SOFTWARE ENGINEERING MINI PROJECT - ||

REPORT

PAYROLL MANAGEMENT SYSTEM

Submitted by:

Name	MIS	Email	Batch
Kunal More	111903043	morekd19.comp@cope.ac.in	T2
Priyanshu Nandagawali	111903065	nandagawalipp19.comp@coep.ac.in	

Problem Statement:

Payroll Management System that uses paper sheets are inefficient and make it very easy for employees to cheat the system by entering incorrect data on the sheet. To avoid these issues, an automatic and flexible system should be implemented, of which the suggested system is one. The old system is outdated and no longer adequately manages the payroll process and the entry of employee timecard information. Therefore, manual intervention is required to process the payroll.

Objectives:

- The aim behind having a payroll management system is to automate and streamline micro tasks such that the HR team has time to focus on the macro tasks. You don't have to worry about handling, managing, and creating pay slips, salaries, and deductions of the employees.
- Efficient payroll systems save time and money by ensuring that paychecks go out on time in the correct amounts each pay period. Once the system is set up, many parts can be automated, to reduce errors and delays.
- The objective is to manage employees' salaries, deductions, other conveyance, net pay, generation of pay slips, etc.

- The macro-objective, which is related to sales, strategy, revenue, etc. Another is micro, which is associated with the daily tasks of the business.

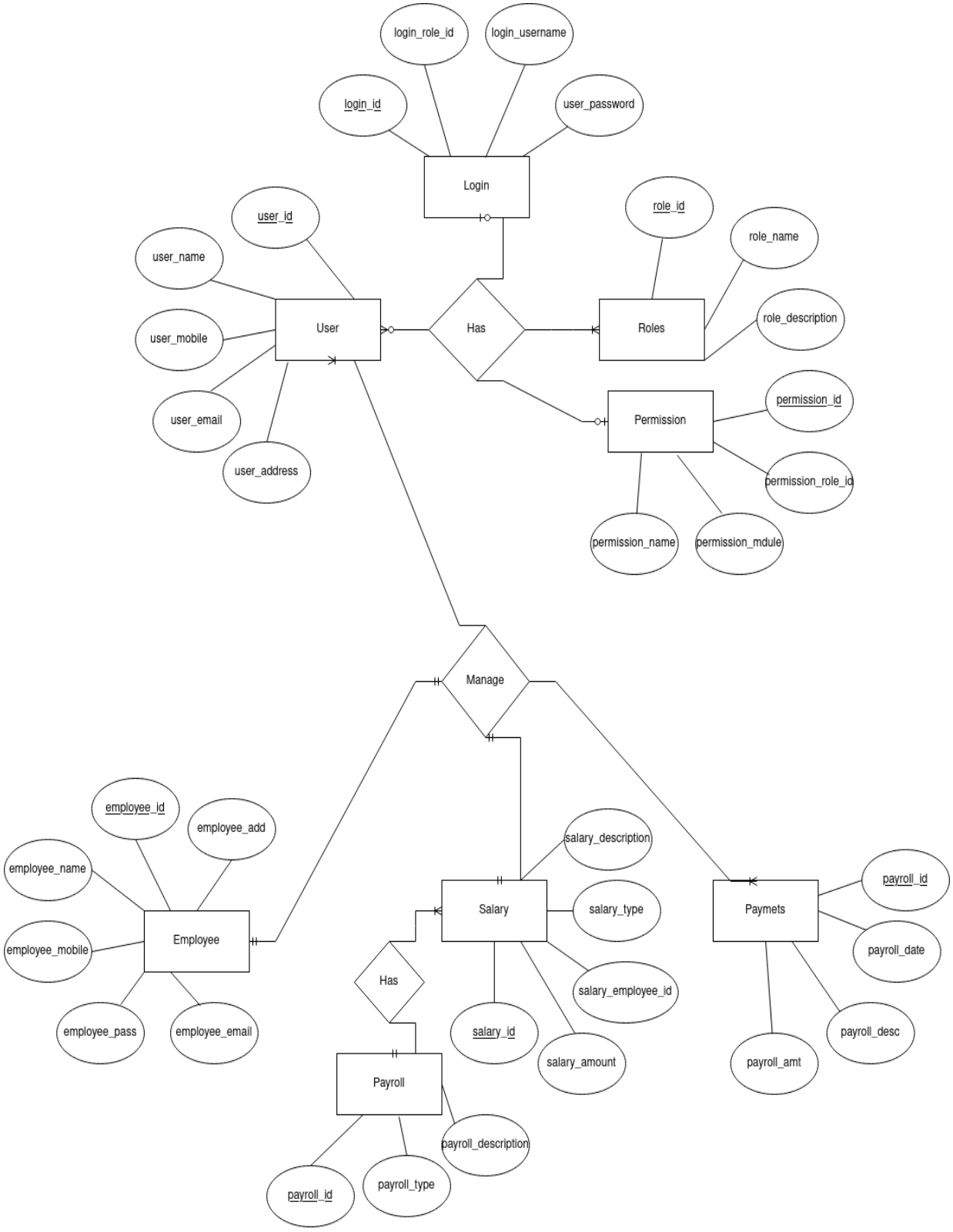
Motivation:

The project was engineered as a software engineering course project. Payroll is usually the most expensive part of a business. Employee attendance systems that use paper sheets are inefficient and make it very easy for employees to cheat the system by entering incorrect data on the sheet. To avoid these issues, an automatic and flexible system should be implemented, of which the suggested system is one. So, to solve this problem we are motivated to generate a payroll management system.

Summary of SRS:

- Main aim of developing Payroll Management System is to provide an easy way to automate all functionalities involved managing leaves and Payroll for the employees of Company.
- This Application works in Multiple PC's installed on multiple Computers but sharing same database by which users of different department can use it sitting at different locations simultaneously.
- Product Functions include – Master Module, Employee Module, Attendance Module, Salary Module.
- Operating environments include user/admin system, will work on Ubuntu/Windows OS, Sql database, using Django webserver, Python, HTML language.
- External Interface Requirements -- Frontend - Django webserver, HTML. Backend - Python, MySQL.
- System Features -- Establish your employee identification number (EID). Choose a payroll schedule. Determine each employee's deductions. Calculate net pay and pay the employees. Keep payroll records and make any necessary connections.
- Functional Requirements – Master, Employee, Search, Salary.
- Non-Functional Requirements – Performance Requirements, Safety Requirements, Security Requirements, Software Quality Attributes.

Entity Relationship Diagram:

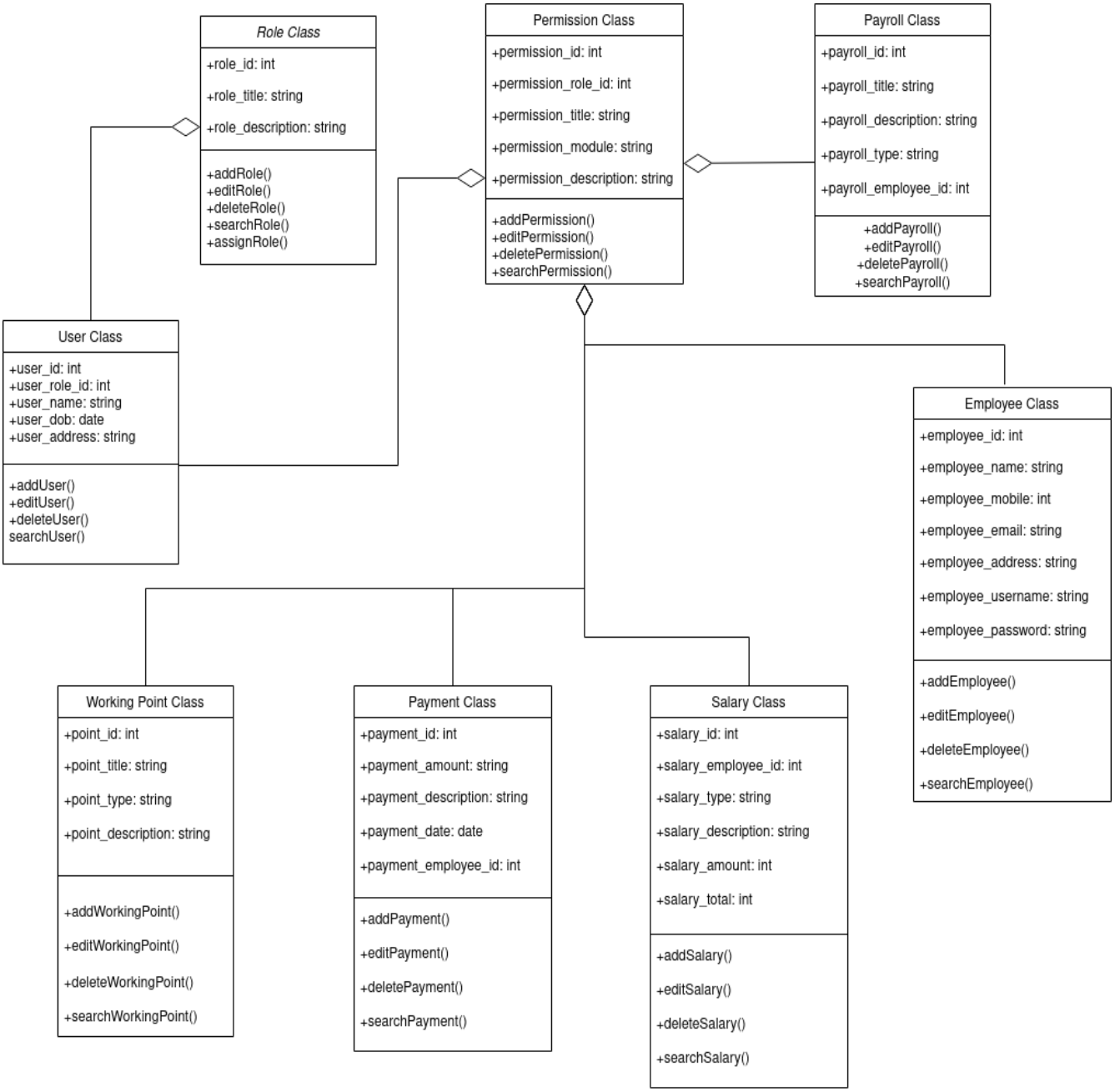


UML Diagram and Explanation:

1. Activity Diagram:

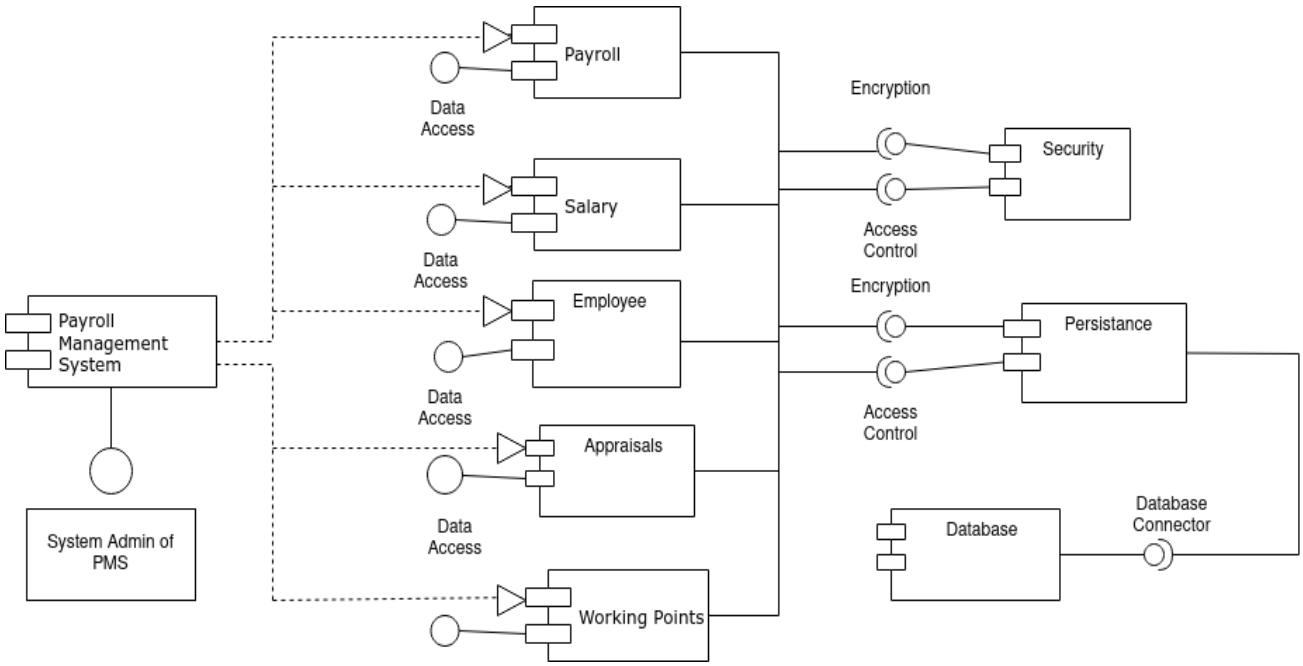
- Admin can search Salary, view description of selected Salary. Add, update and delete salary.
- Shows activity of editing, adding and updating payments.
- Show full description of flows of salary, employee and payments.

2. Class Diagram: describes the structure of PMS classes, attributes, operations and relationship among the objects. The main classes are shown in the diagram below.

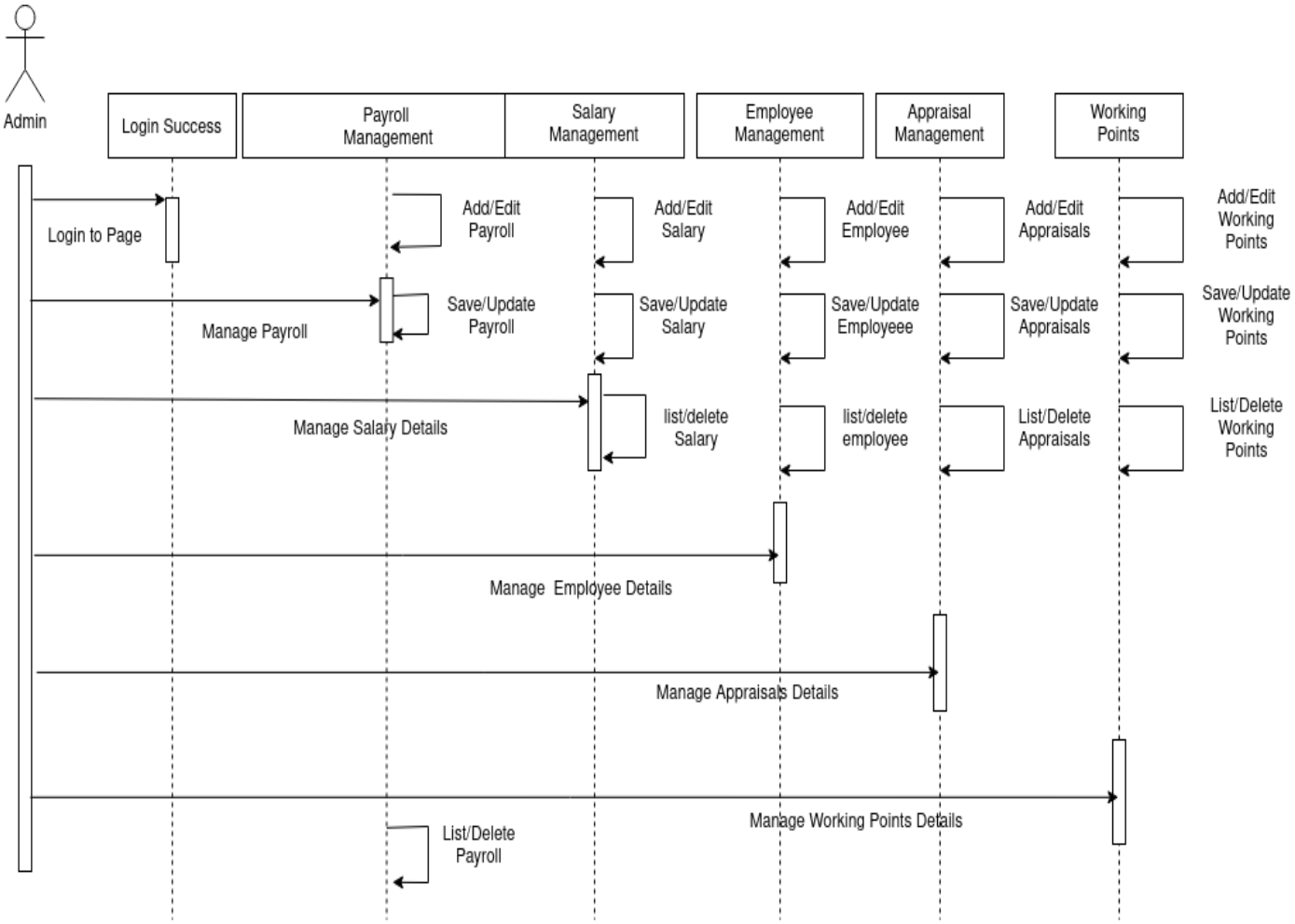


3. Component Diagram: It describes the wiring of the physical components in a system.

- Can show the models, the components of the PMS.
- Model the database schema of PMS.
- Model the executables of an application of an PMS.
- Model the system source code of PMS.

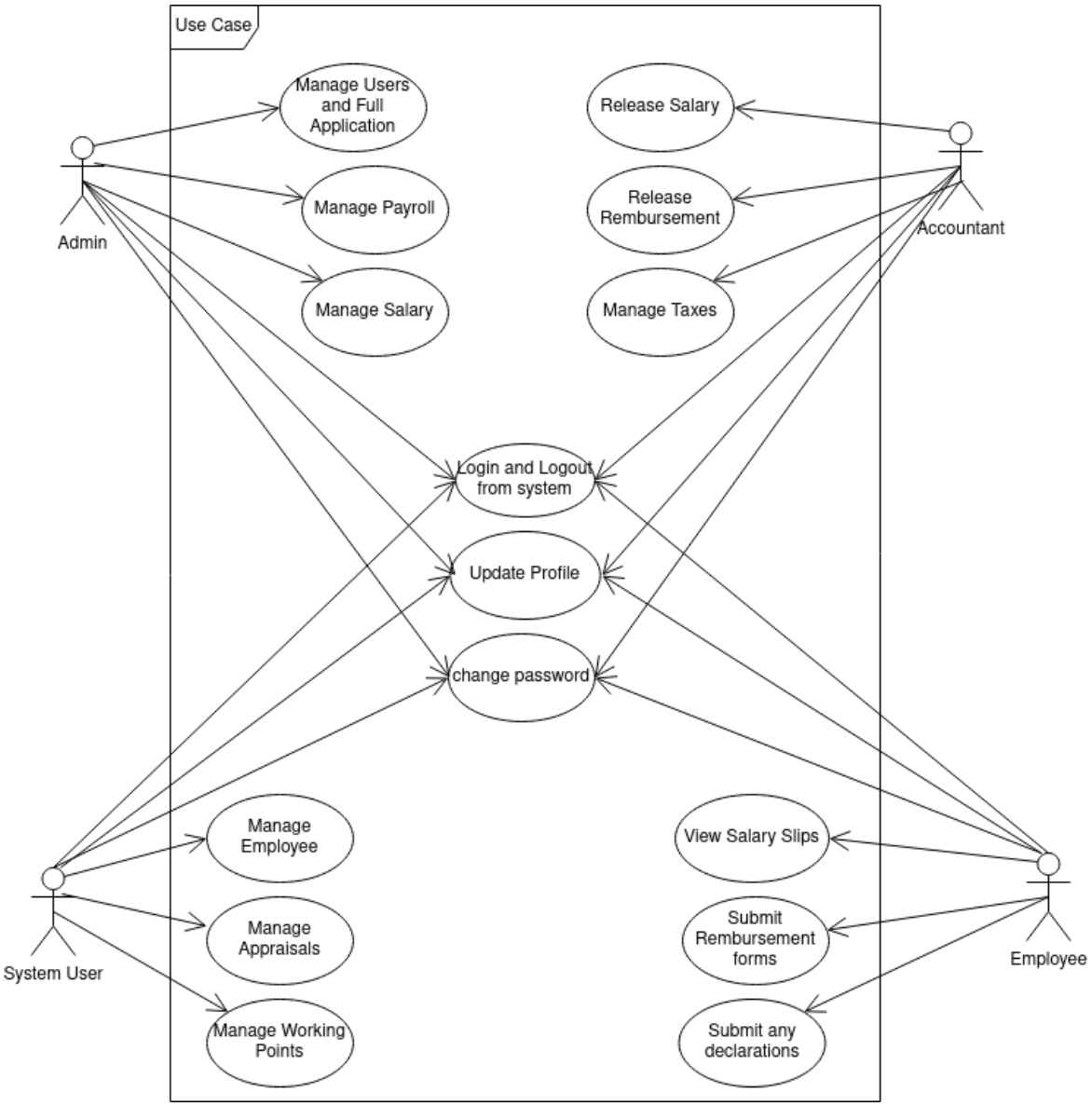


4. Sequence Diagram: It shows the interaction between the objects of Payroll, Salary, Employee, Appraisals, Payments. The instance of class objects is shown below.



5. Use Case Diagram: It is a graphic depiction of interaction among the elements of PMS.

- It represents methodology used in system analysis to identify, clarify and organize system requirements of PMS.
- The actors are Super Admin, Super User, Employee.
- Actors performed different types of use cases such as Manage Payroll, Manage Salary, Employee, Appraisals, Payments, Users.



Coding Screenshots with Result:

Code Snippet:

1. urls.py

urls.py M

payroll_management_sys > urls.py > ...

```python
"""payroll_management_sys URL Configuration
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from payroll_manager import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.index , name='index'),
    path('employee_login/', views.employee_login, name='employee_login'),
    path('employee_dashboard/<int:emp_id>', views.employee_dashboard, name='employee_dashboard'),
    path('employee_dashboard/<int:emp_id>/LeaveApply', views.leaveApply, name='leaveApply'),
    path('employee_dashboard/<int:emp_id>/addressChange', views.changeAddress, name='changeAddress'),
    path('employee_dashboard/<int:emp_id>/payChange', views.changePay, name='changePay'),
    path('employee_dashboard/<int:emp_id>/infoChange', views.changeInfo, name='changeInfo'),
    path('employee_dashboard/<int:emp_id>/achievementChange', views.changeAchievement, name='changeAchievement'),
    path('approval/<int:leave_id>/<int:app_id>/', views.approval, name='approval'),
    path('admin_dashboard/', views.admin_dashboard, name='admin_dashboard'),
    path('admin_dashboard/<int:emp_id>/', views.admin_employee_dashboard, name='admin_employee_dashboard'),
    path('admin_login/', views.admin_login, name='admin_login'),
    path('register/', views.register, name='register'),
    path('logout/', views.logoutUser, name='logout'),
    path('deleteAll', views.deleteAll, name="deleteAll"),
]

```

2. views.py

views.py

payroll_manager > views.py > ...

```python
from django.shortcuts import render, redirect
from .forms import *
from django.contrib.auth import authenticate,login,logout
from datetime import datetime
from .models import *
import datetime
from django.contrib import messages
# Create your views here.
def index(request):
    return render(request,'payroll_manager/index.html')
def employee_dashboard(request,emp_id):
    user_info=user_paygrade=user_pay=user_achieve=user_leave=None
    user = Account.objects.get(user_id = emp_id)
    if user == request.user:
        if MEmployee.objects.filter(employee= user).exists():
            user_info = MEmployee.objects.get(employee= user)
        if MPaygrade.objects.filter(employee= user_info).exists():
            user_paygrade = MPaygrade.objects.filter(employee=user_info).first()
        if MPay.objects.filter(employee= user_info).exists():
            user_pay = MPay.objects.filter(employee=user_info).first()
        if TAchievement.objects.filter(employee= user_info).exists():
            user_achieve = TAchievement.objects.filter(employee=user_info)
        if TLeave.objects.filter(employee= user_info).exists():
            user_leave = TLeave.objects.filter(employee=user_info)
        return render(request,'payroll_manager/employee_dashboard.html', context={'user_info':user_info,'user_paygrade':user_paygrade,'us
    else:
        messages.info(request, 'You Are Not Authorized To Access That Page')
        return redirect('index')
def employee_login(request):
    if request.method == 'POST':
        Acc = Account()
        user_id = request.POST.get('user_id')
        password = request.POST.get('password')
        user = authenticate(username = user_id , password = password )
        if user is not None:
            if Account.objects.filter(user_id=user_id, is_employee=True).exists():
                login(request,user)
```

views.py

payroll_manager > views.py > ...

```
                return redirect('employee_dashboard',emp_id=user_id)
            else:
                messages.info(request, 'Invalid, user not An Employee.')
                form = EmployeeLogin()
                return render(request,'payroll_manager/employee_login.html',context={'form':form})
        else:
            messages.info(request, 'Invalid Credentials.')
            form = EmployeeLogin()
            return render(request,'payroll_manager/employee_login.html',context={'form':form})
    form = EmployeeLogin()
    return render(request,'payroll_manager/employee_login.html',context={'form':form})

def admin_dashboard(request):
    if Account.objects.filter(user_id= request.user.user_id, is_employer=True):
        allEmp = MEmployee.objects.all()
        LeaveRequests = TLeave.objects.filter(is_approved=0)
        return render(request,'payroll_manager/admin_dashboard.html', context={'allEmp':allEmp, 'LeaveR':LeaveRequests})
    else:
        messages.info(request, 'You Are Not Authorized To Access That Page')
        return redirect('index')
def admin_login(request):
    if request.method == 'POST':
        Acc = Account()
        user_id = request.POST.get('user_id')
        password = request.POST.get('password')
        user = authenticate(username = user_id , password = password )
        if user is not None:
            if Account.objects.filter(user_id=user_id, is_employer=True).exists():
                login(request,user)
                return redirect('admin_dashboard')
            else:
                messages.info(request, 'Invalid, user not An Admin.')
                form = EmployeeLogin()
                return render(request,'payroll_manager/admin_login.html',context={'form':form})
        else:
            messages.info(request, 'Invalid Credentials.')
            form = EmployeeLogin()
```

views.py

payroll_manager > views.py > ...

```
75                  return render(request,'payroll_manager/admin_login.html',context={'form':form})
76          form = EmployeeLogin()
77          return render(request,'payroll_manager/admin_login.html',context={'form':form})
78      def register(request):
79          if request.method == 'POST':
80              user = Account()
81              if request.POST.get('password1') == request.POST.get('password2'):
82                  user.user_id = request.POST.get('user_id')
83                  user.set_password(request.POST.get('password1'))
84                  user.is_employee=True
85                  user.is_employer=False
86                  user.date_joined=datetime.datetime.now()
87                  user.save()
88                  add = MEmployee()
89
90                  add.employee = user
91                  add.employee_name = request.POST.get('employee_name')
92                  add.employee_doj = request.POST.get('employee_doj')
93                  add.department = MDepartment.objects.get(department_id=request.POST.get('department'))
94                  add.company = MCompany.objects.get(company_id=request.POST.get('company'))
95                  add.grade = MGrade.objects.get(grade_id=request.POST.get('grade'))
96                  add.save()
97                  return redirect('admin_dashboard')
98          form = RegisterEmployeeForm()
99          formSub = employeeInfoForm()
100         return render(request,'payroll_manager/register.html',context={'form':form,'formSub':formSub})
101
102     def logoutUser(request):
103         logout(request)
104         return redirect('index')
105     def deleteAll(request):
106         Account.objects.all().delete()
107         return redirect('index')
108     def leaveApply(request,emp_id):
109         user = Account.objects.get(user_id = emp_id)
110         if user == request.user:
111             if request.method == 'POST':
```

views.py

payroll_manager > views.py > ...

```
112                 leaveApp = TLeave()
113                 leaveApp.employee = MEmployee.objects.get(employee=user)
114                 leaveApp.fin_year= int(datetime.datetime.now().year)
115                 leaveApp.leave_date = request.POST.get('leave_date')
116                 leaveApp.leave_type=request.POST.get('leave_type')
117                 leaveApp.save()
118                 messages.success(request, 'Leave Application Submitted.')
119                 return redirect('employee_dashboard', emp_id=emp_id)
120             leaveForm = leaveApplyForm()
121             return render(request,'payroll_manager/leaveApply.html',context={'form':leaveForm})
122     def changeAddress(request,emp_id):
123         user = Account.objects.get(user_id = emp_id)
124         if True:
125             if request.method == 'POST':
126                 if MAddress.objects.filter(memployee=MEmployee.objects.get(employee=user)).exists():
127                     add = MAddress.objects.filter(memployee=MEmployee.objects.get(employee=user)).first()
128                 else:
129                     add = MAddress()
130                     add.employee = MEmployee.objects.get(employee=user)
131                 add.building_details = request.POST.get('building_details')
132                 add.road = request.POST.get('road')
133                 add.landmark = request.POST.get('landmark')
134                 add.city = request.POST.get('city')
135                 add.state = MState.objects.get(state_code=request.POST.get('state'))
136                 add.country = request.POST.get('country')
137                 add.save()
138                 messages.success(request, 'Address Details Updated.')
139                 if request.user.is_employer :
140                     return redirect('admin_dashboard')
141                 else:
142                     return redirect('employee_dashboard', emp_id=emp_id)
143             oldData = MAddress.objects.filter(memployee=MEmployee.objects.get(employee=user)).first()
144             AddForm = addressForm(instance=oldData)
145             return render(request,'payroll_manager/addressChange.html',context={'form':AddForm})
146     def admin_employee_dashboard(request,emp_id):
147         user_info=user_paygrade=user_pay=user_achieve=user_leave=None
148         user = Account.objects.get(user_id = emp_id)
```

3. forms.py

forms.py

payroll_manager > forms.py > ...

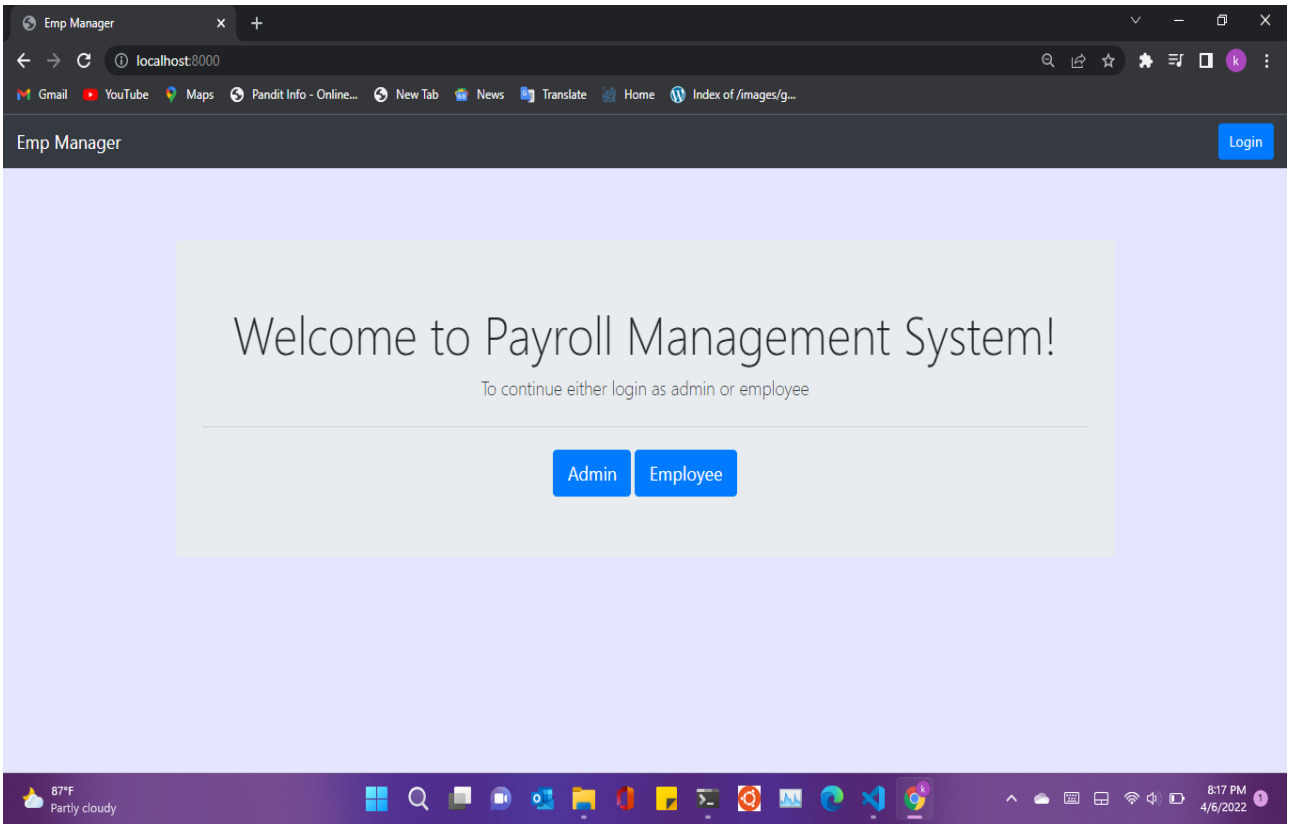
```python
from django import forms
from .models import *
from django.contrib.auth.forms import UserCreationForm
from django.forms import ModelForm

class RegisterEmployeeForm(UserCreationForm):

    class Meta(UserCreationForm.Meta):
        model = Account
        fields=['user_id',]
    def __init__(self, *args, **kwargs):
        super(RegisterEmployeeForm, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'

class EmployeeLogin(forms.Form):
    user_id = forms.IntegerField(required=True)
    password = forms.CharField(widget=forms.PasswordInput)
    def __init__(self, *args, **kwargs):
        super(EmployeeLogin, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'
class DateInput(forms.DateInput):
    input_type = 'date'
class leaveApplyForm(ModelForm):
    class Meta:
        model = TLeave
        fields = ['leave_type','leave_date']
        widgets = {
            'leave_date': DateInput(attrs={'type': 'date'})
        }
    def __init__(self, *args, **kwargs):
        super(leaveApplyForm, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'
class addressForm(ModelForm):
    class Meta:
```

forms.py

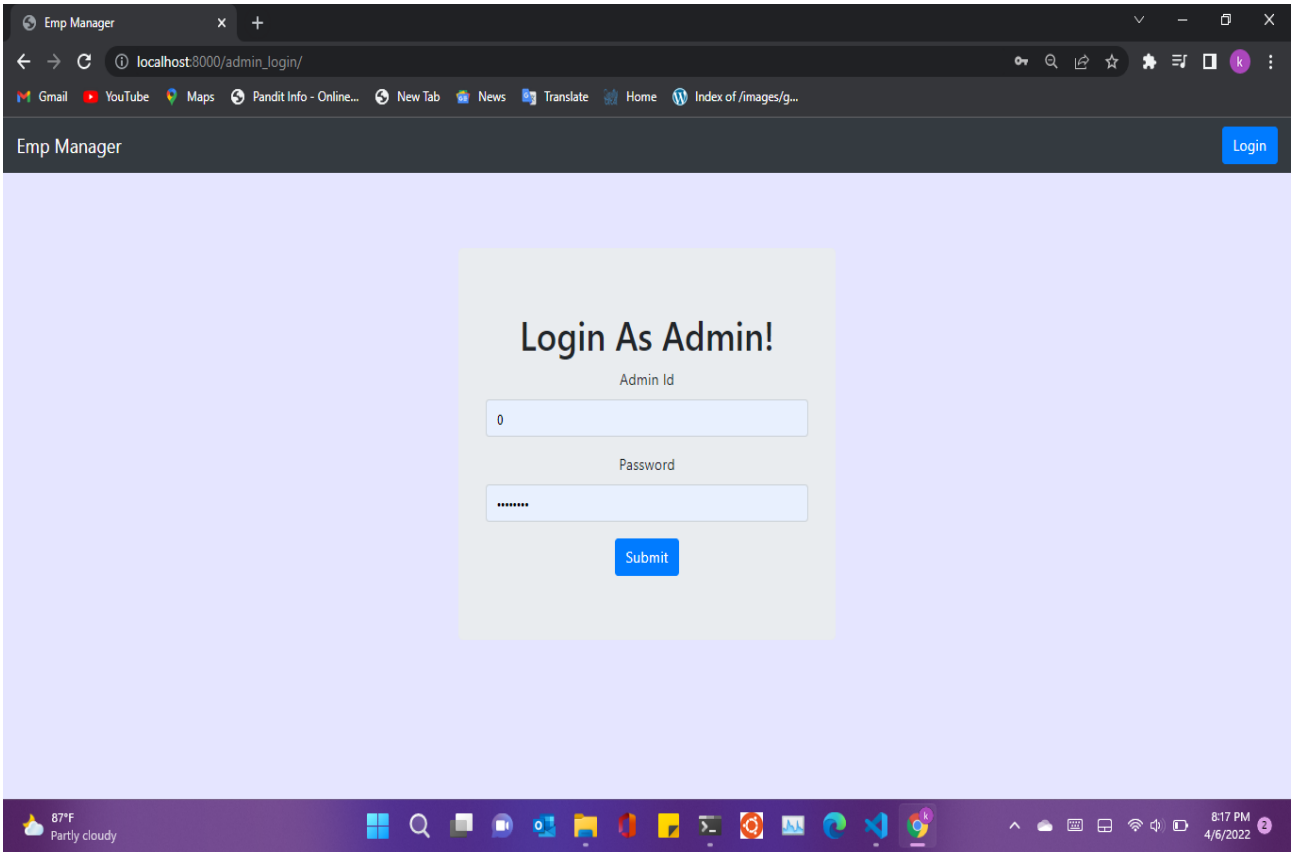
payroll_manager > forms.py > ...

```python
    class Meta:
        model = MAddress
        fields = ['building_details','road','landmark','city','state','country']
    def __init__(self, *args, **kwargs):
        super(addressForm, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'
class paygradeForm(ModelForm):
    class Meta:
        model = MPaygrade
        fields = ['basic_amt','da_amt','pf_amt','medical_amt']
    def __init__(self, *args, **kwargs):
        super(paygradeForm, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'
class payForm(ModelForm):
    class Meta:
        model = MPay
        fields = ['fin_year','gross_salary','gross_dedn','net_salary']
    def __init__(self, *args, **kwargs):
        super(payForm, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'
class employeeInfoForm(ModelForm):
    class Meta:
        model = MEmployee
        fields= ['employee_name', 'department', 'company', 'employee_doj', 'grade']
        widgets = {
            'employee_doj': DateInput(attrs={'type': 'date'})
        }
    def __init__(self, *args, **kwargs):
        super(employeeInfoForm, self).__init__(*args, **kwargs)
        for visible in self.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'
class AchievementForm(ModelForm):
    class Meta:
        model =TAchievement
```

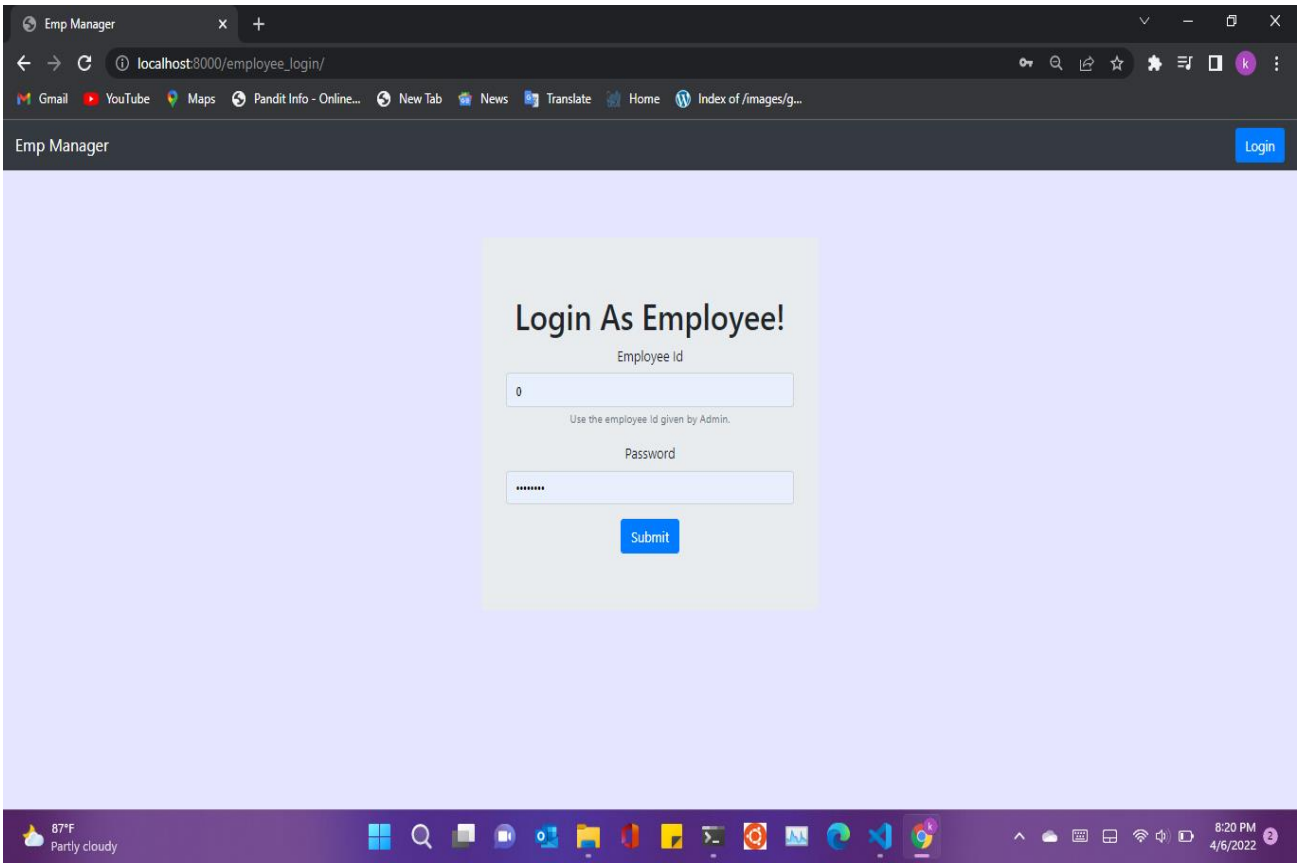
Output Screenshots:


Emp Manager
localhost:8000
Gmail
YouTube
Maps
Pandit Info - Online...
New Tab
News
Translate
Home
Index of /images/g...
Emp Manager
Login
Welcome to Payroll Management System!
To continue either login as admin or employee
Admin
Employee
87°F
Partly cloudy
8:17 PM
4/6/2022

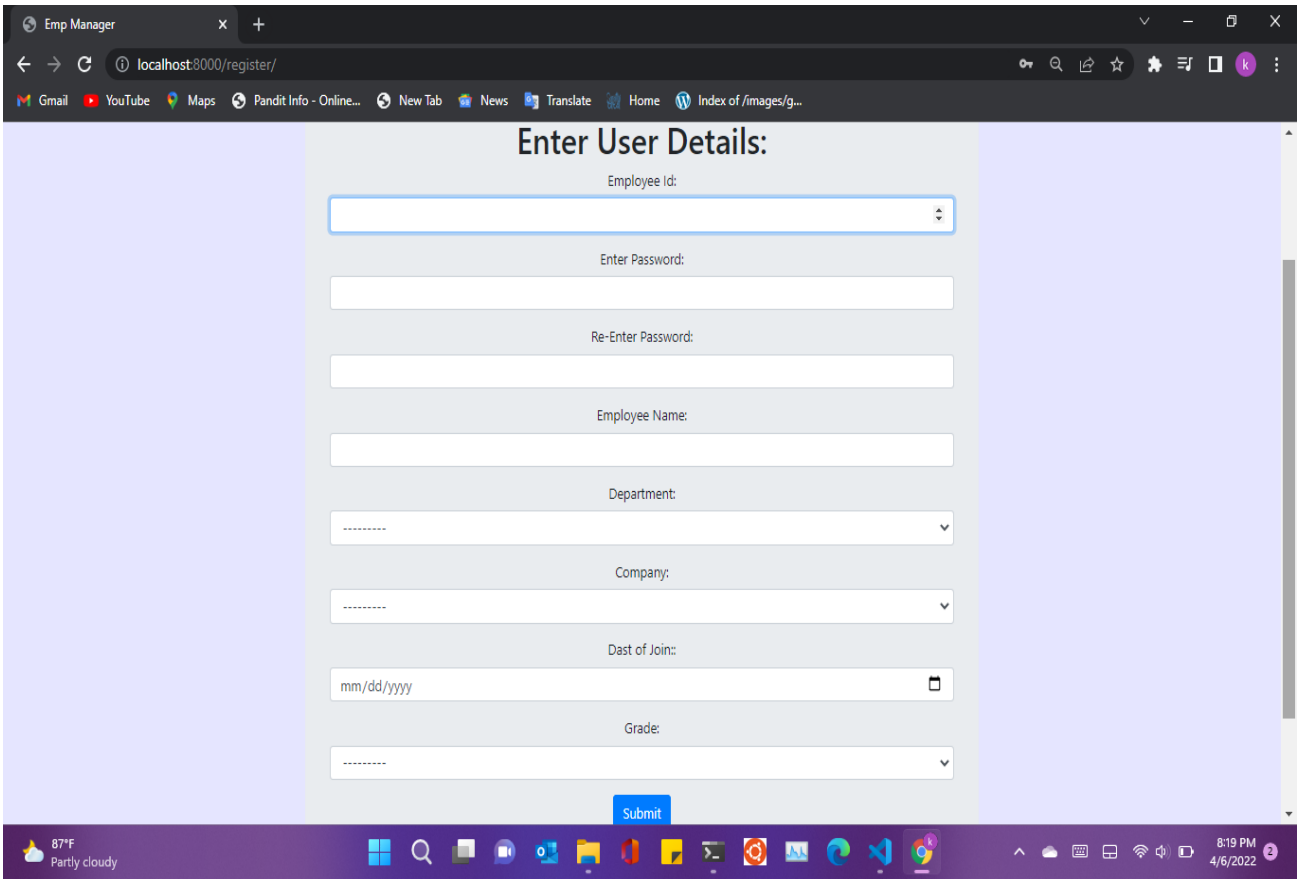
Home Page


Emp Manager
localhost:8000/admin_login/
Gmail
YouTube
Maps
Pandit Info - Online...
New Tab
News
Translate
Home
Index of /images/g...
Emp Manager
Login
Login As Admin!
Admin Id
0
Password
Submit
87°F
Partly cloudy
8:17 PM
4/6/2022

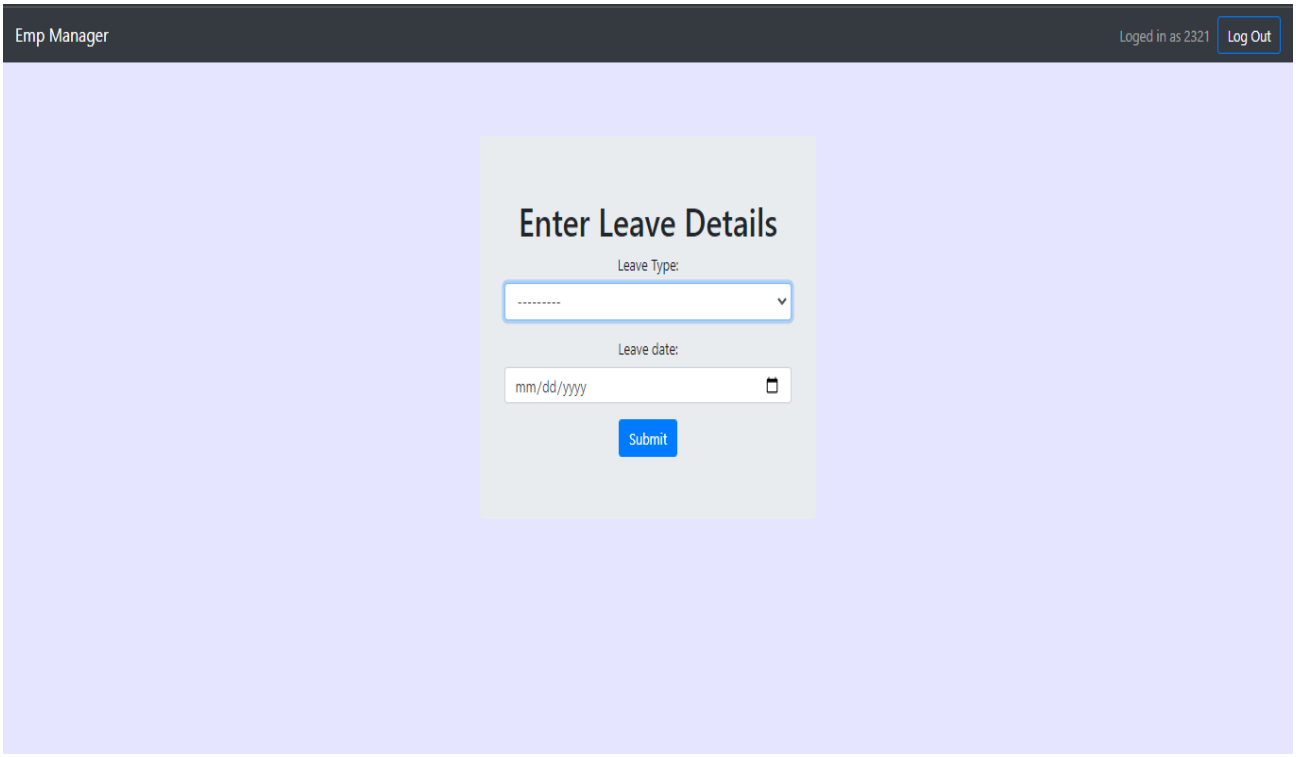
Admin Login



employee login



enter employee details



Employee's Leave Details

Testing:

Admin Module:

Test Case	Input	Process	Output
Login	ID, Password	If administrator enters ID and password correct it goes to the admin services otherwise displays the same page with an error message.	Displays the admin page.
Add new employee	Name, Id, designation, Date of joining.	A new employee can be added to the system and admin can update his details.	Employee added successfully.
Salary Details	Id, designation, basic salary, PF, DA, HRA.	The admin can update the salary details.	Employee salary details will be updated to the database.
Leave Details	Id, Type of leave, Number of Leaves.	Admin updates the leave details.	Employee Leave details will be updated to the database.

Employee Module:

Test Case	Input	Process	Output
Login	ID, Password.	If Employee enters ID and password correct it goes to the other page otherwise displays the same page with an error message.	Displays the Information to be viewed by an employee
Update Profile	Id, Name, Designation, Email-id, Mobile number, Address, qualification	The employee can update his profile if any modifications occur in his details	The details of the employee can be updated.
Change Password	Id, Old password, new password	Employee can be able to change his password	Employee's new password will be updated
View Deductions	ID, Month, Year.	The deductions can be known	The total deductions of an employee for the specified month and year can be viewed

Conclusion:

The Payroll Management System product developed for a company has been designed to achieve maximum efficiency and reduce the time taken to handle the payroll activity. It is designed to replace an existing manual record system, thereby reducing time taken for calculations and for storing data. The system uses Django, HTML as front end and Python, SQL as a backend for the database.

Future Scope:

This project has many future applications like it can be used in any of the Retail Outlet of Any Type companies. This project was built keeping in mind all the requirements of these outlets and they can be implemented in any such type of

organization with very few modifications. It can be further developed to include more operations and analysis, as changes are required in the system to adapt to the external developments. Further enhancements can be made to the system at any later point in time.

GitHub Link:

https://github.com/Kunal2300/Payroll_Management_System

THANK YOU!