## Back End Engineering-II

Project Report

Semester-V (Batch-2023)

**Employee Payroll**

PG-17

A red and white sign

Description automatically generated with low confidence

**Supervised By: Submitted By:**

Dr. Pummy Parmeet, 2310992254 (G26)

Romsha, 2310992275 (G26)

Somiya, 2310991719 (G26)

Swati, 2310992302 (G26)

**Department of Computer Science and Engineering**

**Chitkara University Institute of Engineering & Technology,**

**Chitkara University, Punjab**

**ABSTRACT**

The Employee Payroll System is a software application designed to automate and manage the payroll process efficiently. It aims to simplify salary calculation, manage allowances, deductions, and tax computations while maintaining accurate employee records. The system ensures timely processing of employee salaries, generates payslips, and keeps a record of financial transactions for auditing purposes. By integrating features such as role-based access, employee management, and secure data storage, this project reduces manual effort, minimizes errors, and improves overall organizational efficiency. The system is implemented with a user-friendly interface and supports scalability to accommodate growing workforce needs.

**TABLE OF CONTENTS**

**1. Introduction**

1.1 Background

1.2 Objectives

1.3 Significance

**2. Problem Definition and Requirements**

2.1 Problem Statement

2.2 Features

2.3 Software Requirements

2.4 Functional and Non-Functional Requirements

**3. Proposed Design/Methodology**

3.1 System Architecture Overview

3.2 Middleware Architecture

3.3 Data Model Design

3.4 File Structure

3.5 Security Implementation

**4. Results**

**1.Introduction**

**1.1 Background**

In any organization, managing employee salaries and financial records is a critical task. Traditionally, payroll processing involved manual calculations, which were time-consuming, error-prone, and inefficient, especially for organizations with a large workforce. The **Employee Payroll System** is a software application designed to automate and streamline the entire payroll process, ensuring accuracy, efficiency, and timely salary disbursement.

The system calculates employee salaries by taking into account various components such as basic pay, allowances, bonuses, overtime, and statutory deductions like taxes, provident fund contributions, and insurance premiums. It generates detailed payslips for each employee, providing transparency and easy access to their earnings and deductions. Additionally, the system maintains comprehensive records of employees’ salary history, deductions, and benefits, which can be useful for audits, financial planning, and reporting.

One of the main advantages of this system is the **reduction of manual effort**. Payroll management can be complex due to the need to comply with tax regulations, labor laws, and organizational policies. The automated system minimizes human errors and ensures that all calculations are performed accurately according to predefined rules. It also supports role-based access, allowing administrators to manage employee data securely while providing employees with access to their own information.

The Employee Payroll System is built with a user-friendly interface, making it easy for HR personnel and employees to use without extensive technical knowledge. The system is scalable and can handle the payroll needs of both small businesses and large organizations. By automating payroll processes, organizations can save time, reduce administrative costs, and enhance employee satisfaction through timely and precise salary management.

In conclusion, the Employee Payroll System is an essential tool for modern organizations, combining accuracy, efficiency, and security to manage payroll effectively and support organizational growth.

**1.2 Objectives**

The primary objectives of the PoolUp project are as follows:

 **Automate Salary Calculation**

* To calculate employee salaries accurately by considering basic pay, allowances, bonuses, overtime, and deductions such as taxes, provident fund, and insurance.

 **Generate Payslips**

* To provide detailed payslips for employees, showing earnings, deductions, and net salary for transparency and record-keeping.

 **Maintain Employee Records**

* To store and manage comprehensive employee information including personal details, salary history, and attendance records for easy retrieval and reporting.

 **Ensure Timely Salary Disbursement**

* To process payroll efficiently and ensure employees receive their salaries on time without delays.

 **Reduce Manual Effort and Errors**

* To minimize human errors in payroll processing by automating complex calculations and repetitive tasks.

 **Support Compliance**

* To ensure payroll adheres to statutory regulations, tax rules, and organizational policies.

 **Enhance Security**

* To provide secure access to payroll data through role-based permissions, protecting sensitive employee and financial information.

 **Facilitate Reporting and Auditing**

* To generate payroll reports, summaries, and historical data that help in audits, financial planning, and decision-making.

 **Scalability**

* To support growing organizations by handling increasing numbers of employees and complex payroll structures efficiently.

**1.3 Significance**

The Employee Payroll System plays a crucial role in modern organizations by automating and streamlining the payroll process. Its significance lies in several key areas:

1. **Accuracy in Salary Calculation**
   * It ensures precise computation of salaries, including allowances, bonuses, overtime, and deductions, reducing human errors that are common in manual processing.
2. **Time and Cost Efficiency**
   * Automating payroll reduces the time and effort required by HR personnel, allowing them to focus on other important tasks. It also minimizes administrative costs associated with manual payroll management.
3. **Employee Satisfaction**
   * Timely and accurate salary disbursement increases employee trust and satisfaction. Employees can also easily access their payslips and financial records.
4. **Data Management and Record-Keeping**
   * The system maintains organized and secure records of employee details, salary history, and deductions, which is useful for audits, reporting, and decision-making.
5. **Regulatory Compliance**
   * It helps organizations comply with tax regulations, labor laws, and statutory deductions, reducing the risk of legal penalties.
6. **Security and Confidentiality**
   * Sensitive employee and payroll information is protected through secure access controls, ensuring confidentiality and data integrity.
7. **Scalability for Growth**
   * The system can handle an increasing number of employees and complex payroll structures as the organization grows, making it suitable for both small and large enterprises.

In conclusion, the Employee Payroll System is significant because it enhances organizational efficiency, ensures accurate and timely salary management, and contributes to overall employee satisfaction and compliance with statutory regulations.

**2. Problem Definition and Requirements**

**2.1 Problem Statement**

Managing employee salaries and payroll manually is a time-consuming and error-prone task in most organizations. Traditional payroll processing involves repetitive calculations for basic pay, allowances, overtime, deductions, and taxes, which can lead to mistakes and delays. Maintaining accurate records of employee salaries, payslips, and statutory compliance is challenging, especially in organizations with a large workforce. Manual systems also make it difficult to retrieve historical data for audits, generate reports, or analyze payroll trends efficiently.

There is a need for an automated system that can simplify payroll management, ensure accuracy in salary computation, maintain secure employee records, generate payslips, and comply with statutory regulations. The system should reduce human effort, minimize errors, save time, and provide timely access to payroll information for both HR personnel and employees. By addressing these challenges, the Employee Payroll System can improve organizational efficiency, enhance employee satisfaction, and support better financial and administrative management.

**2.2 Features**

 **Automated Salary Calculation**

* Calculates salaries automatically based on basic pay, allowances, bonuses, overtime, and deductions like taxes, provident fund, and insurance.

 **Payslip Generation**

* Generates detailed payslips for employees, displaying earnings, deductions, and net salary for each pay period.

 **Employee Management**

* Maintains records of employee personal details, salary history, attendance, and leave information in a centralized system.

 **Role-Based Access**

* Provides secure access with different permissions for administrators, HR personnel, and employees.

 **Tax and Statutory Compliance**

* Ensures adherence to income tax rules, labor laws, and other statutory regulations while calculating deductions.

 **Reports and Analytics**

* Generates payroll reports, summaries, and historical data for auditing, analysis, and decision-making.

 **Secure Data Storage**

* Protects sensitive employee and payroll information with encryption and secure database management.

 **Time and Attendance Integration**

* Supports integration with attendance systems to automatically calculate overtime, leave deductions, and other allowances.

 **Scalability**

* Can handle payroll for a growing number of employees and complex organizational structures.

 **User-Friendly Interface**

**2.3 Software Requirements**

2.3.1 Hardware Requirements

* Processor: Minimum Intel i3 or equivalent
* RAM: 4 GB (8 GB recommended for development)
* Storage: 500 MB for application, additional for database
* Network: Stable internet connection

2.3.2 Software Requirements

* Operating System: Windows, macOS, or Linux
* Backend: Node.js with Express.js
* Frontend: HTML5, CSS3, JavaScript (optional React/Bootstrap for UI enhancements)
* Database: MongoDB or any NoSQL database for scalability
* Package Manager: npm (Node Package Manager)
* Version Control: Git/GitHub for source code management
* IDE: Visual Studio Code or any preferred code editor
* Browser: Latest version of Chrome/Firefox/Edge for testing

**2.4** **Functional and Non-Functional Requirements**

**Functional Requirements:**

Functional requirements describe **what the system should do**—the tasks, services, and functions it must perform. For an Employee Payroll System, the key functional requirements include:

1. **Employee Management**
   * Add, edit, and delete employee records.
   * Store personal information, designation, department, and contact details.
   * Maintain salary structure and historical salary data.
2. **Salary Calculation**
   * Automatically compute salaries based on basic pay, allowances, bonuses, overtime, and deductions.
   * Calculate statutory deductions such as taxes, provident fund, and insurance.
3. **Payslip Generation**
   * Generate payslips for each employee for a given pay period.
   * Include details of earnings, deductions, and net salary.
4. **Role-Based Access Control**
   * Allow different access levels for administrators, HR personnel, and employees.
   * Ensure employees can view only their own salary information.
5. **Reporting and Analytics**
   * Generate payroll summary reports, tax reports, and deduction reports.
   * Support export of reports in formats like PDF or Excel.
6. **Leave and Attendance Integration**
   * Integrate with attendance/leave system to calculate deductions for absences and overtime pay.
7. **Secure Login and Authentication**
   * Provide login functionality with username and password.
   * Implement secure authentication to protect sensitive payroll data.
8. **Data Storage and Backup**
   * Store all employee and payroll data in a secure database.
   * Support periodic backups to prevent data loss.
9. **Statutory Compliance**
   * Ensure payroll processing adheres to income tax, labor laws, and other statutory requirements.
10. **User Notifications**
    * Notify employees when payslips are generated or salaries are credited.

**Non-Functional Requirements**

1. **Performance:**
   * The system should load the home page within 2–3 seconds.
   * Search results must be displayed within 1 second for up to 100 ride entries.
2. **Scalability:**
   * The architecture should support additional features like payment gateways, live tracking, and AI-based route optimization in the future.
3. **Usability:**
   * Provide an intuitive interface that minimizes the number of clicks needed for key operations.
   * Ensure responsive design for compatibility with mobile and desktop devices.
4. **Reliability:**
   * Ensure 99% uptime for hosted services.
   * Implement proper error handling to avoid crashes and data loss.
5. **Security:**
   * Follow best practices for authentication, authorization, and data encryption.
   * Protect against common vulnerabilities like SQL injection, XSS, and CSRF attacks.
6. **Maintainability:**
   * Use modular coding practices and proper documentation to simplify future updates.
   * Employ version control for team collaboration and rollback capability.
7. **Portability:**
   * The system should be deployable on multiple platforms (Windows/Linux servers).
   * Support cloud deployment for scaling as user demand grows.

**3. Methodology**

**3.1 System Architecture Overview**

1. **Application Layer (Business Logic):**
   * Implemented using **Node.js with the Express.js framework**.
   * Contains all the routing, request handling, and middleware execution.
   * Divided into controllers and middleware to ensure separation of concerns.
   * Includes features such as authentication, ride management, chat functionality, and admin-level checks.
2. **Data Layer (Persistence):**
   * Powered by **MongoDB** using **Mongoose** ODM (Object Data Modeling).
   * Stores persistent data for users, carpools, and chats.
   * Defines data models with validation, constraints, and relationships (e.g., linking users to carpools).

The three layers interact in the following way:

* A user interacts with the **EJS frontend** in the browser.
* Requests are sent to the **Express.js server** via HTTP routes.
* Middleware authenticates and validates these requests.
* Business logic executes and retrieves/stores data through **Mongoose models**.
* Processed data is rendered back to the frontend via EJS templates.

This modular architecture ensures **scalability, maintainability, and security** while providing a clean separation between the presentation, logic, and data layers.

**3.2 Middleware Architecture**

Middleware in Employee Payroll acts as an intermediate processing layer between incoming requests and the final execution of route handlers. It ensures **security, access control, and session management**. Two custom middleware modules are present in the project:

1. **Authentication Middleware (auth.js):**
   * Ensures only authenticated users can access protected resources.
   * Likely uses JWT or session-based authentication (based on .env secrets).
   * Common functions:
     + Check if a user is logged in.
     + Redirect unauthenticated users to the login page.
2. **Admin Middleware (admin.js):**
   * Provides role-based access control.
   * Ensures only users with the admin role can perform administrative tasks (e.g., approving carpools, managing users).

**Other Middleware (Express-based):**

* **Body-parser / Express JSON parser:** Handles form data and JSON payloads.
* **Cookie/session management:** Maintains user sessions across requests.
* **Error handling middleware:** Provides graceful error messages and prevents sensitive data leaks.

**3.3 Data Model Design**

**User Model (User.js):**

* Stores user account information.
* Fields may include:
  + username
  + email
  + password (hashed)
  + role (e.g., user/admin)
  + Profile details (optional: name, phone, etc.).
* Supports authentication and authorization.

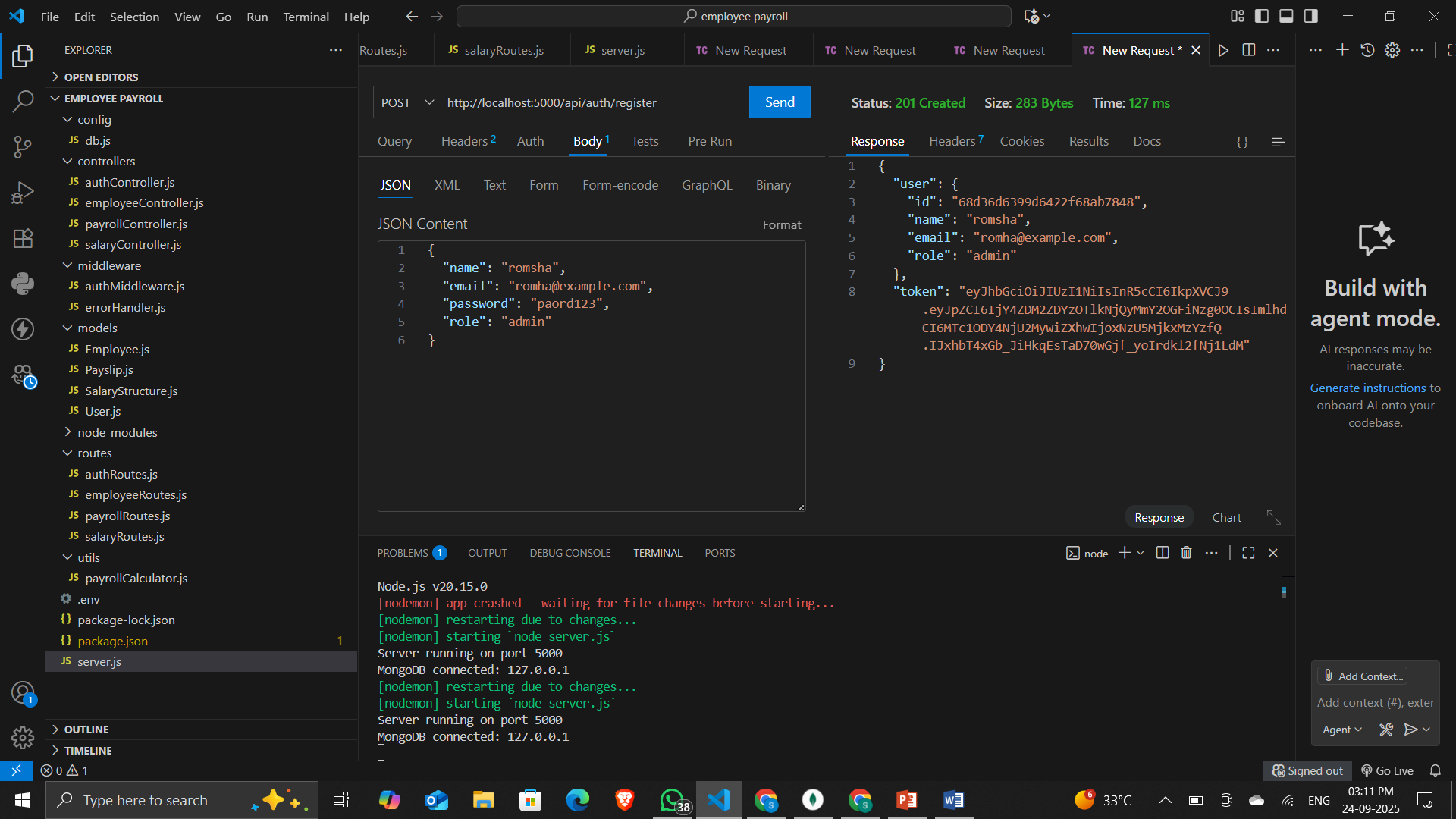
**Carpool Model (Carpool.js):**

* Represents a ride offer/request in the system.
* Fields may include:
  + driver (reference to User)
  + origin and destination
  + date/time
  + availableSeats
  + passengers (array of User references)
* Enables users to create, join, or search for carpools.

**Chat Model (Chat.js):**

* Enables communication between carpool participants.
* Fields may include:
  + sender (User reference)
  + receiver (User reference or group chat ID)
  + message
  + timestamp
* Enhances collaboration and coordination among users.

**3.4 File Structure**



**3.5 Security Implementation**

**Environment Variables (.env):**

* Stores secrets like database URI, JWT keys, and session secrets.
* Prevents hardcoding sensitive data in source code.

**Password Security:**

* User passwords are hashed (likely using **bcrypt**) before storage in MongoDB.
* Prevents exposure of plaintext passwords even if the database is compromised.

**Authentication:**

* Middleware verifies user sessions or JWT tokens.
* Unauthenticated users are redirected to the login page.

**Authorization:**

* Role-based access control via **admin middleware**.
* Ensures that only admins can perform system-wide actions.

**Session & Cookie Security:**

* Uses secure cookies with flags like HttpOnly and Secure.
* Prevents common session hijacking attacks.

**Input Validation & Sanitization:**

* Express middleware validates user input before storing it in MongoDB.
* Reduces risks of injection attacks (e.g., NoSQL Injection, XSS).

**Error Handling & Logging:**

* Centralized error handlers prevent leaking stack traces.
* Security events can be logged for auditing.

**4. Results**