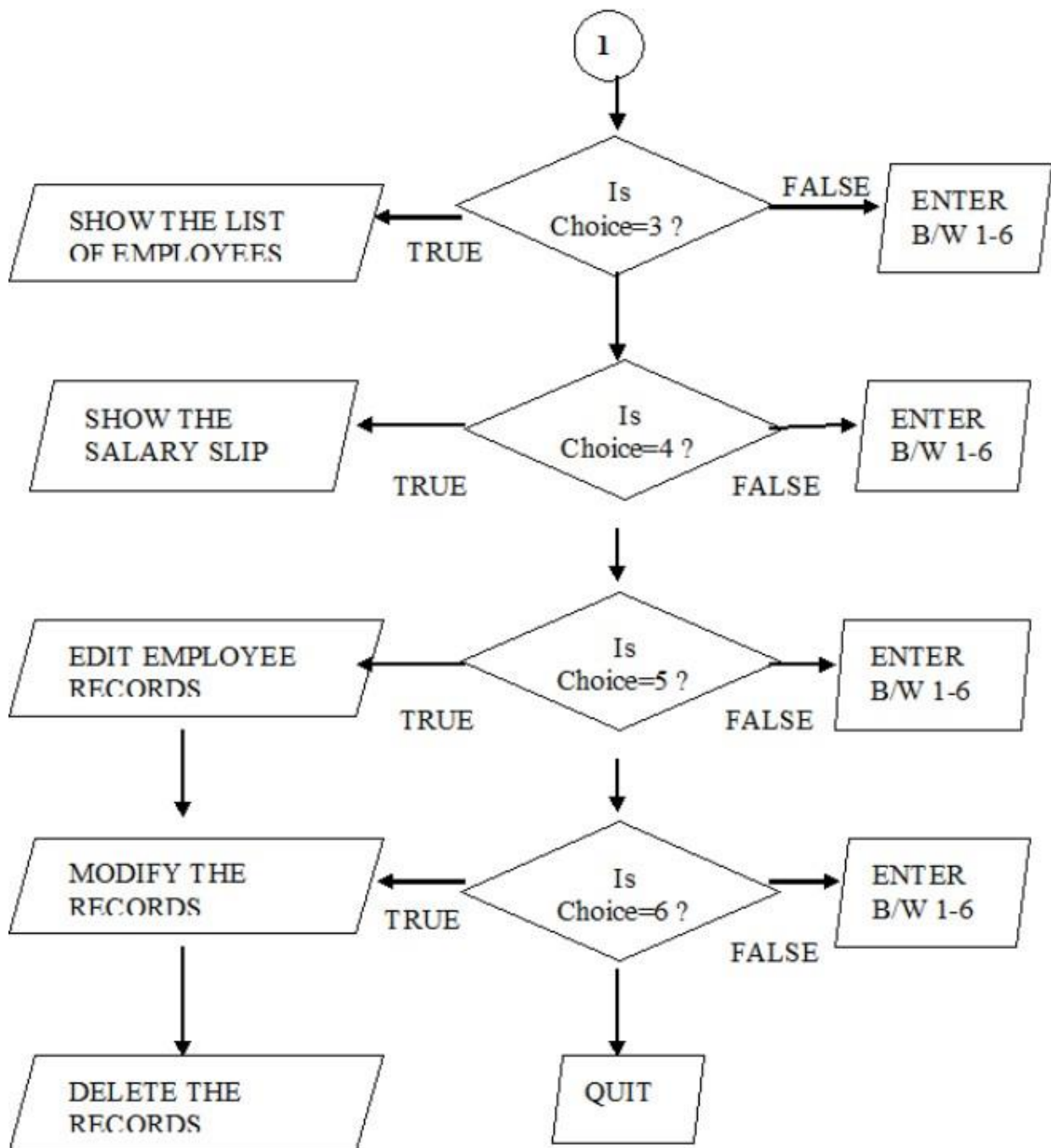
#### Salary Management System:



## Attendance Management System:



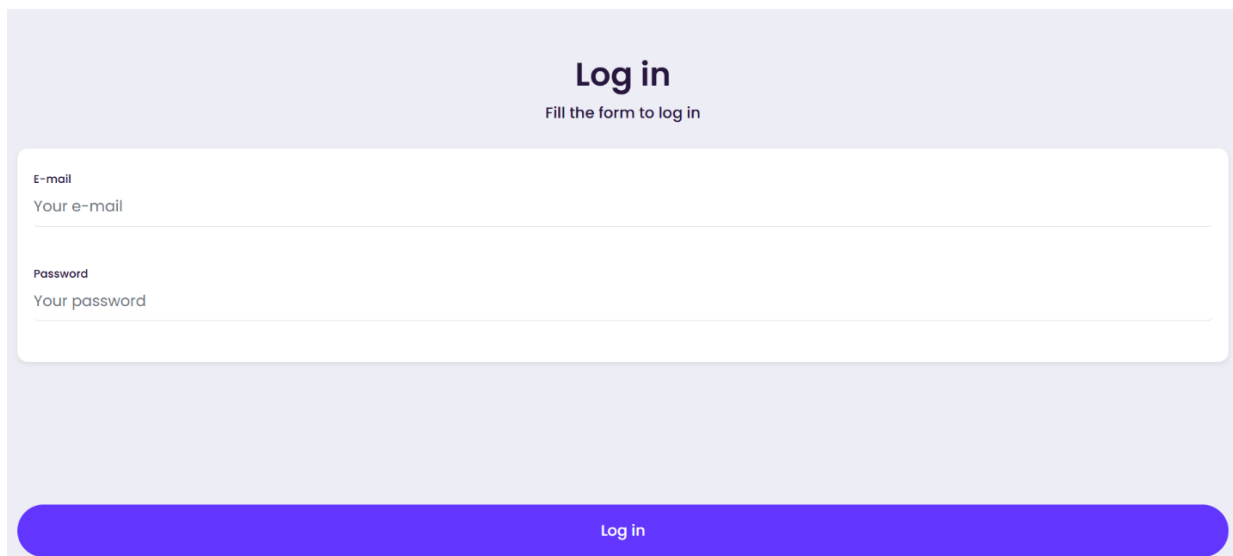
#### 8.4 Flow chart:



# CHAPTER 9

## Results

### 1. LOGIN PAGE



The image shows a login page design. At the top, there is a light purple header bar. Below it, the text "Log in" is centered in a bold, dark font, with the subtitle "Fill the form to log in" in a smaller, lighter font. Below the header, there is a white form box with rounded corners. Inside the form box, there are two input fields. The first field is labeled "E-mail" and "Your e-mail". The second field is labeled "Password" and "Your password". Below the form box, there is a large, rounded, blue button with the text "Log in" in white.

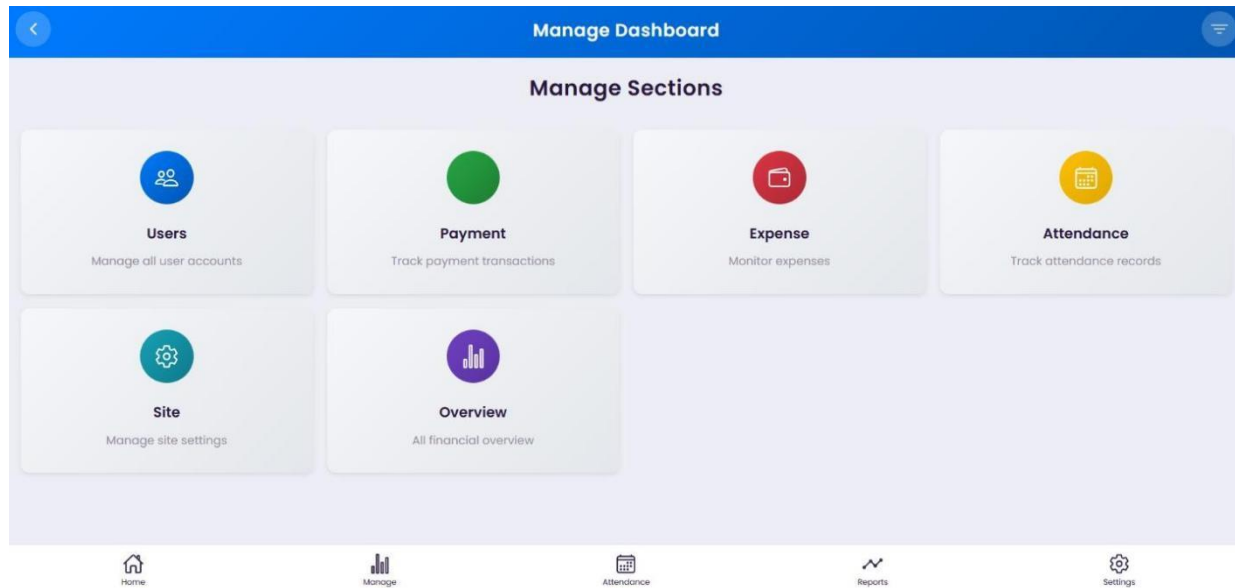
**Log in**  
Fill the form to log in

**E-mail**  
Your e-mail

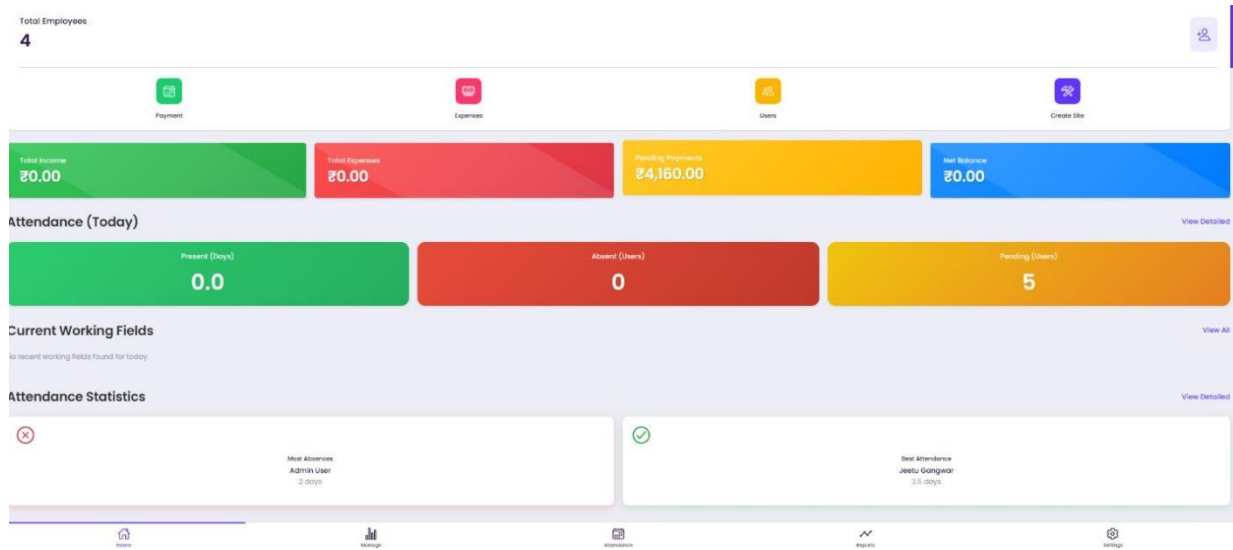
**Password**  
Your password

Log in

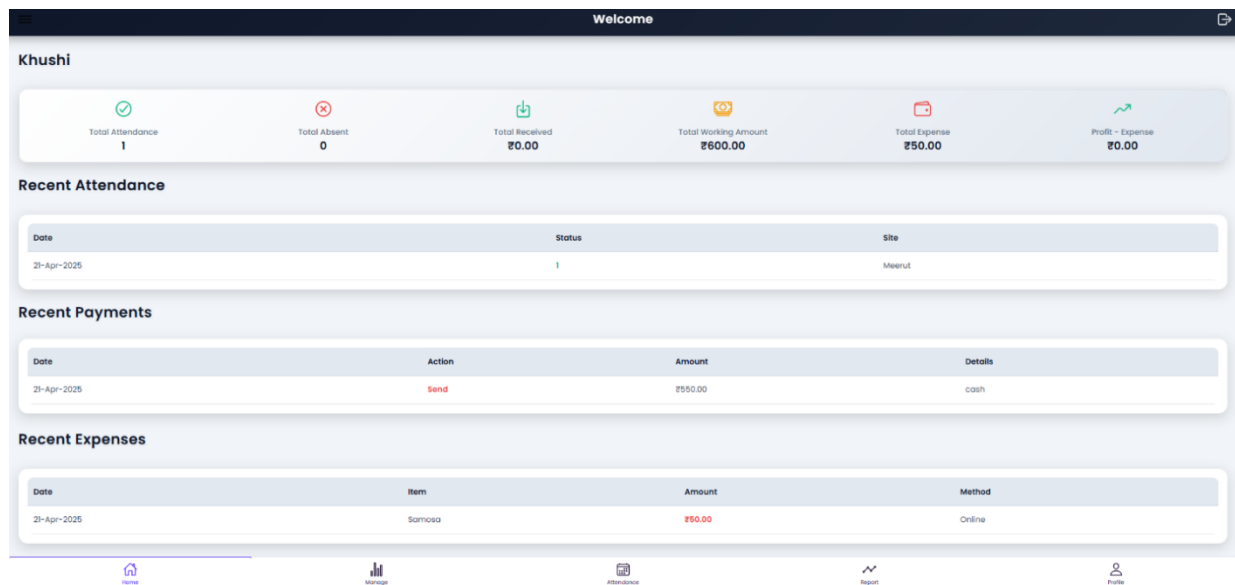
## 2. Dashboard Page



### 3.Admin Page



### 3. User Page





## CHAPTER 10

### CODING

#### Admin

##### 1. Admin controller:

```
<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use App\Models\User;
use App\Models\Payment;
use App\Models\Expense;
use App\Models\Attendance;
use App\Models\Site;
use Illuminate\Http\Request;
use Carbon\Carbon;

class AdminController extends Controller
{
    public function index()
    {
        // Date range for "all time" (Base Controller Method)
        [$startDate, $endDate] = $this->getDateRange('all');

        // Admin-specific data
        $adminSentMoney = Payment::where('action', 'send')
            ->where('sender_type', 'admin')
```

```

->whereBetween('transaction_date', [$startDate, $endDate])

->sum('amount');

$adminReceivedMoney = Payment::where('action', 'receive')

->where('recipient_type', 'admin')

->whereBetween('transaction_date', [$startDate, $endDate])

->sum('amount');

$adminExpense = Expense::where('user_id', 1) // Assuming admin ID is 1

->whereBetween('expense_date', [$startDate, $endDate])

->sum('amount');

$adminIncome = $adminReceivedMoney - $adminSentMoney - $adminExpense;

// All users including admin
$users = User::with([

    'attendances' => fn($query) => $query->whereBetween('attendance_date', [$startDate,
$endDate]),

    'expenses' => fn($query) => $query->whereBetween('expense_date', [$startDate,
$endDate])

])->get();

// User details (Base Controller Method)
$userDetails = $this->calculateUserDetails($users, $startDate, $endDate);

$pendingPayments = $userDetails->sum(function ($user) {

    return $user['total_profit'] > 0 ? $user['total_profit'] : 0;

});

// Today's attendance
$today = Carbon::today()->toDateString();

```

```

$attendances = Attendance::where('attendance_date', $today)->get();

$presentCount = $attendances->sum(function ($attendance) {
    return is_numeric($attendance->status) ? floatval($attendance->status) : 0;
});

$absentCount = $attendances->where('status', 'A')->count();

$totalUsers = User::count();

$markedAttendanceCount = $attendances->count();

$pendingCount = $totalUsers - $markedAttendanceCount;

// Current working fields

$workingFields = Site::whereHas('attendance', function ($query) use ($today) {
    $query->where('attendance_date', $today)
        ->where('status', '!=', 'A');
})->with(['attendance' => function ($query) use ($today) {
    $query->where('attendance_date', $today)->with('user');
}])->get()->map(function ($site) {
    $user = $site->attendance->first()->user ?? null;
    return [
        'site_name' => $site->site_name,
        'user_name' => $user ? $user->name : 'Unknown',
    ];
});

// Attendance Statistics

$attendanceStats = $users->map(function ($user) use ($startDate, $endDate) {
    $absentDays = $user->attendances->where('status', 'A')-
>whereBetween('attendance_date', [$startDate, $endDate])->count();

    $presentDays = $user->attendances->where('status', '!=', 'A')-
>whereBetween('attendance_date', [$startDate, $endDate])->sum(function ($attendance) {
        return is_numeric($attendance->status) ? floatval($attendance->status) : 0;
    });
});

```

```

});
return [
    'name' => $user->name,
    'absent_days' => $absentDays,
    'present_days' => $presentDays,
];
});

$mostAbsences = $attendanceStats->sortByDesc('absent_days')->first();
$bestAttendance = $attendanceStats->sortByDesc('present_days')->first();

// Payment Statistics

$paymentStats = $userDetails->sortBy('total_profit');
$lowestPending = $paymentStats->where('total_profit', '>', 0)->first();
$highestPending = $paymentStats->sortByDesc('total_profit')->first();

// Pending Actions (Using is_repaid column)

$pendingPaymentsList = Payment::where('action', 'send')
    ->where('sender_type', 'admin')
    ->whereBetween('transaction_date', [$startDate, $endDate])
    ->where('is_repaid', false)
    ->orderBy('amount', 'desc')
    ->limit(2)
    ->get()
    ->map(function ($payment) {
        return [
            'type' => 'Payment',
            'name' => $payment->recipient_name,
            'details' => $payment->details,
            'amount' => $payment->amount,
        ];
    });

```

```
});
```

```
$pendingExpenses = Expense::where('user_id', 1)
->whereBetween('expense_date', [$startDate, $endDate])
->orderBy('amount', 'desc')
->limit(2)
->get()
->map(function ($expense) {
    return [
        'type' => 'Expense',
        'name' => 'Admin',
        'details' => $expense->details,
        'amount' => $expense->amount,
    ];
});
```

```
$pendingActions = $pendingPaymentsList->merge($pendingExpenses)-
>sortByDesc('amount')->take(2);
```

```
return view('admin.dashboard', [
    'countUsers' => $this->countUsers, // Base Controller Property
    'adminIncome' => $adminIncome,
    'adminExpense' => $adminExpense,
    'pendingPayments' => $pendingPayments,
    'netBalance' => $adminIncome - $adminExpense,
    'presentCount' => $presentCount,
    'absentCount' => $absentCount,
    'pendingCount' => $pendingCount >= 0 ? $pendingCount : 0,
    'workingFields' => $workingFields,
    'mostAbsences' => $mostAbsences,
```

```
        'bestAttendance' => $bestAttendance,  
        'lowestPending' => $lowestPending,  
        'highestPending' => $highestPending,  
        'pendingActions' => $pendingActions,  
    ];  
}  
  
public function dashboard()  
{  
    return $this->index();  
}  
  
public function manage()  
{  
    return view('admin.manage');  
}  
}
```

## **2. Admin profile controller:**

<?php

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Hash;
```

```
class AdminProfileController extends Controller
{
```

```
    /**
```

```
     * Display the admin profile page.
```

```
    */
```

```
    public function index()
```

```
    {
```

```
        $admin = Auth::user();
```

```
        $adminData = [
```

```
            'name' => $admin->name,
```

```
            'email' => $admin->email,
```

```
            'contact' => $admin->contact,
```

```
            'address' => $admin->address,
```

```
            'profile_picture' => $admin->profile_picture ?? 'default.jpg',
```

```
            'role' => ucfirst($admin->role),
```

```
            'joining_date' => $admin->joining_date ? date('d-M-Y', strtotime($admin->joining_date))
```

```
            : 'Not Set',
```

```
            'setamount' => '₹' . number_format($admin->setamount ?? 300, 2),
```

```
        ];
```

```
        return view('Admin.profile.index', compact('adminData'));
    }
```

```
    /**
```

```
     * Show the edit profile form.
```

```
    */
```

```
    public function edit()
```

```
    {
```

```
        $admin = Auth::user();
```

```
        $adminData = [
```

```
            'name' => $admin->name,
```

```
            'email' => $admin->email,
```

```
            'contact' => $admin->contact,
```

```
            'address' => $admin->address,
```

```
            'profile_picture' => $admin->profile_picture ?? 'default.jpg',
```

```

        'role' => ucfirst($admin->role),
        'joining_date' => $admin->joining_date,
        'setamount' => $admin->setamount,
    ];

    return view('Admin.profile.edit', compact('adminData'));
}

/**
 * Update the admin's profile.
 */
public function update(Request $request)
{
    $admin = Auth::user();

    $validated = $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|email|unique:users,email,' . $admin->id,
        'contact' => 'required|numeric|digits:10',
        'address' => 'required|string|max:255',
        'joining_date' => 'required|date',
        'setamount' => 'required|numeric|min:0',
        'profile_picture' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        'password' => 'nullable|min:6|confirmed',
    ], [
        'name.required' => 'Name is required',
        'email.required' => 'Email is required',
        'email.email' => 'Enter a valid email',
        'email.unique' => 'This email is already registered',
        'contact.required' => 'Contact number is required',
        'contact.numeric' => 'Contact must be a number',
        'contact.digits' => 'Contact must be 10 digits',
        'address.required' => 'Address is required',
        'joining_date.required' => 'Joining date is required',
        'setamount.required' => 'Amount is required',
        'setamount.numeric' => 'Amount must be a number',
        'setamount.min' => 'Amount must be at least 300',
        'profile_picture.image' => 'Profile picture must be an image file',
        'profile_picture.mimes' => 'Allowed formats: jpeg, png, jpg, gif',
        'password.min' => 'Password must be at least 6 characters',
        'password.confirmed' => 'Password confirmation does not match',
    ]);

    if ($request->hasFile('profile_picture')) {
        // Delete old profile picture if it exists
        if ($admin->profile_picture && Storage::disk('public')->exists($admin->profile_picture))
        {
            Storage::disk('public')->delete($admin->profile_picture);
        }
    }
}

```



```

        $file = $request->file('profile_picture');
        $fileName = date('YmdHis') . '_' . $file->getClientOriginalName();
        $admin->profile_picture = $file->storeAs('profile_pictures', $fileName, 'public');
    }

    $admin->name = $validated['name'];
    $admin->email = $validated['email'];
    $admin->contact = $validated['contact'];
    $admin->address = $validated['address'];
    $admin->joining_date = $validated['joining_date'];
    $admin->setamount = $validated['setamount'];

    if ($request->filled('password')) {
        $admin->password = Hash::make($validated['password']);
    }

    $admin->save();

    return redirect()->route('admin.profile.index')->with('success', 'Profile updated
successfully');
}

/**
 * Handle logout from all devices.
 */
public function logout()
{
    Auth::logout();
    return redirect()->route('login')->with('success', 'Logged out successfully');
}
}

```

### **3. Attendance Controller:**

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Attendance;
use App\Models\User;
use App\Models\Site;
use Illuminate\Http\Request;
use Carbon\Carbon;
use Illuminate\Support\Facades\Log;

class AttendanceController extends Controller
{
    public function overview(Request $request)
    {
        $users = User::withCount([
            'attendance as total_present' => function ($query) {
                $query->where('status', '!=', 'A')->selectRaw('SUM(status)');
            },
            'attendance as total_absent' => function ($query) {
                $query->where('status', 'A');
            }
        ])->paginate(10);

        $userAttendanceSummaries = $users->mapWithKeys(function ($user) {
            return [$user->id => [
                'total_present' => $user->total_present ?? 0,
                'total_absent' => $user->total_absent ?? 0,
            ]];
        }->all());

        $selectedDate = now()->toDateString();
        return view('Admin.attendance.overview', compact('users', 'userAttendanceSummaries',
'selectedDate'));
    }

    public function index(Request $request)
    {
        $selectedDate = $request->input('attendance_date', now()->toDateString());

        if (Carbon::parse($selectedDate)->gt(now())) {
            $selectedDate = now()->toDateString();
            return redirect()->route('attendance.index', ['attendance_date' => $selectedDate])
                ->with('error', 'Cannot mark attendance for future dates.');
```

```

$users = User::all();
$sites = Site::all();

$attendanceRecords = Attendance::with('site')
    ->where('attendance_date', $selectedDate)
    ->get()
    ->keyBy('user_id');

$allMarked = $users->count() > 0 && $attendanceRecords->count() === $users->count();

return view('Admin.attendance.index', compact('users', 'sites', 'attendanceRecords',
'selectedDate', 'allMarked'));
}

public function create()
{
    $users = User::all();
    $sites = Site::all();
    return view('Admin.attendance.create', compact('users', 'sites'));
}

public function store(Request $request)
{
    $validated = $request->validate([
        'attendance_date' => 'required|date|before_or_equal:today',
        'attendance' => 'required|array',
        'attendance.*.user_id' => 'required|exists:users,id',
        'attendance.*.status' => 'required|in:0.5,1,1.5,2,A',
        'attendance.*.site_id' => [
            'nullable',
            'exists:sites,id',
            function ($attribute, $value, $fail) use ($request) {
                $userId = explode('.', $attribute)[1];
                $status = $request->input("attendance.$userId.status");
                if (in_array($status, ['0.5', '1', '1.5', '2']) && empty($value)) {
                    $fail("Site is required when status is present.");
                }
                if ($status === 'A' && !empty($value)) {
                    $fail("Site must be empty when status is absent (A).");
                }
            }
        ],
    ],
    );

    $attendanceDate = $validated['attendance_date'];

    try {
        foreach ($validated['attendance'] as $data) {
            Attendance::updateOrCreate(

```

```

        ['user_id' => $data['user_id'], 'attendance_date' => $attendanceDate],
        [
            'status' => $data['status'],
            'site_id' => $data['status'] === 'A' ? null : $data['site_id'],
        ]
    );
}
return redirect()->route('attendance.index', ['attendance_date' => $attendanceDate])
    ->with('success', 'Attendance updated successfully.');
```

```

} catch (\Exception $e) {
    Log::error('Attendance store failed', ['error' => $e->getMessage()]);
    return back()->with('error', 'Failed to update attendance.');
```

```

}
}

public function edit($id)
{
    $attendance = Attendance::with('user', 'site')->findOrFail($id);
    $users = User::all();
    $sites = Site::all();
    return view('Admin.attendance.edit', compact('attendance', 'users', 'sites'));
}

public function update(Request $request, $id)
{
    $attendance = Attendance::findOrFail($id);

    $validated = $request->validate([
        'user_id' => 'required|exists:users,id',
        'attendance_date' => 'required|date|before_or_equal:today',
        'status' => 'required|in:0.5,1,1.5,2,A',
        'site_id' => [
            'nullable',
            'exists:sites,id',
            function ($attribute, $value, $fail) use ($request) {
                $status = $request->input('status');
                if (in_array($status, ['0.5', '1', '1.5', '2']) && empty($value)) {
                    $fail("Site is required when status is present.");
                }
                if ($status === 'A' && !empty($value)) {
                    $fail("Site must be empty when status is absent (A).");
                }
            }
        ],
    ],
    );

    try {
        $attendance->update([
            'user_id' => $validated['user_id'],

```

```

        'attendance_date' => $validated['attendance_date'],
        'status' => $validated['status'],
        'site_id' => $validated['status'] === 'A' ? null : $validated['site_id'],
    );
    return redirect()->route('attendance.view', ['userId' => $validated['user_id'], 'selectedDate'
=> $validated['attendance_date']])
        ->with('success', 'Attendance updated successfully.');
```

```

    } catch (\Exception $e) {
        Log::error('Attendance update failed', ['error' => $e->getMessage()]);
        return back()->with('error', 'Failed to update attendance.');
```

```

    }
}

public function view(Request $request, $userId, $selectedDate = null)
{
    $user = User::findOrFail($userId);
    $filter = $request->input('filter', 'all');

    $query = Attendance::with('site')
        ->where('user_id', $userId)
        ->orderBy('attendance_date', 'desc');

    switch ($filter) {
        case 'week':
            $query->where('attendance_date', '>=', now()->subDays(7));
            break;
        case 'month':
            $query->where('attendance_date', '>=', now()->startOfMonth());
            break;
        case 'year':
            $query->where('attendance_date', '>=', now()->startOfYear());
            break;
    }

    $attendanceRecords = $query->paginate(10);

    $totalPresent = $attendanceRecords->where('status', '!=', 'A')->sum('status');
    $totalAbsent = $attendanceRecords->where('status', 'A')->count();

    return view('Admin.attendance.view', compact('user', 'attendanceRecords', 'totalPresent',
'totalAbsent', 'selectedDate', 'filter'));
}

public function destroy($id)
{
    $attendance = Attendance::findOrFail($id);
    $userId = $attendance->user_id;
    $attendanceDate = $attendance->attendance_date;
    $attendance->delete();
}

```

```

        return redirect()->route('attendance.view', ['userId' => $userId, 'selectedDate' =>
$attendanceDate])
        ->with('success', 'Attendance record deleted successfully.');
```

```

    }

```

```

public function bulkDelete(Request $request)
{

```

```

    $request->validate(['attendance_date' => 'required|date|before_or_equal:today']);

```

```

    $date = $request->attendance_date;

```

```

    Attendance::where('attendance_date', $date)->delete();

```

```

    return redirect()->route('attendance.index', ['attendance_date' => $date])

```

```

        ->with('success', 'All attendance records for ' . Carbon::parse($date)->format('d M Y') . '
deleted.');
```

```

    }

```

```

}

```

#### 4. Expense Controller:

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Expense;
use App\Models\User;
use Illuminate\Http\Request;
use Carbon\Carbon;
use Illuminate\Support\Facades\Auth;

class ExpenseController extends Controller
{
    public function index()
    {
        $expenses = Expense::with('user')->latest()->get();
        return view('Admin.expenses.index', [
            'expenses' => $expenses,
        ]);
    }

    public function create()
    {
        // Admin ke liye saare users, non-admin ke liye khud ka data already form mein handle
        $users = Auth::user()->role === 'admin' ? User::all() : collect([Auth::user()]);
        return view('Admin.expenses.create', [
            'users' => $users,
            'countUsers' => $this->countUsers,
        ]);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'item_name' => 'required|string|max:255',
            'name' => 'required|string|max:255', // Paid by
            'method' => 'required|in:cash,online',
            'expense_date' => 'required|date',
            'amount' => 'required|numeric|min:0',
            'user_id' => 'required|exists:users,id',
        ]);

        // Ensure non-admin can only assign expense to themselves
        if (Auth::user()->role !== 'admin' && $request->user_id != Auth::id()) {
```

```

        return redirect()->back()->withErrors(['user_id' => 'You can only assign expenses to
yourself.']);
    }

    Expense::create($validated);

    return redirect()->route('expenses.index')->with('success', 'Expense added successfully.');
```

}

```

public function view($id)
{
    $expense = Expense::with('user')->findOrFail($id);
    return view('Admin.expenses.view', [
        'expense' => $expense,
        'countUsers' => $this->countUsers,
    ]);
}

public function edit($id)
{
    $expense = Expense::findOrFail($id);
    $users = Auth::user()->role === 'admin' ? User::all() : collect([Auth::user()]);
    return view('Admin.expenses.edit', [
        'expense' => $expense,
        'users' => $users,
        'countUsers' => $this->countUsers,
    ]);
}

public function update(Request $request, $id)
{
    $expense = Expense::findOrFail($id);

    $validated = $request->validate([
        'item_name' => 'required|string|max:255',
        'name' => 'required|string|max:255', // Paid by
        'method' => 'required|in:cash,online',
        'expense_date' => 'required|date',
        'amount' => 'required|numeric|min:0',
        'user_id' => 'required|exists:users,id',
    ]);

    // Ensure non-admin can only update their own expenses
    if (Auth::user()->role !== 'admin' && $request->user_id !== Auth::id()) {
        return redirect()->back()->withErrors(['user_id' => 'You can only update expenses
assigned to yourself.']);
    }

    $expense->update($validated);

```



```

    return redirect()->route('expenses.index')->with('success', 'Expense updated successfully.');
```

```

}

public function destroy($id)
{
    $expense = Expense::findOrFail($id);

    // Restrict non-admin from deleting others' expenses
    if (Auth::user()->role !== 'admin' && $expense->user_id !== Auth::id()) {
        return redirect()->back()->withErrors(['error' => 'You can only delete your own
expenses.']);
    }

    $expense->delete();

    return redirect()->route('expenses.index')->with('success', 'Expense deleted successfully.');
```

```

}
}

```

## **5. Overview Controller:**

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class OverviewController extends Controller
{
    public function index(Request $request)
    {
        return parent::overview($request);
    }
}
```

## **6. Payment Controller:**

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Payment;
use App\Models\Site;
use App\Models\User;
use Illuminate\Http\Request;
use Carbon\Carbon;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Auth;

class PaymentController extends Controller
{
    public function index()
    {
        $payments = Payment::latest()->get();
        return view('Admin.payments.index', [
            'payments' => $payments,
        ]);
    }

    public function send(Request $request)
    {
        if ($request->isMethod('get')) {
            // Admin aur users dono ko fetch karo
            $users = User::with([
                'attendances' => fn($query) => $query->where('status', '!=', 'A'),
                'expenses'
            ])->get();

            [$startDate, $endDate] = $this->getDateRange('all');
            $userDetails = $this->calculateUserDetails($users, $startDate, $endDate);

            // Remaining balances ko name ke against set karo
            $remainingBalances = $userDetails->pluck('total_profit', 'name')->all();

            return view('Admin.payments.send', [
                'users' => $users,
                'remainingBalances' => $remainingBalances,
                'countUsers' => $this->countUsers,
            ]);
        }

        $validated = $request->validate([
            'action' => 'required|in:send',
        ];
```

```

'recipient_id' => 'required|exists:users,id',
'payment_type' => 'required|in:earned,advance',
'details' => 'required|string',
'method' => 'required|in:cash,online',
'transaction_date' => 'required|date',
'amount' => 'required|numeric|min:0',
'sender_type' => 'required|in:admin',
'recipient_type' => 'required|in:user',
'is_out_of_pocket' => 'nullable|boolean',
]);

$recipient = User::findOrFail($request->recipient_id);
[$startDate, $endDate] = $this->getDateRange('all');
$userDetails = $this->calculateUserDetails(collect([$recipient]), $startDate,
$endDate);
$remainingAmount = $userDetails->first()['total_profit'];

if ($request->payment_type === 'earned' && $request->amount >
$remainingAmount) {
    return redirect()->back()
        ->withErrors(['amount' => "Amount exceeds the user's remaining earned
amount of ₹" . number_format($remainingAmount, 2) . ". Consider marking this as an
advance/loan instead."])
        ->withInput();
}

$paymentData = [
    'action' => $request->action,
    'sender_type' => $request->sender_type,
    'sender_name' => Auth::user()->name ?? 'Administrator',
    'recipient_type' => $request->recipient_type,
    'recipient_name' => $recipient->name,
    'payment_type' => $request->payment_type,
    'details' => $request->details,
    'method' => $request->method,
    'transaction_date' => Carbon::parse($request->transaction_date),
    'amount' => $request->amount,
    'user_id' => $request->recipient_id,
    'is_out_of_pocket' => $request->boolean('is_out_of_pocket', false),
    'is_repaid' => false,
];

Log::info('Payment data before creation:', $paymentData);
Payment::create($paymentData);

return redirect()->route('payments.index')->with('success', 'Payment sent
successfully.');
```

```

public function receive(Request $request)
{
    if ($request->isMethod('get')) {
        $users = User::where('role', 'user')->get();
        $contractors = Site::select('contractor_name')
            ->distinct()
            ->pluck('contractor_name')
            ->all();

        return view('Admin.payments.receive', [
            'users' => $users,
            'contractors' => $contractors,
            'countUsers' => $this->countUsers,
        ]);
    }

    $validated = $request->validate([
        'action' => 'required|in:receive',
        'sender_type' => 'required|in:user,contractor',
        'sender_id' => 'nullable|required_if:sender_type,user|exists:users,id',
        'sender_name' => 'nullable|required_if:sender_type,contractor|string|max:255',
        'details' => 'required|string',
        'method' => 'required|in:cash,online',
        'transaction_date' => 'required|date',
        'amount' => 'required|numeric|min:0',
        'recipient_type' => 'required|in:admin',
    ]);

    $sender_name = $request->sender_type === 'user'
        ? User::findOrFail($request->sender_id)->name
        : $request->sender_name;

    $paymentData = [
        'action' => $request->action,
        'sender_type' => $request->sender_type,
        'sender_name' => $sender_name,
        'recipient_type' => $request->recipient_type,
        'recipient_name' => Auth::user()->name ?? 'Administrator',
        'payment_type' => 'earned',
        'details' => $request->details,
        'method' => $request->method,
        'transaction_date' => Carbon::parse($request->transaction_date),
        'amount' => $request->amount,
        'user_id' => $request->sender_type === 'user' ? $request->sender_id : null,
        'is_out_of_pocket' => false,
        'is_repaid' => false,
    ];

    Log::info('Payment data before creation:', $paymentData);
}

```

```

        Payment::create($paymentData);

        return redirect()->route('payments.index')->with('success', 'Payment received
successfully.');
```

```

    }

    public function view($id)
    {
        $payment = Payment::findOrFail($id);
        return view('Admin.payments.view', [
            'payment' => $payment,
            'countUsers' => $this->countUsers,
        ]);
    }

    public function edit($id)
    {
        $payment = Payment::findOrFail($id);
        $users = User::all();
        return view('Admin.payments.edit', [
            'payment' => $payment,
            'users' => $users,
            'countUsers' => $this->countUsers,
        ]);
    }

    public function update(Request $request, $id)
    {
        $payment = Payment::findOrFail($id);

        $validated = $request->validate([
            'action' => 'required|in:send,receive',
            'sender_type' => 'required|in:user,contractor,admin',
            'sender_id' => 'nullable|required_if:sender_type,user|exists:users,id',
            'sender_name' => 'nullable|required_if:sender_type,contractor|string|max:255',
            'recipient_type' => 'required|in:user,admin',
            'recipient_id' => 'nullable|required_if:recipient_type,user|exists:users,id',
            'recipient_name' => 'nullable|string|max:255',
            'payment_type' => 'required|in:earned,advance,repayment',
            'details' => 'required|string',
            'method' => 'required|in:cash,online',
            'transaction_date' => 'required|date',
            'amount' => 'required|numeric|min:0',
            'is_out_of_pocket' => 'nullable|boolean',
        ]);

        $sender_name = $request->sender_type === 'user'
            ? User::findOrFail($request->sender_id)->name

```

```

        : ($request->sender_type === 'admin' ? Auth::user()->name ?? 'Administrator' :
$request->sender_name);
        $recipient_name = $request->recipient_type === 'user'
        ? User::findOrFail($request->recipient_id)->name
        : ($request->recipient_type === 'admin' ? Auth::user()->name ?? 'Administrator' :
$request->recipient_name);

        $paymentData = [
            'action' => $request->action,
            'sender_type' => $request->sender_type,
            'sender_name' => $sender_name,
            'recipient_type' => $request->recipient_type,
            'recipient_name' => $recipient_name,
            'payment_type' => $request->payment_type,
            'details' => $request->details,
            'method' => $request->method,
            'transaction_date' => Carbon::parse($request->transaction_date),
            'amount' => $request->amount,
            'user_id' => $request->sender_type === 'user' ? $request->sender_id : ($request-
>recipient_type === 'user' ? $request->recipient_id : null),
            'is_out_of_pocket' => $request->boolean('is_out_of_pocket', false),
        ];

        Log::info('Payment data before update:', $paymentData);
        $payment->update($paymentData);

        return redirect()->route('payments.index')->with('success', 'Payment updated
successfully.');
```

```

    }

    public function destroy($id)
    {
        $payment = Payment::findOrFail($id);
        $payment->delete();

        return redirect()->route('payments.index')->with('success', 'Payment deleted
successfully.');
```

```

    }
}

```

## User :

### User Dashboard:

<?php

```
namespace App\Http\Controllers\User;

use App\Http\Controllers\Controller;
use App\Models\Payment;
use App\Models\Attendance;
use App\Models\Expense;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class UserDashboardController extends Controller
{
    public function index()
    {
        $user = Auth::user();

        // Attendance Records (Latest 5)
        $attendanceRecords = Attendance::where('user_id', $user->id)
            ->orderBy('attendance_date', 'desc')
            ->limit(5)
            ->get();

        // Payment Records (Latest 5)
        $paymentRecords = Payment::where('user_id', $user->id)
            ->orderBy('transaction_date', 'desc')
            ->limit(5)
            ->get();

        // Expense Records (Latest 5)
        $expenseRecords = Expense::where('user_id', $user->id)
            ->orderBy('expense_date', 'desc')
            ->limit(5)
            ->get();

        // Summary Calculations
        $totalAttendance = Attendance::where('user_id', $user->id)
            ->where('status', '!=', 'A')->get()
            ->sum(function ($record) {
                return (float) $record->status;
            });
        // dd($totalAttendance);
        $totalAbsent = Attendance::where('user_id', $user->id)
            ->where('status', 'A')
```



```

        ->count(); // Count of absent days
    // dd($totalAbsent);
    $totalReceived = Payment::where('user_id', $user->id)
        ->where('action', 'receive')
        ->sum('amount');
    // dd($totalReceived);
    $totalSent = Payment::where('user_id', $user->id)
        ->where('action', 'send')
        ->sum('amount');
    // dd($totalSent);
    $totalWorkingAmount = $totalAttendance * ($user->setamount ?? 0); // Working days *
    daily rate
    $totalExpense = Expense::where('user_id', $user->id)->sum('amount');
    $totalProfitMinusExpense = $totalWorkingAmount + $totalReceived - $totalSent -
    $totalExpense;

    $summary = [
        'total_attendance' => $totalAttendance,
        'total_absent' => $totalAbsent,
        'total_received' => $totalReceived,
        'total_working_amount' => $totalWorkingAmount,
        'total_expense' => $totalExpense,
        'total_profit_minus_expense' => $totalProfitMinusExpense,
    ];

    return view('user.dashboard', compact('user', 'attendanceRecords', 'paymentRecords',
    'expenseRecords', 'summary'));
}

public function attendance(Request $request)
{
    $user = Auth::user();
    $month = $request->input('month', now()->month);
    $year = $request->input('year', now()->year);

    // Get all days in the selected month
    $daysInMonth = \Carbon\Carbon::create($year, $month)->daysInMonth;
    $attendanceRecords = Attendance::where('user_id', $user->id)
        ->whereYear('attendance_date', $year)
        ->whereMonth('attendance_date', $month)
        ->get();

    // Prepare data for the graph
    $attendanceData = [
        'dates' => [],
        'statuses' => []
    ];
    $totalAttendance = 0;
    $totalAbsences = 0;

```

```

for ($day = 1; $day <= $daysInMonth; $day++) {
    $date = \Carbon\Carbon::create($year, $month, $day)->format('Y-m-d');
    $attendanceData['dates'][] = $day;
    $record = $attendanceRecords->firstWhere('attendance_date', $date);

    if ($record) {
        $attendanceData['statuses'][] = $record->status;
        if ($record->status !== 'A') {
            $totalAttendance += (float)$record->status; // Sum numeric values
        } else {
            $totalAbsences++;
        }
    } else {
        $attendanceData['statuses'][] = 'A'; // Absent by default
        $totalAbsences++;
    }
}

return view('user.attendance', compact('attendanceRecords', 'attendanceData',
'totalAttendance', 'totalAbsences'));
}
}

```

## **User Manage Controller:**

```
<?php

namespace App\Http\Controllers\User;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class UserManagerController extends Controller
{
    public function manage(){
        return view('user.manage');
    }
}
```

# **CHAPTER 11**

## **TESTING**

### **10.1 Introduction**

Testing is a critical phase in the development lifecycle of the Salary and Attendance Management System. It ensures that the system performs as expected, meets all functional and non-functional requirements, and is free from defects. Testing helps in identifying errors, gaps, or missing requirements in contrast to the actual requirements. This process guarantees system reliability, performance, and user satisfaction before deployment.

The testing phase involved both manual and automated methods to validate each module—such as employee registration, attendance logging, leave management, and salary generation.

### **10.2 Types of Testing**

#### **10.2.1 Unit Testing**

Each module, like attendance entry, leave application, and salary calculation, was tested individually to ensure correct functionality. This helped detect bugs at the code level early in the development phase.

#### **10.2.2 Integration Testing**

After unit testing, modules were integrated and tested to ensure they work together seamlessly. For example, integration between the attendance module and the salary calculation module was thoroughly verified.

#### **10.2.3 System Testing**

System testing was conducted on the entire application to ensure that all functionalities perform correctly under various conditions. This type of testing covered all use-case scenarios and verified that the system meets the requirements.

#### **10.2.4 User Acceptance Testing (UAT)**

Real users (employees and admin) interacted with the system to validate that it is user-friendly, meets business goals, and behaves as expected in a real-world environment. Feedback was gathered to make final refinements.

### **10.2.5 Performance Testing**

This involved testing how the system performs under load—such as processing salary for 100+ employees at once or logging 50+ attendance entries simultaneously—to ensure scalability and responsiveness.

## **CHAPTER 12**

### **IMPLEMENTATION OF NEED**

The need for implementing a Salary and Attendance Management System arises due to the inefficiencies, errors, and time-consuming nature of manual systems. Below is a structured explanation:

#### **1. Accuracy and Error Reduction**

Manual salary calculations often result in errors due to human oversight. Automating attendance tracking and salary calculations ensures:

- Accurate payroll processing
- Elimination of manual calculation mistakes
- Correct leave deductions and overtime compensation

#### **2. Time Efficiency**

HR and accounts departments spend a significant amount of time managing spreadsheets or paper records. A digital system:

- Automates repetitive tasks
- Reduces time spent on monthly payroll
- Enables quick generation of reports and payslips

#### **3. Transparency and Accessibility**

Employees often lack visibility into their attendance records and salary structure. This system:

- Allows employees to view their attendance and payslips
- Improves transparency and trust
- Enables remote access (if implemented as a web app)

#### **4. Compliance and Record-Keeping**

Legal and organizational policies require proper documentation of employee attendance and salary:

- Maintains accurate logs for audits
- Helps in statutory compliance (PF, ESI, Tax etc.)
- Generates reports for government or internal reviews

## **5. Scalability**

As an organization grows, manual systems become impractical:

- The system can handle hundreds of employees efficiently
- Scalable for multi-department or multi-location companies

## **6. Integration with Other Modules**

Can be integrated with:

- Leave management system
- Biometric devices
- Payroll systems or accounting software

# CHAPTER 13

## DISCUSSIONS

### 1. Functional Importance

The system solves critical HR problems:

- Manual attendance tracking is error-prone and time-consuming.
- Salary processing based on Excel or paper leads to miscalculations.
- Employees often have no clear visibility into their attendance or salary deductions.

With this system:

- Admins can automate repetitive tasks like calculating pay, managing leave, and generating reports.
- Employees can access their own records securely.
- The company ensures transparency, reduces payroll errors, and avoids compliance risks.

### 2. System Design Considerations

#### A. Scalability

If the company scales from 10 to 1,000 employees, the system should:

- Support bulk attendance import (via biometric or CSV)
- Use pagination and filters in UI
- Optimize queries (e.g., indexed emp\_id, monthly salary partitioning)

#### B. Security

Security is essential because salary and attendance data is sensitive:

- Passwords must be hashed (e.g., using bcrypt)
- Use access control (employees can't see others' data)
- Use HTTPS for web-based apps
- Role-based dashboard (admin vs employee)



## **C. Flexibility**

Different companies have:

- Different salary structures (fixed, variable, hourly)
- Various leave policies
- Overtime/bonus rules

So, the system should allow admins to customize settings—define working days, deduction rules, leave types, and holidays.

# CHAPTER 14

## LIMITATIONS

Here are the **limitations** of a typical **Salary and Attendance Management System**, especially when implemented at a basic or intermediate level.

### 1. Limited Biometric Integration

- Most basic systems don't integrate with biometric attendance devices.
- Manual data entry or CSV import is required, which can lead to delays or errors.

### 2. Rigid Salary Calculation

- Salary logic is often hardcoded or simple:
  - Cannot handle complex salary structures (incentives, performance bonuses, shift differentials).
  - Difficult to adjust for retroactive changes (e.g., previous month corrections).

### 3. No Taxation & Statutory Deduction Handling

- Lacks modules for:
  - Income tax (TDS)
  - Provident fund (PF)
  - Employee State Insurance (ESI)
  - Other compliance-related calculations

### 4. Scalability Issues

- Designed for small organizations (e.g., < 200 employees).
- Performance may degrade with:

- Large employee datasets
- Multiple departments or branches
- High transaction volume

## **5. Limited Security Features**

- Basic authentication only (e.g., no 2FA, no session timeout).
- Role management may not be fully secure.
- Sensitive salary data may be exposed if not encrypted or accessed by the wrong role.

## **6. Lack of Mobile Access**

- Many systems are web or desktop-only.
- No mobile app or responsive design for access on phones.

## **7. No Real-Time Notifications**

- No automatic alerts for:
  - Low attendance
  - Late marks
  - Upcoming salary credit
  - Missing attendance data

## **8. Basic Reporting**

- Limited reporting and analytics:
  - Cannot filter by custom date ranges
  - No graphical dashboards
  - No export to advanced formats like Excel with charts

## **CHAPTER 15**

### **CONCLUSION**

The Salary and Attendance Management System is a powerful and essential tool for modern organizations seeking to streamline HR processes, reduce errors, and enhance productivity. By automating attendance tracking and salary calculations, the system eliminates the need for manual record-keeping, minimizes payroll discrepancies, and ensures timely and accurate salary disbursements.

This system not only facilitates effective employee management but also enhances transparency and accountability by providing real-time access to attendance records and payslips. With features like role-based access, centralized data storage, and customizable salary structures, it offers both administrative efficiency and employee satisfaction.

While the current version may have limitations—such as lack of biometric integration, advanced taxation handling, or mobile accessibility—it serves as a robust foundation for future development. Enhancements like mobile app support, integration with payroll software, and advanced analytics can significantly expand its functionality.

Overall, the project successfully demonstrates how technology can be leveraged to improve routine HR functions and lays the groundwork for building a more scalable and secure HR automation solution.

# CHAPTER 16

## References

### 1. Books & Academic Material

- Laudon, K. C., & Laudon, J. P. (2018). *Management Information Systems: Managing the Digital Firm* (15th ed.). Pearson Education.
- Shelly, G. B., & Rosenblatt, H. J. (2012). *Systems Analysis and Design*. Cengage Learning.

### 2. Web-Based Resources

- W3Schools. (n.d.). *PHP Tutorial*. <https://www.w3schools.com/php>
- GeeksforGeeks. (n.d.). *DBMS and SQL Tutorials*. <https://www.geeksforgeeks.org/dbms>
- TutorialsPoint. (n.d.). *Software Engineering - Project Management*. [https://www.tutorialspoint.com/software\\_engineering](https://www.tutorialspoint.com/software_engineering)

### 3. Technology Documentation

- MySQL Documentation. <https://dev.mysql.com/doc>
- PHP Manual. <https://www.php.net/manual/en>
- Bootstrap Framework. <https://getbootstrap.com>

### 4. Open Source & GitHub Examples

- GitHub. (n.d.). *Payroll Management System Projects*. <https://github.com> – Used for understanding real-world code structures and open-source HRM systems