

SSN College of Engineering
Department of Computer Science and Engineering
UCS 1617 Mini Project Lab
End Semester Practical Examination

Name: S. Vishakan
Class: CSE-C
Reg. No.: 18 5001 196
Batch: B
Date: 07 May 2021

Problem Statement:

The aim is to develop a system to automate the payroll management process, wherein the payroll administrator can manage the finances of his/her employees. Employee details are to be stored in the database persistently for bookkeeping and auditing purposes. The process of automation also consists of the administrator being able to credit the basic salary to a specific employee according to his/her designation, and let the payroll system automatically calculate the other allowances like DA, HRA, Provident Fund and deductions as mandated.

The application should enable hassle-free, one-stop solution to all of a company's payroll accounting purposes. Each user of the system must have a secure portal to login and access their details. Administrator must have a privileged login to allow him/her to perform the necessary admin tasks.

The system must also abide by the legal statutory requirements for allotting salaries to employees, and make sure that no staff is underpaid. It should also enable transparency and be a trustworthy and reliable tool for bookkeeping.

Employees should also be able to file taxes from the payroll portal after logging into their accounts. The software must thus provide support to popular third-party tax vendor applications as well. An employee should also be able to view & generate his/her payslip on demand.

Administrators should be able to view the details of his/her employees at will, monitor the tax records, and further be able to add, edit or delete new/existing employees from the database.

The system should be easy to use and must have a low failure rate to ensure smooth usage. It should also emphasize on a low mean-time-to-repair so that the application can be serviced and maintained with low downtime.

It should also be able to handle huge workloads, and must be able to scale effectively with the size of the organization, and provide seamless service to every user.

Classes and Noun Phrases:

1. Tangible Objects:

- a. Employee Record
- b. Pay Slip Receipt
- c. Tax Application
- d. Tax Record

2. Places:

- a. Company
- b. Accounting Office

3. Transactions:

- a. New Employee Addition
- b. Employee Updation
- c. Employee Deletion
- d. Crediting Salary
- e. Paying Tax

4. Roles:

- a. Administrator
- b. Employee
- c. Tax System

5. Containers:

- a. Employee Records
- b. Tax Records

6. Events:

- a. Login
- b. Add Employee
- c. Update Employee
- d. Delete Employee
- e. View Records
- f. View Payslip
- g. Pay Taxes
- h. Logout

7. Abstract Noun Concepts:

- a. Designation
- b. Date of Join
- c. Allowances

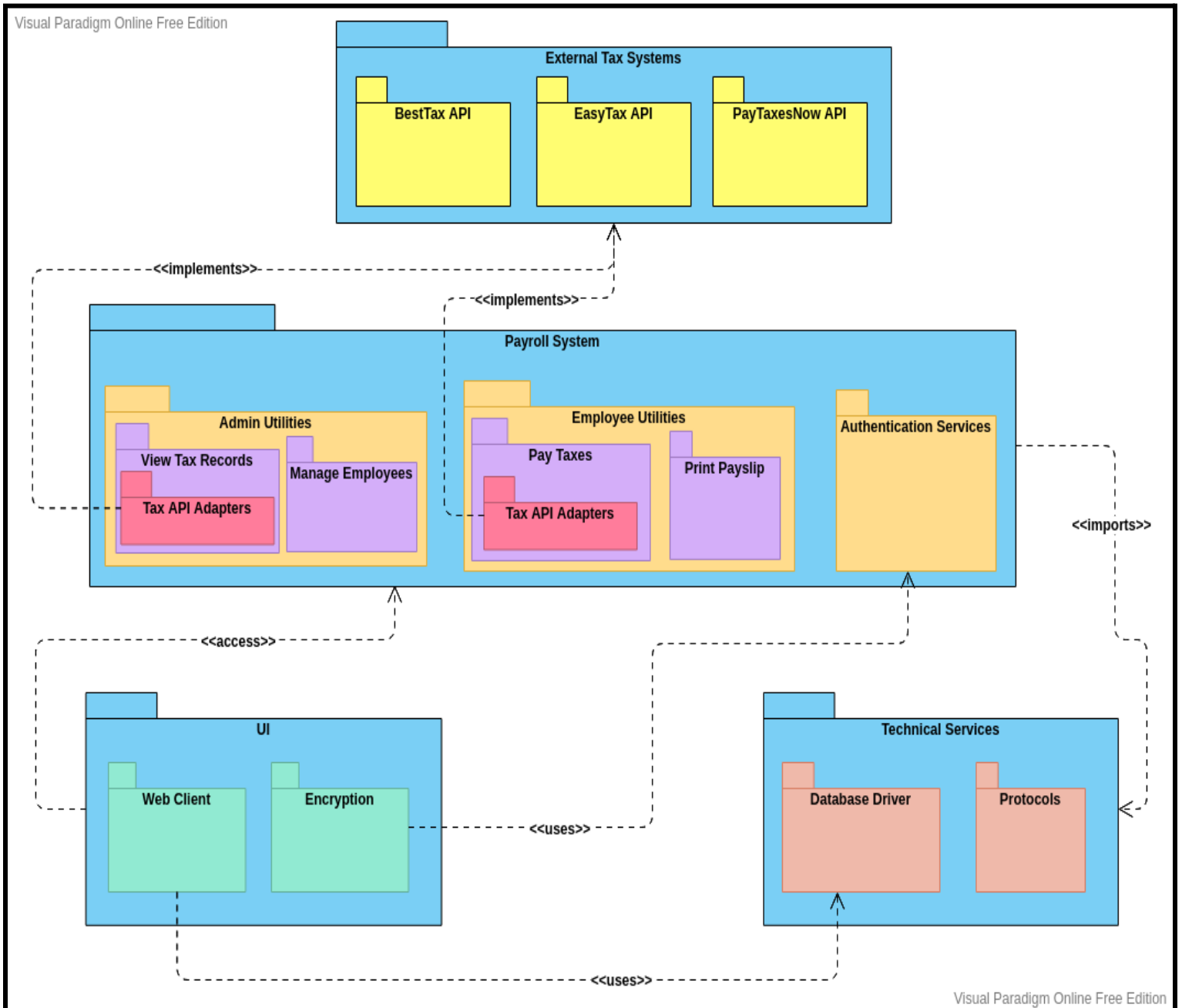
8. Classes:

- a. User (Abstract class)
- b. Administrator (Implemented from User class)
- c. Employee (Implemented from User class)
- d. Payslip
- e. Credentials
- f. TaxClient (Adapter Class)

9. Associations:

- a. Administrator (1:1 implements) User
- b. Employee (1:1 implements) Employee
- c. User (1:1 uses) Credentials
- d. Administrator (1:N manages) Employee
- e. Administrator (1:N grants) Payslip
- f. Employee (1:1 receives) Payslip
- g. Employee (N:M utilizes) TaxClient

Package Diagram:



Test Cases:

Test ID	Scenario	Data	Predicted Result	Actual Result	Status
1.	Admin - Login	Credentials: Invalid	Invalid Login Alert	Invalid Login Alert	Pass
2.	Admin - Login	Credentials: Valid	Proceed to Admin Page	Proceed to Admin Page	Pass
3.	Employee - Login	Credentials: Invalid	Invalid Login Alert	Invalid Login Alert	Pass
4.	Employee - Login	Credentials: Valid	Proceed to Employee Page	Proceed to Employee Page	Pass
5.	Admin - Add Record	Details: Insufficient/Invalid	Alert Insufficient/Invalid Details	Alert Insufficient/Invalid Details	Pass
6.	Admin - Add Record	Details: Valid	Add Success	Add Success	Pass
7.	Admin - Credit Salary	Details: Invalid Employee ID	Alert Invalid Details	Alert Invalid Details	Pass
8.	Admin - Credit Salary	Details: Valid	Salary Credited	Salary Credited	Pass
9.	Admin - View Records	Database: Empty	No Records	No Records	Pass
10.	Admin - View Records	Database: Has Records	Display Records	Display Records	Pass
11.	Admin - Delete Record	Details: Valid Employee ID	Delete Success	Delete Success	Pass

12.	Admin - Delete Record	Details: Invalid Employee ID	Employee Not Found	Employee Not Found	Pass
13.	Admin - Update Record	Details: Valid Employee ID	Update Success	Update Success	Pass
14.	Employee - View Details	Database: Employee Exists	Show Details	Show Details	Pass
15.	Employee - View Payslip	Database: Employee Exists	Show Payslip	Show Payslip	Pass
16.	Employee - Pay Taxes	Database: Employee Exists	Proceed to Tax System	Proceed to Tax System	Pass
17.	Employee - View Tax Records	Database: Employee Exists	Show Tax Records	Show Tax Records	Pass
18.	Admin/ Employee - Change Credentials	Valid Previous Credentials	Change Success	Change Success	Pass
19.	Admin/ Employee - Change Credentials	Invalid Previous Credentials	Change Failure	Change Failure	Pass
20.	Admin - Export Tax Data	Valid Credentials	Export Success	Export Success	Pass
21.	Admin - Export Records	Valid Credentials	Export Success	Export Success	Pass
22.	Employee - File Tax Application	Database: Employee Exists	Proceed to Tax System	Proceed to Tax System	Pass
23.	Admin/ Employee - Logout	-nil-	Logout Success	Logout Success	Pass

Improvisation with Design Patterns:

Design Patterns are very ubiquitous in the realm of **Object-Oriented Design** and it can help developers identify well-known and well-documented patterns that ease the process of implementation and design in the **SDLC**.

These patterns are tried and tested by experts and have proven to be highly useful.

In our application, we can make use of the **MVC Pattern** to separate out concerns to provide **low coupling and high cohesion**. Model is represented by the database, View is represented by the UI, Controller is represented by the underlying backend logic, which in my implementation is done using Python's Flask library. Thus, UI need not worry about modification/manipulation of data, the database need not worry about presentation of its contents and the controller need not worry about data persistence or presentation. This achieves a high degree of **modular code** which can be debugged and maintained in a significantly easier manner.

The **Adapter Pattern** can be made use of here to bring in many third-party tax vendors. Since each vendor might use a different set of protocols and functions, the adapter class can provide seamless integration to all of these vendors and abstract away the connection & implementation logic from the core application. It only provides a single API for the core payroll system to use, which specifies the required vendor, and the rest is taken care of by the implementing objects of the adapter class.

The Employee, Admin and Payslip classes come under the pattern of **Information Expert**. These classes must not be concerned with database access and other functionalities, and must provide data to the other classes. Database connectivity can be handled by a separate Driver class, to keep the actual implementation logic for database querying away from the core control logic.

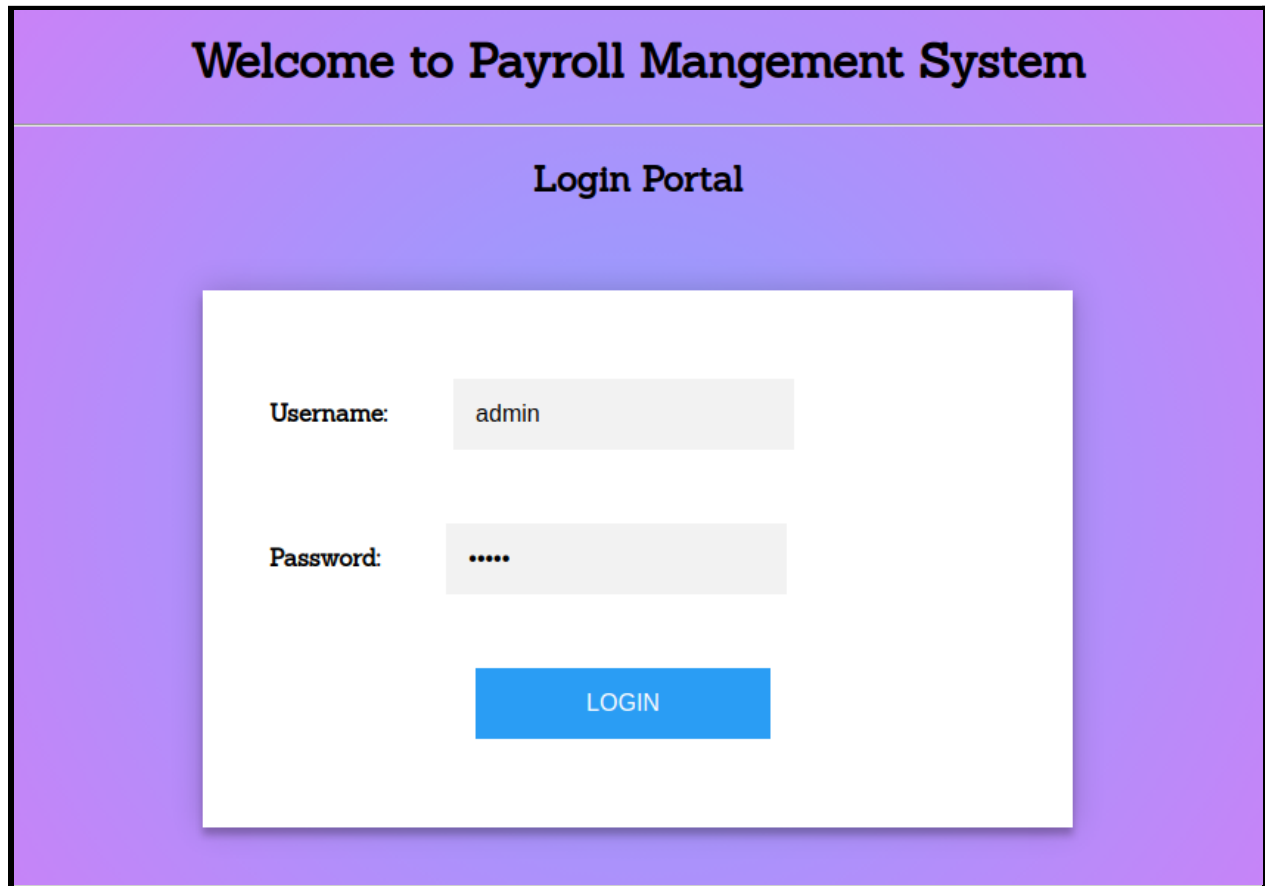
The navigation menus and the tax interface can also make use of the **Facade Pattern**, allowing the client to interact with the system to hide the complexities.

The **Chain of Responsibility Pattern** can be utilized to pass the responsibility of handling tax requests from the payroll system to the external third-party tax system.

This provides a great deal of **extensibility** to the software, as the addition of new features do not require major changes to be made to the other parts of the software.

We also simplify the process of test-case planning and unit-testing can be done to different modules separately and integration testing can be done at a later point of time to check if all the modules work together seamlessly.

Results:



The screenshot displays a web interface for a Payroll Management System. At the top, a purple header bar contains the text "Welcome to Payroll Mangement System" in bold black font. Below this, a light purple section is titled "Login Portal" in bold black font. The central area is a white box with a subtle shadow, containing the login form. The form has two input fields: "Username:" with the text "admin" and "Password:" with five dots. A blue "LOGIN" button is positioned below the password field.

Welcome to Payroll Mangement System

Login Portal

Username: admin

Password:

LOGIN

Login Portal

Payroll Management System

Admin Navigation Menu

ADD AN EMPLOYEE

CREDIT SALARY

VIEW EMPLOYEES

DELETE EMPLOYEE

UPDATE EMPLOYEE
DETAILS

LOGOUT

Admin Menu

Payroll Management System

Add A Record

Employee ID

P2020

Name

Sherlock Holmes

Age

39

Gender

☒ Male ☐ Female ☐ Other

Designation

Manager

Address

221B, Baker Street, London, UK

Phone Number

44321223

Date of Join

19/02/2021



REGISTER

CLEAR ALL

Add New Employee

Payroll Management System

Credit Salary

Employee ID

P2020

Basic Pay

55000

CREDIT SALARY

CLEAR ALL

Payroll Management System

Update Employee Details

Employee ID

P2020

Phone Number

44321222

UPDATE

CLEAR ALL

Credit Salary & Update Employee Details

Employee Details

Employee ID	Name	Age	Gender	Designation	Address	Phone	Date of Join
P2002	John Doe	25	Male	Team Leader	25, Sunset Blvd, NY	1234567890	2021-05-06
P2003	Jane Doe	22	Female	Trainee	23, Sunrise Blvd, NY	22323113	2021-04-30
P2011	James McGill	45	Male	Manager	21, Abcd Road, Chennai	23442122	2021-04-24
P2020	Sherlock Holmes	39	Male	Manager	221B, Baker Street, London, UK	44321222	2021-02-19

Payroll Management System

Delete Employee

Employee ID

P2011

DELETE

CLEAR ALL

[View Employee Records & Delete An Employee](#)

Payroll Management System

Employee Navigation Menu

VIEW PROFILE

VIEW PAYSリップ

LOGOUT

Employee Menu, Employee Details & Employee Payslip

Employee Details	
Employee ID	P2020
Name	Sherlock Holmes
Age	39
Gender	Male
Designation	Manager
Address	221B, Baker Street, London, UK
Phone	44321222
Date of Join	2021-02-19

Employee Details	
Basic Pay	55000.0
DA	11000.0
HRA	6600.0
Provident Fund	3850.0
Deductions	1650.0
Gross Pay	67100.0

Implementation:

App.PY

```
from flask import Flask, render_template, request, redirect
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] =
"mysql://admin:password@localhost/mysql"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

class Emp(db.Model):
    id = db.Column(db.String(10), primary_key=True)
    ename = db.Column(db.String(100), nullable=False)
    age = db.Column(db.Integer, nullable=False)
    gender = db.Column(db.String(10), nullable=False)
    job = db.Column(db.String(80), nullable=False)
    address = db.Column(db.String(300), nullable=False)
    phone = db.Column(db.Integer, nullable=False)
    doj = db.Column(db.DateTime, nullable=False)
    bpay = db.Column(db.Float, nullable=False, default=0.0)
    da = db.Column(db.Float, nullable=False, default=0.0)
    hra = db.Column(db.Float, nullable=False, default=0.0)
    pf = db.Column(db.Float, nullable=False, default=0.0)
    dedn = db.Column(db.Float, nullable=False, default=0.0)
    gpay = db.Column(db.Float, nullable=False, default=0.0)

    def __repr__(self):
        return "<Emp %r>" % self.id

class Creds(db.Model):
    userid = db.Column(db.String(30), primary_key=True)
```



```

password = db.Column(db.String(30), nullable=False)

def __repr__(self):
    return "<User %r>" % self.userid

@app.route("/", methods=['POST', 'GET'])
def index():

    if request.method == "POST":
        uname = request.form["uname"]
        pwd = request.form["pwd"]

        creds = Creds.query.get(uname)

        try:
            if uname == "admin" and creds.password == pwd:
                return redirect("/admin")
            elif creds.password == pwd:
                return redirect("/employee/" + uname)
            else:
                return render_template("login.html")

        except Exception as e:
            return "There was a problem finding your account."

    else:
        return render_template("login.html")

@app.route("/admin", methods=['POST', 'GET'])
def admin_dashboard():
    return render_template("admin.html")

@app.route("/employee/<string:eid>", methods=['POST', 'GET'])
def employee_dashboard(eid):
    return render_template("employee.html",
employee=Emp.query.get_or_404(eid))

```

```

@app.route("/viewprofile/<string:eid>", methods=['POST', 'GET'])
def view_profile(eid):
    emp = [Emp.query.get_or_404(eid)]
    return render_template("view.html", employees=emp, payslip=False,
vertical=True)

@app.route("/viewpayslip/<string:eid>", methods=['POST', 'GET'])
def view_payslip(eid):
    emp = [Emp.query.get_or_404(eid)]
    return render_template("view.html", employees=emp, payslip=True,
vertical=True)

@app.route("/add", methods=['POST', 'GET'])
def add_employee():
    if request.method == "POST":
        eid = request.form["id"]
        ename = request.form["pname"]
        age = int(request.form["age"])
        gender = request.form["gender"]
        job = request.form["job"]
        addr = request.form["addr"]
        phone = int(request.form["phone"])
        doj = request.form["doj"]

        doj = datetime.strptime(doj, '%Y-%m-%d')

        #print(type(eid), type(ename), type(age), type(gender), type(job),
type(addr), type(phone), type(doj))

        new_emp = Emp(id=eid, ename=ename, age=age, gender=gender,
job=job, address=addr, phone=phone, doj=doj)

        new_user = Creds(userid=eid, password=eid)

    try:
        db.session.add(new_emp)
        db.session.add(new_user)

```

```

        db.session.commit()
        return redirect("/admin")

    except Exception as e:
        return "There was a problem adding that employee record."

else:
    return render_template("add.html")

@app.route("/credit/", methods=['POST', 'GET'])
def credit_employee():

    if request.method == "POST":
        emp = Emp.query.get_or_404(request.form["id"])

        emp.bpay = float(request.form["salary"])

        if emp.job == 'Trainee':
            emp.da = 0.1 * emp.bpay
            emp.hra = 0.05 * emp.bpay
            emp.pf = 0.02 * emp.bpay
            emp.dedn = 0.01 * emp.bpay

        elif emp.job == 'Team Leader':
            emp.da = 0.15 * emp.bpay
            emp.hra = 0.1 * emp.bpay
            emp.pf = 0.05 * emp.bpay
            emp.dedn = 0.03 * emp.bpay

        elif emp.job == 'Manager':
            emp.da = 0.2 * emp.bpay
            emp.hra = 0.12 * emp.bpay
            emp.pf = 0.07 * emp.bpay
            emp.dedn = 0.03 * emp.bpay

        elif emp.job == 'HR':
            emp.da = 0.3 * emp.bpay
            emp.hra = 0.15 * emp.bpay
            emp.pf = 0.1 * emp.bpay

```

```

        emp.dedn = 0.05 * emp.bpay

    emp.gpay = emp.bpay + emp.da + emp.hra - emp.pf - emp.dedn

    try:
        db.session.commit()
        return redirect("/admin")

    except Exception as e:
        return "There was a problem crediting salary to that employee."

    else:
        return render_template("credit.html")

@app.route("/delete", methods=['POST', 'GET'])
def delete_employee():
    if request.method == "POST":
        emp = Emp.query.get_or_404(request.form["eid"])
        creds = Creds.query.get_or_404(request.form["eid"])
        try:
            db.session.delete(emp)
            db.session.delete(creds)
            db.session.commit()
            return redirect("/admin")
        except Exception as e:
            return "There was a problem deleting that employee."

    else:
        return render_template("delete.html")

@app.route("/update", methods=['POST', 'GET'])
def update_employee():
    if request.method == "POST":
        emp = Emp.query.get_or_404(request.form["id"])
        try:
            emp.phone = request.form["phone"]
            db.session.commit()
            return redirect("/admin")

```

```
        except Exception as e:
            return "There was a problem updating that employee's record."

    else:
        return render_template("update.html")

@app.route("/viewrecords", methods=['POST', 'GET'])
def view_records():
    employees = Emp.query.order_by(Emp.id).all()
    #print(employees, type(employees))

    return render_template("view.html", employees=employees, payslip=False,
vertical=False)

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

Base.HTML

```
<!DOCTYPE html>
<html>

<head>
  <link rel="stylesheet" href="{{url_for('static',
filename='css/styles.css')}}">
  {% block head %}{% endblock %}
</head>

<body>
  {%block body %}{% endblock %}
</body>

</html>
```

Admin.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS Admin</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Payroll Management System</h1>
<hr>
<h2 class="titleText">Admin Navigation Menu</h2><br><br>
<div align="center" class="reg-page">
    <button class="btn" onclick="window.open('/add')">Add An
Employee</button><br><br>
    <button class="btn" onclick="window.open('/credit')">Credit
Salary</button><br><br>
    <button class="btn" onclick="window.open('/viewrecords')">View
Employees</button><br><br>
    <button class="btn" onclick="window.open('/delete')">Delete
Employee</button><br><br>
    <button class="btn" onclick="window.open('/update')">Update Employee
Details</button><br><br>
    <button class="btn" onclick="logoutUser()">Logout</button><br><br>
</div>

<script>
    logoutUser = () => {
        //redirect to home page
        location.href = "/";
    }
</script>
{% endblock %}
```

Employee.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS Employee</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Payroll Management System</h1>
<hr>
<h2 class="titleText">Employee Navigation Menu</h2><br><br>
<div align="center" class="reg-page">
    <button class="btn"
onclick="window.open('/viewprofile/{{employee.id}}')">View
Profile</button><br><br>
    <button class="btn"
onclick="window.open('/viewpayslip/{{employee.id}}')">View
Payslip</button><br><br>
    <button class="btn" onclick="logoutUser()">Logout</button><br><br>
</div>

<script>
    logoutUser = () => {
        //redirect to home page
        location.href = "/";
    }
</script>
{% endblock %}
```


Login.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS Portal</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Welcome to Payroll Management System</h1>
<hr>
<h2 class="titleText">Login Portal</h2><br><br>
<form align="center" class="reg-page" method="POST">
    <label for="uname">Username:</label>
    <input type="text" name="uname" id="uname" required/><br><br>
    <label for="pwd">Password:</label>
    <input type="password" name="pwd" id="pwd" required/><br><br>
    <input type="submit" value="Login" id="button" class="btn" />
</form>
{% endblock %}
```

Add.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS - Add Employee</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Payroll Management System</h1>
<hr>
<h2 class="titleText">Add A Record</h2>
    <br><br>

    <div class="reg-page">
        <form name="regform" id="regform" method="POST">
            <label for="id">Employee ID</label><br>
            <input type="text" id="id" name="id" placeholder="P2008"
size="10" required pattern="^[P0-9]+$"
                title="Enter in the specified format">
            <br><br>

            <label for="pname">Name</label><br>
            <input type="text" id="pname" name="pname" placeholder="John
Doe" size="40" required pattern="^[a-zA-Z\s]+$"
                title="Enter English alphabets only">
            <br><br>

            <label for="age">Age</label><br>
            <input type="number" id="age" name="age" min="1" max="120"
step="1" placeholder="21" required>
            <br><br>

            <label>Gender</label>
            <br>
            <input type="radio" id="male" name="gender" value="Male"
required>
            <label for="male">Male</label>
```

```

        <input type="radio" id="female" name="gender" value="Female"
required>
        <label for="female">Female</label>
        <input type="radio" id="other" name="gender" value="Other"
required>
        <label for="other">Other</label>

        <br><br>

        <label for="job">Designation</label><br>
        <select id="job" name="job" required>
            <option>Trainee</option>
            <option>Team Leader</option>
            <option>Manager</option>
            <option>HR</option>
        </select>
        <br><br>

        <label for="addr">Address</label><br>
        <textarea cols="30" rows="3" id="addr" name="addr"
required></textarea>
        <br><br>

        <label for="phone">Phone Number</label><br>
        <input type="text" id="phone" name="phone"
placeholder="1234567890" size="20" required>
        <br><br>

        <label for="doj">Date of Join</label><br>
        <input type="date" id="doj" name="doj" required><br><br>

        <input type="submit" class="btn" name="Submit" id="Submit"
value="Register"></input><br>
        <input type="reset" class="btn" name="Reset" id="Reset"
value="Clear All"></input>
    </form>
</div>
{% endblock %}

```

Credit.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS - Credit Employee</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Payroll Management System</h1>
<hr>
<h2 class="titleText">Credit Salary</h2>
    <br><br>

    <div class="reg-page">
        <form name="regform" id="regform" method="POST">
            <label for="id">Employee ID</label><br>
            <input type="text" id="id" name="id" placeholder="P2008"
size="10" required pattern="^P[0-9]+$"
            title="Enter in the specified format">
            <br><br>

            <label for="salary">Basic Pay</label><br>
            <input type="number" id="salary" name="salary"
placeholder="66000" size="20" required>
            <br><br>

            <input type="submit" class="btn" name="Submit" id="Submit"
value="Credit Salary"></input><br>
            <input type="reset" class="btn" name="Reset" id="Reset"
value="Clear All"></input>
        </form>
    </div>
{% endblock %}
```

Delete.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS - Delete Employee</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Payroll Management System</h1>
<hr>
<h2 class="titleText">Delete Employee</h2>
    <br><br>

    <div class="reg-page">
        <form name="regform" id="regform" method="POST">
            <label for="eid">Employee ID</label><br>
            <input type="text" id="eid" name="eid" placeholder="P2008"
size="10" required pattern="^P[0-9]+$"
            title="Enter in the specified format">
            <br><br>

            <input type="submit" class="btn" name="Submit" id="Submit"
value="Delete"></input><br>
            <input type="reset" class="btn" name="Reset" id="Reset"
value="Clear All"></input>
        </form>
    </div>
{% endblock %}
```

Update.HTML

```
{% extends "base.html" %}

{% block head %}
<title>PMS - Update Record</title>
{% endblock %}

{% block body %}
<h1 class="titleText">Payroll Management System</h1>
<hr>
<h2 class="titleText">Update Employee Details</h2>
    <br><br>

    <div class="reg-page">
        <form name="regform" id="regform" method="POST">
            <label for="id">Employee ID</label><br>
            <input type="text" id="id" name="id" placeholder="P2008"
size="10" required pattern="^P[0-9]+$"
            title="Enter in the specified format">
            <br><br>

            <label for="phone">Phone Number</label><br>
            <input type="text" id="phone" name="phone"
placeholder="1234567890" size="20" required>
            <br><br>

            <input type="submit" class="btn" name="Submit" id="Submit"
value="Update"></input><br>
            <input type="reset" class="btn" name="Reset" id="Reset"
value="Clear All"></input>
        </form>
    </div>
{% endblock %}
```

View.HTML

```
{% extends "base.html" %}

{% block head %}
<title>Employee Details</title>
{% endblock %}

{% block body %}
<h1 class='titleText'>Employee Details</h1>
<hr><br>
{% if payslip == False %}
{% if employees|length < 1 %} <h4 class='titleText'>There are no employees
in the database.</h4>
{% else %}
{% if vertical == False %}
<table align='center'>
  <tr>
    <th>Employee ID</th>
    <th>Name</th>
    <th>Age</th>
    <th>Gender</th>
    <th>Designation</th>
    <th>Address</th>
    <th>Phone</th>
    <th>Date of Join</th>
  </tr>
  {% for emp in employees %}
  <tr>
    <td>{{emp.id}}</td>
    <td>{{emp.ename}}</td>
    <td>{{emp.age}}</td>
    <td>{{emp.gender}}</td>
    <td>{{emp.job}}</td>
    <td>{{emp.address}}</td>
    <td>{{emp.phone}}</td>
    <td>{{emp.doj.date()}}</td>
  </tr>
  {% endfor %}

```

```

</table>
{% else %}
<table align="center">
  <tr>
    <th>Employee ID</th>
    <td>{{employees[0].id}}</td>
  </tr>
  <tr>
    <th>Name</th>
    <td>{{employees[0].ename}}</td>
  </tr>
  <tr>
    <th>Age</th>
    <td>{{employees[0].age}}</td>
  </tr>
  <tr>
    <th>Gender</th>
    <td>{{employees[0].gender}}</td>
  </tr>
  <tr>
    <th>Designation</th>
    <td>{{employees[0].job}}</td>
  </tr>
  <tr>
    <th>Address</th>
    <td>{{employees[0].address}}</td>
  </tr>
  <tr>
    <th>Phone</th>
    <td>{{employees[0].phone}}</td>
  </tr>
  <tr>
    <th>Date of Join</th>
    <td>{{employees[0].doj.date()}}</td>
  </tr>
</table>
{% endif %}
{% endif %}
{% else %}
<table align="center">

```



```
|  |  |
| --- | --- |
| Basic Pay | {{employees[0].bpay}} |
| DA | {{employees[0].da}} |
| HRA | {{employees[0].hra}} |
| Provident Fund | {{employees[0].pf}} |
| Deductions | {{employees[0].dedn}} |
| Gross Pay | {{employees[0].gpay}} |


{%endif %}
{% endblock %}

```

Styles.CSS

```
@import
url("https://fonts.googleapis.com/css2?family=Sanchez&display=swap");

body {
    background: rgb(143, 161, 255);
    background: radial-gradient(
        circle,
        rgba(143, 161, 255, 1) 0%,
        rgba(255, 103, 241, 1) 100%
    );
    font-family: "Sanchez", serif;
}

.titleText,
.responseText {
    text-align: center;
}

.reg-page {
    width: 500px;
    position: relative;
    z-index: 1;
    background: #ffffff;
    max-width: 500px;
    margin: 0 auto 100px;
    padding: 45px;
    text-align: left;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0,
0.24);
}

.reg-page input,
.reg-page textarea,
.reg-page select {
    outline: 0;
    background: #f2f2f2;
    width: 100;
```

```
border: 0;
margin: 15px 0px 15px 40px;
padding: 15px;
box-sizing: border-box;
font-size: 16px;
font-family: "Sanchez", serif;
}
```

```
.reg-page label {
  font-weight: 650;
}
```

```
.reg-page textarea {
  resize: none;
}
```

```
.reg-page .btn {
  margin-left: 140px;
  font-family: "Sanchez", serif;
  font-size: 16px;
  font-weight: 300;
  text-transform: uppercase;
  outline: 0;
  background: #2a9df4;
  width: 200px;
  border: 0;
  padding: 15px;
  color: #ffffff;
  transition: all 0.3 ease;
  cursor: pointer;
  user-select: none;
}
```

```
.btn:hover,
.btn:focus,
.btn:active {
  font-weight: bold;
  background: #187bcd;
}
```

```
footer {  
    color: black;  
    font-weight: bold;  
    border-top: 3px solid black;  
    padding: 5px 5px 5px 5px;  
    text-align: center;  
    width: 100%;  
}
```

```
table,  
td,  
th {  
    font-size: 25px;  
    border: 2px solid black;  
    padding: 5px;  
    border-collapse: collapse;  
    text-align: center;  
}
```

```
table {  
    width: 75%;  
}
```

```
tr:hover {  
    background-color: azure;  
}
```

```
.links {  
    text-decoration: none;  
    cursor: pointer;  
    color: white;  
}
```

```
.links:hover {  
    font-weight: bold;  
}
```