



EMPLOYEE MANAGEMENT SYSTEM

---SAKSHI

CONTENTS

Overview of Employee Management system

- What
- Why
- How

Prerequisites

Database Structure

Creating MySQL Database via the Command Line

Manage Database Using Python and Create Features:

- Virtual Environment
- Steps for Creating a Virtual Environment (V.E)
- Activate virtual Environment
- Install the package MySQL-connector -python
- Retrieve the data from database
- Insert the data into database

Frontend Development

- Frontend in Streamlit
- Install Streamlit
- Use images in Streamlit
- Use videos in Streamlit
- Create a menu bar in Streamlit
- [Features For HR](#)
- Create a login form in Streamlit
- View the tables in Streamlit
- [Features For Employees](#)
- Create a feature to see personal details
- Create a feature for updating the password
- [Features For Administrator](#)
- Create feature for adding a new employee in Streamlit
- Create feature for deleting an employee's details in Streamlit
- Create a feature for modifying the details of an employee in Streamlit
- Change the icon and title in Streamlit.

Summary

Conclusion

OVERVIEW OF EMPLOYEE MANAGEMENT SYSTEM

WHAT:

- It is a data management application which can manage employee records, including personal details, job roles, attendance and payroll information.
- It provides features such as:
 - View All Employees Details like Employee ID, Name, and Email ID.
 - Add New Employee Details.
 - Modify Employee Details.
 - Manage attendance.
 - Process payroll.
 - Maintain Department Details like Department name, Department ID.
- This application uses MYSQL as a DBMS.
- This application is a Web-based system that can run on a Local Area Network, allowing HR managers and administrators to manage employees from different locations.
- The system enhances workforce efficiency by automating routine HR tasks, reducing manual errors, and improving data accessibility.

WHY:

- Practical use for Organisations:
 - This project is very useful for businesses, organisation and companies of all size to efficiently manage employee data, attendance and payroll.
 - It helps HR departments in streamlining employee data, tracking attendance, processing payroll, and managing departments efficiently.
- Enhances skills:
 - Working on this project improves programming skills, particularly in Python, SQL, and Streamlit.
 - Enhances database management skills by handling MySQL queries, relational data, and secure access management.

HOW:

- Backend: This system uses MYSQL as the database management system while python handles backend logic, data processing and communication between database and frontend.
- Frontend: The user interface is built using streamlit. Streamlit is a Python tool to create web apps, show data, and build interactive dashboards easily.

PREREQUISITES

Before you begin working on your project, ensure that you have all the necessary components installed on your system. These components are essential for setting up your development environment and ensuring smooth interaction between Python and MySQL.

➤ **Install Python (Recommended Version: 3.x)**

Visit the official Python website: [Download Python](#)

➤ **Install MySQL Database**

Go to the official MySQL website: [Download MySQL](#)

DATABASE STRUCTURE OF EMPLOYEES

A **database structure** refers to the way data is organized, stored, and managed in a database system. It defines how data is related to each other and how it can be efficiently accessed, modified, and maintained. The database structure for an employee management system typically includes a table named **Personal_info**(Employees), **Departments**, **Department_Manager**, **HR**, **Admin** with fields that store information about each employee. Here's a basic structure:

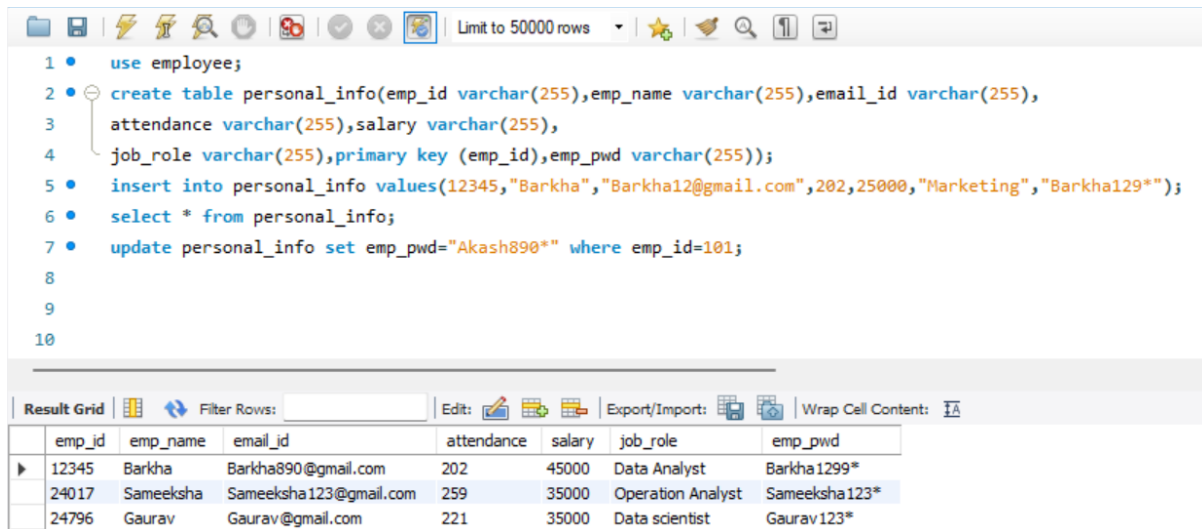
Whereas, (P) indicates the columns with the primary key.

Tables:	Personal_info	Departments	Department_Manager	HR	Admin
Columns:	emp_id(P)	depat_name	depat_name(P)	hr_id(P)	admin_id(P)
	emp_name	depat_id(P)	depat_id	hr_pwd	admin_pwd
	email_id		emp_id		
	attendance		depat_manager		
	salary				
	Job_role				
	emp_pwd				

This **Employee Management System (EMS) database structure** ensures efficient management of employee data by organizing information into well-defined tables.

CREATING MYSQL DATABASE VIA COMMAND LINE

- To create an employee's database, SQL can be used. Here is an example of how to create a database and under the database we can create table with the name of personal information which stores information such as Employee Id, Name, Email-Id, Attendance, Salary, Job role and Employee Password:



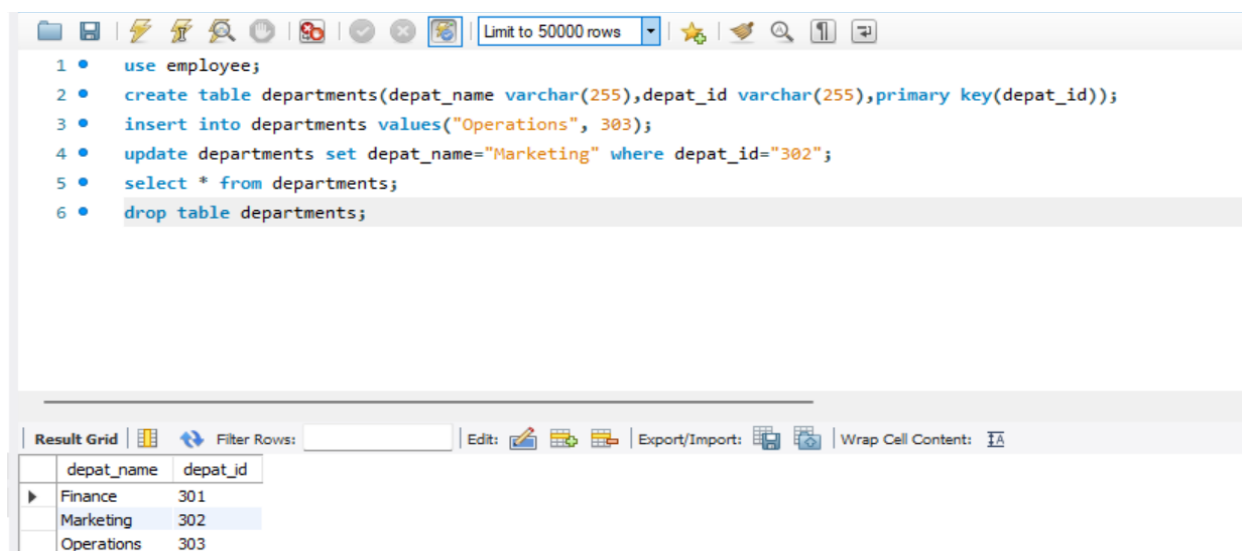
```

1 • use employee;
2 • create table personal_info(emp_id varchar(255),emp_name varchar(255),email_id varchar(255),
3   attendance varchar(255),salary varchar(255),
4   job_role varchar(255),primary key (emp_id),emp_pwd varchar(255));
5 • insert into personal_info values(12345,"Barkha","Barkha12@gmail.com",202,25000,"Marketing","Barkha129*");
6 • select * from personal_info;
7 • update personal_info set emp_pwd="Akash890*" where emp_id=101;
8
9
10

```

emp_id	emp_name	email_id	attendance	salary	job_role	emp_pwd
12345	Barkha	Barkha890@gmail.com	202	45000	Data Analyst	Barkha1299*
24017	Sameeksha	Sameeksha123@gmail.com	259	35000	Operation Analyst	Sameeksha123*
24796	Gaurav	Gaurav@gmail.com	221	35000	Data scientist	Gaurav123*

- To create a department table under the Employee database, we can use the following SQL:



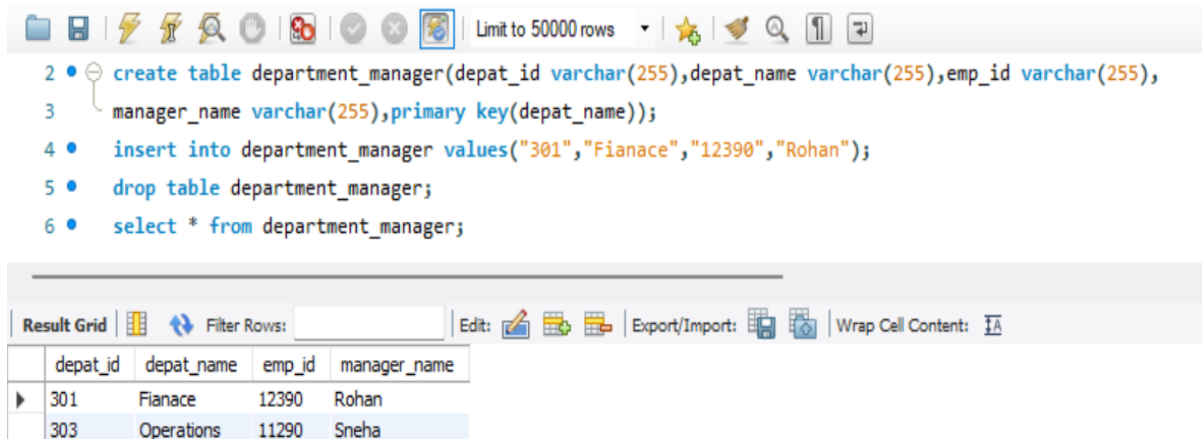
```

1 • use employee;
2 • create table departments(depat_name varchar(255),depat_id varchar(255),primary key(depat_id));
3 • insert into departments values("Operations", 303);
4 • update departments set depat_name="Marketing" where depat_id="302";
5 • select * from departments;
6 • drop table departments;

```

depat_name	depat_id
Finance	301
Marketing	302
Operations	303

- To create a department manager table under the Employee database, we can use the following SQL:



SQL Studio interface showing the execution of SQL commands to create and populate the department_manager table. The toolbar includes icons for file operations, execution, and a 'Limit to 50000 rows' dropdown. The SQL editor contains the following commands:

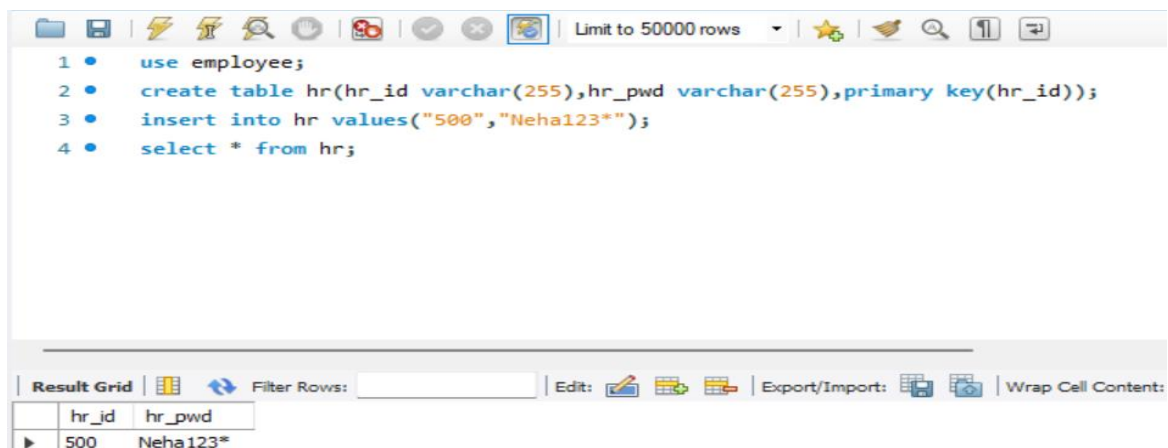
```

2 • create table department_manager(depat_id varchar(255),depat_name varchar(255),emp_id varchar(255),
3   manager_name varchar(255),primary key(depat_name));
4 • insert into department_manager values("301","Fianace","12390","Rohan");
5 • drop table department_manager;
6 • select * from department_manager;
  
```

The Result Grid shows the output of the SELECT statement:

depat_id	depat_name	emp_id	manager_name
301	Fianace	12390	Rohan
303	Operations	11290	Sneha

- To create a HR table under the Employee database, we can use the following SQL:



SQL Studio interface showing the execution of SQL commands to create and populate the hr table. The toolbar includes icons for file operations, execution, and a 'Limit to 50000 rows' dropdown. The SQL editor contains the following commands:

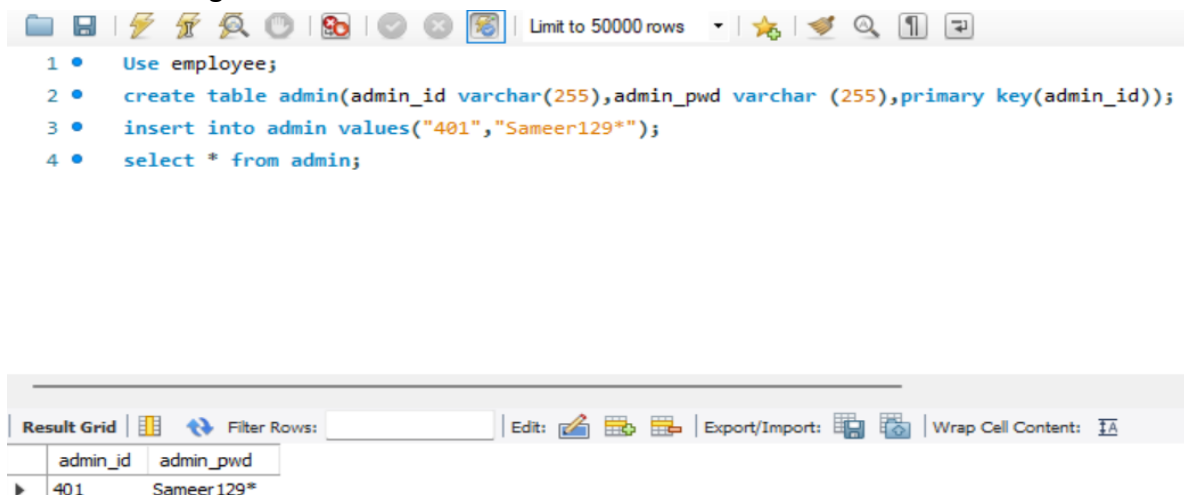
```

1 • use employee;
2 • create table hr(hr_id varchar(255),hr_pwd varchar(255),primary key(hr_id));
3 • insert into hr values("500","Neha123*");
4 • select * from hr;
  
```

The Result Grid shows the output of the SELECT statement:

hr_id	hr_pwd
500	Neha123*

- To create an Administrator table under the Employee database, we can use the following SQL:



SQL Studio interface showing the execution of SQL commands to create and populate the admin table. The toolbar includes icons for file operations, execution, and a 'Limit to 50000 rows' dropdown. The SQL editor contains the following commands:

```

1 • Use employee;
2 • create admin(admin_id varchar(255),admin_pwd varchar (255),primary key(admin_id));
3 • insert into admin values("401","Sameer129*");
4 • select * from admin;
  
```

The Result Grid shows the output of the SELECT statement:

admin_id	admin_pwd
401	Sameer129*

MANAGE DATABASE USING PYTHON AND CREATE

FEATURES

➤ Virtual Environment:

A **virtual environment** (venv) is an isolated workspace for Python projects. It allows you to install dependencies separately from the system-wide Python installation, preventing conflicts between different projects.

Features of Virtual Environment:

- **Dependency Management** – Keeps project-specific libraries separate.
- **Avoid Conflicts** – Different projects can use different versions of the same package.
- **Reproducibility** – Makes it easier to share projects with others.
- **Cleaner Development** – Prevents clutter in the global Python environment.

Steps for Creating a Virtual Environment (V.E)

- Go to C Drive → Create a Project Folder
Inside the Project Folder, create a new folder named Employee.
- This Employee folder will act as the V.E (Virtual Environment).
- You can create multiple virtual environments for different projects in this Project Folder.

Now, create the virtual environment

- Copy the address: C:\Projects\Employee
- Open Command Prompt (cmd)
- Write the address in cmd or navigate to the folder manually
- Run the following command:

```
python -m venv C:\Projects\Employee
```

- If successful, you will see the virtual environment created inside the Employee folder

Inside the Employee Folder, you should see:

- Include
- lib
- Scripts
- pyvenv.cfg

➤ Activate Virtual Environment:

- Navigate to the Scripts folder inside your Employee folder, Click on the address bar and type cmd.
- Run the following command: activate

```
C:\Projects\Employee\Scripts>activate
```

Important Notes:

- You must activate the virtual environment every time you want to use it
- If you don't activate it, Python will use the global environment instead of the virtual one
- If activated correctly, all installed packages will be saved in the virtual environment, not in the main system

Install the package MySQL-connector -python:

- The MySQL Connector allows Python to communicate with the MySQL database. You can install it easily using **pip**, the package manager for Python.
- To install MySQL Connector, open a terminal or command prompt and run:

```
(Employee) C:\Projects\Employee\Scripts>pip install mysql-connector-python
```

Now after successful installation of MYSQL-connector we can connect MYSQL with python which can be done by writing the following code.

Python 1

```
File Edit Format Run Options Window Help
```

```
import mysql.connector # Importing the MySQL connector module to interact with the MySQL database
mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
print(mydb) # Printing the connection object to check if the connection was successful
```

Now that we have successfully set up and activated our virtual environment, we have completed the foundational steps required for a well-structured development setup. This ensures that all dependencies and packages for our project remain isolated, preventing conflicts with other projects.

With the virtual environment in place, we can now shift our focus to the core development of our Employment Management System. This will involve designing the system architecture, implementing key functionalities, and ensuring that it efficiently handles employment records, staff management, and other essential features.

To retrieve the data from my database:

Python 2

- The code retrieves and prints data from two tables: `personal_info` and `departments`.
- It prints both specific column values and entire records.
- It uses two separate cursor objects to execute different queries.

#To retrieve the data from my database:

```
c=mydb.cursor() # Creates a cursor object to interact with the database.
# Executes an SQL query to retrieve all records from the `personal_info` table.
c.execute("select * from personal_info")
for r in c: # Iterates through each row in the result set.
    print(r[1]) # Prints the element at index
    print(r) #Prints the entire row.
c2=mydb.cursor()
# Executes an SQL query to retrieve all records from the `departments` table.
c2.execute("select * from departments")
for r in c2:
    print(r) # Prints the entire row.
```

When we execute the Python script in the command prompt (CMD), the code interacts with a database. The script retrieves data from two tables: `personal_info` and `departments`, and prints the results.

```
(Employee) C:\Projects\Employee\Scripts>backend.py
<mysql.connector.connection_cext.CMySQLConnection object at 0x0000023B5DE2DFD0>
Barkha
('12345', 'Barkha', 'Barkha890@gmail.com', '202', '45000', 'Data Analyst', 'Barkha1299*')
Sameeksha
('24017', 'Sameeksha', 'Sameeksha123@gmail.com', '259', '35000', 'Operation Analyst', 'Sameeksha123*')
Gaurav
('24796', 'Gaurav', 'Gaurav@gmail.com', '221', '35000', 'Data scientist', 'Gaurav123*')
('Finance', '301')
('Marketing', '302')
('Operations', '303')
```

To insert the data into my database:

Python 3

- The script prompts for employment ID, name, email, attendance, salary, and job role, password.
- Initializes a cursor (c3) to interact with the database.
- Inserts the collected data into the personal_info table.
- Saves the data permanently using mydb.commit().
- Prints "Details saved successfully".

```
#To insert the data into my database:
emp_id=input("Enter your employment id: ")
emp_name=input("Enter your name: ")
email_id=input("Enter your email id: ")
attendance=input("Enter your attendance: ")
Salary=input("Enter your salary: ")
Job_role=input("Enter your job role: ")
emp_pwd=input("Enter password: ")
c3=mydb.cursor()# Create a cursor object to execute SQL queries
# Insert the collected data into the 'personal_info' table
c3.execute("insert into personal_info values(%s,%s,%s,%s,%s,%s,%s)", (emp_id,emp_name,email_id,attendance,Salary,Job_role,emp_pwd))
mydb.commit()# Commit the transaction to save changes to the database
print("Details saved successfully")# Confirms that the data has been stored in the database
```

The script collects employee details (ID, name, email, attendance, salary, job role and Password) and inserts them into the database. After committing the data, it displays "Details saved successfully", confirming successful storage. The command prompt then returns to the project directory.

```
Enter your employment id: 34589
Enter your name: Seema
Enter your email id: Seema123@gmail.com
Enter your attendance: 253
Enter your salary: 35000
Enter your job role: Marketing Analyst
Enter password: Seema123*
Details saved successfully
```

Now, MySQL database contains the most recent data added via Python.

Result Grid							
Filter Rows:				Edit:		Export/Import:	
	emp_id	emp_name	email_id	attendance	salary	job_role	emp_pwd
▶	12345	Barkha	Barkha890@gmail.com	202	45000	Data Analyst	Barkha1299*
	24017	Sameeksha	Sameeksha123@gmail.com	259	35000	Operation Analyst	Sameeksha123*
	24796	Gaurav	Gaurav@gmail.com	221	35000	Data scientist	Gaurav123*
	33169	Vishal	Vishal123@gmail.com	234	35000	Operation Analyst	Vishal123*
	34589	Seema	Seema123@gmail.com	253	35000	Marketing Analyst	Seema123*

FRONTEND DEVELOPMENT

WEB APPLICATION

Frontend development is crucial for an Employee Management System (EMS) because it provides an interactive and user-friendly interface for HR teams, Administrators and Employees. It improves efficiency by enabling real-time data visualization, quick employee searches, and seamless form submissions. A good frontend enhances usability, accessibility, and productivity, making EMS more effective in streamlining HR processes and workforce management.

➤ **Frontend in Streamlit:**

Streamlit is an open-source Python framework for building data-driven web applications quickly and easily. Streamlit allows developers to create interactive web apps using just Python.

➤ **Install streamlit:**

Go to scripts folder of virtual environment and open command prompt and run the following command:

```
(Employee) C:\Projects\Employee\Scripts>pip install streamlit
```

Now after successful installation of streamlit, Open the main.py file in your in-code editor, and add the following basic Streamlit code:

```
import streamlit as st # Importing Streamlit for building interactive web applications
st.title("EMPLOYEE MANAGEMENT SYSTEM") # Title of the app
```

Now that you've successfully created your Streamlit app with the title, it opens in the browser and lets you interact with the UI.

EMPLOYEE MANAGEMENT SYSTEM

This is a web application developed by sakshi as a part of training project.

Use Images: Open the main.py file in your in-code editor, and add the following Streamlit code and put the image link.

```
st.image("https://www.cutehr.io/wp-content/uploads/2020/04/Employee-Management-affiliate-page-1184x800.png")
```

Here's an image showcasing in streamlit.

EMPLOYEE MANAGEMENT SYSTEM

This is a web application developed by sakshi as a part of training project.



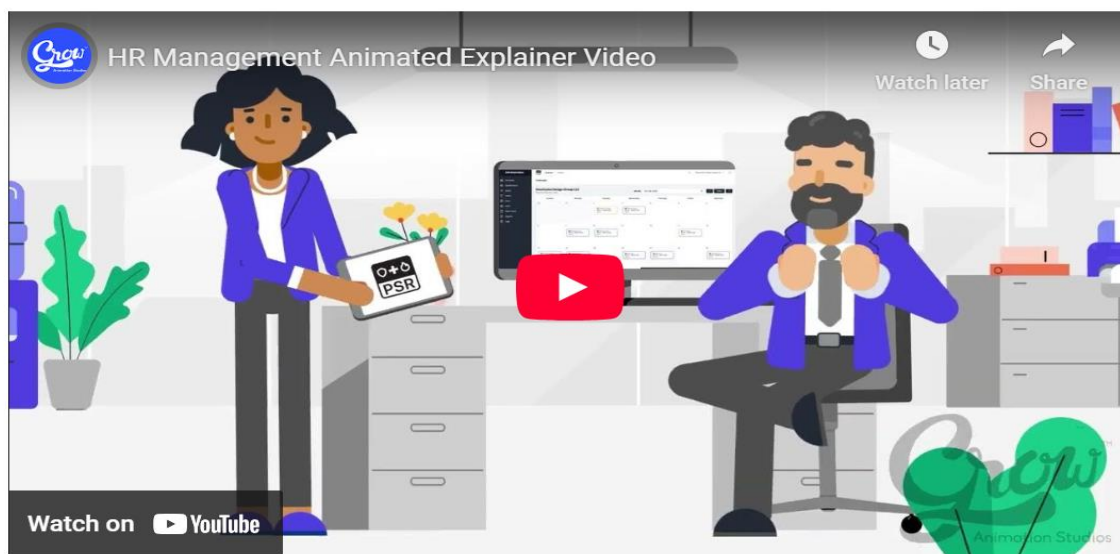
Use video: Open the main.py file in your in-code editor, and add the following Streamlit code and put the video link.

```
st.video("https://youtu.be/QvaQpx1zJHY?si=G1aLKXzy81l6eSG1")
```

Here's a video showcasing in streamlit

EMPLOYEE MANAGEMENT SYSTEM

This is a web application developed by sakshi as a part of training project.

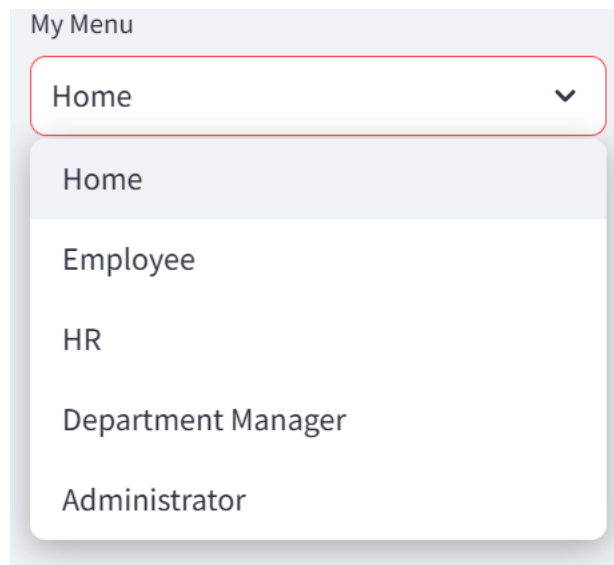


Create Menu Bar: Having a menu bar in a Streamlit app enhances navigation, organization, and user experience by allowing users to easily switch between sections. It keeps the UI clean, improves accessibility, boosts efficiency, enhances interactivity, and makes multi-feature applications more structured and user-friendly

Open the main.py file in your in-code editor, and add the following Streamlit code.

```
choice=st.sidebar.selectbox("My Menu", ("Home", "Employee", "HR", "Department Manager", "Administrator"))
```

Here's Menu Bar in streamlit



Features For HR

Create a login form for HR: Creating a login form for HR in Streamlit enhances security, data privacy, and access control by ensuring only authorized users manage employee records. It improves user experience, accountability, and workflow efficiency, while protecting sensitive HR data from unauthorized access.

Follow these steps to create an HR login system where HR users enter their hr_id and hr_pwd to access the system.

When user select the HR form the menu then it will show HR Login System and then there will be a login validation.

- If credentials are incorrect → Show "Incorrect ID or Password"
- If credentials are correct → Show "Login Successfully"

Open the main.py file in your in-code editor, and add the following Streamlit code for creating a login form for HR.

```
import streamlit as st
import mysql.connector

elif(choice=="HR"): #user selects HR
    st.write("Welcome To Employee Management System")
    bt2=st.button("settings")
    if"login" not in st.session_state: #Initialize session state for login if not already set
        st.session_state['login']=False #st.session_state to maintain login status across reruns.
    hr_id=st.text_input("Enter HR ID") #Input fields for HR ID and Password
    hr_pwd=st.text_input("Enter Password",type="password") #"type" is used to hide password

    btn=st.button("Login") #creating login button
    if btn:
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        c=mydb.cursor() # Establish connection with MySQL database
        c.execute("select * from hr")
        for r in c:
            if(r[0]==hr_id and r[1]==hr_pwd): #searching ID and password in the database
                st.session_state['login']=True
                break # Exit loop once valid credentials are found
            if(not st.session_state['login']):
                st.write("Incorrect ID or Password")#If no matching credentials are found, display an error message
        if(st.session_state['login']):
            st.write("Login Sucessfully")#If login is successful, display confirmation message
```

Steps:

- Design a login form with fields for HR ID and Password.
- Add a login button for users to submit credentials.
- Establish a connection to the MySQL database inside the Streamlit app.
- When the HR user enters login details, the app should query the database to check if the credentials are correct. It starts looping through each record in the HR table.

- If the HR ID and password match, display a success message and redirect them to the HR dashboard
- If the login fails, show an error message like "Incorrect HR ID or Password."
- To maintain login status across pages, we use Streamlit's session state (st.session_state). This allows us to store login information and prevent users from needing to log in again on every interaction.

Here's a login system in streamlit

My Menu

HR

EMPLOYEE MANAGEMENT SYSTEM

This is a web application developed by sakshi as a part of training project.

Welcome To Employee Management System

settings

Enter HR ID

500

Enter Password

.....



Login

Login Sucessfully

View the tables after login: After logging in, the HR professional can access multiple options, including viewing tables for personal information (Employees), department managers, departments, and more. HR can analyse records efficiently, ensuring smooth workforce management.

Open the main.py file in your in-code editor, and add the following Streamlit code for viewing the tables.

```
import streamlit as st # Importing Streamlit for building interactive web applications
import mysql.connector # Importing MySQL Connector to establish a connection with the MySQL database
import pandas as pd # Importing Pandas for data manipulation and displaying tabular data

# Creating a dropdown select box with different HR features
choice2=st.selectbox("Features",("select","View All Employees Information","View All Departments","View All Department Managers"))
if(choice2=="View All Employees Information"): # If the user selects "View All Employees Information"
    mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")# Establishing a database connection
    df=pd.read_sql("select emp_id,emp_name,email_id,attendance,salary,job_role from personal_info",mydb)
    st.dataframe(df) # Fetching employee details using Pandas
elif(choice2=="View All Departments"): #If the user selects "View All Departments"
    mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
    df=pd.read_sql("select * from departments",mydb)# Fetching all department details
    st.dataframe(df)
elif(choice2=="View All Department Managers"): #If the user selects "View All Departments Managers"
    mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
    df=pd.read_sql("select * from department_manager",mydb) # Fetching all department details
    st.dataframe(df)
```

Steps:

- Pandas' library for handling and displaying data.
- Provide a dropdown menu with options to view:
 - Employee information
 - Department details
 - Department manager details
- Establish a connection to the MySQL database using credentials like host, user, password, and database name.
- Fetch Data Using Pandas – Based on the selected feature:
 - Use SQL queries to retrieve data from the relevant table.
 - Load the data into a pandas DataFrame (pd.read_sql()).
- To hide the password column of employees from the HR team while retrieving employee details, we can modify our SQL query to exclude the password column
Display Data in Streamlit – Use st.dataframe() to present the retrieved data in a structured, user-friendly format.

The output of the code displays dynamic information based on the user's login status and their feature:

Here are the tables in streamlit

Features

View All Employees Information



	emp_id	emp_name	email_id	attendance	salary	job_role
0	12345	Barkha	Barkha890@gmail.com	202	45000	Data Analyst
1	24017	Sameeksha	Sameeksha123@gmail.com	259	35000	Operation Analyst
2	24796	Gaurav	Gaurav@gmail.com	221	35000	Data scientist
3	33169	Vishal	Vishal123@gmail.com	234	35000	Operation Analyst
4	34589	Seema	Seema123@gmail.com	253	35000	Marketing Analyst

Features

View All Departments



	depat_name	depat_id
0	Finance	301
1	Marketing	302
2	Operations	303

Features

View All Department Managers



	depat_id	depat_name	emp_id	manager_name
0	301	Finance	100	Aman
1	302	Marketing	105	Akash
2	303	Operations	112	Sneha

Features For Employees

Create features for employees: We have introduced a self-service feature for employees, allowing them to conveniently access their personal details and update their login password whenever needed. To ensure security and privacy, employees must need to log in first using their credentials. This enhancement improves security, transparency, and ease of access for all employees.

Enter Employee ID

12345

Enter Your Password

.....



Login

Login Sucessfully

Select

My Details

Update Password

Employement Agreement

Select



Create feature to see personal details: It allows employees to securely access their personal details anytime. This feature enhances transparency and convenience, enabling employees to view their information without needing to contact HR.

Open the main.py file in your in-code editor, and add the following Streamlit code to view own details.

```
# Creating a dropdown select box with different HR features
choice6=st.selectbox("Menu",("Select","My Details","Update Password","Employement Agreement"))#"Update E-mail ID

if(choice6=="My Details"): # Checks if the user selects "My Details".
    emp_id=st.text_input("Enter Your Employee ID") # User inputs their Employee ID.
    btn14=st.button("Show My details") # Button to fetch and display employee details.
if btn14:
    # Establishes connection to the database.
    mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
    # Retrieves employee details from the database based on the entered Employee ID.
    df=pd.read_sql("select * from personal_info where emp_id=%s",mydb, params=(emp_id,))
    st.dataframe(df)
```

Steps:

- The program checks if the user selects the "My Details" option.
- It prompts the user to enter their Employee ID using a text input field.
- A button labelled "Show My Details" is displayed. When clicked, it triggers the next steps.
- The program establishes a connection to the MySQL database to fetch data.
- A SQL query is executed to retrieve the employee's details based on the entered Employee ID.
- The retrieved data is displayed in a table format using Streamlit's `st.dataframe()`.

Menu

My Details

Enter Your Employee ID

12345

Show My details

	emp_id	emp_name	email_id	attendance	salary	job_role	emp_pwd
0	12345	Barkha	Barkha890@gmail.com	202	45000	Data Analyst	Barkha1299*

Create feature for updating the password: We create the employee password feature to ensure security, authentication. It helps maintain confidentiality.

Open the main.py file in your in-code editor, and add the following Streamlit code to Update Employee's password:

```
if(choice6=="Update Password"):#Checks if the user selects "Update password".
    emp_id=st.text_input("Enter Employee Id") #Input field for employee ID
    old_pwd=st.text_input("Enter Old Password",type="password") # Input field for the old password
    emp_pwd=st.text_input("Enter New Password",type="password")# Input field for the new password

    #password conditions
    if emp_pwd:
        errors = [] # List to store validation errors

        if len(emp_pwd) < 8:
            errors.append("Password should be at least 8 characters long.")

        if not any(c.isupper() for c in emp_pwd):
            errors.append("Password should have at least one uppercase letter.")

        if not any(c.isdigit() for c in emp_pwd):
            errors.append("Password should have at least one digit.")

        if not any(c in "@#$$%^&*" for c in emp_pwd):
            errors.append("Password should have at least one special character (@, #, $, %, ^, &, *, (, ).)")

        # Display validation results
        if errors:
            for error in errors:
                st.error(error) # Display each error message
        else:
            st.success("Password is valid!") # Success message if all conditions are met
    btn12=st.button("Update")
    if(btn12):
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        p=mydb.cursor()
        #Check if the employee ID and old password match in the database
        p.execute("SELECT * FROM personal_info WHERE emp_id = %s and emp_pwd= %s", (emp_id, old_pwd))
        result = p.fetchone()
```

```

if result: # If a matching record is found
    p.execute("UPDATE personal_info SET emp_pwd=%s WHERE emp_id=%s", (emp_pwd, emp_id))
    mydb.commit() # Save changes to database
    st.header("Updated successfully") # Display success message

else:
    st.write("Incorrect Old Password") # Show error if old password is incorrect

```

Steps:

- If employee selects the "Update Password" option.
- The system asks for Employee ID, Old Password, and New Password.
- The new password is checked against validation rules (length, uppercase, digit, special character).
- If the new password is invalid, error messages are displayed; otherwise, a success message appears.
- The user clicks the "Update" button to proceed.
- A connection is made to the MySQL database.
- The system checks if the Employee ID and Old Password match a record in the database.
- If the credentials are correct, the new password is updated and saved.
- A success message is shown if the update is successful.
- If the old password is incorrect, an error message is displayed.

Menu

Update Password



Enter Employee Id

12345

Enter Old Password

.....



Enter New Password

.....



Password is valid!

Update

Updated successfully

Features For Administrator

Create features for admin: The admin is responsible for managing employee records and overseeing database operations. After logging in with their Admin ID and Password, they gain access to various functionalities, such as adding new employees, deleting employee details, and modifying employee-related information etc.

This is a web application developed by sakshi as a part of training project.

Enter Admin ID

401

Enter Password

.....

Login

Login Sucessfully

select

Add New Employee

Delete Employee Details

Modify Employee Table

Modify Department Manager Table

select

Create features for adding new employee: Open the main.py file in your in-code editor, and add the following Streamlit code for adding details of new employee.

```
import streamlit as st # Importing Streamlit for building interactive web applications
import mysql.connector # Importing MySQL Connector to establish a connection with the MySQL database
import pandas as pd # Importing Pandas for data manipulation and displaying tabular data
import random # Importing Random module to generate random numbers

choice3=st.selectbox("Menu",("select","Add New Employee","Delete Employee Details","Modify Employee Table",
                             "Modify Department Manager Table"))# Dropdown menu to select an operation.
if(choice3=="Add New Employee"): # Checks if the admin selects "Add New Employee"
    l=list(range(10000,90000)) # Generates a list of employee IDs in the range 10000 to 90000.
    emp_id=random.choice(l) # Randomly selects an employee ID.
    emp_name=st.text_input("Enter Employee Name")# Collects user input for employee details
    email_id=st.text_input("Enter Email ID")
    attendance=st.text_input("Enter Attendance")
    salary=st.text_input("Enter Salary")
    job_role=st.text_input("Enter Job Role")
    emp_pwd=st.text_input("Create Password")
    btn4=st.button("Add Details") # Button to add details to the database.
    if btn4: # When the button is clicked, the following code executes.
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        p=mydb.cursor()
        p.execute("insert into personal_info values(%s,%s,%s,%s,%s,%s,%s)",(emp_id,emp_name,email_id,attendance,salary,job_role,emp_pwd))
        mydb.commit() # Commits the transaction to the database.
        st.header("Details added successfully") # Displays a success message.
```

Steps:

- The admin logs into the system using their credentials.
- The admin selects the "Add New Employee" option to proceed with adding a new employee to the database.
- After selecting the option, the system directs the admin to a form designed for entering employee details.
- The admin fills in the following required details of an employee:
- It will also Generate a list of possible employee IDs ranging from 10000 to 90000
- Randomly select an Employee ID from the generated list
- They then click on the "Add Details" button to submit the form.
- Once the form is submitted, the system processes the entered information.
- The system connects to the database and inserts the new employee details into the relevant table.
- After successfully storing the data, the system generates a confirmation message.

Employee Form:

Features

Add New Employee

Enter Employee Name

Akansha

Enter Email ID

Akansha123@gmail.com

Enter Attendance

253

Enter Salary

35000

Enter Job Role

Data Analyst

Create Password

.....



Add Details

Details added successfully

Create features for deleting an employee detail: Open the main.py file in your in-code editor, and add the following Streamlit code for deleting details of an employee

```
if(choice3=="Delete Employee Details"): # Checks if the admin selects "Delete Employee Details" from the dropdown.
    emp_id=st.text_input("Enter Employee ID") # Admin inputs the Employee ID to be deleted.
    btn5=st.button("Delete") # Button to confirm the deletion.
    if btn5:
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        p=mydb.cursor()
        p.execute("delete from personal_info where emp_id=%s", (emp_id,))
        mydb.commit() # Commits the transaction to permanently remove the record.
        st.header("Details Deleted successfully")# Displays a success message.
```

STEPS:

- The admin selects the “Delete Employee Details” option to proceed with deleting the details of an employee from the database.
- After selecting the option, the system directs the admin to a form designed for entering employee Id.
- They then click on the "Delete" button.
- Once the button is clicked, the system processes the entered information.
- The system connects to the database and delete the employee details from the relevant table.
- After deleting the data, the system generates a confirmation message.

Features

Delete Employee Details ▼

Enter Employee ID

64902

Delete

Details deleted successfully

Create features for modifying an employee’s detail: Open the main.py file in your in-code editor, and add the following Streamlit code for modifying the details of an employee.

```
if(choice3=="Modify Employee Table"): # Checks if the user selects "Modify Employee Table" from the menu.
    # Provides a dropdown menu for selecting the type of modification.
    choice4=st.selectbox("Employee Options", ("select", "Update Name", "Update Contact Details", "Update Salary", "Update Job Role"))
```

- The dropdown provides four options:
- The user selects an option from the menu.

Menu

Modify Employee Table

select

Update Name

Update Contact Details

Update Salary

Update Job Role

select

Admin can modify employee details like Name, Contact Details, Salary, and Job Role in the company's employee database.

Open the main.py file in your in-code editor, and add the following Streamlit code for modifying the details of an employee.

```
if(choice4=="Update Name"): # Checks if the user selects "Update Name".
    emp_id=st.text_input("Enter Employee Id") # User inputs the Employee ID.
    emp_name=st.text_input("Enter Employee Name") # User inputs the Employee Name.
    btn8=st.button("Update") # Button to execute the update operation.
    if btn8:
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        c=mydb.cursor()
        # Updates the employee name in the database for the given employee ID.
        c.execute("UPDATE personal_info SET emp_name=%s WHERE emp_id=%s", (emp_name, emp_id))
        mydb.commit() # Commits the transaction to apply the changes.
        st.header("Updated Successfully") # Displays a success message

if(choice4=="Update Contact Details"): # If the user selects "Update Contact Details"
    emp_id=st.text_input("Enter Employee Id") # Input field for Employee ID
    email_id=st.text_input("Enter Email Id") # Input field for new Email ID
    btn9=st.button("Update")
    if btn9:
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        c=mydb.cursor()

        c.execute("UPDATE personal_info SET email_id=%s WHERE emp_id=%s", (email_id, emp_id))
        mydb.commit() # Commits the transaction to apply the changes.
        st.header("Updated Successfully") # Displays a success message

if(choice4=="Update Salary"): # If the user selects "Update Salary"
    emp_id=st.text_input("Enter Employee Id") # Input field for Employee ID
    salary=st.text_input("Enter Salary") # Input field for new salary
    btn10=st.button("Update")
    if btn10: #If the update button is clicked
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        c=mydb.cursor()
        c.execute("UPDATE personal_info SET salary=%s WHERE emp_id=%s", (salary, emp_id))
        mydb.commit() # Commit changes to the database
        st.header("Updated Successfully") # Display a success message

if(choice4=="Update Job Role"): # If the user selects "Update Job Role"
    emp_id=st.text_input("Enter Employee Id") # Input field for Employee ID
    job_role=st.text_input("Enter Job Role") # Input field for new job role
    btn11=st.button("Update")
    if btn11: # If the update button is clicked
        mydb=mysql.connector.connect(host="localhost",user="root",password="welcome@123",database="employee")
        c=mydb.cursor()
        c.execute("UPDATE personal_info SET job_role=%s WHERE emp_id=%s", (job_role, emp_id))
        mydb.commit() # Commit changes to the database
        st.header("Updated Successfully") # Display a success message
```


- The admin selects the "Modify Employee Table" option to proceed with deleting the details of an employee from the database.
- Select the specific action you want to perform from the available options:
 - Update Employee Name
 - Update Contact Details
 - Update Salary
 - Update Job Role
- Enter the Employee ID to identify the record.
- Click the Update button to save changes.
- A success message will confirm the update.

Employee Options

Update Contact Details

Enter Employee Id

12345

Enter Email Id

Barkha890@gmail.com

Update

Employee Options

Update Salary

Enter Employee Id

12345

Enter Salary

45000

Update

Updated Sucessfully

Employee Options

Update Job Role

Enter Employee Id

12345

Enter Job Role

Data Analyst

Update

Updated Sucessfully

Updated Sucessfully

Employee Options

Update Name

Enter Employee Id

12345

Enter Employee Name

Barkha

Update

Updated Sucessfully

To change the icon and title in Streamlit: To change the title and icon in Streamlit, first set the title for the browser tab. Then, specify an icon to appear in the tab, which can either be an emoji or an image.

Open the main.py file in your in-code editor, and add the following Streamlit code and put the icon link.

```
st.set_page_config(page_title="Employee Management System",page_icon=("https://www.codester.com/static/uploads/items/000/028/28925/icon.png"))
```

This code changed the icon and also the update the title with "Employment Management System" in streamlit.



SUMMARY OF PROJECT

Summary of Achievement:

We have successfully designed and developed an application that meets the specified requirements while incorporating various essential functionalities. The highlights of our application include:

- **User-Friendly Interface:** The application is designed with a simple and intuitive interface, making it easy to navigate for users of all levels. Minimal training is required to use the application effectively.
- **Efficient Functioning:** The system is optimized for performance, ensuring quick response times and accurate results. It streamlines operations, reduces manual effort, and enhances productivity.

Difficulties Encountered During the Project:

Throughout the development process, we faced several challenges that required careful problem-solving and research. Some of the key difficulties encountered were:

- **Learning Curve:** The project required learning new technologies and frameworks, which initially posed a challenge. A significant amount of time was invested in understanding the new functionalities.
- **Integration Issues:** Merging different components of the application and ensuring seamless interaction between them required extensive debugging and testing.

Limitations of the Project:

- **Limited Scope:** Due to time and resource constraints, the application is developed with basic functionalities. Some advanced features had to be postponed for future development.
- **Feature Restrictions:** Some features that could enhance user experience, such as AI-based automation, real-time analytics, or multi-platform compatibility, are not yet included.

Future Scope of the Project:

The application has great potential for future enhancements, and the following improvements can be implemented in upcoming versions:

- **Additional Functionalities:** More features can be incorporated to enhance the usability and effectiveness of the application.

- **Document Scanning:** A document scanning feature can be added to facilitate digital record-keeping of employee documents, reducing paperwork and improving accessibility.
- **Notification System:** Future iterations of the application can include real-time notifications via **SMS and Email**, ensuring better communication and timely updates.

By incorporating these future enhancements, the application can be made more robust, efficient, and valuable for users.

CONCLUSION

The Employee Management System is an efficient and scalable solution designed to streamline employee data management using Python, MySQL, and Streamlit. By integrating a structured MySQL database, the system ensures secure and organized data storage, allowing for easy retrieval and management of employee records. The backend, built with Python and MySQL-Connector, enables seamless execution of CRUD operations, while the frontend, developed using Streamlit, provides an intuitive and interactive user experience with features like image and video embedding, a navigation menu, and real-time data visualization.

This system significantly reduces manual workload, enhances data accuracy, and improves decision-making by providing well-structured employee records. Additionally, it lays a foundation for automation, making HR and Administrators processes more efficient. While the project successfully meets its objectives, there is great potential for future enhancements, such as document scanning, real-time notifications via SMS and Email, advanced security with role-based access control, cloud storage for scalability, and a mobile-friendly version. These improvements will further enhance the system's usability and adaptability to evolving business needs.

Overall, this project serves as a strong foundation for a more advanced and intelligent Employee Management System, demonstrating the power of integrating database management, backend processing, and frontend development into a single streamlined application. With further enhancements, it can become a comprehensive solution for modern businesses looking to automate and optimize their workforce management.