**PROJECT REPORT**

ON  
”**EMPLOYEE MANAGEMENT SYSTEM**”

Submitted to

**Sant Gadge Baba Amravati University, Amravati.**

In partial fulfillment of the requirements of

**M.Sc. (Computer Software) Final Year Examination**

Submitted by

**Miss Sakshi V. Shekar**

Under the guidance of

**Mr. B. R. Jalamkar**

**(Department of Computer Science)**



**Shri Shivaji Education Society Amravati’s**

**SHRI SHIVAJI SCIENCE COLLEGE**

**Amravati.**

**2023-2024**

**CERTIFICATE**

This is to certify that the Project Report entitled **EMPLOYEE MANAGEMENT SYSTEM** being submitted by **Miss** **Sakshi V.Shekar** in partial fulfillment for the award of Master of Science in Computer Software (Final Year) **Sant Gadge Baba Amravati University, Amravati** is a record of work carried out for the session 2023-24.

To the best of my knowledge, the matter presented in this project has not been presented earlier for a similar degree/diploma.

**Place : Amravati Project Guide**

**Date : Mr. B. R. Jalamkar**

**Internal Examiner External Examiner**

**Head  
Dept. of Computer Science**

**DECLARATION**

**To,**

**The Principal**

**Shri Shivaji Science College,**

**Amravati.**

**Respected Sir,**

I the undersigned, hereby declare that the Project work entitled **EMPLOYEE MANAGEMENT SYSTEM** submitted to **Sant Gadge Baba Amravati University, Amravati** is my independent work. This is my original work and has not been submitted anywhere for any degree/diploma. The system presented herein has not been duplicated from any other source.

I understand that any such copying is liable to be punished in any way the University authority may deem fit.

**Thanking You.**

**Place : Amravati**

**Date :**

**Yours Sincerely,**

**Sakshi V. Shekar**

**M.Sc. IInd (Sem-IV)**

**ACKNOWLEDGMENT**

We wish to express our sincere thanks to the many persons who helped us to develop our original project work.

First, we express our sincere thanks to Principal **Dr. G. V. Korpe**, Shri Shivaji Science College, Amravati for providing the infrastructure and facilities without which it would have been impossible to complete this hard task.

Foremost this is to**, Dr. M. M. Bhonde** , Head of the Department of Computer Science who has aided us in completing this project report. And I am thankful to **Mr.** **B. R. Jalamkar**, for their constant inspiration and guidance throughout this project work.

Our foremost thanks are to Other Staff, who have guided us in completing this project report, We take the opportunity to express our deep sense of gratitude and wholehearted thanks for his inspiration and guidance throughout this project.

I express my gratitude to all members of the teaching and non-teaching staff of the Department of Computer Science for their cooperation during the course.

Finally, we thank our friends and especially those who helped us in our endeavors.

**Place:** Amravati

**Date:**

| INDEX | | |
| --- | --- | --- |
| Sr. No. | Name of Topics | Page No. |
| 1 | INTRODUCTION  1.1 Abstract  1.2 Introduction  1.3 Objectives  1.4 Scope of Study  1.5 Significance | 1 |
| 2 | REQUIREMENT AND ANALYSIS  2.1 Purpose  2.2 Project Scope  2.3 Existing System  2.4 Proposed System  2.5 System Overview | 7 |
| 3 | IMPLEMENTATION ISSUES | 13 |
| 4 | SYSTEM DESIGN  4.1 Use Case Diagram  4.2 Class Diagram  4.3 Activity Diagram  4.3 Sequence Diagram  4.5 Data Flow Diagram | 15 |
| 5 | USER SCREENS | 21 |
| 6 | CODING | 26 |
| 7 | CONCLUSION | 40 |
| 8 | REFERENCES | 43 |

**INTRODUCTION**

**ABSTRACT**

The Employee Management System (EMS) is a comprehensive solution designed to streamline the management of employee data within an organization. Developed as a distributed application using Python Django framework, EMS serves as a centralized repository for storing and managing crucial information about employees, including personal details.

In today's dynamic business environment, the efficient management of human resources is paramount to organizational success. EMS aims to simplify this process by providing a user-friendly interface accessible to administrators and employees alike. With modules such as Administrator Login, Add New Employee, Update Employee, Delete Employee, View Employee, EMS offers a range of functionalities to cater to the diverse needs of organizations.

Employing a technology stack consisting of Python, HTML, CSS, and JavaScript, EMS ensures scalability, flexibility, and robustness. Leveraging the Django web framework and SQLite database, the system offers a reliable platform for storing and retrieving employee information securely.

This abstract provides an overview of the Employee Management System, highlighting its significance in enhancing organizational efficiency and productivity. Through its intuitive design and comprehensive features, EMS empowers organizations to effectively manage their workforce and drive business growth.

**INTRODUCTION**

The Employee Management System (EMS) is a crucial component in the modern workplace, facilitating the efficient handling of employee data and streamlining administrative processes. In today's dynamic business landscape, organizations are constantly seeking ways to enhance operational efficiency and productivity. The effective management of human resources plays a pivotal role in achieving these objectives.

The purpose of this project report is to provide a comprehensive overview of the Employee Management System developed using Python Django framework. EMS serves as a centralized platform for storing, managing, and accessing vital information about employees within an organization. By automating routine tasks and providing easy access to critical data, EMS empowers administrators to focus on strategic initiatives and drive organizational success.

This introduction sets the stage for understanding the significance of EMS in modern organizations. It outlines the objectives of the project, discusses the scope of the system, and provides an overview of the subsequent sections of the report. Through the development of EMS, the aim is to address the challenges associated with manual employee management processes and deliver a scalable, user-friendly solution that meets the needs of diverse organizations

**OBJECTIVES**

The Employee Management System (EMS) project is developed with the following objectives in mind:

1. Efficient Employee Data Management: The primary objective of EMS is to provide a centralized platform for storing and managing employee data. This includes personal details such as name, contact information, and employment history, as well as payroll information.

2. Streamlined Administrative Processes: EMS aims to streamline administrative tasks related to employee management, such as adding new employees, updating existing records, and deleting outdated information. By automating these processes, the system reduces manual effort and minimizes the risk of errors.

3. User-Friendly Interface: One of the key objectives of EMS is to provide a user-friendly interface that is accessible to administrators and employees alike. The system is designed to be intuitive and easy to navigate, ensuring that users can perform tasks with minimal training or assistance.

4. Enhanced Security: Security is a top priority in EMS. The system is designed to protect sensitive employee data from unauthorized access or tampering. This includes implementing robust authentication mechanisms, data encryption, and access controls to safeguard information.

5. Scalability and Flexibility: As organizations grow and evolve, their employee management needs may change. EMS is designed to be scalable and flexible, allowing it to adapt to the changing requirements of organizations of all sizes. Whether an organization has ten employees or ten thousand, EMS can accommodate its needs.

6. Integration with Existing Systems: EMS is developed with the goal of seamless integration with existing systems and processes within an organization. Whether it's integrating with payroll software, time tracking systems, or HRIS (Human Resources Information Systems), EMS aims to provide a cohesive solution that enhances overall efficiency.

**SCOPE OF STUDY**

The scope of the Employee Management System (EMS) project encompasses several key areas aimed at providing a comprehensive solution for managing employee data within an organization. The scope includes:

1. Employee Information Management: EMS enables the storage and management of various types of employee information, including personal details (such as name, contact information etc.), employment

3. Administrative Functions: EMS provides a range of administrative functions to manage employee data effectively. This includes functionalities such as adding new employees, updating existing records, deleting outdated information, and viewing employee profiles.

4. User Authentication and Authorization: The system includes robust user authentication mechanisms to ensure that only authorized personnel can access sensitive employee information. It allows administrators to manage user roles and permissions to control access to different parts of the system.

5. Security and Data Privacy: EMS prioritizes security and data privacy to protect sensitive employee information from unauthorized access or tampering. It implements encryption techniques, access controls, and audit trails to safeguard data integrity and confidentiality.

6. Scalability and Flexibility: The system is designed to be scalable and flexible, allowing it to accommodate the needs of organizations of varying sizes and complexities. Whether an organization has a few dozen employees or several thousand, EMS can adapt to its requirements.

7. Integration with External Systems: EMS supports integration with external systems and software applications, such as payroll systems, time tracking software, and human resources information systems (HRIS). This enables seamless data exchange and interoperability with existing organizational infrastructure.

8. User Interface and Experience: EMS offers a user-friendly interface that is intuitive and easy to navigate. It ensures that both administrators and employees can perform tasks efficiently without requiring extensive training or technical expertise.

**SIGNIFICANCE OF STUDY**

The development and implementation of the Employee Management System (EMS) carry significant importance in the context of modern organizations. Several key aspects highlight the significance of this study:

1. Enhanced Efficiency: EMS streamlines administrative processes related to employee management, reducing the time and effort required for tasks such as data entry, record-keeping.

2. Improved Accuracy: Manual handling of employee data is prone to errors and inconsistencies. EMS minimizes the risk of errors by ensuring data accuracy and integrity through centralized storage and automated processes.

3. Enhanced Security and Compliance: EMS prioritizes data security and compliance with privacy regulations by implementing robust security measures and access controls. This helps safeguard sensitive employee information from unauthorized access or misuse, reducing the risk of data breaches and ensuring compliance with legal requirements.

4. Strategic Decision-Making: Access to comprehensive employee data enables organizations to make data-driven decisions regarding workforce management, talent acquisition, training and development initiatives, and succession planning. EMS provides valuable insights into employee demographics, skills, performance, and trends, empowering organizations to align their human capital strategies with business objectives.

5. Improved Employee Experience: By providing employees with easy access to their personal information, payroll details, and self-service options, EMS enhances the employee experience and engagement. Employees can update their information, view their pay stubs, and request leave or other benefits conveniently through the system, leading to greater satisfaction and retention.

6. Scalability and Adaptability: As organizations evolve and grow, EMS can scale to accommodate changing needs and requirements. Whether it's expanding the workforce, integrating with new systems, or adapting to regulatory changes, EMS offers flexibility and scalability to support organizational growth and evolution.

**REQUIREMENT AND ANALYSIS**

**PURPOSE**

The purpose of developing the Employee Management System (EMS) is multifaceted and aimed at addressing various challenges faced by organizations in managing their workforce effectively. The primary purposes of EMS include:

1. Efficient Employee Data Management: EMS serves as a centralized repository for storing and managing comprehensive employee data, including personal details, employment history. The purpose is to streamline the collection, storage, and retrieval of employee information, thereby eliminating the need for disparate systems and manual record-keeping processes.

2. Streamlined Administrative Processes: The purpose of EMS is to automate routine administrative tasks related to employee management, such as adding new employees, updating records, and generating reports.

3. Enhanced Security and Compliance: The purpose of EMS is to ensure the security and integrity of employee data by implementing robust security measures, access controls, and encryption techniques.

4. Employee Self-Service: EMS offers self-service functionalities that enable employees to access and manage their own information, such as updating personal details, viewing pay stubs, and requesting leave or benefits.

5. Scalability and Flexibility: The purpose of EMS is to provide a scalable and flexible solution that can adapt to the evolving needs and requirements of organizations of all sizes.

**PROJECT SCOPE**

The scope of the Employee Management System (EMS) project encompasses the development and implementation of a comprehensive solution for managing employee data within an organization. The project scope includes the following components:

1. Functional Requirements:

* Administrator Login: Implementation of secure authentication mechanisms for administrators to access the system.
* Add New Employee: Functionality to add new employee records, including personal details and employment information.
* Update Employee: Capability to update existing employee records with changes in personal information or employment status.
* Delete Employee: Functionality to remove outdated or redundant employee records from the system.
* View Employee: Feature to view detailed information about individual employees or generate reports on employee data.

2. User Interface Design:

* Intuitive Interface: Designing a user-friendly interface that is easy to navigate and requires minimal training for users to operate.
* Responsive Design: Ensuring that the system interface is responsive and accessible across different devices and screen sizes.

3. Database Design:

* Database Schema: Designing a database schema to store employee information securely, ensuring data integrity and normalization.
* Integration with SQLite: Utilizing SQLite as the database management system to store and retrieve employee data efficiently.

4. Technology Stack:

* Programming Languages: Using Python for backend development, HTML, CSS, and JavaScript for frontend design.
* Web Framework: Employing Django, a high-level Python web framework, for rapid development and scalability.
* Development Environment: Utilizing PyCharm Integrated Development Environment (IDE) for coding and debugging purposes.

**EXISTING SYSTEM**

Before embarking on the development of the Employee Management System (EMS), it's essential to conduct an analysis of the existing system or processes in place within the organization. Understanding the strengths, weaknesses, and limitations of the current system provides valuable insights into the requirements and objectives of the new system. Here's an analysis of the existing system:

1. Manual Data Management: In many organizations, employee data management is primarily done manually using spreadsheets, paper-based records, or legacy systems.

2. Limited Accessibility: Access to employee data may be restricted to specific individuals or departments, leading to delays in information sharing and collaboration.

3. Security Concerns: Manual data management systems may lack adequate security measures to protect sensitive employee information from unauthorized access or misuse.

4. Lack of Integration: Existing systems may lack integration with other organizational systems such as payroll software, time tracking systems, or HRIS.

5. Limited Reporting and Analysis: Generating reports and analyzing employee data may be labor-intensive and time-consuming, requiring manual data extraction and manipulation.

6. Scalability and Flexibility Issues: Legacy systems or manual processes may lack scalability and flexibility to accommodate the evolving needs and growth of the organization.

7. User Experience Challenges: Users may experience challenges in navigating and using the existing system, leading to frustration and reduced productivity.

**PROPOSED SYSTEM**

The proposed Employee Management System (EMS) aims to address the limitations of the existing manual processes and legacy systems by providing a comprehensive, automated, and user-friendly solution for managing employee data within the organization. The proposed system includes the following key features and functionalities:

1. Centralized Employee Database: The core feature of the proposed system is a centralized database to store comprehensive employee data, including personal details, employment history, qualifications, skills.

2. Intuitive User Interface: The proposed system features a user-friendly interface with intuitive navigation and design elements, ensuring ease of use for both administrators and employees.

3. Authentication and Access Control: The system includes robust authentication mechanisms to verify the identity of users and prevent unauthorized access.

4. Employee Management Functionalities:

* Add New Employee: Administrators can easily add new employee records to the system, capturing relevant personal and employment details.
* Update Employee: The system allows for seamless updating of employee information, ensuring that records are always accurate and up to date.
* Delete Employee: Administrators can remove outdated or redundant employee records from the system with ease, maintaining data hygiene and compliance.
* View Employee: Comprehensive search and filter functionalities enable users to quickly locate and view employee profiles, facilitating informed decision-making.

6. Reporting and Analytics: The system includes robust reporting and analytics capabilities, allowing administrators to generate various reports and analyze employee data effectively.

7. Security and Compliance: The proposed system prioritizes data security and compliance with privacy regulations, implementing encryption, access controls, and audit trails to safeguard sensitive employee information.

**SYSTEM OVERVIEW**

The Employee Management System (EMS) is a comprehensive software solution designed to streamline the management of employee data within an organization. Built on a robust technological framework, EMS offers a range of functionalities to simplify administrative tasks, enhance data accuracy, and improve decision-making processes.

1. User Interface: EMS features an intuitive and user-friendly interface accessible to administrators and employees. The interface is designed for ease of navigation and provides seamless access to various modules and features of the system.

2. Database Management: A centralized database serves as the backbone of EMS, storing all employee-related information securely. The database is designed for scalability, ensuring that it can accommodate the growing needs of the organization.

3. Authentication and Access Control: Robust authentication mechanisms ensure secure access to the system. Role-based access control allows administrators to define user roles and permissions, controlling access to different parts of the system based on user roles.

4. Employee Management Modules:

* Add New Employee: Administrators can easily add new employee records to the system, capturing detailed information about each employee.
* Update Employee: The system allows for seamless updating of employee information, ensuring that records are always accurate and up to date.
* Delete Employee: Administrators can remove outdated or redundant employee records from the system, maintaining data hygiene and compliance.
* View Employee: Comprehensive search and filter functionalities enable users to quickly locate and view employee profiles, facilitating informed decision-making.

5. Integration with External Systems: EMS integrates seamlessly with existing systems and software applications within the organization. This integration ensures data consistency and eliminates the need for manual data entry across multiple systems.

**IMPLEMENTATION ISSUE**

**IMPLEMENTATION ISSUE**

During the development and implementation of the Employee Management System (EMS), several key issues may arise that need to be addressed to ensure the successful deployment and adoption of the system. These implementation issues include:

1.Technical Challenges: Integration with existing systems: Ensuring seamless integration with legacy systems and HRIS may pose technical challenges due to differences in data formats and protocols. Scalability: Designing the system to accommodate future growth and scalability requirements may require careful planning and architectural considerations.

2. Data Migration: Migrating existing employee data from legacy systems or manual records to the new EMS database may be a complex and time-consuming process, requiring data cleansing and validation.

3. User Training and Adoption: - User training: Providing comprehensive training to administrators and employees on how to use the new system effectively may be necessary to ensure adoption and user satisfaction. Change management: Overcoming resistance to change and managing expectations among stakeholders may require proactive communication and support throughout the implementation process.

4. Security and Compliance: Data security: Implementing robust security measures to protect sensitive employee information from unauthorized access or data breaches is crucial for maintaining compliance with privacy regulations. Regulatory compliance: Ensuring that the system complies with relevant data protection laws and regulations, such as GDPR or HIPAA, may require thorough auditing and documentation.

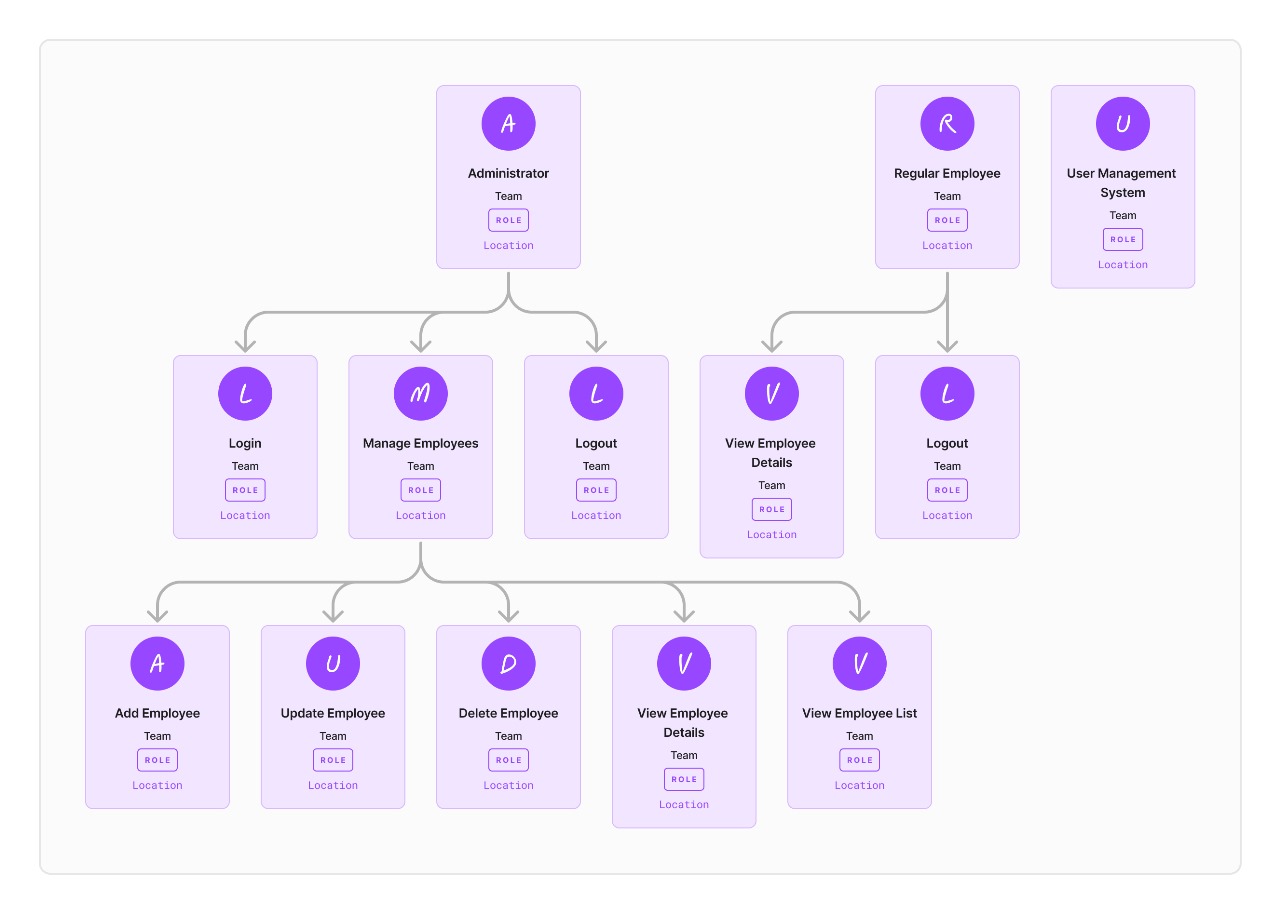
5. Performance Optimization: System performance: Optimizing system performance to ensure fast response times and scalability under heavy loads may require performance tuning and optimization of database queries and system architecture.

6. User Interface Design: User experience: Designing an intuitive and user-friendly interface that meets the needs of administrators and employees while adhering to usability best practices may require iterative design and testing.

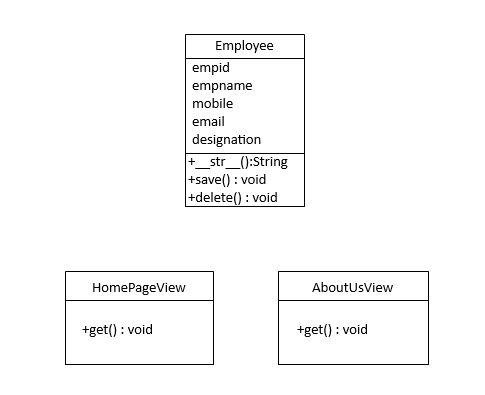
7. Testing and Quality Assurance: Testing: Conducting comprehensive testing, including unit testing, integration testing, and user acceptance testing, to identify and address any bugs or issues before deployment. Quality assurance: Ensuring that the system meets quality standards and requirements through rigorous testing and validation processes.

**SYSTEM DESIGN**

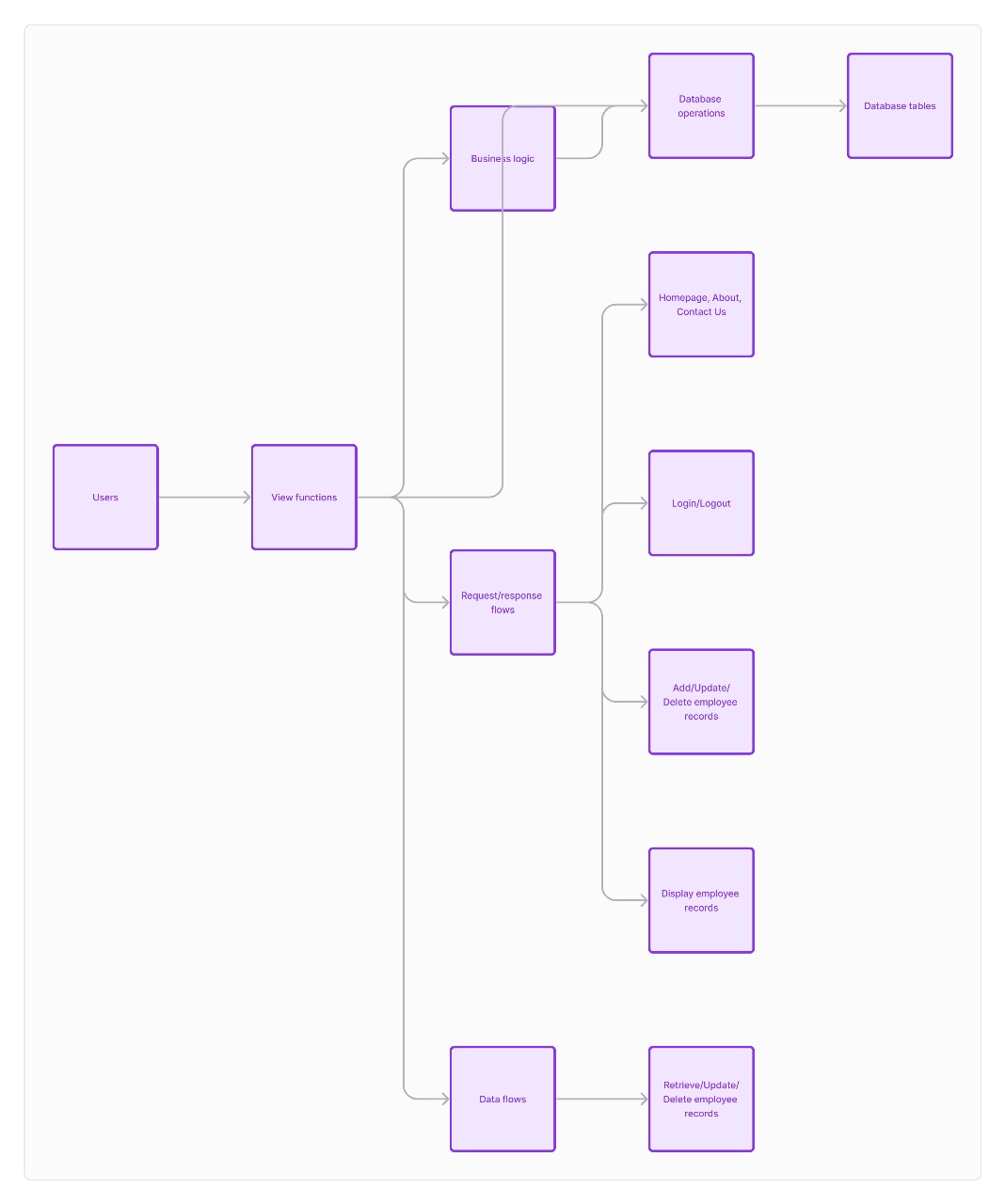
**USE CASE DIAGRAM**

****

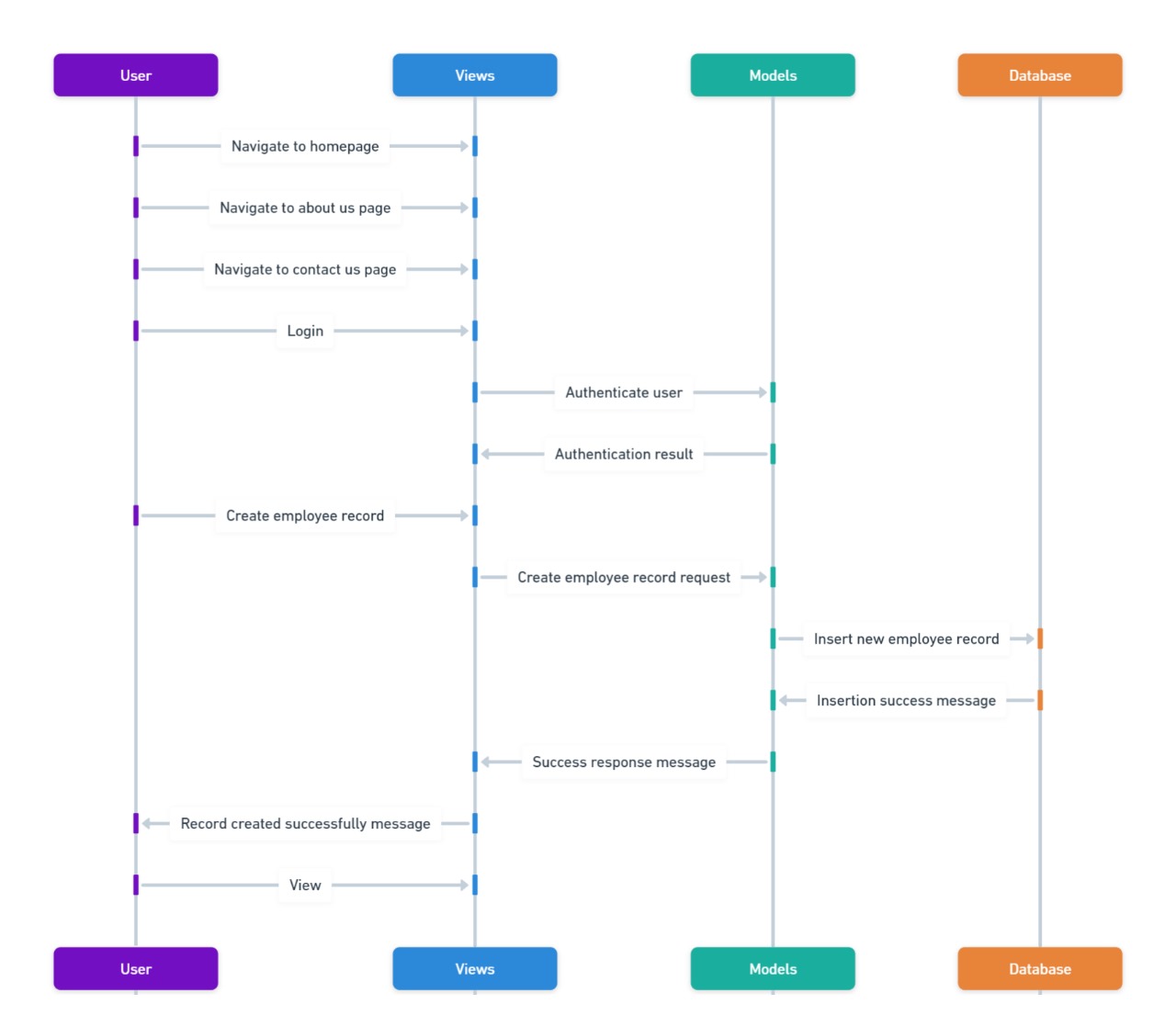
**CLASS DIAGRAM**

****

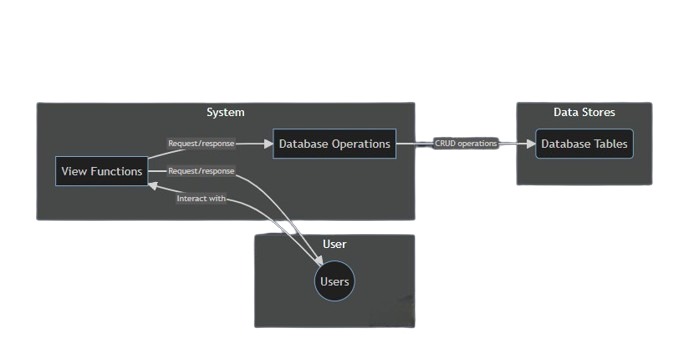
**ACTIVITY DIAGRAM**



**SEQUENCE DIAGRAM**

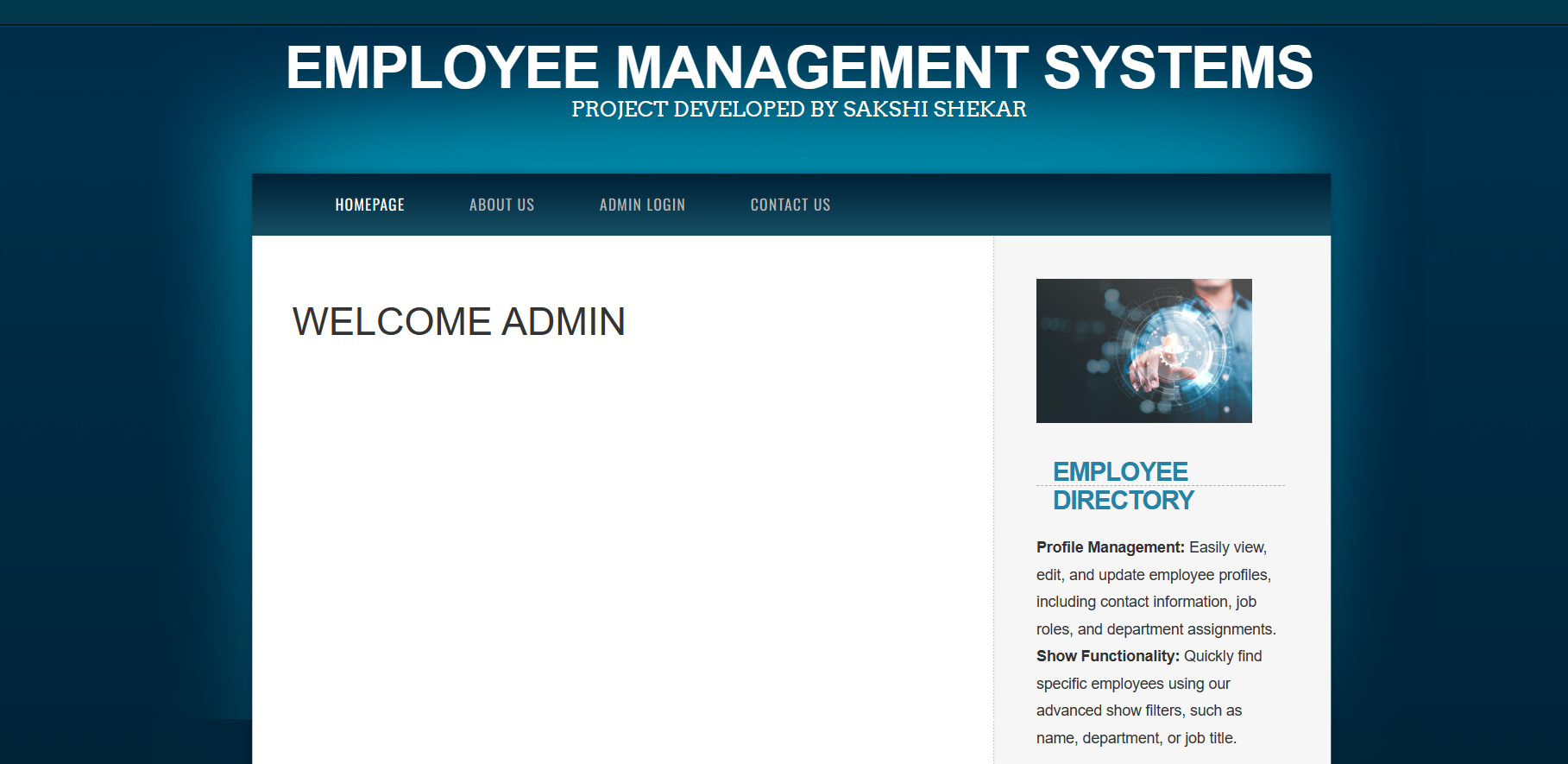


**DATA FLOW DIAGRAM**

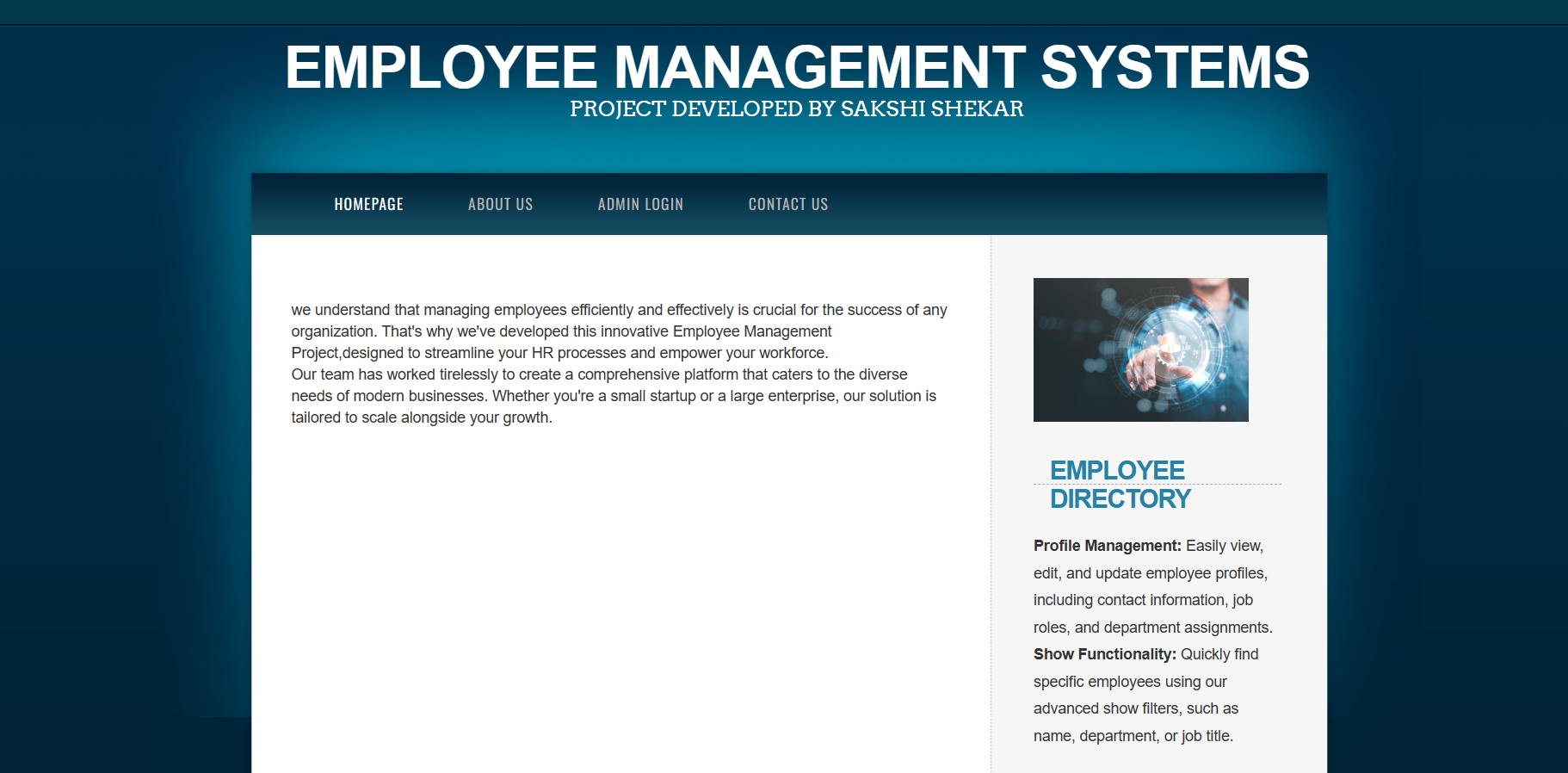


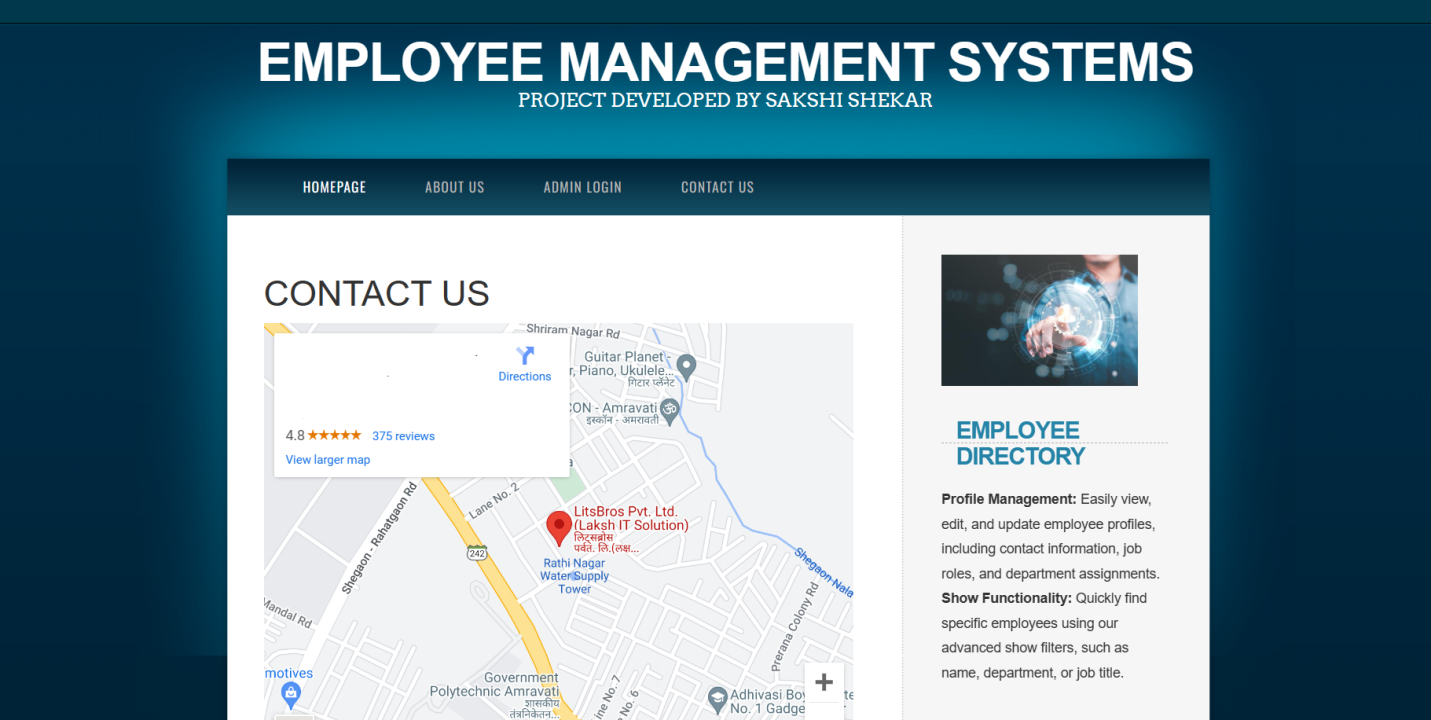
**USER SCREENS**

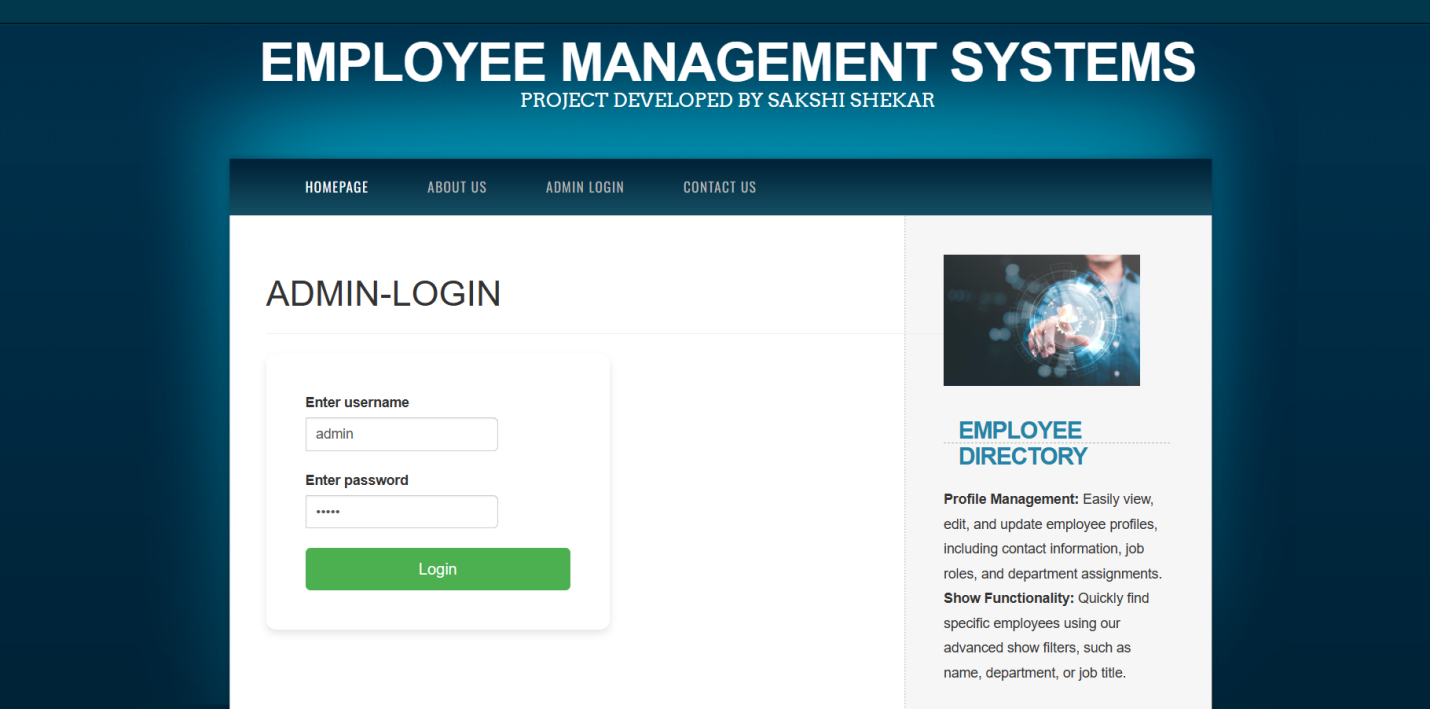
* **Home Page**



* **About us Page**



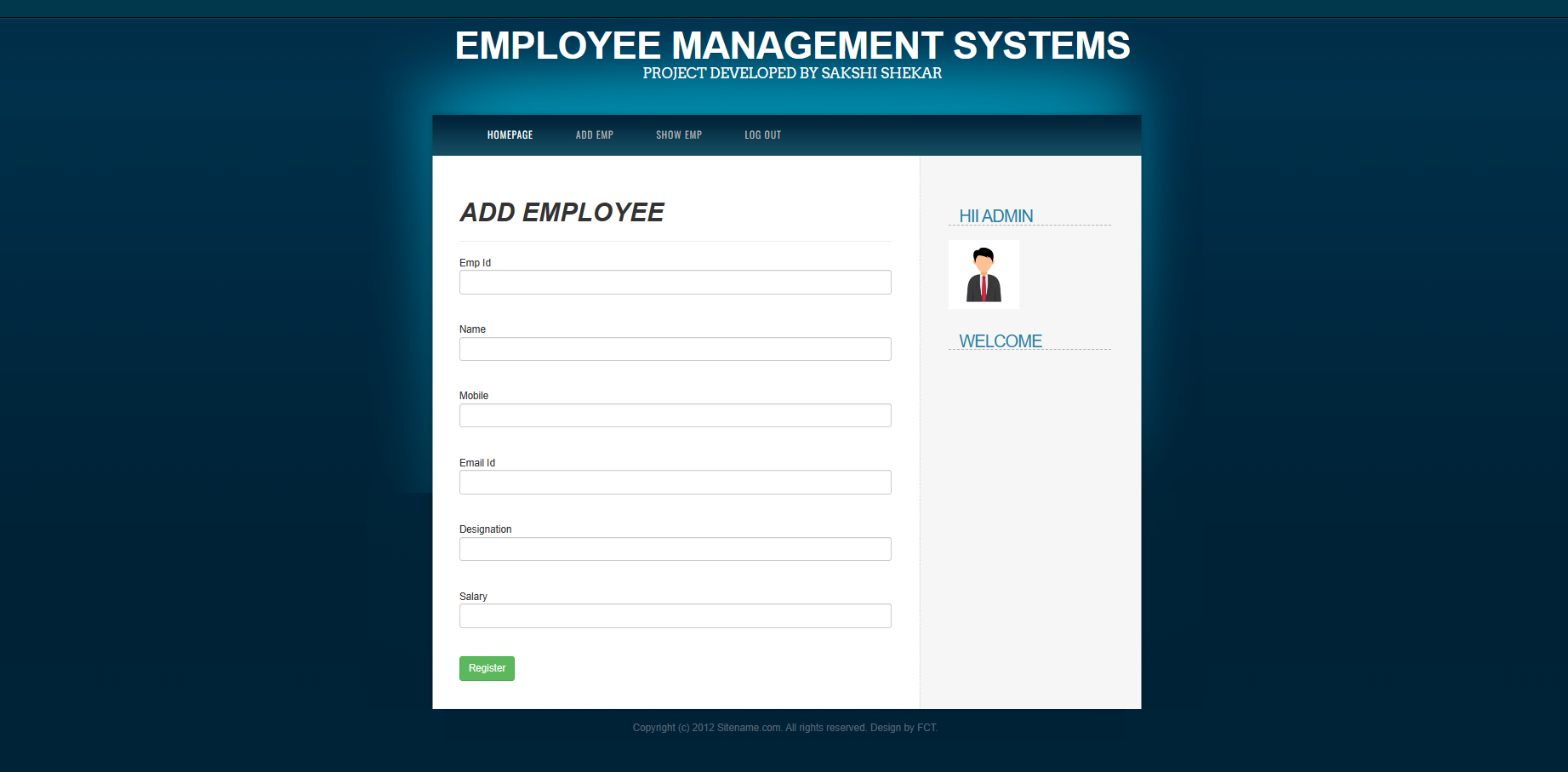
* **Contact Us Page**
* **Log-in Page**



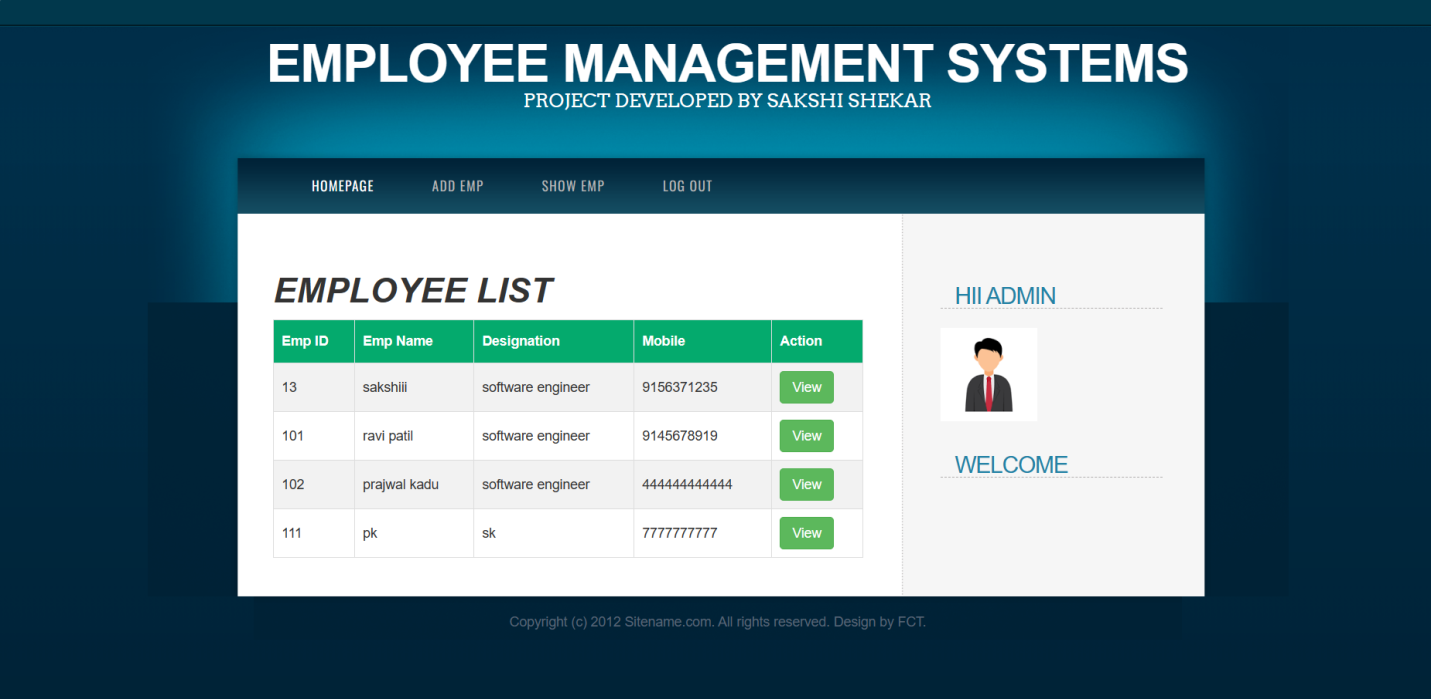
* **Admin Home Page**



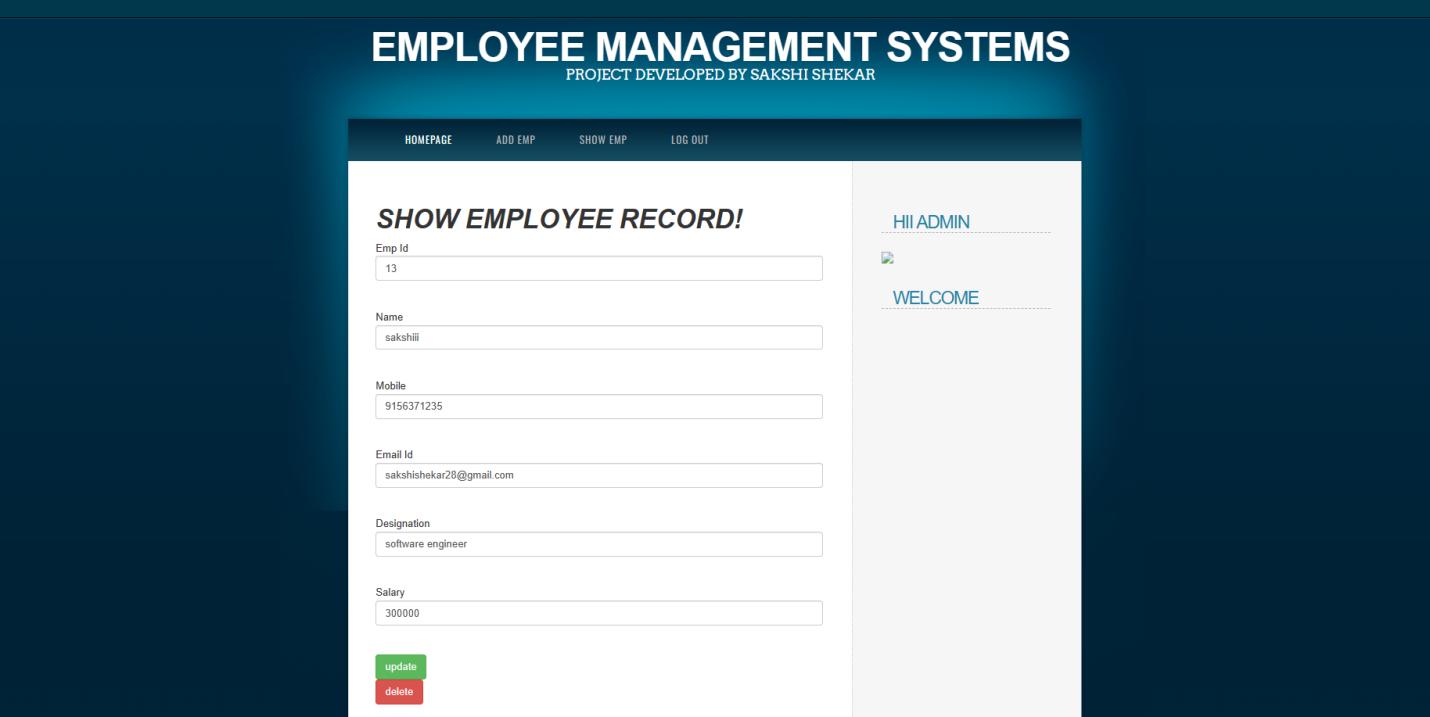
* **Add** Employee Page



* **Show Emp Page**



* **Single Record Update Page**



**CODING**

* **Views.py**

from django.shortcuts import render

from .models import Employee

# Create your views here.

def homepage(request):

return render (request,'index.html')

def about(request):

return render (request,'aboutus.html')

def Contact\_us(request):

return render (request,'contactus.html')

def welcome(request):

n=request.POST["txtname"]

print("the name is : "+ n)

return render (request,'welcome.html',{'name':n})

def adminlogin(request):

return render (request,'loginnew.html')

def admindashboard(request):

u = request.POST["txtusername"]

p = request.POST["txtpassword"]

if(u =="admin" and p =="super"):

return render (request,'admindashboard.html')

else:

return render(request, 'invalid.html')

def invalid(request):

return render (request,'invalid.html')

def addemployee(request):

return render (request,'addemployee.html')

def showemp(request):

x = Employee.objects.all()

return render (request,'showemp.html',{'emplist':x})

def logout(request):

return render (request,'index.html')

def regsuccess(request):

if request.method =='POST':

i = request.POST['txtEmpid']

n = request.POST['txtName']

e = request.POST['txtEmail']

m = request.POST['txtMobile']

d = request.POST['txtDesignation']

s = request.POST['txtSal']

x = Employee(request.POST)

x.empid = i

x.empname = n

x.emailid = e

x.mobile = m

x.empdesignation = d

x.empsalary = s

x.save()

return render (request,'adminRegSuccess.html')

def showsingleemployee(request,empid):

x = Employee.objects.get(empid=empid)

return render (request,'adminSingleRecord.html',{'emplist':x})

def updateemployee(request,empid):

if request.method == 'POST':

i = request.POST['txtEmpid']

n = request.POST['txtName']

e = request.POST['txtEmail']

m = request.POST['txtMobile']

d = request.POST['txtDesignation']

s = request.POST['txtSal']

x=Employee.objects.get(empid = empid)

x.empid = i

x.empname = n

x.emailid = e

x.mobile = m

x.empdesignation = d

x.empsalary = s

x.save()

return render (request,'AdminUpdateSuccess.html')

def deleteemployee(request,empid):

if request.method == 'POST':

x = Employee.objects.get(empid=empid)

x.delete()

return render(request, 'adminDeleteSuccess.html')

* **Index.html**

{% include "header.html"%}

<div id="menu-wrapper">

<div id="menu">

<ul>

<li class="current\_page\_item"><a href="/">Homepage</a></li>

<li><a href="about\_us">About Us</a></li>

<li><a href="admin\_login">Admin Login</a></li>

<li><a href="Contact\_us">Contact Us</a></li>

</ul>

</div>

</div>

<!-- end #menu -->

<div id="page">

<div id="page-bgtop">

<div id="page-bgbtm">

<div id="page-content">

<div id="content">

<h1> Welcome Admin</h1>

<div style="clear: both;">&nbsp;</div>

</div>

<!-- end #content -->

{% include "sidebar.html" %}

<!-- end #sidebar -->

</div>

<div style="clear: both;">&nbsp;</div>

</div>

</div>

</div>

<!-- end #page -->

</div>

<div id="footer">

<p>Copyright (c) 2012 Sitename.com. All rights reserved. Design by <a href="http://www.freecsstemplates.org/">FCT</a>.</p>

</div>

<!-- end #footer -->

</body>

</html>

* **Models.py**

from django.db import models

# Create your models here.

class Employee(models.Model):

empid = models.IntegerField(default=1,primary\_key=True)

empname = models.CharField(max\_length=200)

mobile = models.CharField(max\_length=100)

emailid = models.CharField(max\_length=200)

empdesignation = models.CharField(max\_length=200)

empsalary = models.IntegerField()

* **Urls.py**

from django.contrib import admin

from django.urls import path

from django.contrib import admin

from django.urls import path

from.import views

urlpatterns = [

path('admin/', admin.site.urls),

path('',views.homepage,name="my homepage"),

path('about\_us',views.about,name="aboutus page"),

path('Contact\_us',views.Contact\_us,name="contact"),

path('welcome', views.welcome, name="welcome"),

path('admin\_login',views.adminlogin, name="Admin login"),

path('admin\_dashboard',views.admindashboard, name="admin dashboard"),

path('invalid', views.invalid, name="invalid"),

path('add\_employee', views.addemployee, name="emp"),

path('show\_emp', views.showemp, name="showemp"),

path('log\_out', views.logout, name="logout"),

path('reg\_success',views.regsuccess, name="admin reg success"),

path('single-record/<int:empid>/',views.showsingleemployee, name="show single record"),

path('update\_employee/<int:empid>/',views.updateemployee, name="update emp"),

path('delete\_employee/<int:empid>/',views.deleteemployee, name="delete emp"),]

* **Loginnew.html**

{% include "header.html"%}

<div id="menu-wrapper">

<div id="menu">

<ul>

<li class="current\_page\_item"><a href="/">Homepage</a></li>

<li><a href="about\_us">About Us</a></li>

<li><a href="admin\_login">Admin Login</a></li>

<li><a href="Contact\_us">Contact Us</a></li>

</ul>

</div>

</div>

<!-- end #menu -->

<div id="page">

<div id="page-bgtop">

<div id="page-bgbtm">

<div id="page-content" style="display: flex;">

<div id="content" style="flex-grow: 1;"> <!-- Adjusted styles -->

<h1 style="margin-top: 20px;">Admin-login</h1>

<hr color="orange">

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

}

#page-content {

display: flex;

justify-content: center;

align-items: flex-start; /\* Adjusted alignment \*/

height: 100vh;

}

.form-box {

background-color: #ffffff;

padding: 40px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

width: 350px;

}

.input-group {

margin-bottom: 20px;

}

.input-group input {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

.input-group label {

display: block;

margin-bottom: 5px;

color: #333;

font-weight: bold;

}

.submit-btn {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

width: 100%;

font-size: 16px;

}

.submit-btn:hover {

background-color: #45a049;

}

</style>

<div class="login-wrapper">

<form method="post" action="admin\_dashboard" class="form">

{% csrf\_token %}

<div class="form-box">

<div class="input-group">

<label for="txtusername">Enter username</label>

<input type="text" name="txtusername" id="txtusername" class="form-control" required>

</div>

<div class="input-group">

<label for="txtpassword">Enter password</label>

<input type="password" name="txtpassword" id="txtpassword" class="form-control" required>

</div>

<input type="submit" value="Login" class="submit-btn">

</div>

</form>

</div>

<div style="clear: both;">&nbsp;</div>

</div>

<!-- end #content -->

{% include "sidebar.html" %}

<!-- end #sidebar -->

</div>

<div style="clear: both;">&nbsp;</div>

</div>

</div>

</div>

<!-- end #page -->

</div>

<div id="footer">

<p>Copyright (c) 2024. All rights reserved. Design by SS.

</p>

</div>

<!-- end #footer -->

</body>

</html>

* **Admindashboard.html**

{% include "adminheader.html" %}

<div id="page">

<div id="page-bgtop">

<div id="page-bgbtm">

<div id="page-content">

<div id="content">

<h1><b><i> Welcome Administrator </b></i></h1>

<div style="clear: both;">&nbsp;</div>

</div>

<!-- end #content -->

{% include "adminsidebar.html" %}

<!-- end #sidebar -->

</div>

<div style="clear: both;">&nbsp;</div>

</div>

</div>

</div>

<!-- end #page -->

</div>

<div id="footer">

<p>Copyright (c) 2012 Sitename.com. All rights reserved. Design by <a href="http://www.freecsstemplates.org/">FCT</a>.</p>

</div>

<!-- end #footer -->

</body>

</html>

**Addemployee.html**

{% include "adminheader.html"%}

<div id="page">

<div id="page-bgtop">

<div id="page-bgbtm">

<div id="page-content">

<div id="content">

<h1><b><i>Add Employee </i></b></h1>

<hr color="orange">

<form method="post" action="reg\_success">

{% csrf\_token %}

<tabel class=" tabel tabel-hover">

Emp Id<input type ="text" name="txtEmpid"class="form-control"><br><br>

Name<input type ="text" name="txtName"class="form-control"><br><br>

Mobile<input type ="text" name="txtMobile"class="form-control"><br><br>

Email Id<input type ="text" name="txtEmail"class="form-control"><br><br>

Designation<input type ="text" name="txtDesignation"class="form-control"><br><br>

Salary<input type ="text" name="txtSal"class="form-control"><br><br>

<input type="submit" value="Register" class="btn btn-success">

</tabel>

</form>

<div style="...">&nbsp;</div>

</div>

<!-- end #content -->

{% include "adminsidebar.html" %}

<!-- end #adminsidebar -->

</div>

<div style="clear: both;">&nbsp;</div>

</div>

</div>

</div>

<!-- end #page -->

</div>

<div id="footer">

<p>Copyright (c) 2012 Sitename.com. All rights reserved. Design by <a href="http://www.freecsstemplates.org/">FCT</a>.</p>

</div>

<!-- end #footer -->

</body>

</html>

* **Showemp.html**

{% include "adminheader.html" %}

<style>

#customers {

font-family: Arial, Helvetica, sans-serif;

border-collapse: collapse;

width: 100%;

}

#customers td, #customers th {

border: 1px solid #ddd;

padding: 8px;

}

#customers tr:nth-child(even){background-color: #f2f2f2;}

#customers tr:hover {background-color: #ddd;}

#customers th {

padding-top: 12px;

padding-bottom: 12px;

text-align: left;

background-color: #04AA6D;

color: white;

}

</style>

<div id="page">

<div id="page-bgtop">

<div id="page-bgbtm">

<div id="page-content">

<div id="content">

<h1><b><i>Employee List</i></b></h1>

<table id="customers">

<tr>

<th>Emp ID</th>

<th>Emp Name</th>

<th>Designation</th>

<th>Mobile</th>

<th>Action</th>

</tr>

{% for n in emplist %}

<tr>

<td>{{n.empid}}</td>

<td>{{n.empname}}</td>

<td>{{n.empdesignation}}</td>

<td>{{n.mobile}}</td>

<td> <a href="/single-record/{{n.empid}}" class ="btn btn-success">View </a></td>

</tr>

{% endfor %}

</table>

<div style="clear: both;">&nbsp;</div>

</div>

<!-- end #content -->

{% include "adminsidebar.html" %}

<!-- end #sidebar -->

</div>

<div style="clear: both;">&nbsp;</div>

</div>

</div>

</div>

<!-- end #page -->

</div>

<div id="footer">

<p>Copyright (c) 2012 Sitename.com. All rights reserved. Design by <a href="http://www.freecsstemplates.org/">FCT</a>.</p>

</div>

<!-- end #footer -->

</body>

</html>

* **AdminSingleRecord.html**

{% include "adminheader.html" %}

<div id="page">

<div id="page-bgtop">

<div id="page-bgbtm">

<div id="page-content">

<div id="content">

<h1><b><i> Show Employee Record! </b></i></h1>

<tabel class=" tabel tabel-hover">

<form method="post" action="/update\_employee/{{emplist.empid}}/">

{% csrf\_token %}

Emp Id<input type ="text" name="txtEmpid"class="form-control" value="{{emplist.empid}}"><br><br>

Name<input type ="text" name="txtName"class="form-control" value="{{emplist.empname}}"><br><br>

Mobile<input type ="text" name="txtMobile"class="form-control" value="{{emplist.mobile}}"><br><br>

Email Id<input type ="text" name="txtEmail"class="form-control" value="{{emplist.emailid}}"><br><br>

Designation<input type ="text" name="txtDesignation"class="form-control"value="{{emplist.empdesignation}}"><br><br>

Salary<input type ="text" name="txtSal"class="form-control"value="{{emplist.empsalary}}"><br><br>

<tr>

<td></td>

<td>

<input type="submit" value="update" class="btn btn-success">

</tr>

</td>

</form>

<form method="post" action="/delete\_employee/{{emplist.empid}}/">

{% csrf\_token %}

<input type="submit" value="delete" class="btn btn-danger">

</form>

</tabel>

<div style="clear: both;">&nbsp;</div>

</div>

<!-- end #content -->

{% include "adminsidebar.html" %}

<!-- end #sidebar -->

</div>

<div style="clear: both;">&nbsp;</div>

</div>

</div>

</div>

<!-- end #page -->

</div>

<div id="footer">

<p>Copyright (c) 2012 Sitename.com. All rights reserved. Design by <a href="http://www.freecsstemplates.org/">FCT</a>.</p>

</div>

<!-- end #footer -->

</body>

</html>

**CONCLUSION**

**FEATURE OF EMS**

1. Employee Database Management:

- Maintain a centralized repository of employee information including personal details, contact information, employment history, qualifications, skills, and performance records.

2. User Authentication and Access Control:

- Implement secure login mechanisms with username and password authentication.

3. Employee Lifecycle Management:

- Add, update, and delete employee records as per organizational requirements.

4. Self-Service Portals:

- Allow employees to access and update their own information such as contact details, emergency contacts, and bank information.

5. Performance Management:

- Capture performance evaluation data, including goals, achievements, feedback, and performance ratings.

6. Reporting and Analytics:

- Generate customizable reports and dashboards to analyze employee data, trends, and metrics.

10. Document Management:

- Store and manage employee documents and records such as resumes, contracts, certifications, and performance reviews.

**RECOMMENDATION**

1. User Feedback and Iterative Improvements: Gather feedback from users (administrators, HR personnel, and employees) regularly to identify pain points and areas for improvement.

2. Continuous Training and Support: Provide ongoing training and support to users to ensure they are proficient in using the system effectively.

3. Data Quality Assurance: Implement data validation checks and data cleansing processes to maintain data accuracy and integrity.

4. Enhanced Reporting and Analytics: Continuously enhance reporting and analytics capabilities to provide deeper insights into employee data and HR metrics.

5. Mobile Accessibility: Develop mobile-friendly interfaces or dedicated mobile apps to enable access to the EMS from smartphones and tablets.

6. Integration with Third-Party Systems: Explore opportunities for further integration with third-party systems such as accounting software, project management tools, and collaboration platforms.

7. Enhanced Security Measures: Stay updated with the latest security standards and best practices to address emerging cybersecurity threats. Implement multi-factor authentication, encryption, and regular security audits to safeguard sensitive employee data.

8. Compliance Monitoring and Auditing: Conduct regular compliance audits to ensure adherence to data protection regulations such as GDPR, HIPAA, or industry-specific standards.Maintain detailed records of data processing activities and implement mechanisms for data subject access requests and consent management.

9. Scalability and Performance Optimization: Monitor system performance and scalability to accommodate growing data volumes and user loads. Optimize database queries, caching mechanisms, and server infrastructure to maintain optimal performance levels.

10. Feedback Mechanism for Continuous Improvement: Establish a feedback mechanism within the EMS for users to submit suggestions, report issues, and track the status of their feedback.

**REFERENCES**

**REFERENCES**

References for the Employee Management System (EMS) project:

1. Django Documentation - Official documentation for the Django web framework. Available at: [https://docs.djangoproject.com/en/stable/](https://docs.djangoproject.com/en/stable/)

2. Python Documentation - Official documentation for the Python programming language. Available at: [https://docs.python.org/3/](https://docs.python.org/3/)

3. SQLite Documentation - Official documentation for the SQLite database management system. Available at: [https://sqlite.org/docs.html](https://sqlite.org/docs.html)

4. HTML Tutorial - W3Schools HTML tutorial for web development. Available at: [https://www.w3schools.com/html/](https://www.w3schools.com/html/)

5. CSS Tutorial - W3Schools CSS tutorial for web development. Available at: [https://www.w3schools.com/css/](https://www.w3schools.com/css/)

6. JavaScript Documentation - Mozilla Developer Network (MDN) JavaScript documentation. Available at: [https://developer.mozilla.org/en-US/docs/Web/JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript)