ELECTIVE-I

MINI PROJECT REPORT ON

“Employee Management System”

Submitted to the

Savitribai Phule Pune University

In partial fulfilment for the award of the Degree of

Bachelor of Engineering in

Information Technology

By

Name Of Student	Seat No
Akash Shinde	T1902608503
Ashish Vidhate	T1902608504
Gagan Dhanapune	T1902608517
Om Barvekar	T1902608506
Rushikesh Chame	T1902608512

Under the guidance of

Prof. Priyanka Kaurav

Department Of Information Technology

PDEA’s College Of Engineering Manjari (BK)

Pune, Maharashtra

2024-25

CERTIFICATE

This is to certify that the Mirco Project report entitled **“Employee Management System”** being submitted by Akash shinde,Ashish vidhate,Gagan Dhanapune,Chame Rushikesh,Om Baravekar (Seat No: T1902608503, T1902608504, T1902608517 , T1902608512, T1902608506| Class: TE/IT) is a record of bonafide work carried out by her under the supervision and guidance of Prof. Priyanka Kaurav in partial fulfilment of the requirement for TE Information Technology Engineering 2019 Pattern of Savitribai Phule Pune University, Pune in the academic year 2024-25

Date:

Place:

Name of the Mentor	Name of the HOD
Prof.Priyanka Kaurav	Dr.D.S.Hirolikar

This MicroProject report has been examined by us as per the Savirtibai Phule Pune University, Pune, requirements at Pune district education association’s College of Engineering Manjari (BK), Pune.

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to **Prof. Priyanka Kaurav** for her guidance and constant supervision as well as for providing necessary information regarding the project & also for their support.

We would like to express our gratitude towards my parents & member of the Information Technology Department of PDEA collage of Engineering Manjari (BK) Pune, for their kind cooperation and encouragement which help us in the completion of this project.

We would like to express our special gratitude and thanks to All other Professors in the Department for giving us such attention and time.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

Thanking You

!Akash shinde

Ashish Vidhate

Gagan Dhanapune

Chame Rushikesh

Om Baravekar

INDEX

1. ABSTRACT

The Employee Management System is a digital tool designed to help organizations efficiently manage employee data. It allows administrators to add, update, view, and delete employee information such as names, ages, positions, departments, and salaries. By using MongoDB for data storage and Express.js on Node.js for the backend, the system ensures that data is stored securely and can be easily accessed or modified.

To keep the system secure, it includes a login feature with JWT (JSON Web Tokens) authentication, ensuring only authorized users can access or make changes to employee records. The frontend is designed to be user-friendly and interacts with the backend API, making it easy for users to perform operations and retrieve information as needed.

Overall, the Employee Management System reduces manual paperwork, improves accuracy, and saves time, making it a reliable and efficient solution for modern organizations.

2. INTRODUCTION

1. Problem Statement

Managing employee information accurately and efficiently is crucial for organizations. Traditional methods often result in errors, delayed updates, and limited scalability, leading to inefficiencies in record-keeping. This system addresses these challenges by providing a secure, centralized solution to manage employee data effectively.

2. Motivation

In today's fast-paced work environment, organizations need reliable tools that streamline data management and minimize errors. A modern, digital system for managing employee records ensures that organizations can keep information accurate, secure, and up-to-date, helping them operate more smoothly and make informed decisions.

3. Objectives

1.Secure Data Storage : Implement a database to securely store all employee information, ensuring data is well-organized and protected. MongoDB is used for flexible and scalable storage, handling various data types with ease. This structure supports reliable, centralized data management.

2. Real-Time Operations: Enable the ability to add, view, update, and delete employee records instantly. This keeps information accurate and up-to-date without delays, benefiting operational efficiency. All actions are executed directly in the database, ensuring data consistency.

3. Access Control: Protect employee data using JWT-based authentication, allowing only authorized users to access or edit records. This enhances data security by restricting unauthorized access. User roles can further control permissions as needed.

4.User-Friendly Design: Provide a simple and intuitive interface to streamline employee data management. The responsive design allows easy access from different devices, making it convenient for users. Clear navigation improves productivity by simplifying data entry and retrieval.

3. Software and Hardware Requirement

Software Requirements:

- Operating System: Windows 11
- Programming Language:

  Front-end : HTML , CSS , Java script
  Back-end : express.js , node.js
  Database Management System: Mongo DB database (NoSQL)

Hardware Requirements:

- Processor : Intel i5 used
- RAM : 8 GB used
- Storage: At least 256 GB
- Network Connectivity : A stable internet connection is required

4. Core Functionalities

 Employee Records Management

- Adding, updating, and deleting employee details (name, age, position, department, salary, etc.)
- Storing and retrieving personal and professional data securely.

 Authentication & Authorization

- Secure user login using JWT (JSON Web Tokens).
- Differentiating roles (e.g., admin, employee) with specific permissions.

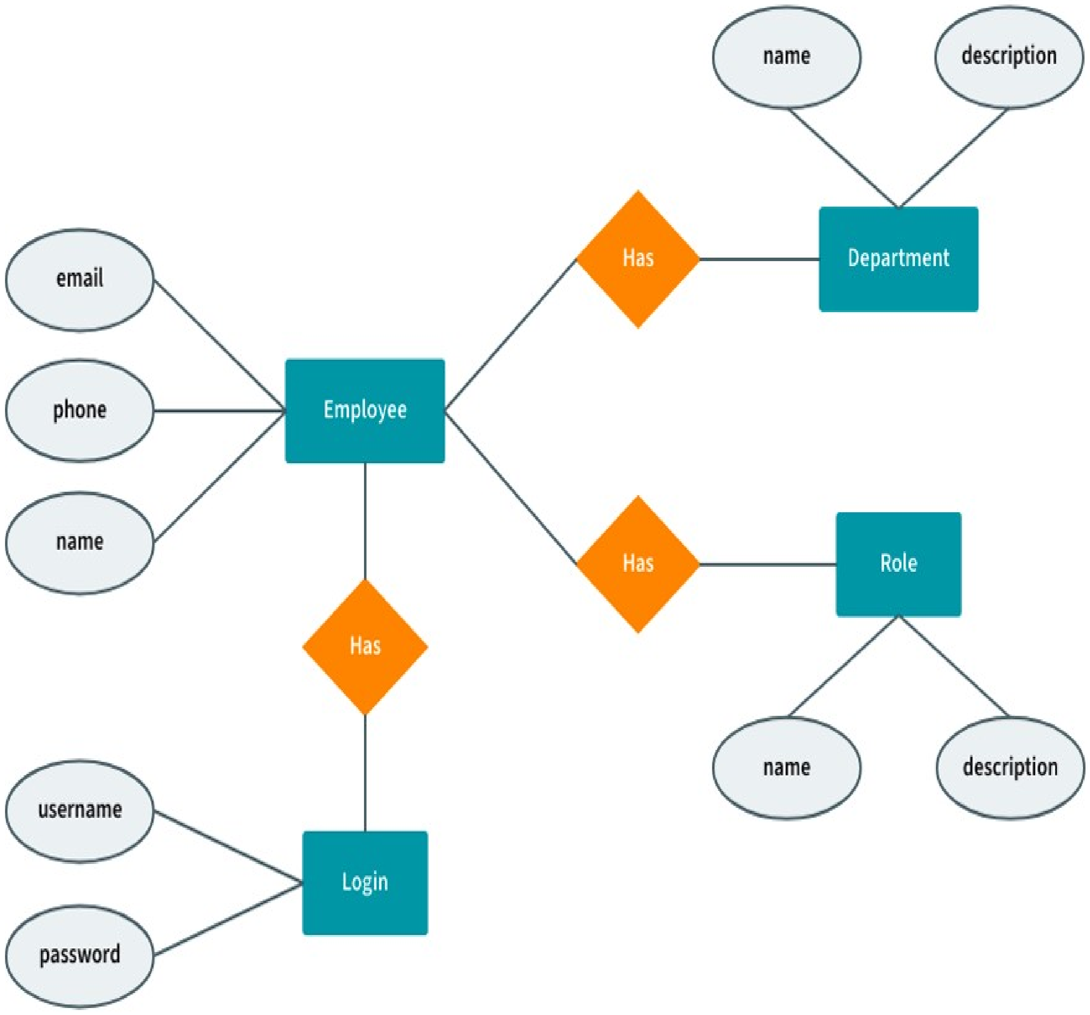
 Search and Filtering

- Search employees by criteria such as name, position, department, or age.
- Advanced filtering for more complex queries (e.g., by salary range or status).

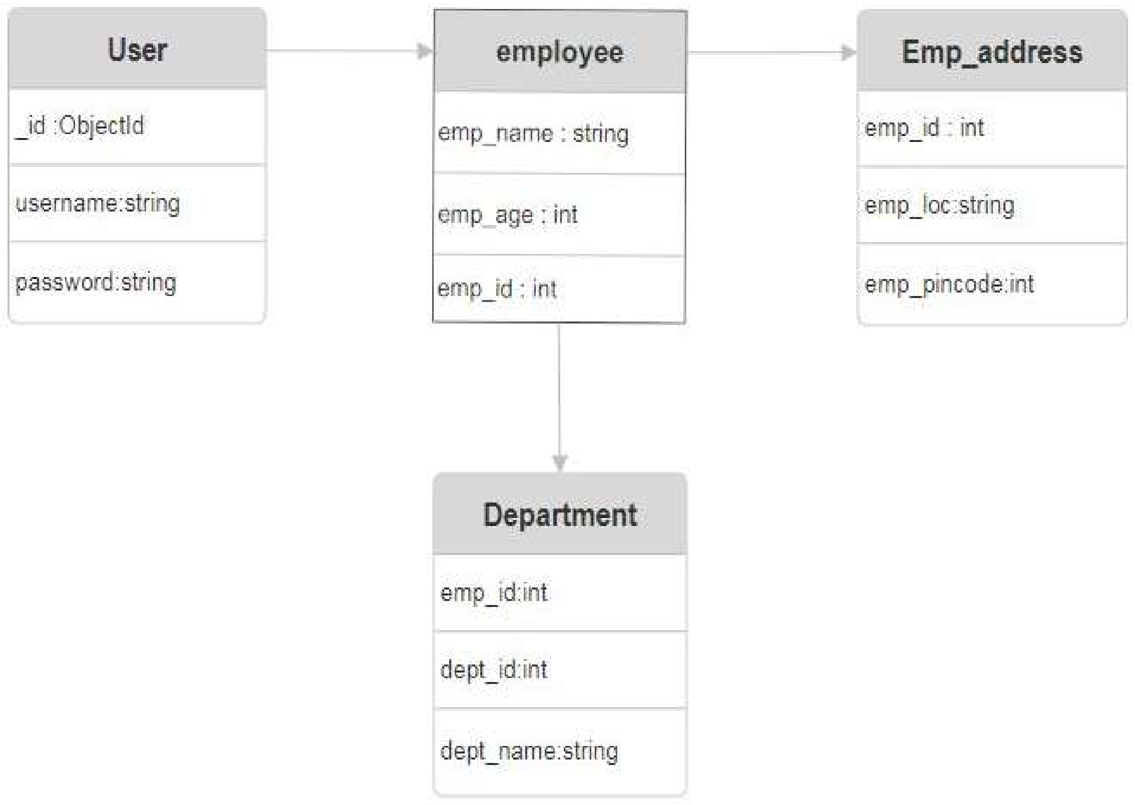
 CRUD Operations

- Support for Create, Read, Update, and Delete operations for employee records.

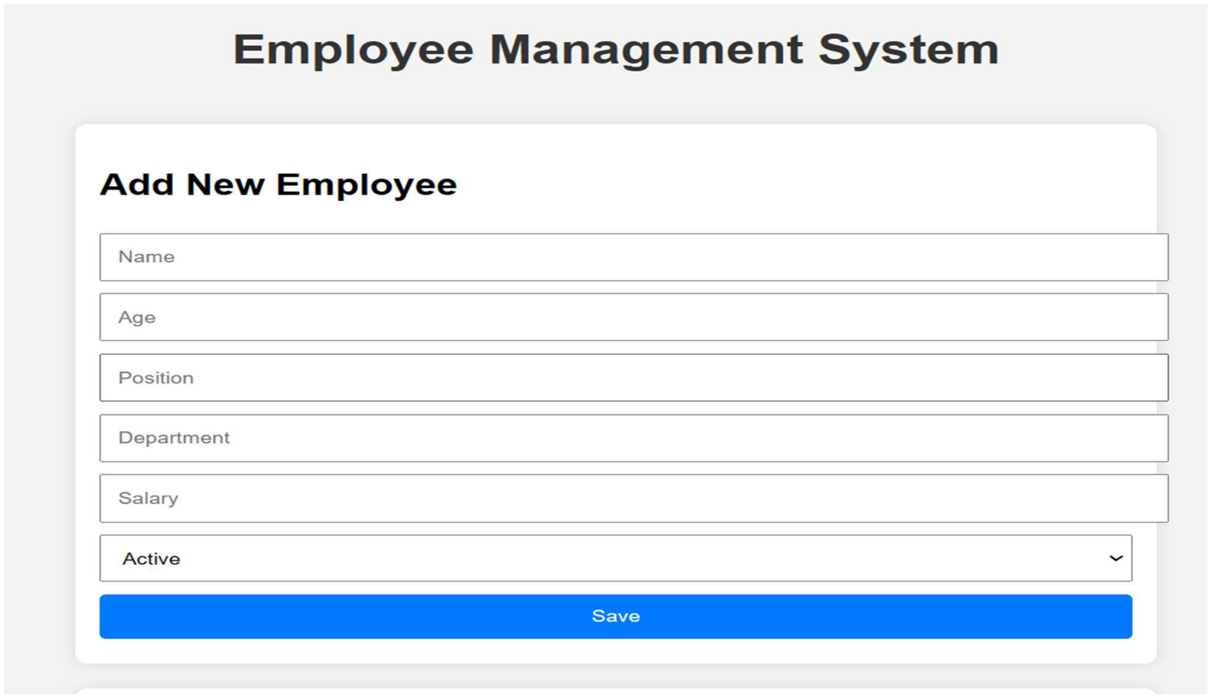
5. E-R DIAGRAM


name
description
email
Has
Department
phone
Employee
name
Has
Role
Has
username
name
description
Login
password

6. SCHEMA DIAGRAM

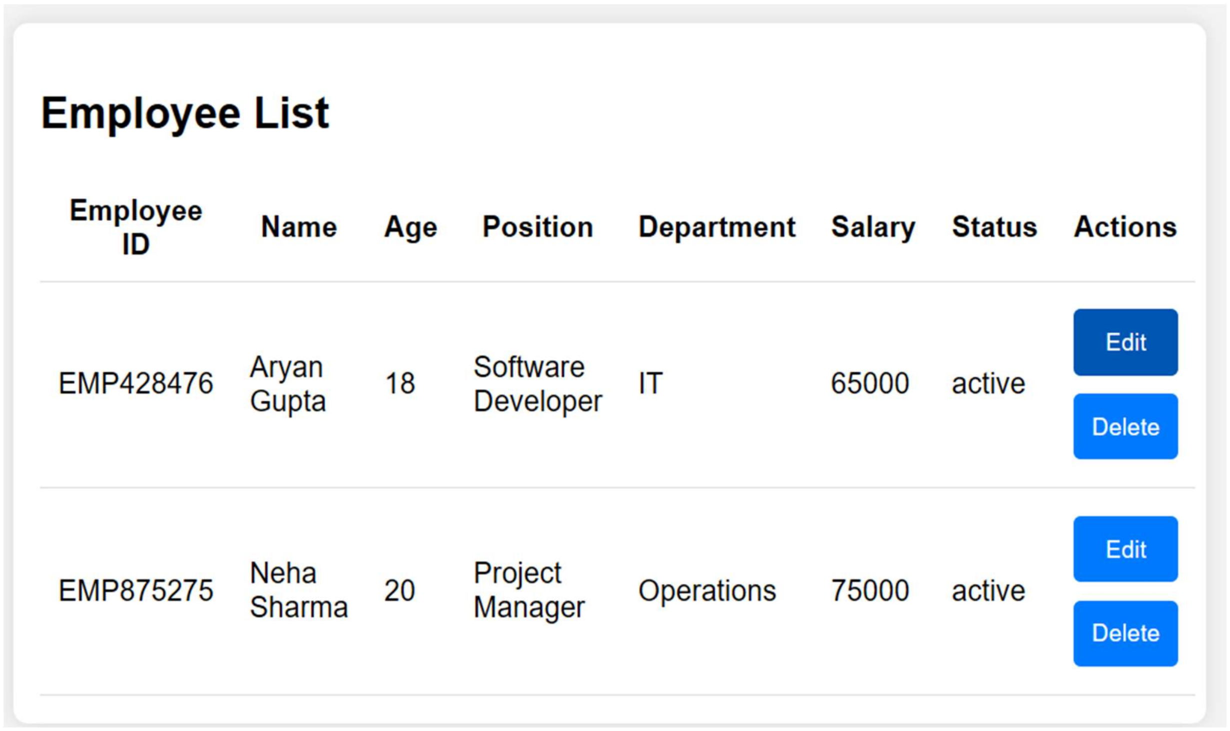


7. GUI Implementation



Employee Management System

Add New Employee



Employee List

Employee ID	Name	Age	Position	Department	Salary	Status	Actions
EMP428476	Aryan Gupta	18	Software Developer	IT	65000	active	Edit Delete
EMP875275	Neha Sharma	20	Project Manager	Operations	75000	active	Edit Delete

8. CREATING DATABASE USING MONGODB

1. Connect to MongoDB:

mongoose.connect('mongodb://localhost:27017/employee_management', { useNewUrlParser: true, useUnifiedTopology: true });

2. Create Employee Collection:

db.createCollection("employees");

3. Insert Multiple Student Records

```
db.employees.insertMany([
   {
      name: "Aryan gupta",
      employee_id: "EMP007",
      age: 18,
      address: "123 MG Road, Pune",
      contact: "9856780934",
      position: "Software Developer",
      department: "IT",
      status: "active",
      salary: 65000
   },
   {
      name: "Neha sharma",
      employee_id: "EMP008",
      age: 20,
      address: "456 FC Road, Pune",
      contact: "8967452309",
      position: "Project Manager",
      department: "Operations",
      status: "active",
      salary: 75000
   },
   {
      name: "Aditya Verma",
      employee_id: "EMP009",
      age: 21,
                address: "789 JM Road, Pune",
                contact: "9689814678",
                position: "Data Analyst",
                department: "Data Science",
                status: "active",
                salary: 60000
            }
]);
```

4. Query for Searching Employees by Name and Age

```
db.employees.find({
   name: "Aryan Gupta",
   age: { $gt: 18 }
})

db.employees.find({
   name: "Neha Sharma",
   age: { $gt: 20 }
})

db.employees.find({
   name: "Aditya Verma",
   age: { $gt: 21 }
})
```

9. CRUD Operations (Test Queries)

1. Create: Add a New Employee

```
db.employees.insertOne(
      {
       name: "Riya Patel",
       employee_id: "EMP0010",
       age: 27,
       address: "789 RJ Road, Satara",
       contact: "9689814678",
       position: "Data Analyst",
       department: "Data Science",
       status: "active",
       salary: 70000
   })
```

2. Read Operations

1.Search Employees by Employee ID

```
db.employees.find({ employee_id: "EMP007" })
```

2. Search Employees by Name

```
db.employees.find({ name: "Aryan Gupta" })
```

3. Search Employees by Age

```
db.employees.find({ age: { $gt: 18 } }) // Employees older than 18
```

4. Search Employees by Position

```
db.employees.find({ position: "Software Developer" })
```

3. Update Operations

1.Update Employee's Contact Information

```
db.employees.updateOne(
   { employee_id: "EMP007" },
   { $set: { contact: "9876543210" } }
)
```

2. Increase Employee's Age by 1 Year

```
db.employees.updateOne(
   { employee_id: "EMP007" },
   { $inc: { age: 1 } }
)
```

3. Change Employee's Status to "Inactive"

```
db.employees.updateOne(
   { employee_id: "EMP009" },
   { $set: { status: "inactive" } }
)
```

4. Delete Operations

1. Delete Employees Below a Certain Age

```
db.employees.deleteMany({ age: { $lt: 18 } }) // Delete employees younger than 18
```

2. Delete Employee by Employee ID

```
db.employees.deleteOne({ employee_id: "EMP007" })
```

5. Aggregate Queries for Employee Management System

1.Total Number of Employees by Department

```
db.employees.aggregate([
   {
        $group: {
            _id: "$department",  // Group by department
            total_employees: { $sum: 1 }  // Count the number of employees per department
        }
    }
])
```

2 .Average Age of Employees by Department

```
db.employees.aggregate([
    {
        $group: {
            _id: "$department",  // Group by department
            average_age: { $avg: "$age" }  // Calculate the average age per department
         }
    }
  ]
)
```

10. CONNECTING DATABASE USING MONGODB

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const Employee = require('./models/employee');

const app = express();
const PORT = 3000;

// Middleware
app.use(bodyParser.json());
app.use(cors());
app.use(express.static('public'));

// MongoDB Connection
mongoose.connect('mongodb://localhost:27017/employeeDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('MongoDB connected'))
.catch((err) => console.log(err));

// CRUD Routes

// Get all employees
app.get('/api/employees', async (req, res) => {
  try {
    const employees = await Employee.find();
    res.json(employees);
  } catch (error) {
    res.status(500).json({ message: 'Failed to fetch employees', error });
  }
});

// Get a specific employee by ID
app.get('/api/employees/:id', async (req, res) => {
  try {
    const employee = await Employee.findById(req.params.id);
    if (!employee) {
      return res.status(404).json({ message: 'Employee not found' });
    }
    res.json(employee);
  } catch (error) {
    res.status(500).json({ message: 'Failed to fetch employee', error });
  }
});
```

```
// Create a new employee
app.post('/api/employees', async (req, res) => {
  try {
    const { name, age, position, department, salary, status, contact, address } = req.body;
    const newEmployee = new Employee({
      name,
      age,
      position,
      department,
      salary,
      status,
      contact,   // Save contact
      address    // Save address
    });

    await newEmployee.save();
    res.status(201).json(newEmployee);
  } catch (error) {
    res.status(500).json({ message: 'Failed to add employee', error });
  }
});

// Update an employee
app.put('/api/employees/:id', async (req, res) => {
  try {
    const updatedEmployee = await Employee.findByIdAndUpdate(req.params.id, req.body, { new:
true });
    if (!updatedEmployee) {
      return res.status(404).json({ message: 'Employee not found' });
    }
    res.json(updatedEmployee);
  } catch (error) {
    res.status(500).json({ message: 'Failed to update employee', error });
  }
});

// Delete an employee
app.delete('/api/employees/:id', async (req, res) => {
  try {
    const deletedEmployee = await Employee.findByIdAndDelete(req.params.id);
    if (!deletedEmployee) {
      return res.status(404).json({ message: 'Employee not found' });
    }
    res.json({ message: 'Employee deleted' });
  } catch (error) {
    res.status(500).json({ message: 'Failed to delete employee', error });
  }
});
```

11. Scope for Future Enhancements

 Employee Performance Management:

- Adding features for performance reviews, goal-setting, and tracking employee achievements.

 Payroll Integration:

- Integration with payroll systems for salary processing, tax calculations, and generating pay slips.

 Leave Management System:

- Employees can request leaves, and HR can approve or deny requests.
- Track leave balances and generate reports.

 Attendance Tracking:

- Integration with time-tracking systems for monitoring employee attendance.

 Document Management:

- A file upload system for storing important employee documents (e.g., resumes, contracts, certifications).

 Notifications:

- Alerts for admins about upcoming contract expirations, leave approvals, or performance reviews.

12. FEATURE

1.Employee Record Management

Store detailed employee information (name, age, position, department, salary, contact details, etc.).

Ability to add, edit, and delete employee records.

2.Search and Filter Employees

Search employees by various criteria such as name, position, department, age, etc.

Advanced search with filters for better efficiency in finding employees.

3.Role-Based Access Control

Different access levels for users (admin, HR, manager, etc.) with permissions to add, edit, or view employee data.

Secure login with JWT authentication to ensure only authorized users can access sensitive data.

4.Employee Performance Tracking

Track salary details and updates, including bonuses and deductions.

Generate reports on salary distribution and growth within the company.

Leave and Attendance Management

13. CONCLUSION

The Employee Management System allows organizations to securely manage employee data in one place. It uses MongoDB for storing information and JWT authentication for secure access. This project is easy to scale and can grow with the organization's needs. It shows how to build a full-stack application, combining both front-end and back-end technologies. The system allows users to add, update, search, and delete employee records. It also includes features to search employees by different criteria, such as name, position, and department. This project is a great starting point for improving and expanding with more features in the future.

14.REFERENCES

1. MongoDB Documentation : Learn about MongoDB's schema design, aggregation framework, and CRUD operations.
   https://www.mongodb.com/docs/

2. Express.js Documentation: This will help you with routing and building RESTful APIs for the employee management system.

3. "MongoDB: The Definitive Guide" Book by Kristina Chodorow.

4. Youtube : code with Yousaf

5. Mongodb Atlas