

HRPULSE

A Project Report for Industrial Training and Internship

submitted by

ROHIT PRASAD

NILOY DEY

KUNDAN KUMAR

NITISH YADAV

In the partial fulfilment of the award of the degree of

B.Tech

in the

ELECTRONICS AND COMMUNICATION ENGINEERING

Of

B.P.PODDAR INSTITUTE OF MANAGEMENT AND TECHNOLOGY



at

Ardent Computech Pvt. Ltd.





CERTIFICATE FROM SUPERVISOR

This is to certify that **NILOY DEY, ROHIT PRASAD, KUNDANKUMAR, NITISH YADAV** , 11500322037, 11500322040, 11500322032, 11500322022 have completed the project titled HRPULSE under my supervision during the period from 04-07-2025 to 16-8-2025 which is in partial fulfilment of requirements for the award of the **B.Tech** degree and submitted to the Department of **ELECTRONICS AND COMMUNICATION ENGINEERING** of **B.P. PODDAR INSTITUTE OF MANAGEMENT AND TECHNOLOGY**.

_____ **Signature of the Supervisor**

Date: 16-8-2025

Name of the Project Supervisor: Subhojit Santra





BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

“HR PULSE” is the Bonafide wo

<i>Name of the student</i>	<i>Signature</i>
NILOY DEY	<i>Niloy Dey</i>
ROHIT PRASAD	<i>Rohit Prasad</i>
KUNDAN KUMAR	<i>Kundan Kumar</i>
NITISH YADAV	<i>Nitish Yadav</i>

SIGNATURE

Name :

PROJECT MENTOR

SIGNATURE

Name:

EXAMINERS

Ardent Original Seal



ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, **Subhojit Santra** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

CONTENT PAGE

1. COMPANY PROFILE	8
2. INTRODUCTION	9
2A.OBJECTIVE	10
2B.SCOPE	11
3. SYSTEM ANALYSIS	12
3A.IDENTIFICATION OF NEED	13
3B.FEASIBILITY STUDY	14
3C.WORKFLOW	15
3D .STUDY OF THE SYSTEM	17
3E.INPUT AND OUTPUT	19
3F.SOFTWARE REQUIREMENT SPECIFICATIONS	20
3G.SOFTWARE ENGINEERING PARADIGM APPLIED	21
4. SYSTEM DESIGN	22
4A. DATA FLOW DIAGRAM	23
4B.SEQUENCE DIAGRAM	25
4C.USE CASE DIAGRAM	26
4D.SCHEMA DIAGRAM	27
5. UI SNAPSHOT	29
❖ FRONTEND :	30
1)Login page	30

✓ CODE-----	30
2)Admin dashboard-----	34
✓ CODE-----	34
3)Manage employees-----	39
✓ CODE-----	39
4)add new employee-----	41
✓ CODE-----	41
5)salary history-----	44
✓ CODE-----	44
6)manage history-----	49
✓ CODE-----	49
❖ BACKEND:-----	55
1)package.json-----	56
2)server.js-----	57
6. CONCLUSION-----	59
7. FUTURE SCOPE & FURTHER ENHANCEMENTS-----	60
8. BIBLIOGRAPHY-----	61

1. COMPANY PROFILE

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO 9001:2015 certified Software Development and Training Company based in India. Operating independently since 2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global outreach and service offerings.

ARDENT Technologies

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core competencies lie in the development of customized application software, encompassing end-to-end solutions including system analysis, design, development, implementation, and training. The company also provides expert consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies, resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

ARDENT Collaborations

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt. Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt. Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing students may qualify for stipends, scholarships, and additional benefits based on performance and mentor recommendations.

Associations and Accreditations

ARDENT (Ardent Computech Pvt. Ltd.) is affiliated with the National Council of Vocational Training (NCVT) under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment, Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is dedicated to bridging the gap between academic knowledge and industry skills through innovative training programs.

2. INTRODUCTION

HRPulse is a comprehensive job portal web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), designed to bridge the gap between job seekers and employers. This platform provides a seamless interface where users can create profiles, upload resumes, search and apply for jobs, and receive notifications about relevant opportunities. Employers can post job openings, manage applications, and shortlist candidates efficiently.

The project emphasizes real-time data management, user-friendly navigation, and responsive design, making it accessible across devices. By integrating modern web technologies, HRPulse ensures secure authentication, dynamic content rendering, and smooth interaction between the front-end and back-end.

In essence, HRPulse is a robust solution for managing recruitment processes digitally, helping organizations find the right talent while empowering job seekers to discover suitable career opportunities.

2A.OBJECTIVE

The primary objective of the HRPulse project is to develop a seamless and efficient platform that connects job seekers with potential employers, simplifying the recruitment and job search process. It aims to enable job seekers to create profiles, upload resumes, and apply for relevant jobs easily, while allowing employers to post job openings, manage applications, and shortlist candidates effectively. By leveraging the MERN stack, the project ensures a secure, dynamic, and responsive web application that offers real-time data management and smooth user interaction. Additionally, HRPulse focuses on enhancing user experience through an intuitive interface accessible across devices and automating recruitment tasks to reduce manual effort, thereby improving the overall efficiency of the hiring process.

2B.SCOPE

- To provide a **digital platform connecting job seekers and employers** efficiently.
- To allow **job seekers** to create profiles, upload resumes, and apply for relevant jobs easily.
- To enable **employers** to post job vacancies, track applications, and shortlist candidates effectively.
- To develop a **secure, scalable, and dynamic web application** using the MERN stack.
- To ensure a **responsive and user-friendly interface** accessible across devices.
- To **automate recruitment processes**, reducing manual effort for both employers and candidates.
- To allow **future enhancements** such as AI-based job recommendations, analytics, and advanced search/filtering options.

3. SYSTEM ANALYSIS

3A.IDENTIFICATION OF NEED

System analysis is a crucial phase in the development of our project HRPulse, involving the creation of an efficient and user-friendly job portal. With the rapid growth of the digital recruitment industry and the increasing demand for streamlined hiring processes, there is a growing need for a flexible, efficient, and interactive platform that connects job seekers and employers. Traditional recruitment methods have limitations in terms of time, accessibility, and manual effort. The HRPulse project addresses these challenges by providing a scalable digital solution for candidates and recruiters.

Key Needs Identified:

1. **Efficient Job Search and Application:**
 - Job seekers require a platform to easily search for jobs that match their skills and qualifications.
 - Users need the flexibility to apply for multiple jobs quickly and track the status of their applications.
2. **Streamlined Recruitment for Employers:**
 - Employers need an efficient way to post vacancies, manage applications, and shortlist candidates.
 - Automation of candidate management reduces manual effort and speeds up the hiring process.
3. **Centralized and Secure Data Management:**
 - Both job seekers and employers need a secure system to store profiles, resumes, and job postings.
 - Real-time updates and notifications help maintain effective communication between applicants and recruiters.
4. **Future Expansion and Advanced Features:**
 - The system should allow for future enhancements such as AI-based job recommendations, analytics, and advanced filtering options.
 - A scalable architecture ensures the platform can handle increasing users and job postings efficiently.

By addressing these needs, HRPulse aims to bridge the gap between job seekers and employers, making recruitment more efficient, accessible, and user-friendly.

3B.FEASIBILITY STUDY

Feasibility study is an important phase in the development of the **HRPulse project**, as it helps determine whether the project can be successfully developed and implemented within the constraints of time, cost, and technology. The study evaluates different aspects of the project to ensure its practicality and effectiveness in solving the identified needs of both job seekers and employers.

Key Aspects of Feasibility:

1. **Technical Feasibility:**
 - The HRPulse project uses the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, which supports dynamic web applications and efficient data management.
 - Cloud deployment and responsive design make the platform accessible across devices, ensuring smooth performance and scalability.
2. **Economic Feasibility:**
 - The project development cost is minimal as it leverages open-source technologies.
 - The platform reduces manual recruitment efforts, saving time and operational costs for employers in the long run.
3. **Operational Feasibility:**
 - HRPulse is designed to be user-friendly, enabling both job seekers and employers to navigate the portal easily.
 - Automated workflows, such as resume submission, job posting, and application tracking, improve operational efficiency.
4. **Schedule Feasibility:**
 - The development can be completed within a reasonable timeframe using agile methodologies, allowing for iterative testing and improvements.
 - Clear project milestones ensure timely delivery and functional deployment.
5. **Legal and Security Feasibility:**
 - Data privacy and secure authentication mechanisms are incorporated to protect user information.
 - Compliance with standard digital regulations ensures safe and lawful operation of the platform.

By analyzing these factors, it is evident that the **HRPulse project is feasible**, technically, economically, and operationally, making it a practical solution to modernize the recruitment process and efficiently connect job seekers with employers

3C.WORKFLOW

This document plays a vital role in the Software Development Life Cycle (SDLC) as it describes the complete requirements and architecture of the HRPulse job portal system. It is intended for use by the developers and serves as a reference during the testing phase. Any changes made to the requirements in the future must go through a formal change approval process to maintain system integrity.

The Waterfall Model was the first SDLC approach widely introduced in software engineering. Also known as a linear-sequential life cycle model, it is simple to understand and use. In this model, each phase must be completed before the next phase begins, and there is no overlapping of phases. The Waterfall Model ensures a sequential flow of the software development process, where the outcome of one phase acts as the input for the next.

Iterative Waterfall Model:

The Iterative Waterfall Model is a variation of the traditional Waterfall approach. While maintaining the linear structure, it allows small, manageable cycles to revisit and refine phases before progressing to the next stage. This model combines the systematic structure of Waterfall with the flexibility of iterative development.

Sequential Phases in Iterative Waterfall Model for HRPulse:

1. Requirement Gathering and Analysis:
 - All possible requirements of the HRPulse system are captured and documented in a requirement specification document.
 - Functional and non-functional requirements are clearly identified for both job seekers and employers.
2. System Design:
 - The requirement specifications are studied to prepare the system design.
 - This phase specifies hardware and software requirements and defines the overall architecture of HRPulse, including the MERN stack setup and database structure.
3. Implementation:
 - The system is developed in small units (modules), such as user registration, job posting, application tracking, and admin panel.
 - Each unit is independently tested for functionality (Unit Testing) before integration.
4. Integration and Testing:
 - All modules are integrated into a complete HRPulse system.
 - The integrated system is tested for faults, bugs, and overall performance.
5. Deployment of the System:
 - After functional and non-functional testing, the HRPulse platform is deployed on a cloud server for real-world use by job seekers and employers.
6. Maintenance:
 - Post-deployment, issues arising in the live environment are addressed through

patches.

- Enhancements, such as AI-based job recommendations or advanced analytics, can be added to improve functionality.

Advantages of Iterative Waterfall Model in HRPulse:

1. Flexibility: Iterations permit adjustments based on feedback from users and stakeholders.
2. Early Delivery: Partial modules can be tested and delivered incrementally.
3. Risk Management: Issues are identified and addressed early in the development process.

Disadvantages:

1. Increased Complexity: Iterative cycles can make management more complex.
2. Potential for Scope Creep: Frequent iterations may result in unplanned changes to requirements.
3. Resource Intensive: Continuous revisiting of phases may require additional resources.

Applications:

The Iterative Waterfall Model is suitable for the HRPulse project as requirements may evolve during development. It allows partial delivery of modules for stakeholder feedback, helping refine the system effectively while ensuring that the final product is robust, scalable, and user-friendly.

3D .STUDY OF THE SYSTEM

The HRPulse job portal system is designed as a modular web application to efficiently manage interactions between job seekers and employers. The system is divided into several modules, each serving a specific purpose to ensure a smooth and user-friendly experience.

Modules:

1. Register:
 - User Register: Job seekers can register to create profiles, upload resumes, and apply for jobs.
 - Employer/Admin Register: Employers or admin users can register to post job vacancies and manage applications.
2. Verify Account:
 - During login or registration, an OTP (One-Time Password) is sent to the user's email to verify the authenticity and validity of the account.
3. Login:
 - User Login: Registered job seekers can log in to browse job listings, apply for jobs, and manage their profiles.
 - Admin/Employer Login: Admins or employers log in to manage job postings, track applications, and handle user roles.
4. Home:
 - This page provides an overview of the portal, including feedback from users and recent job postings.
5. Jobs/Courses (Job Listings Module):
 - User Interface: Job seekers can view available job opportunities, search for jobs by keywords, and filter based on preferences.
 - Admin Interface: Admins can add, update, or remove job postings to keep the portal up-to-date.
6. Applications/Details (Lectures Module Equivalent):
 - User Interface: Users can track their job applications and view details of applied positions.
 - Admin Interface: Admins can manage applications, view candidate profiles, and update the status of applications.

7. About:

- This page provides information about the HRPulse development team and the objectives of the platform.

8. Account:

- User Interface:
 - My Profile: Users can view and update personal information and resume details.
 - Dashboard: Displays the status of applications, shortlisted jobs, and relevant notifications.
- Admin Interface:
 - Admin Profile: Admins can view and manage their account information.
 - Admin Dashboard: Admins can manage job postings, monitor applications, and control user roles.

The modular structure of HRPulse ensures efficient management, clear separation of functionalities, and a user-friendly experience for both job seekers and employers. Each module is designed to handle specific tasks while maintaining seamless integration across the platform.

3E.INPUT AND OUTPUT

The HRPulse system is designed to handle inputs from both **job seekers** and **employers/admins**, and generate appropriate outputs to facilitate seamless interaction and efficient management of recruitment processes.

Input:

1. Users can log in by entering their **credentials** (email/username and password) on the login page.
2. Job seekers provide additional information such as **profiles, resumes, and application details**.
3. Employers or admins input **job postings, candidate information, and updates** to the system.

Output:

1. Job seekers can **view available job opportunities**, apply for jobs, and track the status of their applications.
2. Employers/admins can **access a centralized dashboard** containing user details, manage job postings, update application statuses, and oversee overall system operations.
3. The system generates **real-time notifications and feedback**, ensuring users and employers remain informed about updates, approvals, or new opportunities.

The HRPulse platform efficiently manages inputs and outputs, ensuring a **smooth workflow**, accurate data processing, and **enhanced user experience** for both job seekers and employers.

3F. Software Requirement Specifications (SRS)

The **Software Requirement Specification (SRS)** provides a comprehensive overview of the HRPulse project. It describes the functional and non-functional requirements needed to develop a robust and efficient job portal. The SRS ensures that the developers have a clear understanding of the system, its components, and the expected outcomes. It is prepared after thorough communication with the project team and stakeholders.

The **developer is responsible for:**

- Developing the HRPulse system to meet all requirements specified in the SRS.
- Demonstrating and installing the system for end-users after successful acceptance testing.
- Providing a detailed **user manual** explaining system interfaces and operations.

Functional

Requirements:

A. User Registration and Authentication:

1. Users (job seekers and employers) should be able to **create accounts securely**.
2. The system should **authenticate users** and manage login sessions effectively.

B. Browse and Search Jobs:

1. Users should be able to **browse and search** for job postings using filters like location, role, and skills.

C. Job Listings Display:

1. Job seekers should see detailed job descriptions, company information, and application deadlines.
2. Employers should be able to view applicant details and manage job postings efficiently.

Hardware Requirements:

1. Computer with **Intel i3 Processor** or higher
2. **8 GB RAM**
3. **SSD Storage** for faster read/write operations

Software Requirements:

1. **Windows 11 OS**
2. **Visual Studio Code** (for development)
3. **MongoDB Atlas** (cloud database for backend storage)
4. **Node.js and Express.js** for server-side development
5. **React.js** for front-end development

3G. Software Engineering Paradigm Applied

A **software engineering paradigm** refers to the methodology and systematic approach used while designing and developing software. It provides a framework for ensuring that the software meets both functional and reliability requirements. In the HRPulse project, the **Iterative Waterfall Model** has been applied as the primary development paradigm, combined with modern web development practices.

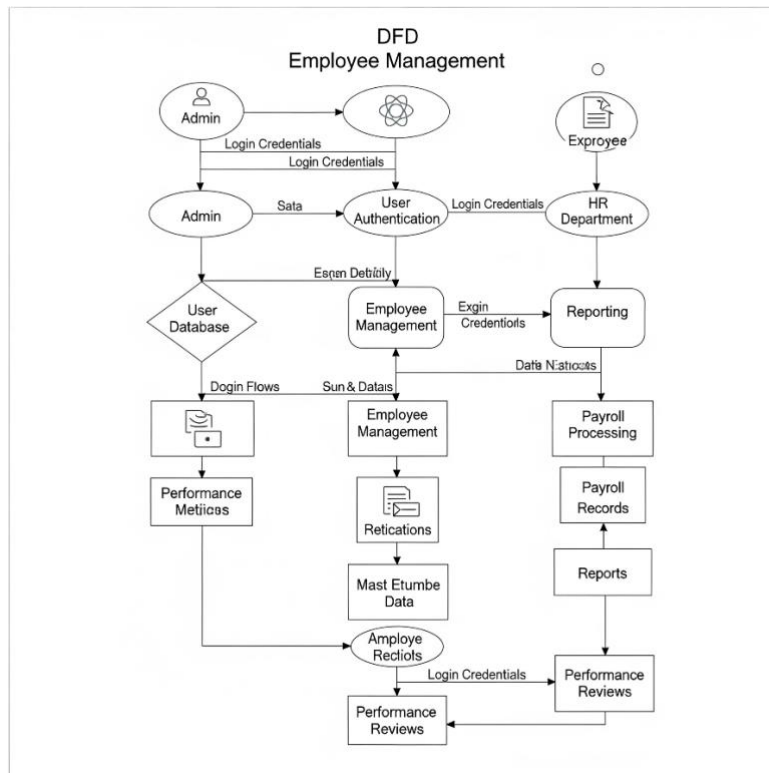
Reliability Approaches in HRPulse:

1. **Error Avoidance:** The system is designed to prevent errors through proper validation, authentication, and secure coding practices.
2. **Error Detection and Correction:** Any errors encountered during testing or deployment are identified and corrected promptly to ensure smooth operation.
3. **Error Tolerance:** The system is built to handle minor errors gracefully without interrupting user operations, maintaining degraded performance if needed.

By applying these paradigms, the HRPulse system ensures **robustness, efficiency, and reliability**, meeting user expectations while providing a stable platform for job seekers and employers.

4. SYSTEM DESIGN

4A. Data Flow Diagram (DFD)



A **Data Flow Diagram (DFD)** is a graphical representation of how data flows through a system. It helps model the processes of the HRPulse job portal, illustrating what information is input, how it is processed, where it is stored, and what outputs are generated. DFDs are used both during **analysis and design phases** of SDLC to visualize the system for stakeholders and developers.

DFD Notations:

- **Process:** Represented by circles, showing the activity or function.
- **Data Store / Database:** Represented by parallel lines or a rectangle, storing information.
- **Input / Output:** Represented by arrows showing data movement.
- **Flow:** Shows direction of data movement between processes, stores, and external entities.

•

Steps to Construct a DFD:

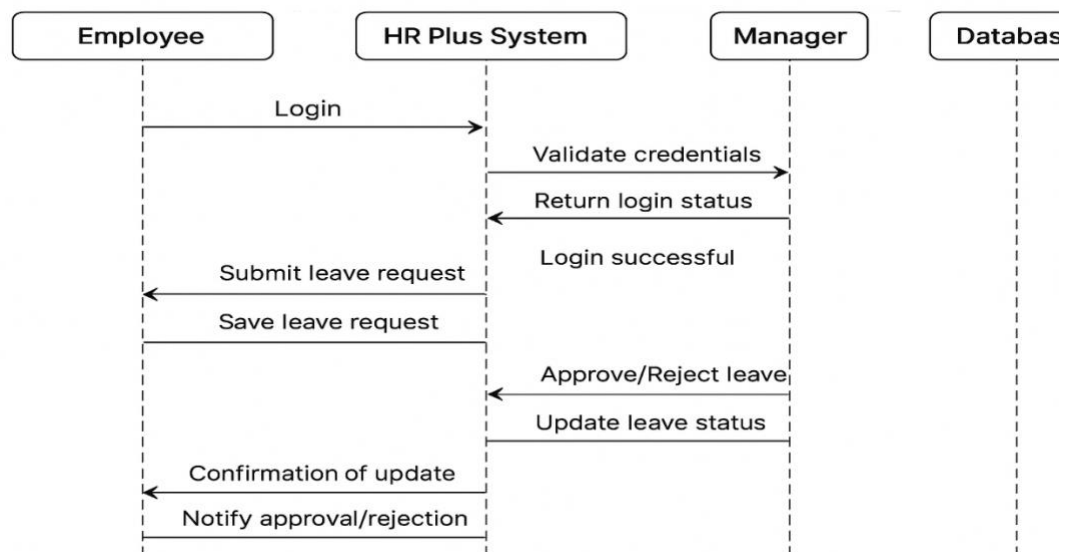
1. Name each process clearly for easy reference.
2. Data flows from top to bottom and left to right.
3. Number lower-level processes when decomposing a high-level process.
4. Data stores, sources, and destinations are named in **capital letters**.
5. Avoid crossing arrows and ensure clarity of relationships.

DFD Levels for HRPulse:

- **Level 0 (Context Diagram):** Shows a high-level overview of the system with external entities (Job Seekers, Employers) interacting with the system (login, job posting, application submission).

- **Level 1 DFD:** Breaks down major subsystems such as **User Registration, Job Search, Job Application, Admin Management**, and shows data stores like **User Data, Job Data, Application Data**.
- **Level 2 DFD (Optional):** Further decomposes Level 1 processes, e.g., **User Profile Management, Resume Upload, Job Status Tracking**.

4B. Sequence Diagram



A **Sequence Diagram** is an interaction diagram that shows how objects or processes interact over time. For HRPulse, the sequence diagram captures the flow of messages between **job seekers, employers, and the system**.

Key Features:

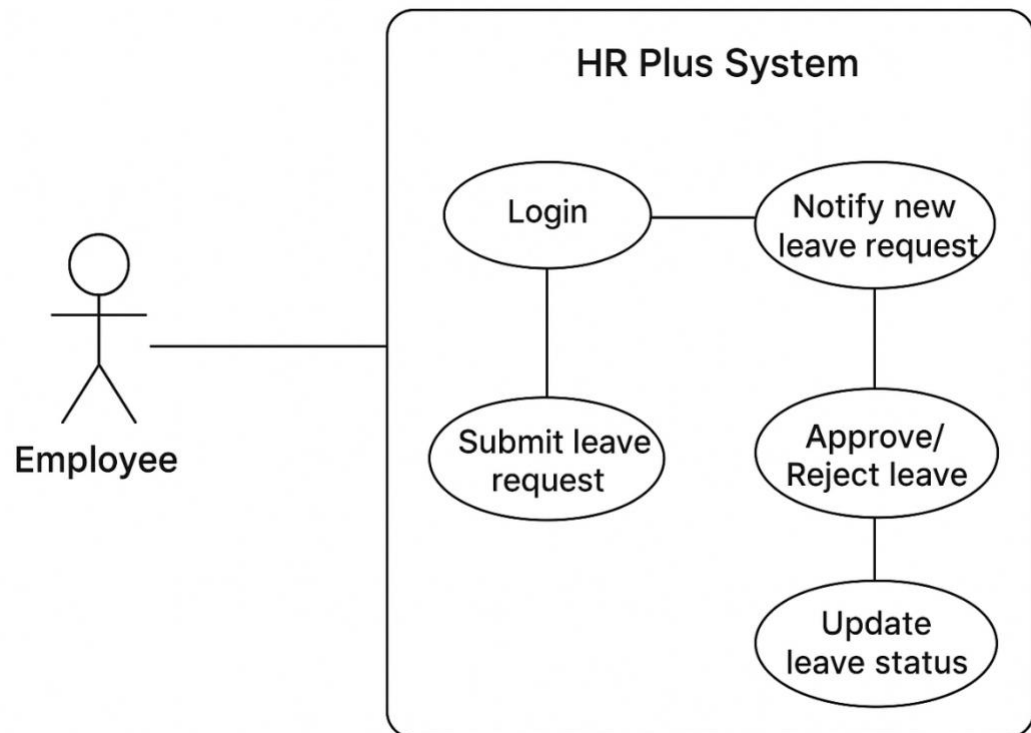
- **Lifelines:** Represent objects or participants (User, System, Admin).
- **Messages:** Horizontal arrows showing interactions, e.g., login requests, job applications, and notifications.
- **Execution Specifications:** Represent the duration an object is active during a process.

Example Scenario (Job Application):

1. User logs in → System verifies credentials.
2. User searches for jobs → System retrieves relevant job listings.
3. User applies for a job → System stores application in the database and notifies the employer.
4. Employer reviews applications → System updates the status and sends notifications to the user.

This diagram ensures a **step-by-step understanding** of system interactions and message sequences.

4C. Use Case Diagram



A **Use Case Diagram** captures the dynamic functionality of HRPulse, showing the interactions between users (actors) and system functionalities (use cases).

Actors:

- **Job Seeker (User)** – Searches jobs, applies, updates profile.
- **Employer/Admin** – Posts jobs, manages applications, updates job status.

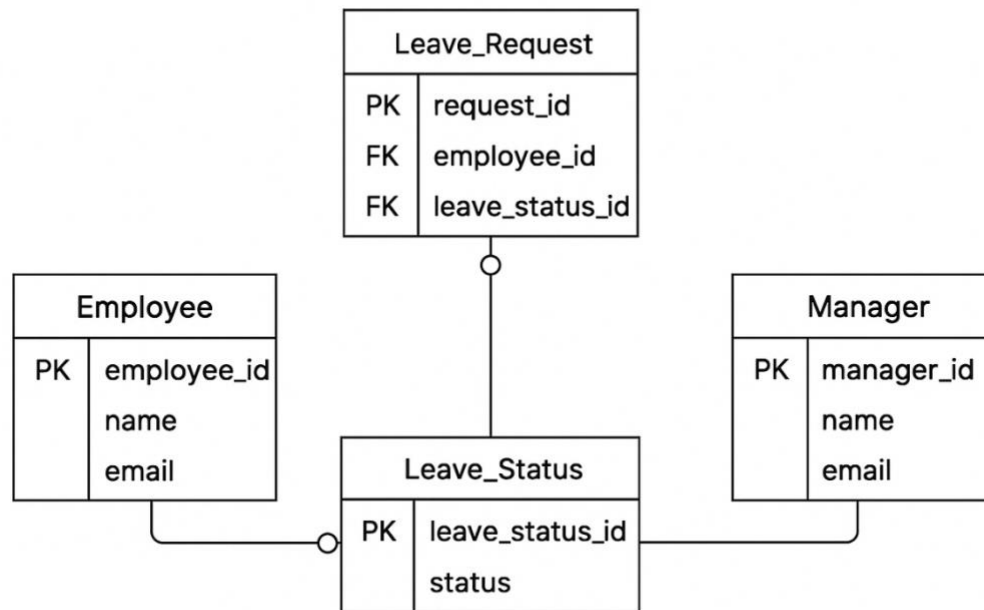
Use Cases:

1. User Registration & Login
2. Job Search & Filter
3. Job Application Submission
4. Application Status Tracking
5. Admin Job Posting & Management
6. Notifications & Feedback

Guidelines for Use Case Diagram:

- Use clear names for actors and use cases.
- Show relationships (associations, includes, extends) where necessary.
- Focus on capturing **functional requirements** rather than implementation details.

4D. Schema Diagram (Database Design)



The **Schema Diagram** represents the logical structure of the HRPulse database, showing entities, attributes, and relationships. It organizes data for easy retrieval and ensures referential integrity.

Entities and Relationships:

1. **Users Table:** Stores user details (UserID, Name, Email, Password, Role).
2. **Jobs Table:** Stores job postings (JobID, EmployerID, Title, Description, SkillsRequired, Location).
3. **Applications Table:** Tracks applications (ApplicationID, JobID, UserID, Status, DateApplied).
4. **Admin Table:** Stores admin credentials and roles for system management.

Relationships:

- One **User** can apply to many **Jobs** (1-to-Many relationship between Users and Applications).
- One **Job** can have multiple **Applications**.
- **Admin** manages multiple jobs and user accounts.

This schema ensures **data consistency, efficient queries, and scalability** for the job portal system.

Flowchart for HRPulse System

High-Level Flow:

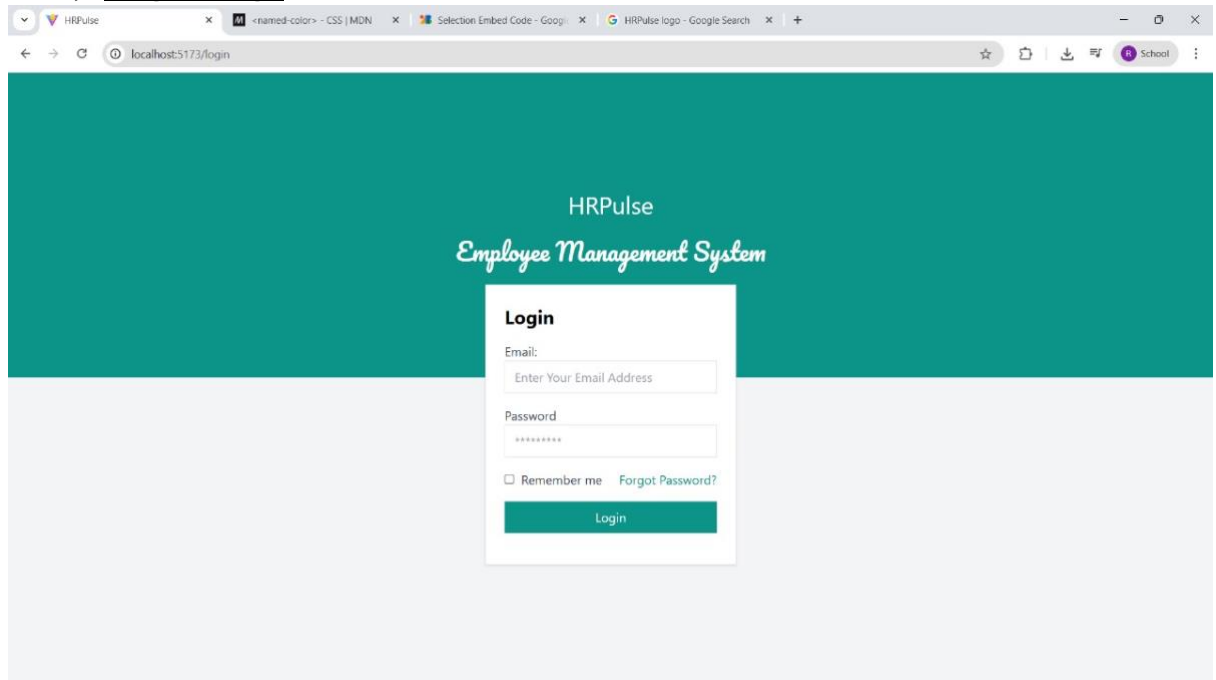
1. **Start** → User or Admin selects **Register/Login** → OTP Verification

2. **Login Successful** → Show Dashboard (User / Admin)
3. **User Flow:** Search Jobs → Apply → Track Application → Logout
4. **Admin Flow:** Post Job → View Applications → Update Status → Logout
5. **End**

5.UI SNAPSHOT

❖ FRONTEND :-

1) Login Page:



✓ CODE:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>HR Plus - Login</title>
```

```
<style>
```

```
  body {
```

```
    font-family: Arial, sans-serif;
```

```
background: linear-gradient(135deg, #2c3e50, #3498db);
margin: 0;
padding: 0;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
}
```

```
.login-container {
  background: #fff;
  padding: 40px;
  border-radius: 10px;
  box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.3);
  width: 350px;
}
```

```
h2 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}
```

```
input {
  width: 100%;
  padding: 12px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 14px;
```

```
}
```

```
button {  
  width: 100%;  
  padding: 12px;  
  border: none;  
  border-radius: 6px;  
  background: #3498db;  
  color: white;  
  font-size: 16px;  
  cursor: pointer;  
}
```

```
button:hover {  
  background: #2980b9;  
}
```

```
.error {  
  color: red;  
  font-size: 13px;  
  display: none;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="login-container">
```

```
<h2>HR Plus Login</h2>
```

```
<input type="text" id="username" placeholder="Username">
```

```
<input type="password" id="password" placeholder="Password">
```

```
<p class="error" id="error-msg">Please enter username and password</p>
<button onclick="login()">Login</button>
</div>
```

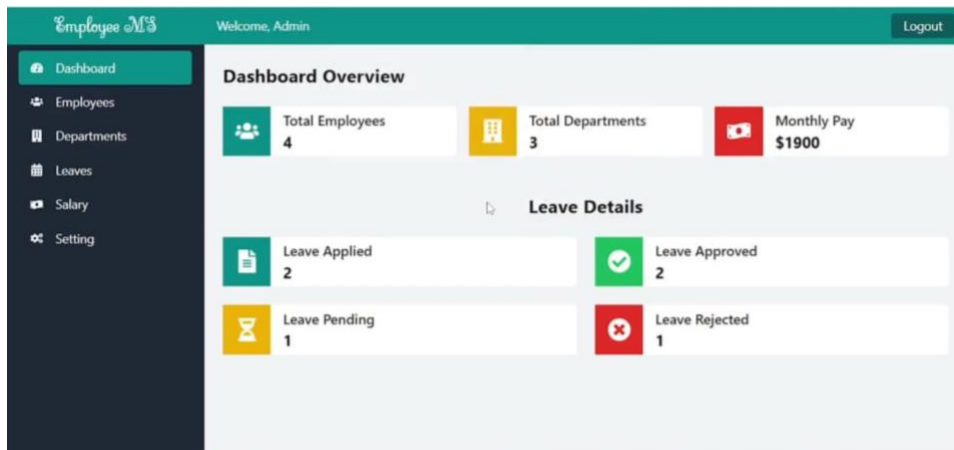
```
<script>
function login() {
    const user = document.getElementById('username').value.trim();
    const pass = document.getElementById('password').value.trim();
    const errorMsg = document.getElementById('error-msg');

    if (user === "" || pass === "") {
        errorMsg.style.display = "block";
    } else {
        errorMsg.style.display = "none";
        alert("Login Successful! Redirecting...");
        // You can add redirect logic here
        window.location.href = "dashboard.html";
    }
}
</script>
```

```
</body>
```

```
</html>
```


2)Admin dashboard:



✓ CODE:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Admin Dashboard</title>
```

```
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
  <link rel="stylesheet" href="style.css">
```

```
</head>
```

```
<body>
```

```
<div class="d-flex" id="wrapper">
```

```
  <!-- Sidebar -->
```

```
  <div class="bg-dark text-white p-3" id="sidebar">
```

```
    <h2 class="text-center mb-4">Admin</h2>
```

```
    <ul class="nav flex-column">
```

```
      <li class="nav-item"><a href="#" class="nav-link text-white">🏠 Dashboard</a></li>
```

```
      <li class="nav-item"><a href="#" class="nav-link text-white">👤 Users</a></li>
```

```

<li class="nav-item"><a href="#" class="nav-link text-white"><img alt="Product icon" data-bbox="662 88 685 105"/> Products</a></li>
<li class="nav-item"><a href="#" class="nav-link text-white"><img alt="Report icon" data-bbox="662 118 685 135"/> Reports</a></li>
<li class="nav-item"><a href="#" class="nav-link text-white"><img alt="Settings icon" data-bbox="662 148 685 165"/> Settings</a></li>
</ul>
</div>

```

```

<!-- Page Content -->

```

```

<div id="page-content-wrapper" class="w-100">

```

```

<!-- Top navbar -->

```

```

<nav class="navbar navbar-expand-lg navbar-light bg-light border-bottom px-4">

```

```

<button class="btn btn-primary" id="menu-toggle"><img alt="Menu icon" data-bbox="375 396 398 413"/></button>

```

```

<form class="d-flex ms-auto">

```

```

<input class="form-control me-2" type="search" placeholder="Search...">

```

```

<button class="btn btn-outline-success">Search</button>

```

```

</form>

```

```

<div class="ms-3">

```

```



```

```

</div>

```

```

</nav>

```

```

<!-- Dashboard Content -->

```

```

<div class="container-fluid p-4">

```

```

<!-- Cards -->

```

```

<div class="row mb-4">

```

```

<div class="col-md-3">

```

```

<div class="card text-bg-primary p-3">

```

```

<h4>Total Users</h4>

```

```

<p>1,245</p>

```

```

    </div>
  </div>
  <div class="col-md-3">
    <div class="card text-bg-success p-3">
      <h4>Orders</h4>
      <p>563</p>
    </div>
  </div>
  <div class="col-md-3">
    <div class="card text-bg-warning p-3">
      <h4>Revenue</h4>
      <p>$12,340</p>
    </div>
  </div>
  <div class="col-md-3">
    <div class="card text-bg-danger p-3">
      <h4>Alerts</h4>
      <p>8</p>
    </div>
  </div>
</div>

<!-- Table -->
<div class="card">
  <div class="card-header">
    Latest Users
  </div>
  <div class="card-body">
    <table class="table table-striped">
      <thead>

```

```
<tr>
  <th>Name</th>
  <th>Email</th>
  <th>Role</th>
  <th>Joined</th>
</tr>
</thead>
<tbody>
  <tr>
    <td>John Doe</td>
    <td>john@example.com</td>
    <td>Admin</td>
    <td>2025-08-10</td>
  </tr>
  <tr>
    <td>Jane Smith</td>
    <td>jane@example.com</td>
    <td>User</td>
    <td>2025-08-12</td>
  </tr>
  <tr>
    <td>Mark Lee</td>
    <td>mark@example.com</td>
    <td>Moderator</td>
    <td>2025-08-14</td>
  </tr>
</tbody>
</table>
</div>
</div>
```

```

    </div>

</div>

</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

<script src="script.js"></script>

</body>

</html>

body {
    overflow-x: hidden;
}

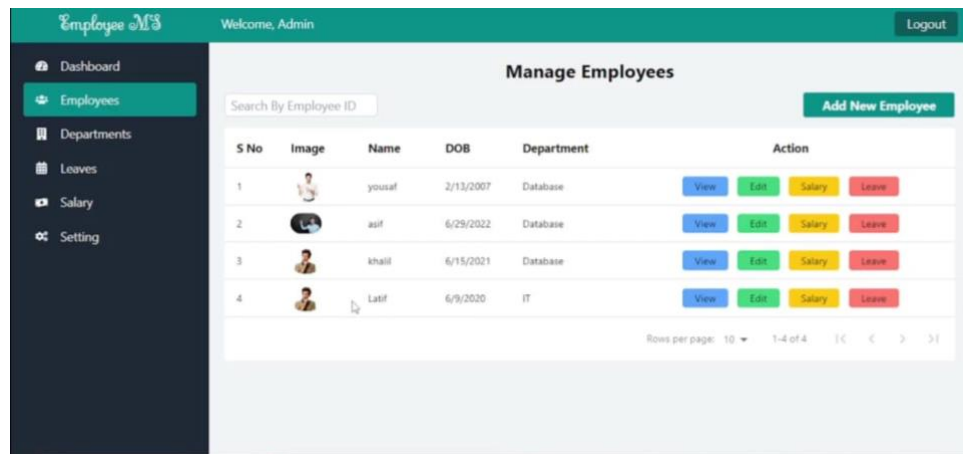
#sidebar {
    min-width: 250px;
    max-width: 250px;
    height: 100vh;
}

#page-content-wrapper {
    flex: 1;
}

.card {
    border-radius: 10px;
} document.getElementById("menu-toggle").addEventListener("click", function () {
    document.getElementById("sidebar").classList.toggle("d-none");
});

```

3)Manage Employees



✓ **CODE:**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Admin Dashboard - Manage Employees</title>
```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
<link rel="stylesheet" href="style.css">
```

```
</head>
```

```
<body>
```

```
<div class="d-flex" id="wrapper">
```

```
<!-- Sidebar -->
```

```
<div class="bg-dark text-white p-3" id="sidebar">
```

```
<h2 class="text-center mb-4">Admin</h2>
```

```
<ul class="nav flex-column">
```

```
<li class="nav-item"><a href="#" class="nav-link text-white">🏠 Dashboard</a></li>
```

```
<li class="nav-item"><a href="#" class="nav-link text-white" id="manage-employees-link">👤 Manage Employees</a></li>
```

```
<li class="nav-item"><a href="#" class="nav-link text-white">📊 Reports</a></li>
```

```
<li class="nav-item"><a href="#" class="nav-link text-white">⚙️ Settings</a></li>
```

```
</ul>
```

```
</div>
```

```
<!-- Page Content -->
```

```
<div id="page-content-wrapper" class="w-100">
```

```
<!-- Top navbar -->
```

```
<nav class="navbar navbar-expand-lg navbar-light bg-light border-bottom px-4">
```

```
<button class="btn btn-primary" id="menu-toggle">☰</button>
```

```
<form class="d-flex ms-auto">
```

```
<input class="form-control me-2" type="search" placeholder="Search...">
```

```
<button class="btn btn-outline-success">Search</button>
```

```
</form>
```

```
<div class="ms-3">
```

```

```

```
</div>
```

```
</nav>
```

```
<!-- Dashboard Content -->
```

```
<div class="container-fluid p-4" id="main-content">
```

```
<h2>Welcome to Admin Dashboard</h2>
```

```
<p>Select an option from the sidebar.</p>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- Bootstrap JS -->

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

<script src="script.js"></script>

</body>

</html>
```


4)Add new employee

✓ CODE:

// Load Manage Employees Page

```
document.getElementById("manage-employees-link").addEventListener("click", function (e)
{
```

```
    e.preventDefault();
```

```
    document.getElementById("main-content").innerHTML = `
```

```
        <h2>Manage Employees</h2>
```

```
        <form id="employee-form" class="mb-3">
```

```
            <div class="row g-2">
```

```
                <div class="col-md-3">
```

```
                    <input type="text" id="emp-name" class="form-control" placeholder="Name"
required>
```

```
                </div>
```

```
                <div class="col-md-3">
```

```
                    <input type="email" id="emp-email" class="form-control" placeholder="Email"
required>
```

```
                </div>
```

```
                <div class="col-md-3">
```

```
                    <input type="text" id="emp-role" class="form-control" placeholder="Role" required>
```

```
                </div>
```

```
            </div>
```

```

        <button type="submit" class="btn btn-success w-100">Add Employee</button>
    </div>
</div>
</form>

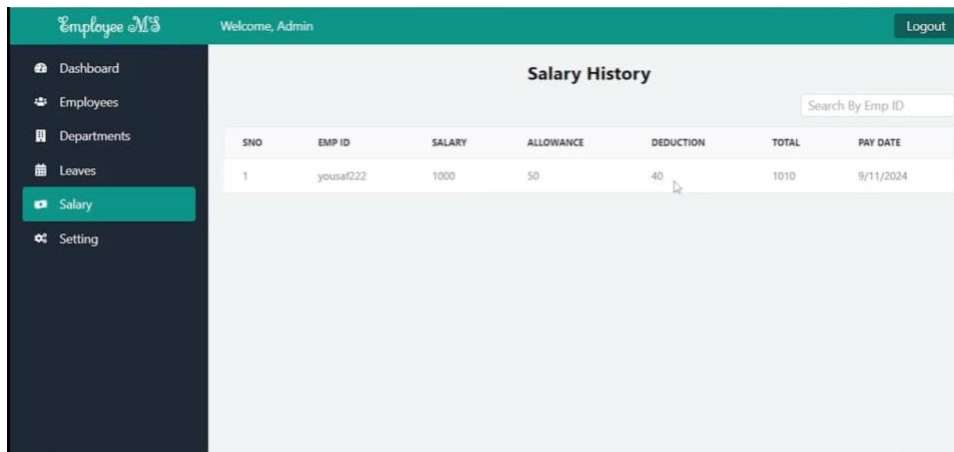
<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Role</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody id="employee-table">
        <tr>
            <td>John Doe</td>
            <td>john@example.com</td>
            <td>Manager</td>
            <td>
                <button class="btn btn-warning btn-sm" onclick="editEmployee(this)">Edit</button>
                <button class="btn btn-danger btn-sm"
onclick="deleteEmployee(this)">Delete</button>
            </td>
        </tr>
    </tbody>
</table>
`;

// Form submission event
document.getElementById("employee-form").addEventListener("submit", function (e) {

```

```
e.preventDefault();  
  
let name = document.getElementById("emp-name").value;  
let email = document.getElementById("emp-email").value;  
let role = document.getElementById("emp-role").value;  
  
let table = document.getElementById("employee-table");  
let row = table.insertRow();  
row.innerHTML = `  
  <td>${name}</td>  
  <td>${email}<
```

5)Salary history:



SNO	EMP ID	SALARY	ALLOWANCE	DEDUCTION	TOTAL	PAY DATE
1	yousaf222	1000	50	40	1010	9/11/2024

✓ CODE:

// Employee salary history storage

```
let salaryData = {  
  "John Doe": [  
    { date: "2025-01-01", amount: 50000 },  
    { date: "2025-06-01", amount: 55000 }  
  ]  
};
```

// Load Manage Employees Page

```
document.getElementById("manage-employees-link").addEventListener("click", function (e)  
{e.preventDefault();
```

```
document.getElementById("main-content").innerHTML = `
```

```
<h2>Manage Employees</h2>
```

```
<form id="employee-form" class="mb-3">
```

```
<div class="row g-2">
```

```
<div class="col-md-3">
```

```
<input type="text" id="emp-name" class="form-control" placeholder="Name"  
required>
```

```
</div>
```

```
<div class="col-md-3">
```

```
<input type="email" id="emp-email" class="form-control" placeholder="Email"  
required>
```

```
</div>
```

```

<div class="col-md-3">
  <input type="text" id="emp-role" class="form-control" placeholder="Role" required>
</div>

<div class="col-md-3">
  <button type="submit" class="btn btn-success w-100">Add Employee</button>
</div>
</div>
</form>

<table class="table table-bordered table-striped">
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Role</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody id="employee-table">
    <tr>
      <td>John Doe</td>
      <td>john@example.com</td>
      <td>Manager</td>
      <td>
        <button class="btn btn-primary btn-sm" onclick="viewSalaryHistory('John Doe')">Salary History</button>
        <button class="btn btn-warning btn-sm" onclick="editEmployee(this)">Edit</button>
        <button class="btn btn-danger btn-sm"
onclick="deleteEmployee(this)">Delete</button>
      </td>
    </tr>
  </tbody>

```

```

</table>

<!-- Salary History Modal -->

<div class="modal fade" id="salaryModal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Salary History</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
      </div>
      <div class="modal-body" id="salary-modal-body">
        </div>
      <div class="modal-footer">
        <input type="number" id="new-salary" placeholder="New Salary" class="form-control w-50">
        <button class="btn btn-success" id="add-salary-btn">Add Salary</button>
      </div>
    </div>
  </div>
</div>

`;

// Add employee form submit
document.getElementById("employee-form").addEventListener("submit", function (e) {
  e.preventDefault();
  let name = document.getElementById("emp-name").value;
  let email = document.getElementById("emp-email").value;
  let role = document.getElementById("emp-role").value;
  let table = document.getElementById("employee-table");
  let row = table.insertRow();
  row.innerHTML = `
    <td>${name}</td>
    <td>${email}</td>

```

```

        <td>${role}</td>

        <td>

            <button class="btn btn-primary btn-sm"
onclick="viewSalaryHistory('${name}')">Salary History</button>

            <button class="btn btn-warning btn-sm" onclick="editEmployee(this)">Edit</button>

            <button class="btn btn-danger btn-sm"
onclick="deleteEmployee(this)">Delete</button>

        </td>

    `;

    // Initialize salary history for new employee
    salaryData[name] = [];
    document.getElementById("employee-form").reset();
    });
});

// Show salary history in modal
function viewSalaryHistory(name) {
    let history = salaryData[name] || [];
    let historyHtml = `<ul class="list-group">`;
    history.forEach(record => {
        historyHtml += `<li class="list-group-item">${record.date} - ₹${record.amount}</li>`;
    });
    historyHtml += `</ul>`;
    document.getElementById("salary-modal-body").innerHTML = historyHtml;
    // Handle adding new salary
    document.getElementById("add-salary-btn").onclick = function () {
        let amount = document.getElementById("new-salary").value;
        if (amount) {
            let today = new Date().toISOString().split("T")[0];
            salaryData[name].push({ date: today, amount: amount });
            viewSalaryHistory(name); // Refresh modal content
        }
    }
}

```

```

};

// Show modal
let modal = new bootstrap.Modal(document.getElementById("salaryModal"));
modal.show();
}

// Edit Employee
function editEmployee(button) {
  let row = button.closest("tr");
  let name = prompt("Edit Name:", row.cells[0].innerText);
  let email = prompt("Edit Email:", row.cells[1].innerText);
  let role = prompt("Edit Role:", row.cells[2].innerText);
  if (name && email && role) {
    row.cells[0].innerText = name;
    row.cells[1].innerText = email;
    row.cells[2].innerText = role;
  }
}

// Delete Employee
function deleteEmployee(button) {
  if (confirm("Are you sure you want to delete this employee?")) {
    let row = button.closest("tr");
    let name = row.cells[0].innerText;
    delete salaryData[name]; // Remove salary history too
    row.remove();
  }
}

```


6)Manage history:

SNO	LEAVE TYPE	FROM	TO	DESCRIPTION	APPLIED DATE	STATUS
1	Sick Leave	9/1/2024	9/5/2024	High fever and flu	9/11/2024	Approved
2	Casual Leave	9/14/2024	9/15/2024	casual leave	9/14/2024	Approved

✓ CODE:

```
let salaryData = {};  
let historyLogs = [];  
// Helper: Add log entry  
function addHistory(action, details) {  
    let timestamp = new Date().toLocaleString();  
    historyLogs.unshift({ time: timestamp, action, details }); // newest first  
}  
// Load Manage Employees Page  
document.getElementById("manage-employees-link").addEventListener("click", function (e)  
{  
    e.preventDefault();  
    document.getElementById("main-content").innerHTML = `  
        <h2>Manage Employees</h2>  
  
        <form id="employee-form" class="mb-3">  
            <div class="row g-2">  
                <div class="col-md-3">  
                    <input type="text" id="emp-name" class="form-control" placeholder="Name"  
required>
```

```

</div>

<div class="col-md-3">

  <input type="email" id="emp-email" class="form-control" placeholder="Email"
required>

</div>

<div class="col-md-3">

  <input type="text" id="emp-role" class="form-control" placeholder="Role" required>

</div>

<div class="col-md-3">

  <button type="submit" class="btn btn-success w-100">Add Employee</button>

</div>

</div>

</form>

<table class="table table-bordered table-striped">
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Role</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody id="employee-table"></tbody>
</table>
`;

document.getElementById("employee-form").addEventListener("submit", function (e) {
  e.preventDefault();
  let name = document.getElementById("emp-name").value;
  let email = document.getElementById("emp-email").value;
  let role = document.getElementById("emp-role").value;

```

```

let table = document.getElementById("employee-table");
let row = table.insertRow();
row.innerHTML = `
    <td>${name}</td>
    <td>${email}</td>
    <td>${role}</td>
    <td>
        <button class="btn btn-primary btn-sm"
onclick="viewSalaryHistory('${name}')">Salary History</button>
        <button class="btn btn-warning btn-sm" onclick="editEmployee(this)">Edit</button>
        <button class="btn btn-danger btn-sm"
onclick="deleteEmployee(this)">Delete</button>
    </td>
`;
salaryData[name] = [];
addHistory("Employee Added", `${name} (${role})`);
document.getElementById("employee-form").reset();
});
});

```

// Manage History Page

```

document.getElementById("manage-history-link").addEventListener("click", function (e) {
    e.preventDefault();
    let rows = historyLogs.map(log => `
        <tr>
            <td>${log.time}</td>
            <td>${log.action}</td>
            <td>${log.details}</td>
        </tr>
    `).join("");

```

```

document.getElementById("main-content").innerHTML = `
<h2>Activity History</h2>
<table class="table table-bordered table-striped">
  <thead>
    <tr>
      <th>Date & Time</th>
      <th>Action</th>
      <th>Details</th>
    </tr>
  </thead>
  <tbody>
    ${rows || "<tr><td colspan='3'>No history available.</td></tr>"}
  </tbody>
</table>
`;

});

// Salary History Modal
function viewSalaryHistory(name) {
  let history = salaryData[name] || [];
  let historyHtml = `<ul class="list-group">`;
  history.forEach(record => {
    historyHtml += `<li class="list-group-item">${record.date} - ₹${record.amount}</li>`;
  });
  historyHtml += `</ul>`;

  document.getElementById("salary-modal-body").innerHTML = historyHtml;

  document.getElementById("add-salary-btn").onclick = function () {
    let amount = document.getElementById("new-salary").value;

```

```

    if (amount) {
        let today = new Date().toISOString().split("T")[0];
        salaryData[name].push({ date: today, amount: amount });
        addHistory("Salary Updated", `${name} salary set to ₹${amount}`);
        viewSalaryHistory(name);
    }
};

let modal = new bootstrap.Modal(document.getElementById("salaryModal"));
modal.show();
}

// Edit Employee
function editEmployee(button) {
    let row = button.closest("tr");
    let oldName = row.cells[0].innerText;
    let name = prompt("Edit Name:", oldName);
    let email = prompt("Edit Email:", row.cells[1].innerText);
    let role = prompt("Edit Role:", row.cells[2].innerText);
    if (name && email && role) {
        row.cells[0].innerText = name;
        row.cells[1].innerText = email;
        row.cells[2].innerText = role;
        addHistory("Employee Edited", `${oldName} → ${name} (${role})`);
    }
}

// Delete Employee
function deleteEmployee(button) {
    let row = button.closest("tr");

```

```
let name = row.cells[0].innerText;
if (confirm(`Delete ${name}?`)) {
  delete salaryData[name];
  row.remove();
  addHistory("Employee Deleted", name);
}
}
```

❖ **BACKEND:-**

admin-backend/

├─ package.json

├─ server.js

├─ .env # MONGO_URI, JWT_SECRET, PORT

├─ src/

| └─ config/

| | └─ db.js

| └─ middleware/

| | └─ auth.js

| | └─ errorHandler.js

| └─ models/

| | └─ Employee.js

| | └─ HistoryLog.js

| └─ controllers/

| | └─ employeeController.js

| | └─ historyController.js

| └─ routes/

| | └─ employeeRoutes.js

| | └─ historyRoutes.js

| └─ utils/

| └─ asyncHandler.js

└─ README.md

1) package.json

```
{  
  "name": "admin-backend",  
  "version": "1.0.0",  
  "main": "server.js",  
  "type": "module",  
  "scripts": {  
    "dev": "node --watch server.js",  
    "start": "node server.js"  
  },  
  "dependencies": {  
    "cors": "^2.8.5",  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "helmet": "^7.1.0",  
    "jsonwebtoken": "^9.0.2",  
    "mongoose": "^8.6.1",  
    "morgan": "^1.10.0",  
    "zod": "^3.23.8"  
  }  
}
```


2) server.js

```
import express from "express";
import cors from "cors";
import helmet from "helmet";
import morgan from "morgan";
import dotenv from "dotenv";
dotenv.config();

import { connectDB } from "../src/config/db.js";
import employeeRoutes from "../src/routes/employeeRoutes.js";
import historyRoutes from "../src/routes/historyRoutes.js";
import { notFound, errorHandler } from "../src/middleware/errorHandler.js";

const app = express();

// Core middleware
app.use(helmet());
app.use(cors({ origin: true, credentials: true }));
app.use(express.json());
app.use(morgan("dev"));

// DB
await connectDB();

// Health
app.get("/api/health", (_req, res) => res.json({ ok: true, ts: new Date() }));

// Routes
```

```
app.use("/api/employees", employeeRoutes);
app.use("/api/history", historyRoutes);

// 404 + Error handler
app.use(notFound);
app.use(errorHandler);

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log(` API running on http://localhost:${PORT} `));
```

5. Conclusion

The MERN stack-based Employee Management System successfully delivers an efficient and modern solution for human resource management. By leveraging the power of JavaScript across the entire stack, the project ensures a cohesive and maintainable codebase. The application provides a clear, scalable, and secure platform for managing employee data, replacing outdated manual methods with an automated, digital solution.

7. Future Scope & Enhancements

The project has a wide range of opportunities for future growth and innovation. Potential enhancements include:

- **AI-Powered Analytics:** Integrate AI to generate insightful reports on employee performance, turnover, and salary trends.
- **Payroll and Leave Management:** Add modules for tracking employee leaves, calculating payroll, and generating payslips.
- **Mobile Application:** Develop a native mobile application using React Native to allow for on-the-go access
- **Gamification:** Implement features like badges or leaderboards to boost employee engagement and recognition.
- **Improved Security:** Introduce two-factor authentication and advanced logging for enhanced security.
- **Real-time Notifications:** Use WebSockets for real-time notifications for events like new hires or leave approvals.

8. Bibliography

1. www.w3schools.com
2. www.youtube.com
3. www.mongodb.com/docs
4. www.expressjs.com
5. www.react.dev/learn
6. www.nodejs.org/en/docs
7. www.github.com
8. www.npmjs.com