HUMAN RESOURCE MANAGEMENT SYSTEM

MINI-PROJECT REPORT

Submitted by

RAGHUL A	220701209
PREM KUMAR D	220701204

In partial fulfilment for the award of the Degree of

BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023-24

BONAFIDE CERTIFICATE

Certified that this project report **"HUMAN RESOURCE MANAGEMENT SYSTEM"** is the bonafide work of **"RAGHUL A (220701209), PREM KUMAR D (220701204)"** who carried out the project work under my supervision

Submitted for the Practical Examination held on _____

SIGNATURE

Mrs. K. Maheshmeena
Assistant Professor (SG),
Computer Science and Engineering
Rajalakshmi Engineering College (Autonomous),
Thandalam, Chennai - 602 105

ABSTRACT

This project proposes an HR Management System (HRMS) to digitally store and manage employee data. It aims to streamline HR processes and improve data accessibility within an organisation.

Employee Database Stores and manages all relevant employee information like IDs, names, contact details, experience, and department assignments.Data Retrieval: Allows authorised users to search and access employee data for informed decision-making.Salary Management Facilitates salary record keeping and calculations, including increments and deductions.Managerial Hierarchy- Maps employee-manager relationships for better communication and clarity on reporting lines

TABLE OF CONTENTS

CHAPTER 1

1. Introduction

This Human Resource Management System (HRMS) project aims to address the inefficiencies and limitations of traditional HR processes, such as manual data handling and limited accessibility. By creating a centralised and automated system for managing employee information, the project seeks to streamline HR operations and improve data management within an organisation. The HRMS will consist of key modules designed to facilitate tasks like employee data storage, retrieval, salary management, and mapping of managerial hierarchies.

2. Objectives

- **Streamlining HR Processes:** Automate routine HR tasks to improve efficiency and free up HR personnel for strategic initiatives.
- **Enhancing Data Accuracy and Accessibility:** Establish a central and reliable database for employee information, eliminating manual record-keeping and ensuring consistency across HR functions.
- **Facilitating Data-Driven Decision Making:** Provide easy access to employee analytics and reports to support informed decision-making in HR.
- **Improving Transparency:** Increase transparency in HR processes for both employees and managers.

3. Modules

- **Employee Database:** This module acts as the central repository for storing and managing all relevant employee information, including IDs, names, contact details, experience, and department assignments. It might also encompass functionalities for adding, editing, and deleting employee records.
- **Data Retrieval:** This module empowers authorised users (HR personnel, managers, and potentially employees for self-service access) to efficiently search and access employee data based on various criteria (e.g., ID, name, department). This retrieved information can be used for informed decision-making within HR.
- **Salary Management:** Designed to streamline salary processes, this module facilitates the calculation, recording, and (potentially) disbursement of employee salaries. It might handle increments and deductions, and could even integrate with existing payroll systems for automation.
- **Managerial Hierarchy:** This module maps the reporting structure within the organisation by capturing employee-manager relationships. It provides clarity on who reports to whom, enabling better communication and collaboration between employees and their managers within the HRMS platform.

CHAPTER 2

1. Software Description

This section dives into existing HRMS software. We'll explore functionalities offered by popular solutions (employee data management, reporting, etc.) to understand their strengths and weaknesses.

- **Feature Analysis:** We'll assess completeness (do they cover core HR needs?), scalability (can they handle different company sizes?), user-friendliness (are they easy to use?), security (do they protect data?), and cost of existing options.

- **Inspiration for Our Project:** By identifying best practices and potential gaps (missing functionalities), we'll draw inspiration for our own HRMS, aiming for a comprehensive and user-centric solution (easy to use for both HR and employees).

This analysis will inform our project's development, including feature selection (which functionalities to include), design choices (creating a user-friendly interface), and user experience (ensuring a smooth interaction for all users).

2. Languages

Python & Flask

Focuses on: Python's beginner-friendliness and Flask's flexibility for rapid prototyping.

Benefits:

- Easier learning curve for new developers
- Faster development cycles for a basic HRMS
- Control over project structure and future customisation

CHAPTER 3

1. Functional Requirements

- **Employee Management:**
    - Add, edit, and delete employee information (ID, name, contact details, experience, department, etc.).
    - Search and filter employee data using various criteria.
    - View detailed employee profiles.

- **Data Retrieval and Reporting:**
    - Allow authorised users (HR personnel, managers) to access employee data for informed decision-making.
    - Generate reports based on employee information (e.g., department breakdown, experience levels).

- **Security:**
    - Implement user authentication and authorisation to control access to sensitive employee data.
    - Secure data storage and transmission protocols.

2. Non-Functional Requirements

- **Performance:**
  - The system should be responsive and provide fast loading times for user interactions.
  - Ability to handle a reasonable number of concurrent users without performance degradation.

- **Scalability:**
  - The system should be adaptable to accommodate future growth in the organisation (number of employees, data volume).

- **Usability:**
  - The user interface should be intuitive and easy to navigate for authorised users (HR personnel, managers, potentially employees for self-service).
  - Provide clear instructions and user guidance within the system.

- **Reliability:**
  - The system should be available and operational during regular business hours.
  - Minimise downtime and implement data backup and recovery procedures.

- **Maintainability:**
  - The code should be well-documented and follow coding standards for future modifications and maintenance.
  - The system should be easily adaptable to future changes in requirements or functionalities.

3. Hardware and Software Requirements

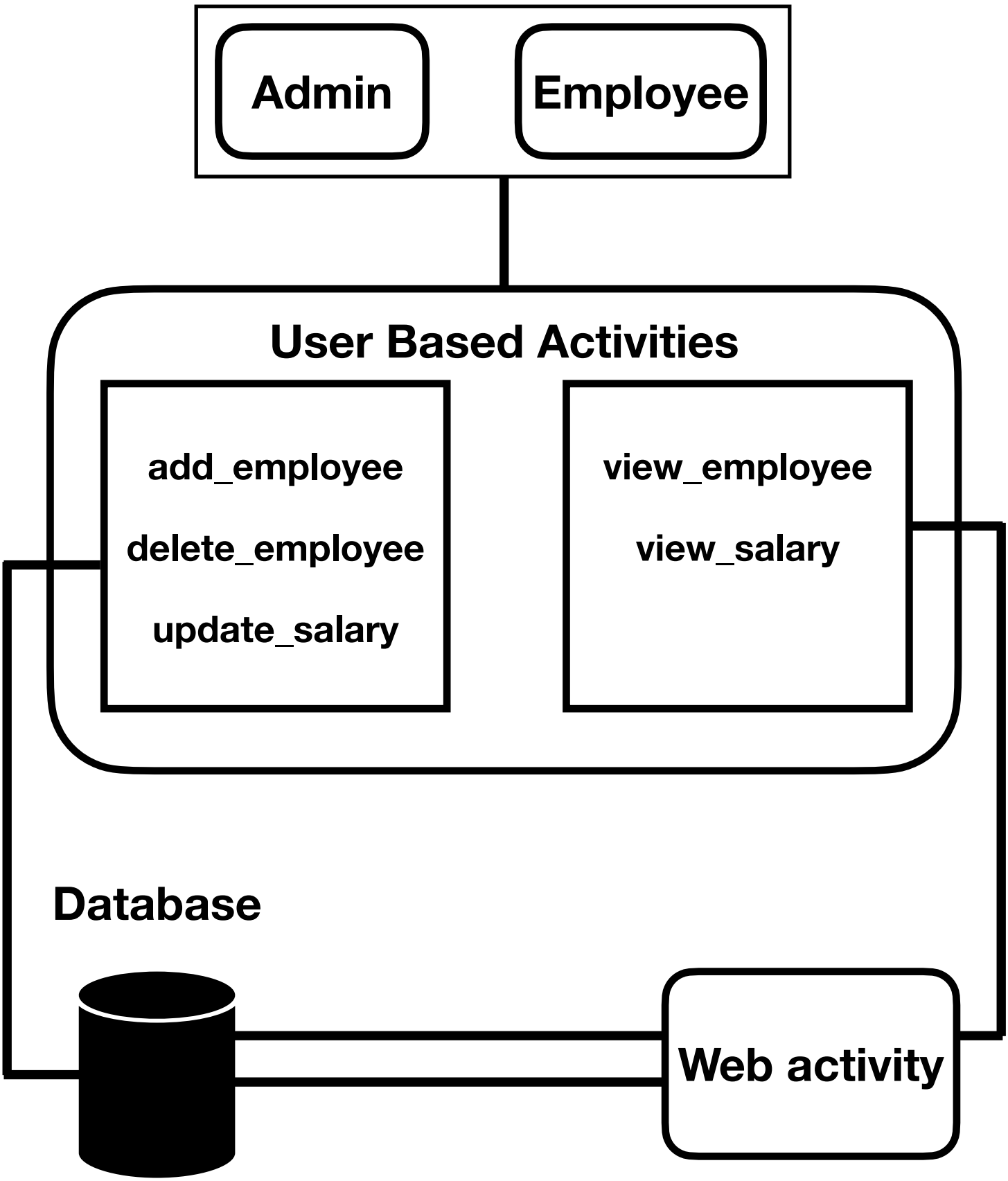
Hardware:

- **Server (physical or virtual):** Handles user load and data storage.
- **Storage:** Secure space for employee data, system files, and backups.
- **Client Computers:** Standard desktops or laptops with internet access for authorised users.

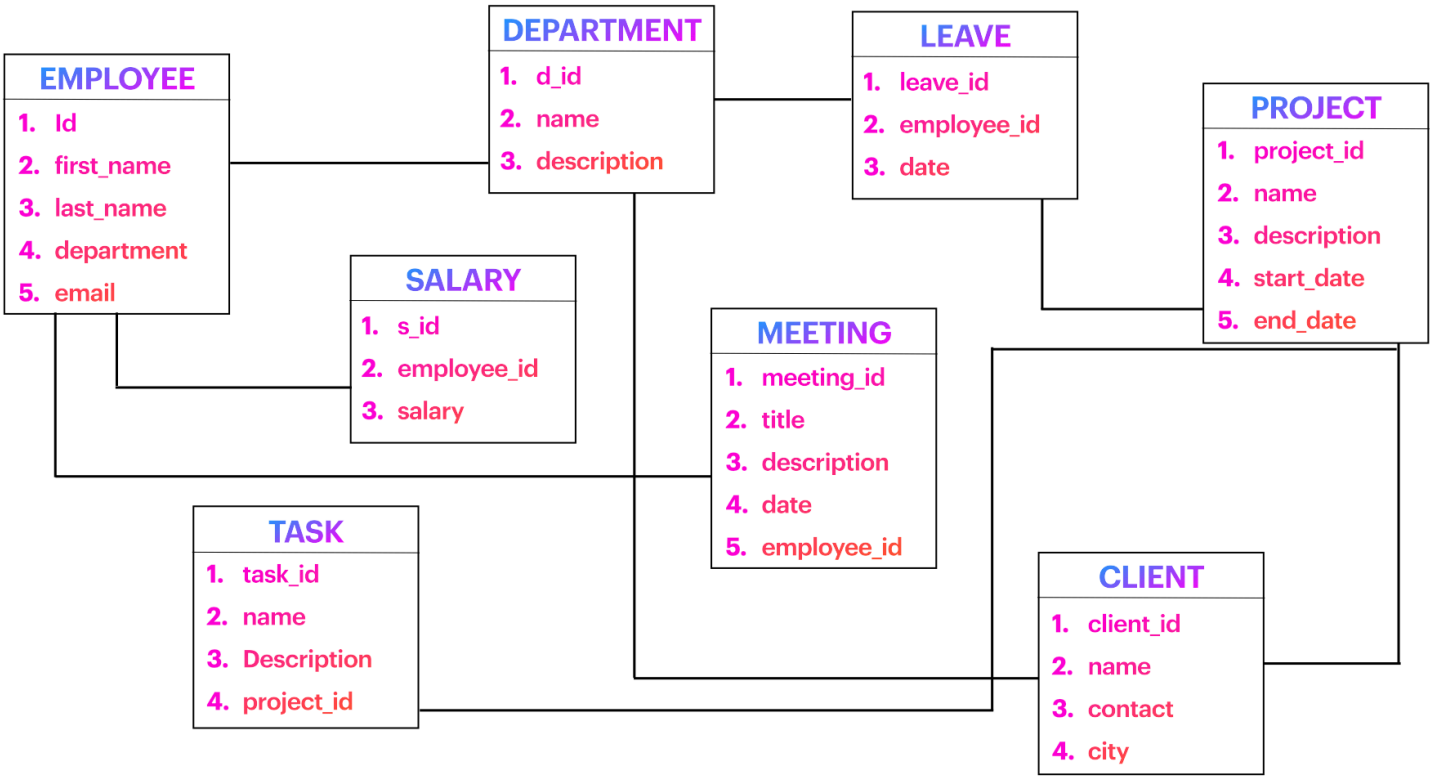
Software:

- **Server OS:** Stable and secure operating system (e.g., Linux, Windows Server).
- **Web Server:** Software to deliver the HRMS web application (e.g., Apache, Nginx).
- **Database:** Manages employee data (e.g., MySQL, PostgreSQL).
- **Python & Flask:** Programming language and framework for building the HRMS.

Architecture Diagram



ER-Diagram


EMPLOYEE
1. Id
2. first_name
3. last_name
4. department
5. email
DEPARTMENT
1. d_id
2. name
3. description
LEAVE
1. leave_id
2. employee_id
3. date
PROJECT
1. project_id
2. name
3. description
4. start_date
5. end_date
SALARY
1. s_id
2. employee_id
3. salary
MEETING
1. meeting_id
2. title
3. description
4. date
5. employee_id
TASK
1. task_id
2. name
3. Description
4. project_id
CLIENT
1. client_id
2. name
3. contact
4. city

Program code

```
from flask import Flask, request, redirect, url_for
from flask import render_template

from modules import seasonLocal, Employee, Salary

app = Flask(__name__)


@app.route('/')
def hello_world():
    db = seasonLocal()
    return render_template('homepage.html',
employees=db.query(Employee).all())


@app.route('/about')
def about_us():
    return render_template('about.html')


@app.route('/add')
def add_employee():
    return render_template('add.html')
```

```
@app.route('/add_employee', methods=['POST'])

def add_employee_post():

    first_name = request.form['firstName']

    last_name = request.form['lastName']

    department = request.form['department']

    email = request.form['email']

    db = seasonLocal()

    new_employee = Employee(first_name=first_name, last_name=last_name,
email=email, dept=department)

    db.add(new_employee)

    db.commit()

    db.close()

    return redirect(url_for('hello_world'))

 @app.route('/delete_employee/<int:employee_id>', methods=['GET', 'POST'])
 def delete_employee(employee_id):
     if request.method == 'GET':
         # Confirmation page before deletion (optional)
         return render_template('delete_confirmation.html',
 employee_id=employee_id)
     elif request.method == 'POST':
         # Delete employee from database
         db = seasonLocal()
         employee = db.query(Employee).get(employee_id)
         if employee:
             db.delete(employee)
             db.commit()
             db.close()
```

```
            return redirect(url_for('hello_world'))
        else:
            return redirect(url_for('hello_world'))
    else:
        return "Method not allowed", 405


@app.route('/show')
def show_employees():
    db = seasonLocal()
    filtered_employees =
db.query(Employee.first_name,Employee.email,Salary.amount).join(Salary,Empl
oyee.dept == Salary.dept).all()
    return render_template('salary.html', employees=filtered_employees)



@app.route('/delete')
def delete_page():
    db = seasonLocal()

    return render_template('delete.html', employees=db.query(Employee).all())



@app.route('/er')
def show_employees_er():
    return render_template('er.html')



@app.route('/search')
def search_page():
    return render_template('search.html')
```

```
@app.route('/search_employees', methods=['POST'])
def search_employees():
    search_term = request.form['search_term']  # Get search term and convert to lowercase
    results = search_employee(search_term)
    return render_template('search.html', search_term=search_term, results=results)



def search_employee(search_term):
    session = seasonLocal()
    try:
        query = session.query(Employee).filter(Employee.first_name == search_term)
        search_results = query
        return search_results
    finally:
        session.close()  # Always close the session



@app.route('/search')
def search_page():
    return render_template('search.html')



if __name__ == '__main__':
    app.run(debug=True)
```

Classes (Entities)

```
from sqlalchemy import Column, Integer, String, create_engine, ForeignKey, DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship

from config import SQLALCHEMY_DATABASE_URI


Base = declarative_base()


class Employee(Base):
    __tablename__ = 'employee'
    e_id = Column(Integer, primary_key=True)
    first_name = Column(String(50), nullable=False)
    last_name = Column(String(50), nullable=False)
    email = Column(String(50), nullable=False)
    dept = Column(String(50), ForeignKey('salary.dept'), nullable=False)

    def __str__(self):
        return f"{self.first_name} {self.last_name} ({self.dept}) - {self.email}"
```

```
class Project(Base):
    __tablename__ = 'project'
    project_id = Column(Integer, primary_key=True)
    name = Column(String(50), nullable=False)
    description = Column(String(500), nullable=False)
    start_date = Column(DateTime, nullable=False)
    end_date = Column(DateTime, nullable=False)


class Task(Base):
    __tablename__ = 'task'
    task_id = Column(Integer, primary_key=True)
    name = Column(String(50), nullable=False)
    description = Column(String(500), nullable=False)
    due_date = Column(DateTime, nullable=False)
    project_id = Column(Integer, ForeignKey('project.project_id'))


class Client(Base):
    __tablename__ = 'client'
    client_id = Column(Integer, primary_key=True)
    name = Column(String(50), nullable=False)
    contact = Column(String(50), nullable=False)
    city = Column(String(50), nullable=False)

class Leave(Base):
    __tablename__ = 'leave'
    leave_id = Column(Integer, primary_key=True)
    employee_id = Column(Integer, ForeignKey('employee.id'))
    date = Column(DateTime, nullable=False)
```

```
class Salary(Base):
    __tablename__ = 'salary'
    dept = Column(String(50), nullable=False,primary_key=True)
    amount = Column(Integer, nullable=False)


engine = create_engine(SQLALCHEMY_DATABASE_URI)


seasonLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
```

Initialise Table from sql

```
# Import necessary functions (outside models.py)
from modules import Base, engine

# Create all tables defined in the Base class (including Employee)
Base.metadata.create_all(engine)
```

Configuration On Flask

```
import os

# Database configuration
SQLALCHEMY_DATABASE_URI = 'postgresql://raguru:lmao@localhost:5433/postgres'

# Other settings (optional)
DEBUG = True  # Enable debug mode for development
SECRET_KEY = os.environ.get('SECRET_KEY') or 'your_secret_key'
```

Database is configured using an psql URI with the username and password with the port number 5433

Debugging Option is available in flask and it is enabled with a secret key

OUTPUT

New Employee Details

First Name:

Enter First Name

Last Name:

Enter Last Name

Department:

Select Department

Email:

Enter Email Address

Add Employee

Salary Details

Name	Email	Salary
Raghul	sraraghul@gmail.com	10000
Rghul	sra@gmail.com	3000

Employees

Name	Department	Email	Delete
Rghul A	Marketing	sra@gmail.com	Delete
Raghul A	Engineering	sraraghul@gmail.com	Delete

Results and Discussion

1. Findings

In this section, you'll present the key results obtained from developing and implementing your Human Resource Management System (HRMS). Here are some potential areas to focus on, depending on your project's specific goals:

- **System Functionality:** Describe the functionalities successfully implemented in the HRMS. This could include modules for employee onboarding, payroll management, performance tracking, leave management, etc.
- **Data Migration:** Discuss the process and success rate of migrating existing HR data to the new HRMS.
- **User Adoption:** Report on the level of user adoption among HR personnel and employees. Did they find the system easy to learn and use?
- **Efficiency Improvements:** If a core objective was to improve efficiency, present data on how the HRMS streamlined processes. This could involve metrics like reduced processing times for tasks or increased automation.
- **Error Reduction:** Did the HRMS minimize errors in data entry or calculations compared to the previous system?Quantify the improvements if possible.

2. Analysis

Here, you'll delve deeper into the findings and discuss their significance for your HRMS project.

- **Strengths and Weaknesses:** Analyze the strengths and weaknesses of the implemented HRMS. What features worked well? Were there any functionalities that fell short of expectations?
- **Impact on HR Processes:** Discuss how the HRMS has impacted various HR processes. Did it lead to more efficient workflows or improved decision-making?
- **Challenges and Solutions:** Describe any challenges encountered during development or implementation, and what solutions were implemented to address them.
- **Future Improvements:** Based on your analysis, propose potential areas for future improvements to the HRMS.This could involve additional functionalities, user interface enhancements, or integration with other systems.

Conclusion

Your Human Resource Management System (HRMS) project has reached a critical milestone. In this concluding section,you'll summarize the key takeaways, reiterate the project's value proposition, and potentially discuss future directions.Here's a breakdown of the points you can cover:

Recap of Project Goals:

- Briefly remind the audience of the main objectives you set out to achieve with the HRMS project.
- What specific HR challenges were you hoping to address?

Summary of Key Findings:

- Highlight the most significant findings from your "Results and Discussion" section.
- Focus on the functionalities successfully implemented, improvements in efficiency or error reduction, and user adoption rates.
- Briefly mention any challenges encountered and how they were overcome.

Impact on HR Management:

- Discuss the broader impact of the HRMS on the organization's HR management practices.
- How has it streamlined workflows, improved decision-making, or enhanced employee experience?
- Did it free up HR personnel's time for more strategic tasks?

Project Significance:

- Emphasize the overall significance of the HRMS project for the organization.
- How does it contribute to a more efficient and effective HR function?
- Consider potential cost savings or return on investment (ROI) if applicable.

Closing Statement:

- Conclude your presentation with a strong and impactful statement that summarizes the project's success and its positive implications for the organization's HR management.

References

https://flask.palletsprojects.com/en/3.0.x/

Python 3 Documentation