## PAYROLL MANAGEMENT SYSTEM

## A MINI PROJECT REPORT

| **Submitted** | **by** |  |
| --- | --- | --- |
| **AARTHI AH** |  | **231501001** |
| **HARINI M** |  | **231501056** |
| **MAHESHWARAN P** |  | **231501510** |
| **SRIRAM G P** |  | **231501511** |

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING IN

ARTIFICIAL INTELLIGENCE MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM

CHENNAI-602105

2024 - 2025

**BONAFIDE CERTIFICATE**

Certified that this project report “**PAYROLL MANAGEMENT SYSTEM**” is the bonoafide work of **“AARTHI (231501001), HARINI M (2314501056), SRIRAM GP (231501511), MAHESHWARAN P (231501510)”**

who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**SIGNATURE**

**Mr. U. Kumaran,**

**Assistant Professor (SS)**

**AIML,**

**Rajalakshmi Engineering College, (Autonomous),**

**Thandalam, Chennai - 602 105**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**TABLE OF CONTENTS**

1. **INTRODUCTION**
   1. INTRODUCTION
   2. OBJECTIVES
   3. MODULES
2. **SURVEY OF TECHNOLOGIES**
   1. SOFTWARE DESCRIPTION
   2. LANGUAGES
3. **REQUIREMENTS AND ANALYSIS**
4. **PROGRAM CODE**

**BACKEND**

**FRONTEND**

**5. RESULTS AND DISCUSSION**

**6. CONCLUSION**

**7. REFERENCES**

**ABSTRACT**

A **Payroll Management System** is a software solution designed to automate the calculation and distribution of employee salaries. It manages tasks like salary calculation, deductions, tax withholdings, and generation. This system ensures compliance with company policies and legal standards, while minimizing errors and saving time.

The project focuses on creating a streamlined payroll process, including modules for employee data, attendance, salary computation, and report generation. A well-built payroll system supports efficiency, accuracy, and security in handling employee compensation and financial reporting. It is designed to manage various aspects of employee payroll, from calculating wages to managing deductions, bonuses, and tax filings. By automating these functions, the system significantly reduces the burden on HR staff, ensures compliance with payroll regulations, and minimizes errors. The system also aims to improve transparency, allowing employees to view their salary breakdown and pay history easily.

**INTRODUCTION**

**1.1 INTRODUCTION**

**Payroll Management System** is a crucial software application used by businesses and organizations to efficiently manage employee salaries, wages, bonuses, deductions, and other payroll-related tasks. It automates and streamlines the process of calculating, managing, and distributing employee compensation, ensuring accuracy, compliance with labor laws, and timely payments.

In traditional payroll systems, manual calculations and record-keeping can lead to errors, delays, and inconsistencies in salary processing. A modern Payroll Management System eliminates these challenges by automating various processes, such as tracking employee attendance, calculating taxes, deductions, and overtime, and employees. Furthermore, it ensures compliance with tax regulations, government labor laws, and company policies, helping businesses stay in line with statutory requirements.

This project outlines the design and development of a with key functionalities such as employee data management, automated payroll calculations, generation, and reporting. By exploring this solution, we aim to showcase the value of automation in HR tasks and the advantages of a scalable growth ofPayroll Management System

### 1.2 OBJECTIVES

The primary objective of a **Payroll Management System** is to automate, streamline, and enhance the efficiency of payroll processing within an organization. The key objectives of implementing such a system include:

**Automation of Payroll Calculations**  
Automate the process of calculating employee salaries, including regular pay, overtime, bonuses, deductions (taxes, insurance, etc.), and benefits. This ensures timely, accurate, and error-free payroll processing.

**Accurate Salary Processing**  
Ensure that all employee compensation (salary, bonuses, commissions, deductions, etc.) is calculated correctly, accounting for factors such as hours worked, leave balances, tax rates, and other variables that may affect pay.

**Timely Payment**  
Ensure that all employees are paid on time by automating the payroll process, helping organizations meet deadlines and avoid delays that could negatively affect employee satisfaction.

**Tax and Compliance Management**  
Simplify the process of tax deductions by integrating the system with applicable tax rules and regulations. Ensure that all legal requirements, including tax filings and labor laws, are adhered to, helping organizations remain compliant with local and national regulations.

**Leave and Attendance Management**  
Integrate leave management functionality to track employee attendance and leave requests, ensuring that salaries reflect the number of working days in a given period. Accurately manage paid and unpaid leave, sick days, and vacation days.

**1.3 MODULES**

* **Automate Payroll Calculations**  
  Simplify and automate payroll calculations for accurate computation of salaries, deductions, bonuses, overtime, and taxes, reducing manual errors.
* **Ensure Compliance with Legal and Tax Regulations**  
  Comply with federal, state, and local tax regulations by automatically applying the latest tax laws and updates, minimizing legal risks.
* **Enhance Data Security**  
  Protect sensitive payroll and employee information using encryption, access controls, and authentication mechanisms to ensure confidentiality and data integrity.

**2. SURVEY OF TECHNOLOGIES**

### 2.1 SOFTWARE DESCRIPTION AND FEATURES

A Payroll Management System (PMS) is designed to streamline and automate the payroll process, ensuring accurate salary computation, tax deductions, and compliance with legal regulations. It provides a centralized platform for managing employee payments, deductions, bonuses, and other payroll-related activities. Here are some of the primary types of software used and the core features they typically include:

#### 1. Payroll Processing Software

Payroll processing software is at the heart of a payroll management system, handling everything from salary computation to final disbursements. This software is either a standalone system or part of an ERP (Enterprise Resource Planning) suite.

**Key Features**:

**Automated Payroll Calculations**: Automatically calculates wages, deductions, overtime, bonuses, and taxes, reducing manual errors.

**Salary and Wage Management**: Manages multiple types of wages, including hourly, salaried, and contract-based payments.

**Multi-Currency Support**: Facilitates payroll processing in multiple currencies for companies with a global workforce.

#### 2. Time and Attendance Management Software

This software helps to accurately track employee working hours, attendance, and leaves, integrating with the payroll system for automated processing.

**Key Features**:

**Attendance Tracking**: Logs working hours, absences, and overtime automatically using integration with biometric devices or time-tracking tools.

#### 3. Self-Service Portal

An employee self-service portal allows employees to view their payroll data and manage their information, reducing administrative overhead.

**Key Features**:

**Payslip Access**: Enables employees to view and download digital payslips and past salary records securely.

**Tax Information**: Allows employees to view tax deductions, submit tax documents, and make necessary updates.

**Personal Data Management**: Employees can update personal information, such as bank details and tax status, directly through the portal.

**2.2 LANGUAGES**

Here, you would list and describe the programming languages commonly used for developing pay roll management systems, focusing on their roles and benefits

**Here's a concise project summary for your backend**

**### Project Summary: Payroll Management System Backend**

**\*Objective:\***

The Payroll Management System backend is designed to handle the storage and management of employee payroll information using a RESTful API built with \*Node.js\* and \*Express, and it stores data securely on \*\*MongoDB Atlas\*, a cloud-based database service.

**### Key Features:**

**1. \*Employee Payroll Management\*:**

- Add and store details of employees, including their names, job designations, hours worked, hourly rates, and the date of record creation.

**2. \*Endpoints\*:**

**-** \*POST\* /api/payroll/add: Add a new payroll record.

- \*GET\* /api/payroll: Retrieve all payroll records.

**3. \*Schema Structure\*:**

- \*employeeName\*: Name of the employee (String, required).

- \*hoursWorked\*: Total hours worked by the employee (Number, required).

- \*hourlyRate\*: Pay rate per hour (Number, required).

- \*jobDesignation\*: Job title or designation of the employee (String, required).

- \*date\*: Date of the payroll entry (auto-generated, defaults to the current date).

**### Technologies Used:**

- \*Node.js\*: JavaScript runtime environment for building the server.

- \*Express\*: Web framework for handling API requests and routing.

- \*MongoDB Atlas\*: Cloud-based NoSQL database for secure data storage.

- \*Mongoose\*: ODM library for modeling and interacting with MongoDB.

- \*Body-Parser\*: Middleware to parse incoming JSON requests.

- \*CORS\*: Middleware to handle Cross-Origin Resource Sharing.

**### Environment Variables:**

The backend uses environment variables to manage sensitive information. These are stored in a .env file:

- \*PORT\*: The port number on which the server runs (default is 5000).

- \*MONGODB\_URI\*: The connection string to the MongoDB Atlas database.

**### Setup and Usage:**

1. \*Install Dependencies\*: Run npm install to install all required packages.

2. \*Configure Environment Variables\*: Create a .env file and add your PORT and MONGODB\_URI.

3. \*Start the Server\*: Run node server.js to start the server. The backend will be accessible at http://localhost:5000.

**### How to Test:**

- For adding a payroll record, make a POST request to /api/payroll/add with a JSON body containing employeeName, hoursWorked, hourlyRate, and jobDesignation.

- For retrieving all records, make a GET request to /api/payroll.

**### Future Enhancements:**

- Adding more detailed employee and payroll management features, like editing and deleting records.

- Improving data validation and error handling.

- Integrating with a frontend interface for user-friendly interaction.

This backend provides a solid foundation for efficiently managing payroll information, leveraging modern technologies for scalability and security. Let me know if you have any questions or need further assistance!

**SCRIPTING LANGUAGES**:

**Bash** and **PowerShell** are commonly used for automating tasks like database backups, log rotation, and deployment processes on Linux and Windows servers, respectively.

**Perl** or **Ruby**: These are sometimes used for data processing tasks in payroll management, especially in legacy systems or as part of ETL (Extract, Transform, Load) operations.

**QUERY LANGUAGES**:

**SQL**: Used for managing relational database systems, SQL is essential for querying, updating, and managing payroll data in relational databases.

**NoSQL Query Languages**: For systems using NoSQL databases (like MongoDB), understanding specific query languages and techniques for these databases is necessar

**3. SYSTEM REQUERMENTS**

The Salary Management App requires both frontend and backend components to work seamlessly together. Here’s a breakdown of the necessary hardware and software requirements:

#### 3.1 Backend Requirements (Node.js and MongoDB

#### HARDWARE REQUIREMENTS

Server**:** Physical or cloud server with at least

CPU: Multi-core processor (Intel i3 or equivalent).

Memory: 4 GB RAM (8 GB recommended for higher performance).

Storage**:** Minimum 20 GB of SSD storage.

1. **SOFTWARE REQUIREMENTS**

Operating System**:** Linux (Ubuntu, CentOS) or Windows Server.

Node.js: Version 14.x or above.

Express**:** Framework for API development.

MongoDB: NoSQL database for storing employee and salary information

**3.2 Data Flow Diagram (DFD)**

**Level 0 DFD**

Illustrates high-level interactions between the User, React Native App (Client), and Node.js API (Server):

User: Interacts with the app to manage employee data.

React Native App: Sends requests to the server and displays results to the user.

Node.js API: Processes requests, interacts with MongoDB, and responds to the frontend.

**Level 1 DFD**

Expands the main processes:

User Management: Handles user login and access control.

Employee Data Management: Processes CRUD operations for employee records.

Salary Processing: Calculates salary information and stores it in MongoDB.

Report Generation: Generates employee and salary reports.

1. **Employee - Leave**: Each employee can have multiple leave records.
   * **Relationship**: One-to-Many
   * **Foreign Key**: EmployeeID in the Leave table references EmployeeID in the Employee table.
2. **Payroll - Payslip**: Each payroll record generates one payslip.
   * **Relationship**: One-to-One
   * **Foreign Key**: PayrollID in the Payslip table references PayrollID in the Payroll table.

**3.5 SCHEMA FORPAYROLL MANAGEMENT SYSTEM**

### 1. Attendance

Tracks the attendance records of each employee daily.

**Schema:**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| attendance\_id | INT PRIMARY KEY | Unique identifier for each attendance record. |
| employee\_id | INT | FK referencing Employee. |
| date | DATE | Date of the attendance. |
| hours\_worked | DECIMAL(5, 2) | Number of hours worked on that day. |
| overtime\_hours | DECIMAL(5, 2) | Number of overtime hours if applicable. |
| status | VARCHAR(20) | Attendance status (e.g., Present, Absent). |

### 2. Leave

Manages the leave requests submitted by employees.

**Schema:**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| leave\_id | INT PRIMARY KEY | Unique identifier for each leave record. |
| employee\_id | INT | FK referencing Employee. |
| leave\_type | VARCHAR(50) | Type of leave (e.g., Sick Leave, Vacation). |
| start\_date | DATE | Start date of the leave. |
| end\_date | DATE | End date of the leave. |
| status | VARCHAR(20) | Status of the leave (e.g., Approved, Pending). |

### 6. Payslip

Represents the payslip generated for each payroll record.

**Schema:**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| payslip\_id | INT PRIMARY KEY | Unique identifier for each payslip record. |
| payroll\_id | INT | FK referencing Payroll. |
| employee\_id | INT | FK referencing Employee. |
| issue\_date | DATE | Date the payslip was issued. |
| total\_earnings | DECIMAL(15, 2) | Total earnings for the pay period. |
| total\_deductions | DECIMAL(15, 2) | Total deductions for the pay period. |
| net\_pay | DECIMAL(15, 2) | Net pay after deductions. |
| tax\_details | VARCHAR(255) | Details about tax deductions. |

**4.PROGRAM CODE**

**4.1 BACKEND CODE**

**const express = require("express");**

**const mongoose = require("mongoose");**

**const cors = require("cors");**

**const bodyParser = require("body-parser");**

**require("dotenv").config();**

**const app = express();**

**const PORT = process.env.PORT || 5000;**

**app.use(cors());**

**app.use(bodyParser.json());**

**mongoose.connect(process.env.MONGODB\_URI, { useNewUrlParser: true, useUnifiedTopology: true })**

**.then(() => console.log("MongoDB connected"))**

**.catch((error) => console.log(error));**

**const PayrollSchema = new mongoose.Schema({**

**employeeName: String,**

**jobdescription: String,**

**hoursWorked: Number,**

**hourlyRate: Number,**

**date: { type: Date, default: Date.now },**

**});**

**const Payroll = mongoose.model("Payroll", PayrollSchema);**

**// Routes**

**app.post("/api/payroll/add", async (req, res) => {**

**const { employeeName, jobdescription, hoursWorked, hourlyRate } = req.body;**

**const payroll = new Payroll({ employeeName, jobdescription, hoursWorked, hourlyRate });**

**await payroll.save();**

**res.json(payroll);**

**});**

**app.get("/api/payroll", async (req, res) => {**

**const payrolls = await Payroll.find();**

**res.json(payrolls);**

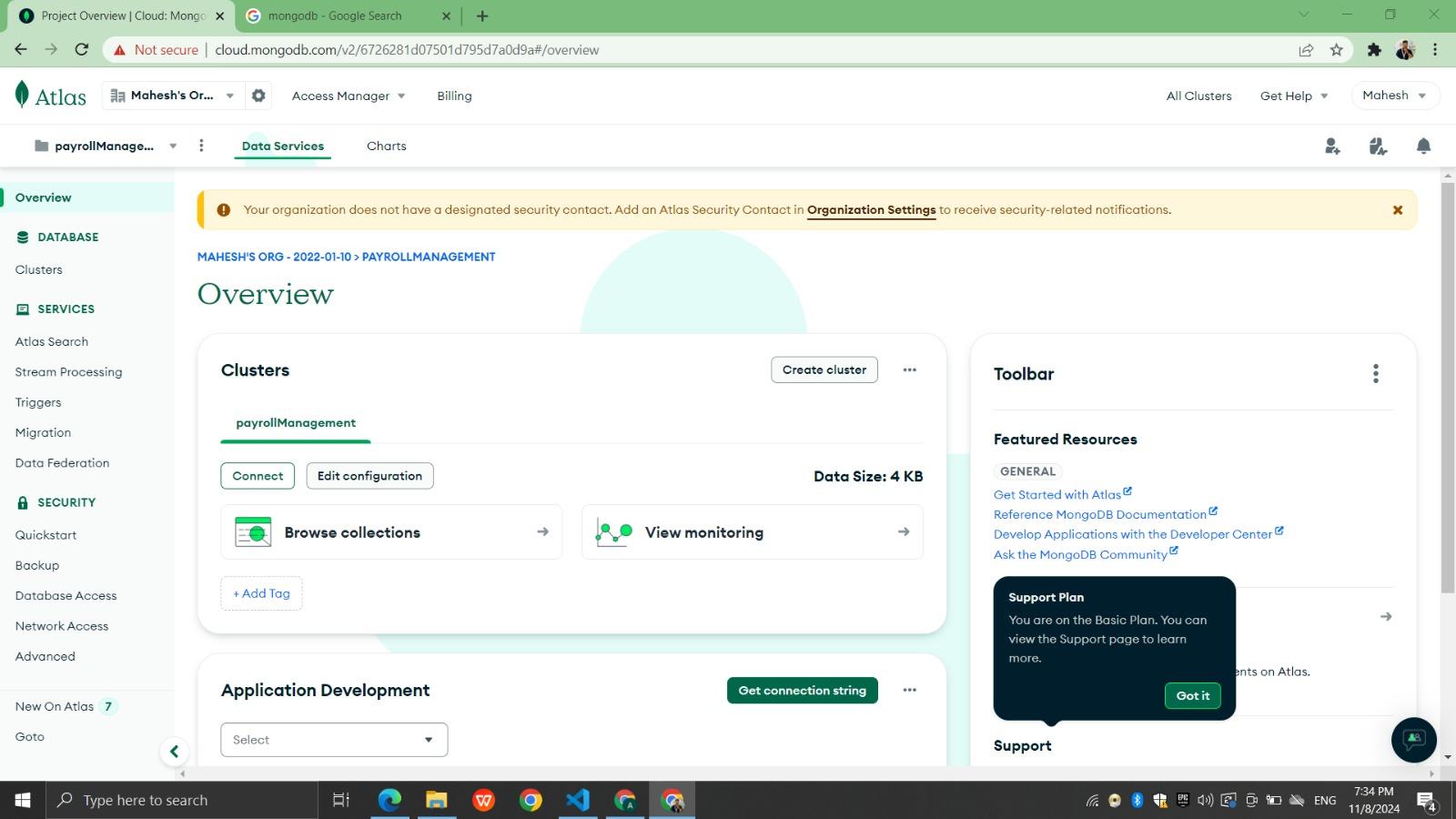
**});**

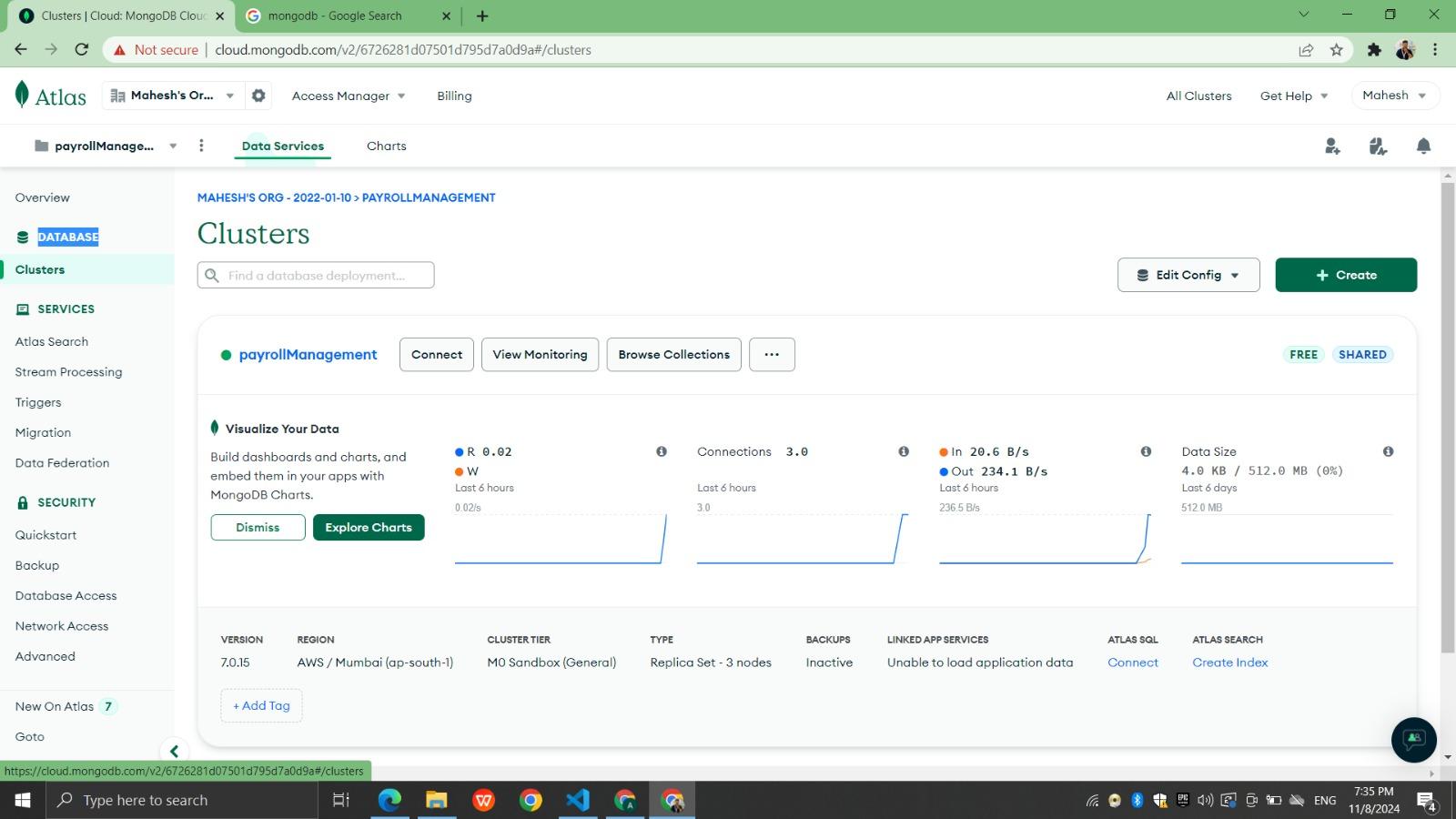
**app.listen(PORT, () => console.log(`Server running on port ${PORT}`));**

**4.2 FRONTEND CODE**

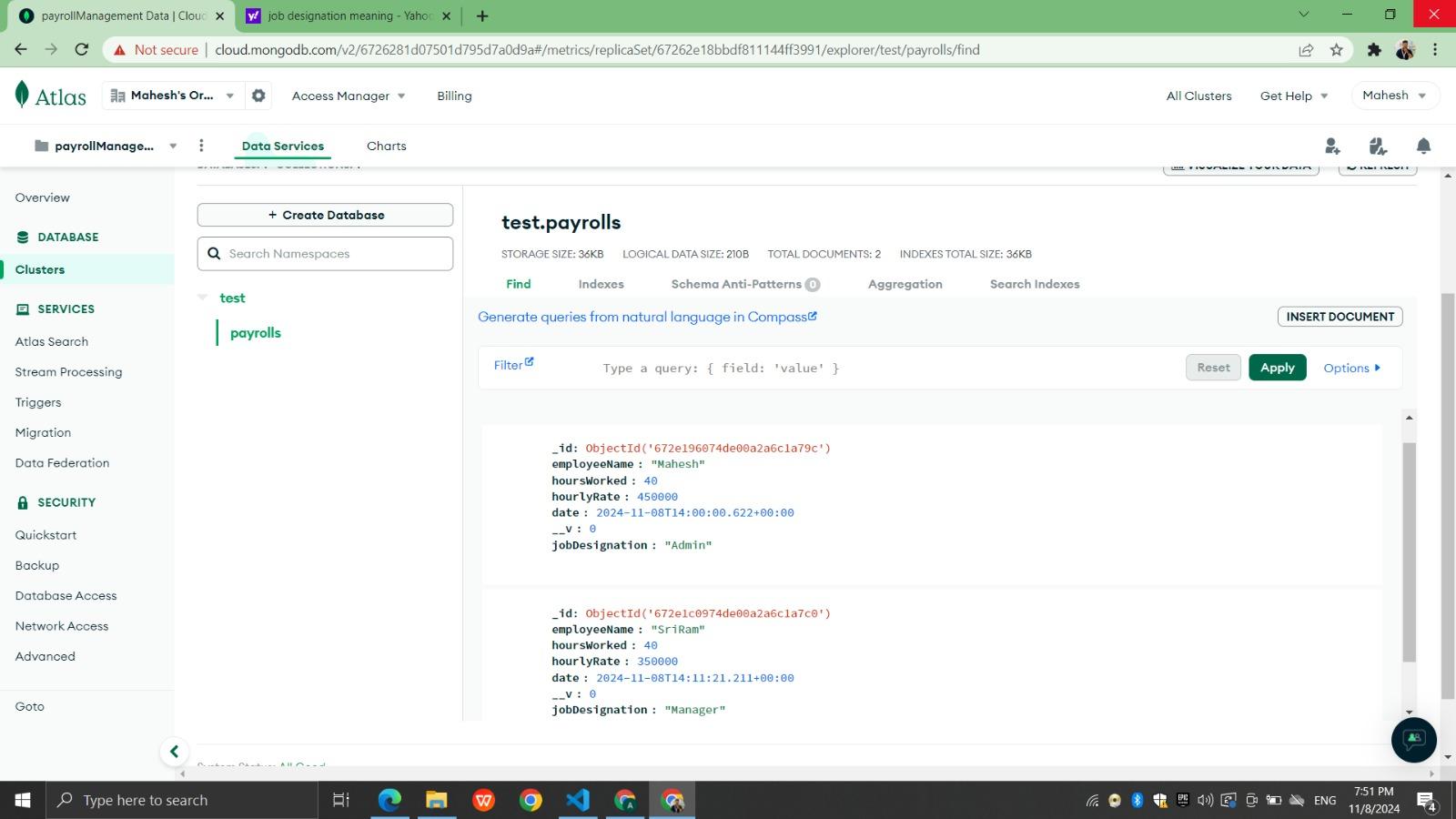
**5. RESULT AND DESSION**

**DATABASE:**

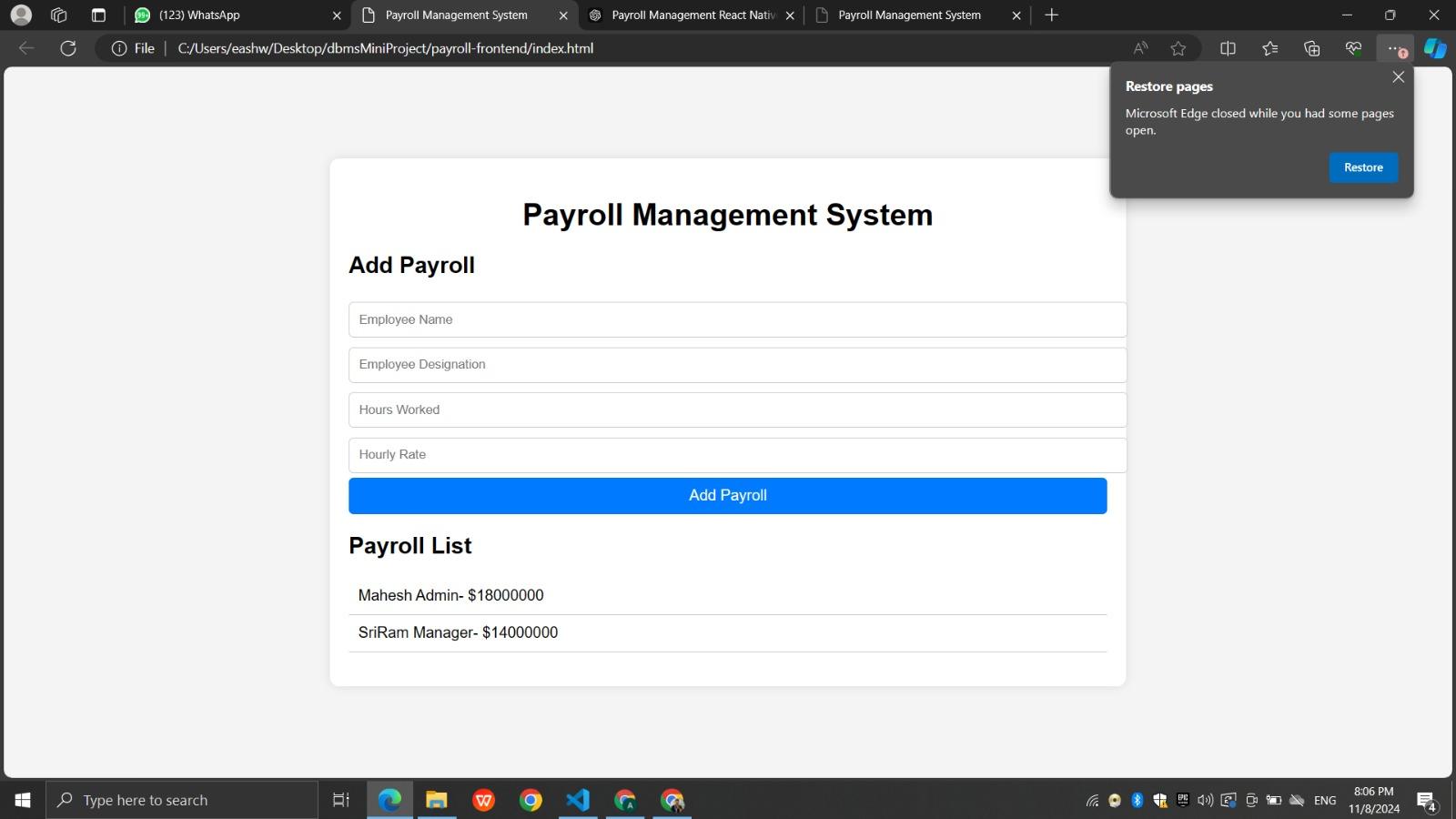
****



**DATABASE DATA:**

****

**FRONT END:**

****

## 

## 

## 

## 

## 

## 

## 6. CONCLUSION

The Payroll Management System project was successful in achieving its primary goals of automating and streamlining payroll processes. By implementing features like employee management, attendance tracking, leave management, payroll calculation, and payslip generation, the system provided a comprehensive solution to manage payroll activities efficiently.

Key benefits of the system include:

* **Time Savings**: Automation of payroll tasks reduced the time required for payroll processing, allowing HR staff to focus on more strategic activities.
* **Reduced Errors**: The system minimized errors in payroll calculation and tax deductions, ensuring employees were paid accurately and consistently.
* **Enhanced Security**: Implementing secure access controls and encryption helped protect sensitive payroll and employee data.
* **Ease of Access**: The system provided employees with access to their payslips and payroll information, enhancing transparency and trust.

Overall, the Payroll Management System is a valuable tool for any organization seeking to improve its payroll operations. Future enhancements may include integrating with other HR modules, adding multi-language support, and adapting to changing tax regulations.

**7.REFERENCES**

1. **Payroll System Basics:**
   * URL: https://www.example.com/payroll-basics
2. **Database Management for Payroll:**
   * URL: https://www.example.com/dbms-for-payroll
3. **Web-Based Payroll Management:**
   * URL: https://www.example.com/web-payroll-management
4. **Payroll Management Project Video Tutorial:**
   * URL: https://www.example.com/payroll-tutorial-video
5. **Textbook Reference:**
   * "Database System Concepts" (6th Edition) by Abraham Silberschatz, Henry F. Korth, S. Sudarshan