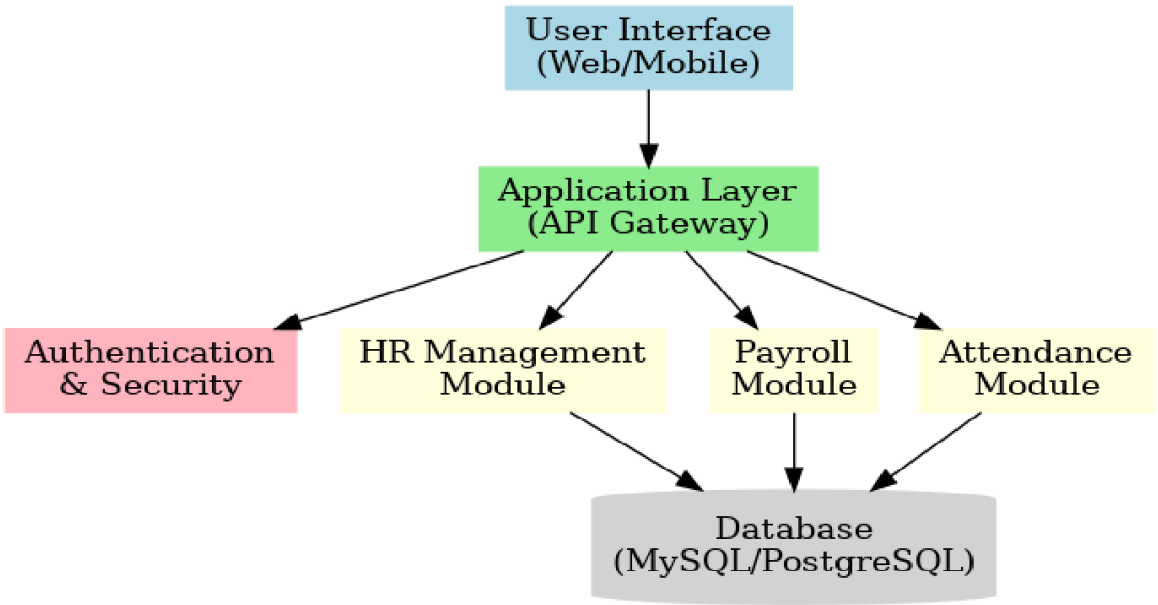
Software Design Document: HR Management System (HRMS)

Problem statement

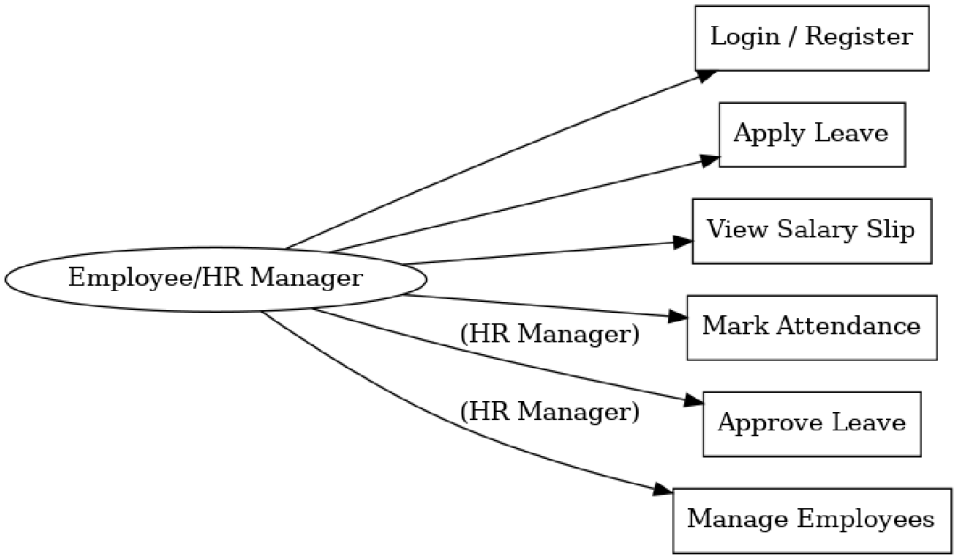
The **HRMS** enables organizations to manage their workforce efficiently by automating employee records, leave processing, attendance tracking, payroll, recruitment, and employee performance monitoring. The system supports employees, HR staff, and administrators, ensuring security, modularity, and auditability of data, while integrating notifications and reporting

Part A : High-level Design (HLD)

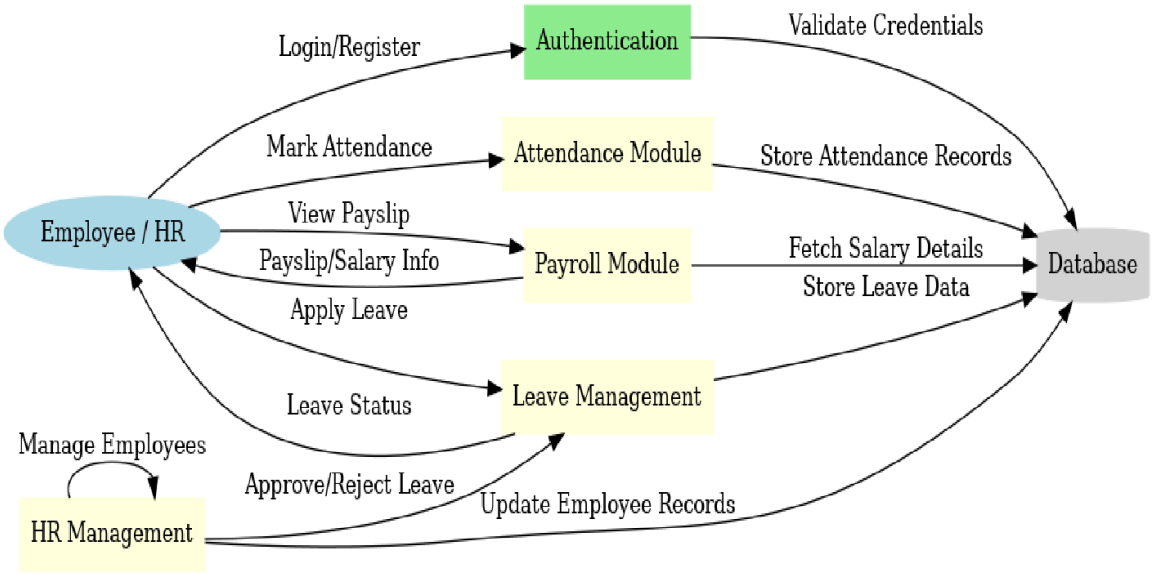
1.Architecture Overview



2.Use Case Diagram



3 Data Flow Diagram(Level 1)

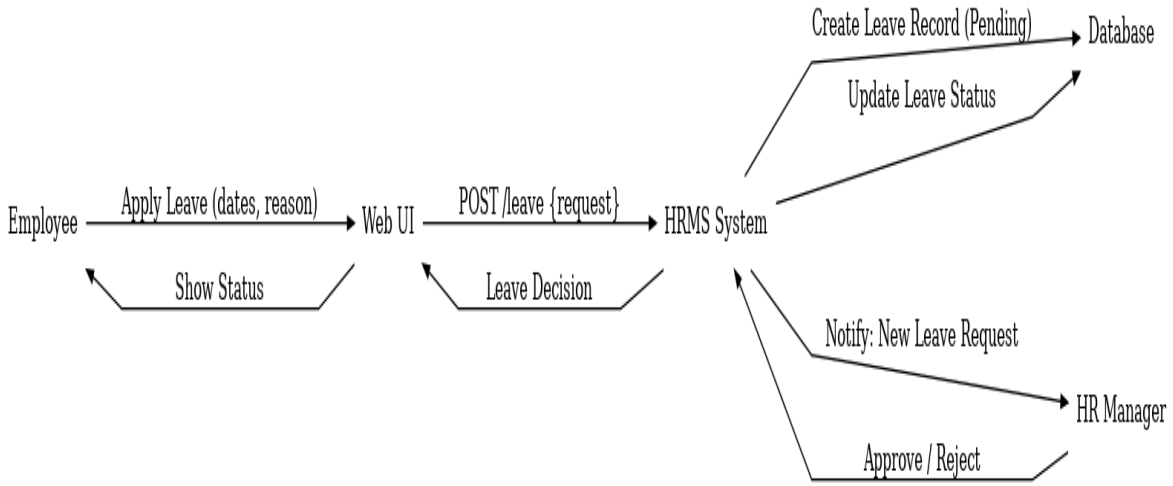


4 Technology Stack

- Frontend (UI): React.js / Angular (Web), Flutter / React Native (Mobile)
- Backend (API Layer): Node.js (Express) / Java (Spring Boot) / Python (Django/Flask)
- Database: PostgreSQL / MySQL/FIREBASE
- Authentication & Security: JWT / OAuth 2.0
- Cloud Hosting: AWS / Azure / GCP
- Version Control: GitHub / GitLab

Part B: Low-Level Design (LLD)

1.Sequence Diagram



Explanation :

Employee -> System: Login

System -> Database: Validate Employee

Database -> System: Success

Employee -> System: Apply Leave(startDate, endDate)

System -> Database: Save Leave Request

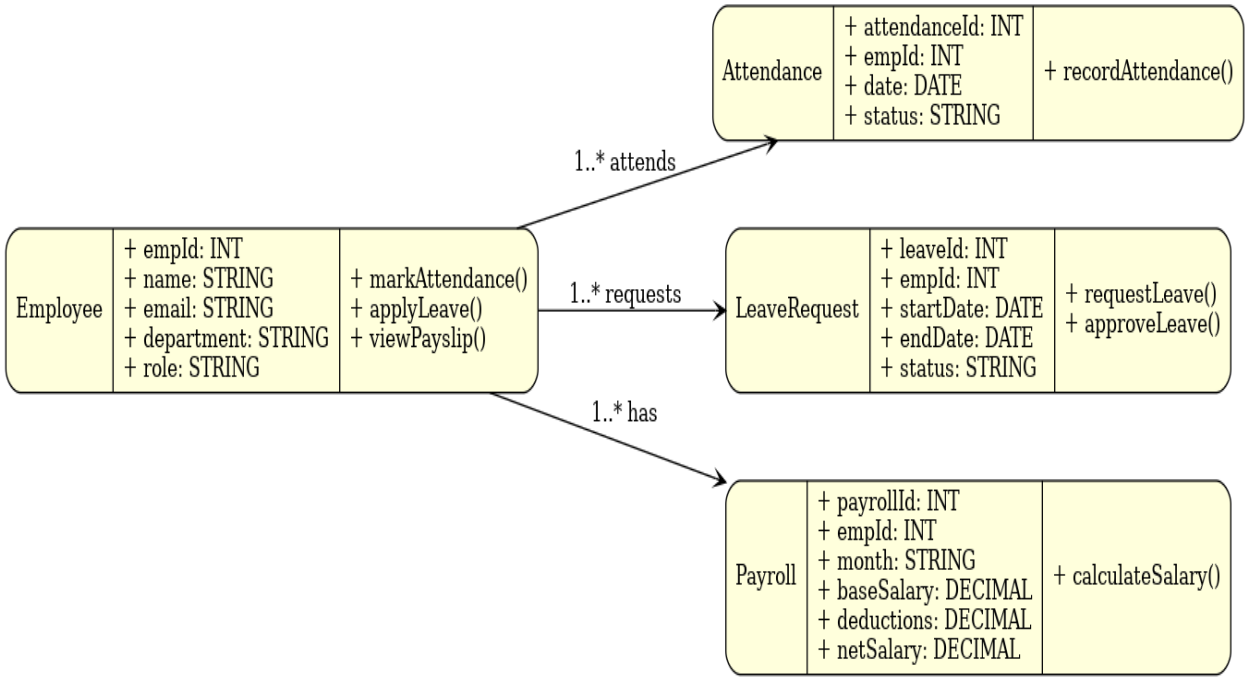
System -> HR Manager: Notify New Request

HR Manager -> System: Approve/Reject Leave

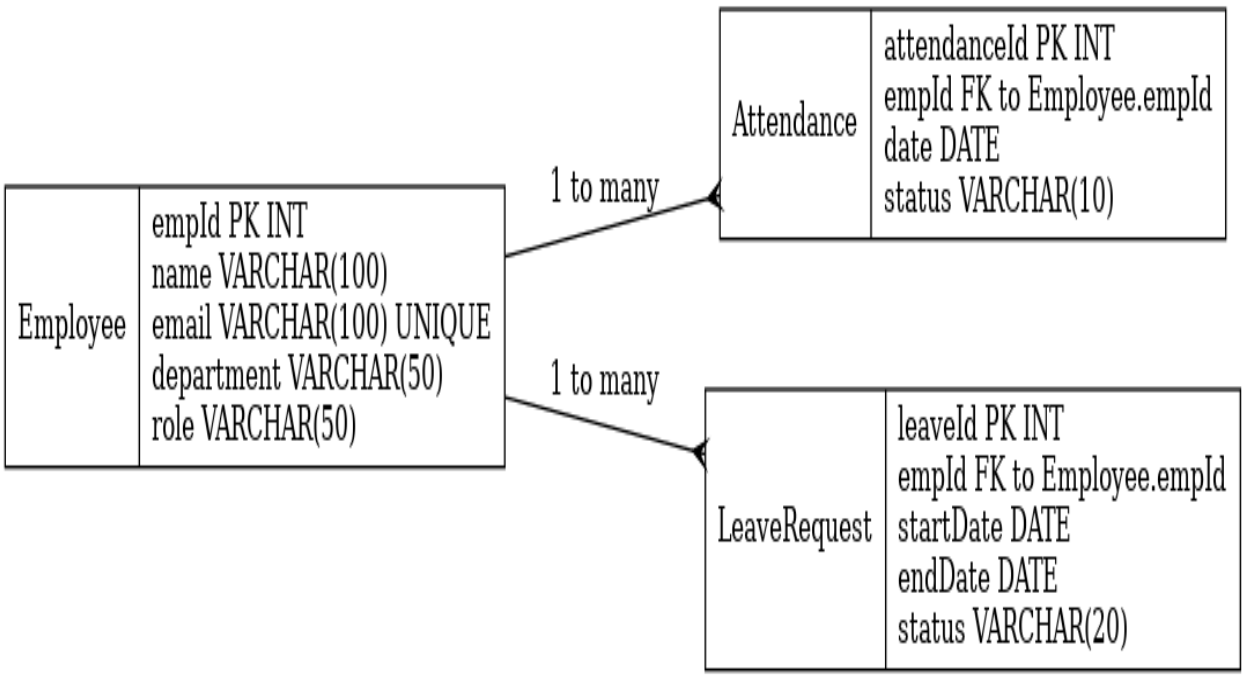
System -> Database: Update Leave Status

System -> Employee: Leave Status Notification

2.Class Diagram



3.Database schema



Explanation :

Employee Table

```
CREATE TABLE Employee (
   empId INT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100) UNIQUE,
   department VARCHAR(50),
   role VARCHAR(50)
);
```

Attendance Table

```
CREATE TABLE Attendance (
   attendanceId INT PRIMARY KEY,
   empId INT,
   date DATE,
   status VARCHAR(10),
```

```
    FOREIGN KEY (empId) REFERENCES Employee(empId)
);
```

Leave Table

```
CREATE TABLE LeaveRequest (
    leaveId INT PRIMARY KEY,
    empId INT,
    startDate DATE,
    endDate DATE,
    status VARCHAR(20),
    FOREIGN KEY (empId) REFERENCES Employee(empId)
);
```

Pseudo-code

```
FUNCTION calculateTotalSalary(employeeId):

  # Step 1: Fetch base salary for employee
  baseSalary = DB.getBaseSalary(employeeId)

  # Step 2: Fetch earnings (bonuses, allowances, overtime, etc.)
  earnings = DB.getEarnings(employeeId)

  # Step 3: Fetch deductions (tax, provident fund, loan, etc.)
  deductions = DB.getDeductions(employeeId)

  # Step 4: Calculate totals
  totalEarnings = 0
```

```
FOR each e IN earnings:
    totalEarnings = totalEarnings + e.amount
END FOR

totalDeductions = 0
FOR each d IN deductions:
    totalDeductions = totalDeductions + d.amount
END FOR

# Step 5: Calculate net salary
netSalary = baseSalary + totalEarnings - totalDeductions

RETURN netSalary
```

Part 3 Explanation

System Architecture

- UI (Web / Mobile) communicates with an API Gateway (Application Layer).
- API passes requests to modules: Authentication, HR Management, Attendance, Payroll.
- Modules persist and read data from a central relational DB (Postgres/MySQL).
- Authentication uses JWT/OAuth for security; services can be hosted on cloud (AWS/Azure/GCP).

Use Case Diagram

- Actors: Employee and HR Manager.
- Employee activities: Login, Mark Attendance, Apply Leave, View Payslip.
- HR Manager activities (higher privilege): Approve/Reject Leaves, Manage Employees.

Data Flow Diagram

- Shows how user actions (attendance, leave, payroll queries) flow into the system.
- Authentication validates credentials; Attendance/Leave/Payroll modules store/retrieve from DB.
- HR Manager reviews and triggers updates (e.g., leave approval updates DB & notifies user).