PROJECT REPORT

ON

"EMPLOYEE MANAGEMENT SYSTEM"

IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE DEGREE OF

BACHELOR OF COMPUTER APPLICATIONS (BCA)

SUBMITTED TO: - MRS. SUNANDA LAL

GGN INSTITUTE OF MANAGEMENT AND TECHNOLOGY,

CIVIL LINES, LUDHIANA

SUBMITTED BY: -

Name: HARSH SOOD

Roll No. – 1823104

Batch: 2018-21

IK GUJRAL PUNJAB TECHNICAL UNIVERSITY (PTU)

JALANDHAR

April 2021

DECLARATION

I **Harsh Sood** being a student of BCA 6th semester of **GUJRANWALA GURU NANAK INSTITUTE OF MANAGEMENT AND TECHNOLOGY, LUDHIANA** hereby declare that all the work presented in the Major Project Report titled "**EMPLOYEE MANAGEMENT SYSTEM**" is an authentic record of my work carried under the guidance of Mrs. Sunanda Lal. It is my own work and original piece of work. I have not submitted it earlier elsewhere.

HARSH SOOD

CERTIFICATE

This is to certify that project titled "**EMPLOYEE MANAGEMENT SYSTEM**" submitted by **Harsh Sood (1823104)** at GGNIMT, Civil Lines, Ludhiana affiliated to I.K.G Punjab Technical University, Jalandhar in partial fulfilment of requirement for the degree of BCA has been approved.

Date – 26/6/2021

(Signature of Project Guide)

ACKNOWLEDGEMENT

I **Harsh Sood** extend my sincere thanks to all those people who have been giving me any kind of assistance in the making and completion of this project report.

I express my gratitude to my faculty guide **Mrs. Sunanda Lal**, who through her vast experience and knowledge has been able to guide me at every step both ably and successfully towards the completion of this major project. This project report would not have been successful without her help and continuous guidance throughout. I express my gratitude to **Gujranwala Guru Nanak Institute of Management and Technology, Civil Lines, Ludhiana**.

I would hereby, make most of the opportunity by expressing my sincerest thanks to all my faculties whose teachings gave me conceptual understanding and clarity of comprehension which ultimately made my job easier. Credit also goes to all my friends and people from various fields for giving me their precious time whose encouragement kept me in good stead. Their continuous support has given me the strength and confidence to complete my project without any difficulty.

HARSH SOOD

INTRODUCTION

Today in the times of the modern era when everything seems to be going online due to the rising demand of the population towards the advancement in the technology there has been a huge boom in the IT sector for fulfilling the requirements of the general population and make everything automatized in the real world. This automatisation in turn has made the lives of the human beings a bit easier as all they can now find on all the solutions to their work related problems by just browsing through their electronic devices.

Since this is the modern time, and everything is being digitalised making up to a reduction in the manual works which has affected the general public and most commonly in all the forms of organisations such as: - the banks, the offices, the railway stations, the colleges, the schools, the malls, the shops, etc., every place we can think of there is a usage of internet or digitisation in one or the other way.

So here I have tried to design a software named **EMPLOYEE MANAGEMENT SYSTEM** which relates to the organisation in one or the other ways. Since there are a huge number of the Multi-National Companies working on the larger scales so in order to reduce the burden of the manpower working therein there can be a use of such a system which would help in the recording the data in a systematic manner, help make the interaction between the employee and the admin staff directly without any of the intermediator in between, store the data and can refer to the same data in the future.

Online Employee Management Software in simple words is managing the records of an organization. As the organizations often require an expert who can plan, organize, analyse, interpret and make decisions regarding the strategies for the effective working of the organization. Moreover, the HR specialist usually always performs these roles. However, the industry and market are beginning to witness a gradual change from manual record management to digital record management system adoption.

The "Employee Management System" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for managing the particular need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly. Employee Management System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources.

Every organization, whether big or small, has challenges to overcome and manage the information of Leaves, Employee’s Personal Details, Payroll Management, Attendance Record, Regularisation of Attendance, etc. Every Employee has different needs in terms of the requirement of the data in the organisation at a particular point of time; therefore exclusive employee management systems are designed that are adapted to fulfil all managerial requirements of the organisation. This is designed to assist in strategic planning, and will help you ensure that the organization is equipped with the right level of information and details for achieving the future goals. Also, for those busy executives who are always on the go, such systems can help them with remote access features inbuilt in the system, which will allow the user to manage the workforce anytime, at all times. These systems will ultimately allow the users to better manage resources.

The purpose of Employee Management System is to automate the existing manual system by the help of computerized tools and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Employee Management System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

PROPOSED SOLUTIONS

The technology used to create this system is **Python (for Front End)** and **SQL (for Back End Database Connectivity)**.

TOOLS/PLATFORM, HARDWARE AND SOFTWARE REQUIREMENT SPECIFICATIONS:

Following are the hardware and software requirements that are required for the system to work in a good and appropriate manner.

SOFTWARE REQUIREMENTS:

Name of component	**Specification**
Operating System	Windows
Language	Python Runtime Environment
Database	SQL
Browser	Any of Mozilla, Opera, Chrome, etc.
Web Server	Python
Software Development Kit	Python
Scripting Language Enable	Python
Database JDBC Driver	DB Browser

HARDWARE REQUIREMENTS:

Name of component	Specification
Processor	Intel Core
RAM	2GB
Hard disk	500GB
Monitor	On any screen
Keyboard	No specific requirement

INTRODUCTION TO THE TECHNOLOGIES USED

Introduction to Python: -

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).Following are some of the important features of the **Python programming** language.

Features of Python: -

1. Easy to Code

Python is a very developer-friendly language which means that anyone and everyone can learn to code it in a couple of hours or days. As compared to other object-oriented programming languages like Java, C, C++, and C#, Python is one of the easiest to learn.

2. Open Source and Free

Python is an open-source programming language which means that anyone can create and contribute to its development. Python has an online forum where thousands of coders gather daily to improve this language further. Along with this Python is free to download and use in any operating system, be it Windows, Mac or Linux.

3. Support for GUI

GUI or Graphical User Interface is one of the key aspects of any programming language because it has the ability to add flair to code and make the results more visual. Python has support for a wide array of GUIs which can easily be imported to the interpreter, thus making this one of the most favourite languages for developers.

4. Object-Oriented Approach

One of the key aspects of Python is its object-oriented approach. This basically means that Python recognizes the concept of class and object encapsulation thus allowing programs to be efficient in the long run.

5. High-Level Language

Python has been designed to be a high-level programming language, which means that when you code in Python you don’t need to be aware of the coding structure, architecture as well as memory management.

6. Integrated by Nature

Python is an integrated language by nature. This means that the python interpreter executes codes one line at a time. Unlike other object-oriented programming languages, we don’t need to compile Python code thus making the debugging process much easier and efficient. Another advantage of this is, that upon execution the Python code is immediately converted into an intermediate form also known as byte-code which makes it easier to execute and also saves runtime in the long run.

7. Highly Portable

Suppose you are running Python on Windows and you need to shift the same to either a Mac or a Linux system, then you can easily achieve the same in Python without having to worry about changing the code. This is not possible in other programming languages, thus making Python one of the most portable languages available in the industry.

8. Highly Dynamic

As mentioned in an earlier paragraph, Python is one of the most dynamic languages available in the industry today. What this basically means is that the type of a variable is decided at the run time and not in advance. Due to the presence of this feature, we do not need to specify the type of the variable during coding, thus saving time and increasing efficiency.

9. Extensive Array of Library

Out of the box, Python comes inbuilt with a large number of libraries that can be imported at any instance and be used in a specific program. The presence of libraries also makes sure that you don’t need to write all the code yourself and can import the same from those that already exist in the libraries.

Introduction to SQL: -

SQL is a database computer language designed for the retrieval and management of data in a relational database. **SQL** stands for **Structured Query Language**.

Features of SQL: -

1. High Performance

SQL provide high performance programming capability for highly transactional, heavy workload and high usage database system. SQL programming gives various ways to describe the data more analytically.

2. High Availability

SQL is compatible with databases like MS Access, Microsoft SQL Server, MySQL, Oracle Database, SAP HANA, SAP Adaptive Server, etc. All of these relational database management systems support SQL and it is easy to create an application extension for procedural programming and various other functions which is additional features thus converting SQL into a powerful tool.

3. Scalability and Flexibility

SQL provide Scalability and Flexibility. It is very easy to create new tables and previously created or not used tables can be dropped or deleted in a database.

4. Robust Transactional Support

With SQL programming can handle large records and manage numerous transactions.

5. High Security

It is very easy to provide permissions on tables, procedures, and views hence SQL give security to your data.

6. Comprehensive Application Development

SQL is used by many programmers to program apps to access a database. No matter what is the size of organization, SQL works for every small or large organization.

7. Management Ease

SQL is used in almost every relational database management system. “Select“, “Create”, “Insert”, “Drop”, “Update”, and “Delete” are the standard and common SQL commands that helps us to manage large amount of data from a database very quickly and efficiently.

8. Open Source

SQL is an open-source programming language for building relational database management system

Python - The most important benefits of using this programming language: -

1. Versatile, Easy to Use and Fast to Develop

Python focuses on code readability. The language is versatile, neat, easy to use and learn, readable, and well-structured.

2. Open Source with a Vibrant Community

You can download Python for free and writing code in a matter of minutes. Developing with Python is hassle-free.

What's more, the Python programmer's community is one of the best in the world - it's very large and active. Some of the best IT minds in the world are contributing to both the language itself and its support forums.

3. Has All the Libraries You Can Imagine

You can find a library for basically anything you could imagine: from web development, through game development, to machine learning.

4. Great for Prototypes - You Can Do More with Less Code

As it was mentioned before, Python is easy to learn and fast to develop with. You can do more with less code, which means you can build prototypes and test out ideas much quicker in Python than in other languages. This means that using Python not only to saves a lot of time, but also reduce your company's costs.

REVIEW OF THE PROJECT

The project ‘Employee Management System’ is the one which is designed for basically the large Multinational Companies where there is a huge amount of data that is to be recorded, updated and transferred on the daily basis.

The ‘Employee Management System’ is both an Employee and an Administrative based system i.e., it is a system which is designed in a manner that anyone who is authorised to the system from that particular organisation can have an access to the system and it’s working.

The important features of the system are as follows: -

- The employee can register into the system and access his panel where he/she can find his/her own personal information, mark attendance, apply for the leaves, apply for the short leaves, view the status for the approval and denial of the leaves, view the salary slip, get the access to fill the form of attendance regularisation, etc.
- The admin can on the other hand access his/her panel too where they can view the detail of all the employee’s who are working in the organisation, view the attendance for the particular day/date, view the salary status, view the list of leaves that have been applied by the employees whose status needs to be updated, approve the leaves, approve the regularisation requests, etc.

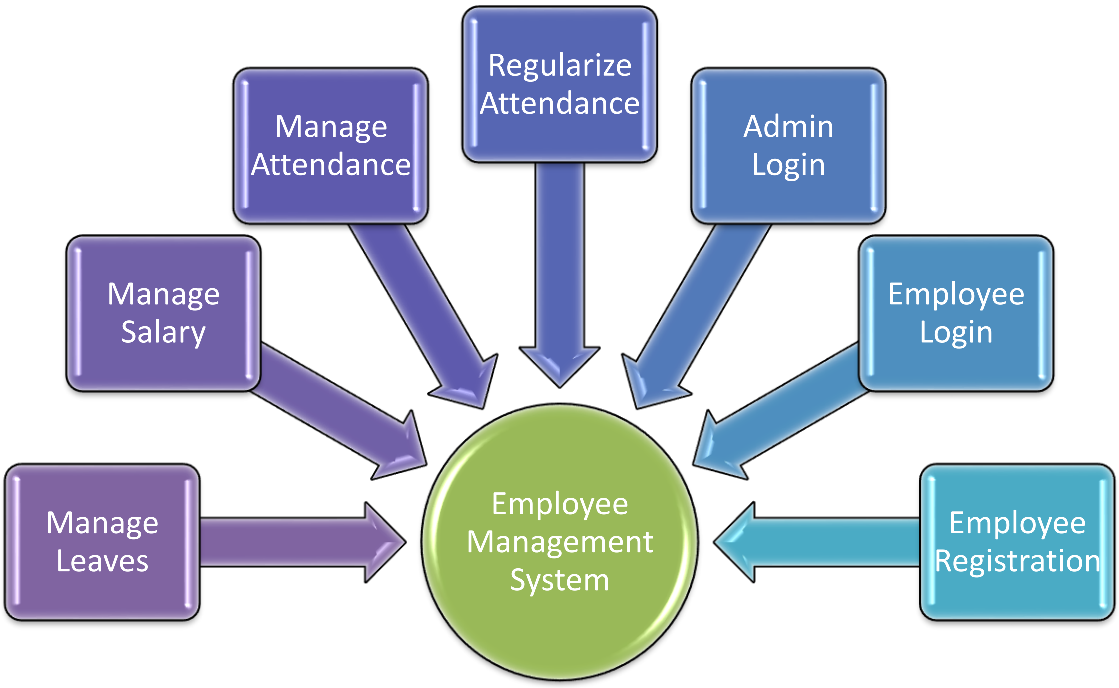
OBJECTIVES OF THE PROJECT

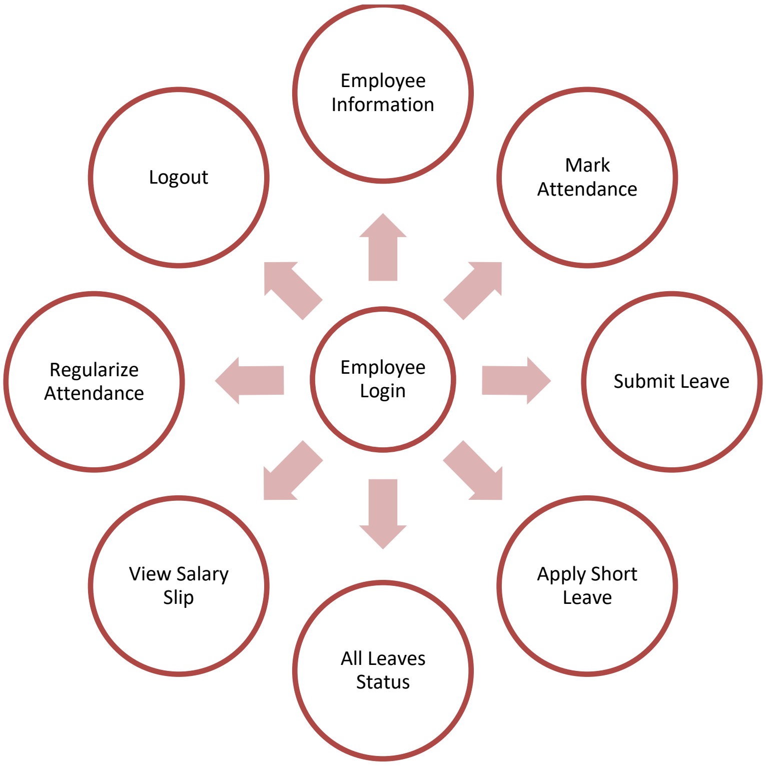
The 'Employee Management System' consists of the four different modules which have their separate working functionality, which is as follows: -

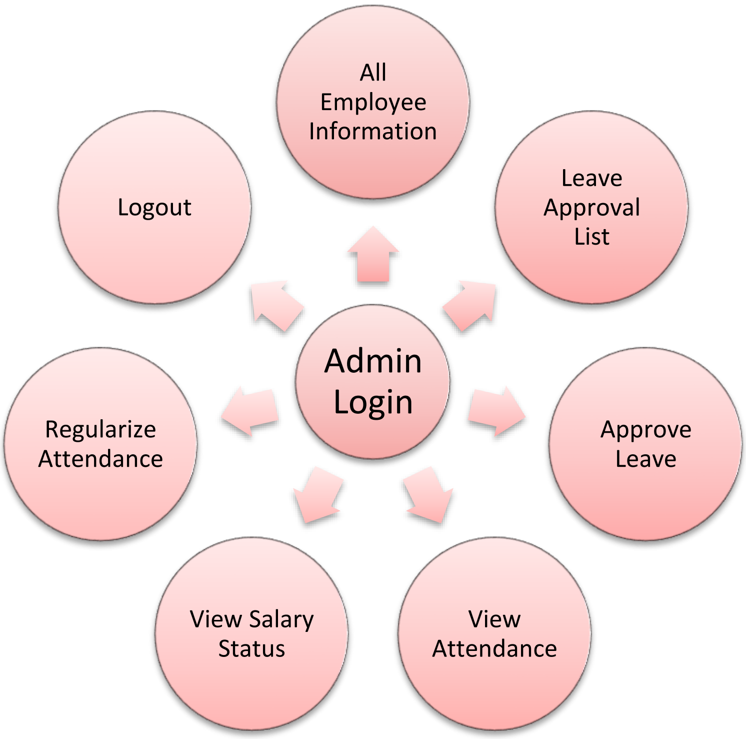
- Employee Registration – This is a form which is required to be filled by the employee in order to register himself/herself and generate his/her Employee ID and Password which can help them to get the access to their panel in the system.
- Manage Leaves – This module can be accessed by both the workforce employees and the admin staff as in this they both can have the access to their panels and therein operate in the system as per their requirements. On the employee's side the respective employee can apply his/her leaves, short leaves, attendance regularisation requests, etc.
- Manage Salary – This module can be accessed only by the admin staff or the HR Management, as the task of salary computation is handled only by the Admin/HR Staff. And in the, the employees only have the right to view the Salary Slip in their respective logins.
- Attendance Module – This module keeps the record of all the employees on that particular date/day.
- Regularize Attendance Module – This module can be accessed by both the employees as well as the admin staff as the employee who are coming late and want to mark their attendance when they arrive, then in that case they are required to fill this particular form and if the reason is the something genuine then only it can be approved by the admin or otherwise it would not be set valid regularization.

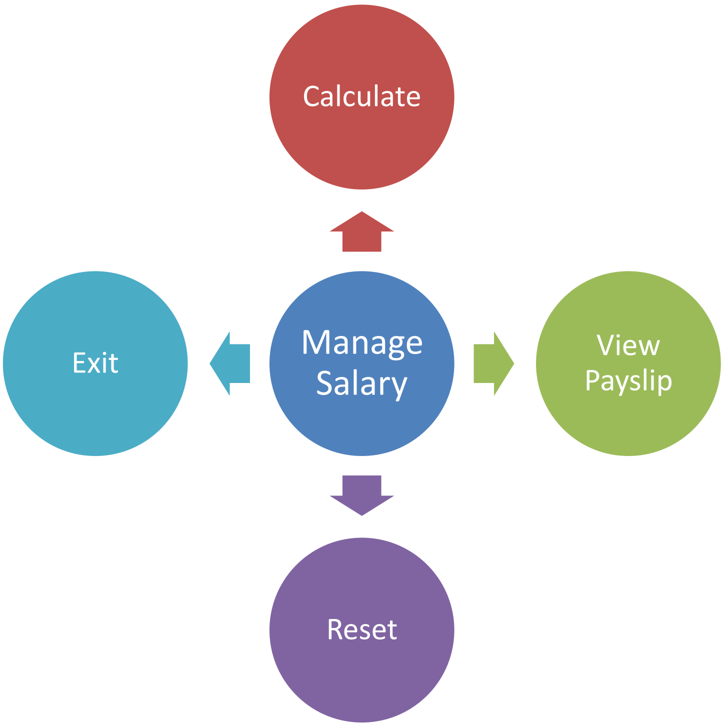
DATA FLOW DIAGRAM

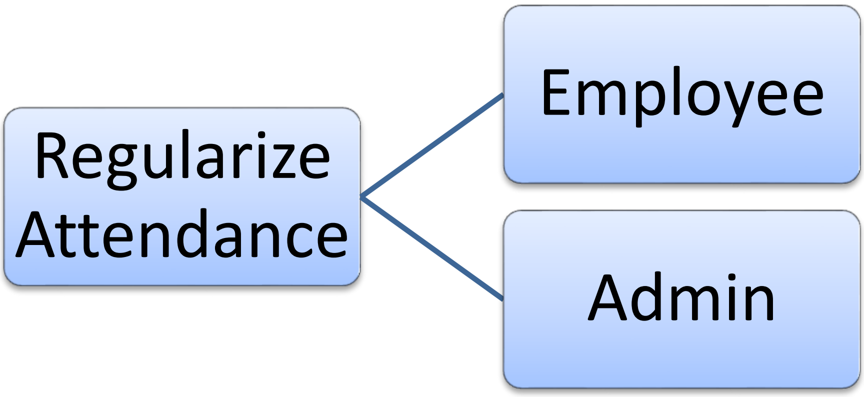
DFD is the abbreviation used for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modelling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.




Employee Information
Logout
Mark Attendance
Regularize Attendance
Employee Login
Submit Leave
View Salary Slip
Apply Short Leave
All Leaves Status


All Employee Information
Logout
Leave Approval List
Admin Login
Regularize Attendance
Approve Leave
View Salary Status
View Attendance





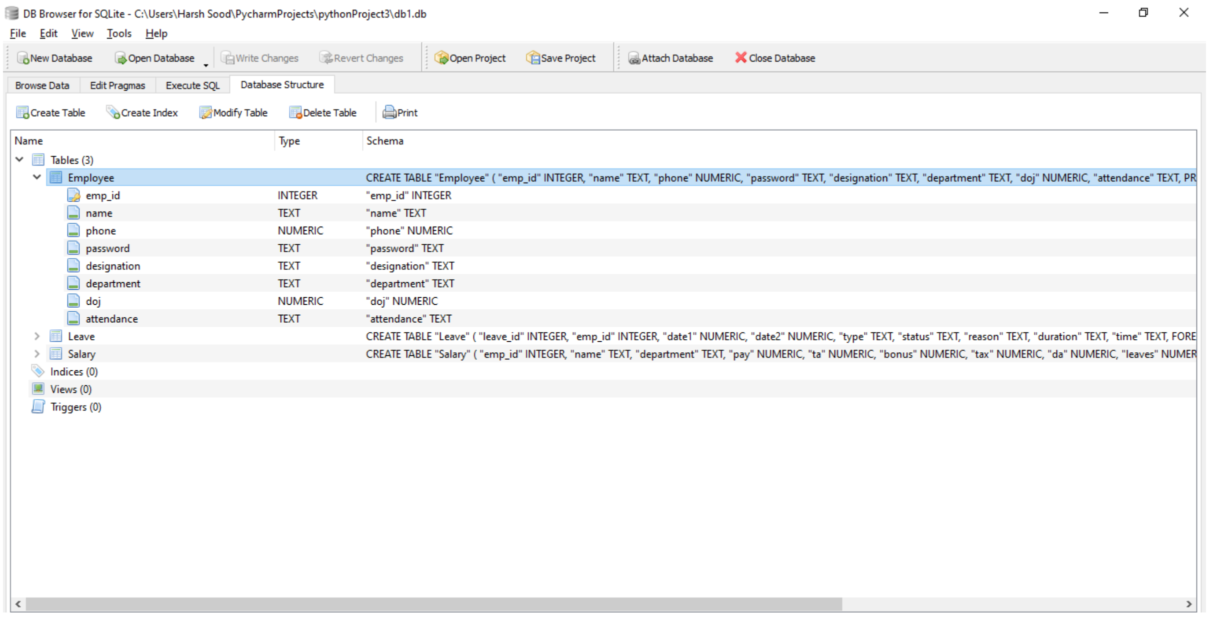
The above shown images are of the Data Flow Diagram of the Employee Management System. It gives the detail of the data flow and also provides the description of how the system will work. It describes about all the different functionalities that will be occurring in the system at all the points.

TABLES (included in Database)

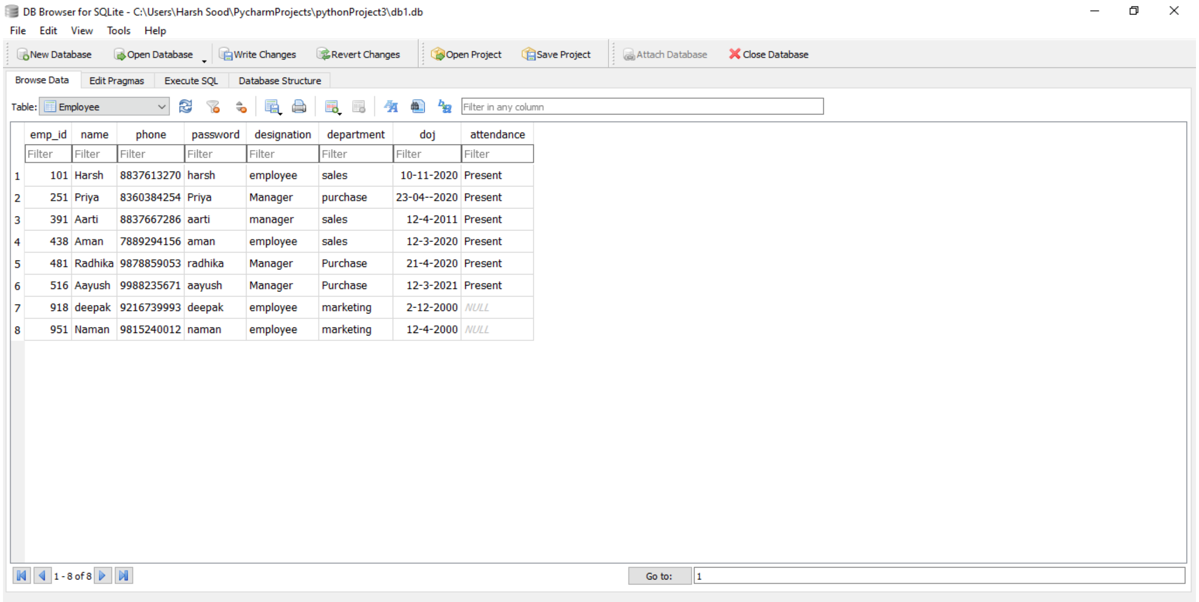
The database of the system includes three different table named as: - Employee, Leave and Salary, etc., which comprise of all the fields that are related to the "Employee" and are being used in the system. The fields included are: -

- Employee Table – [emp_id, name, phone, password, designation, department, doj, attendance].
- Leave Table – [leave_id, emp_id, date1, date2, type, status, reason, duration, time].
- Salary Table – [emp_id, name, department, pay, ta, bonus, tax, da, leaves, pf, net].

The structure of 'Employee Table' with its data types for each of the fields are as under:

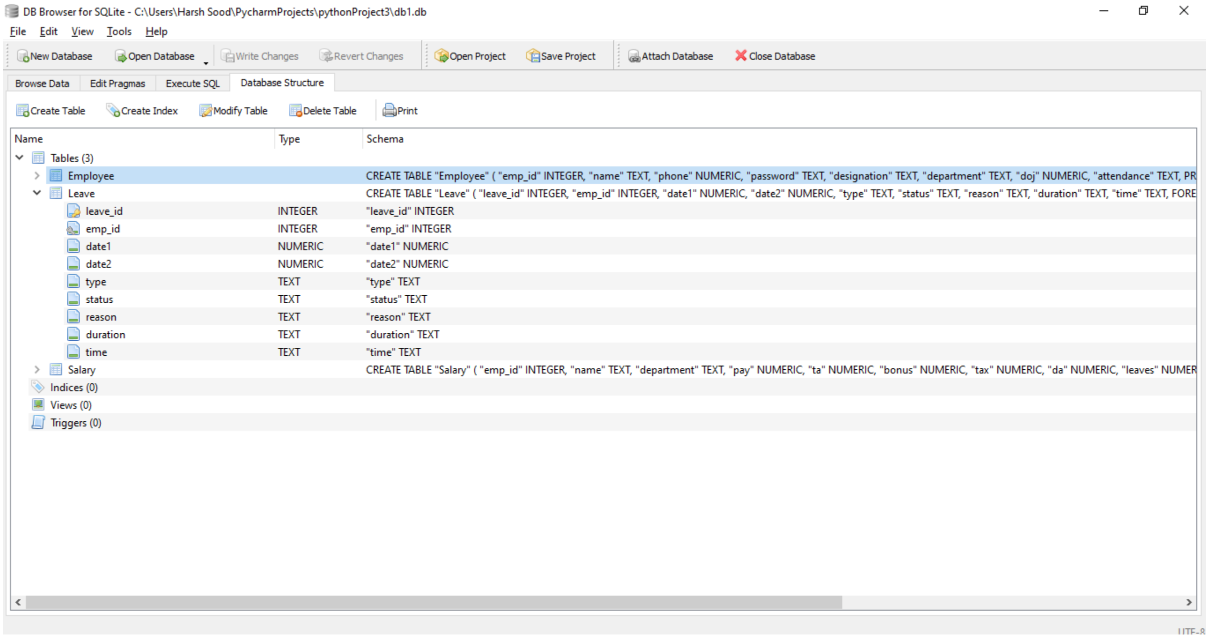


Name	Type	Schema
Tables (3)		
Employee		CREATE TABLE "Employee" ("emp_id" INTEGER, "name" TEXT, "phone" NUMERIC, "password" TEXT, "designation" TEXT, "department" TEXT, "doj" NUMERIC, "attendance" TEXT, PR
emp_id	INTEGER	"emp_id" INTEGER
name	TEXT	"name" TEXT
phone	NUMERIC	"phone" NUMERIC
password	TEXT	"password" TEXT
designation	TEXT	"designation" TEXT
department	TEXT	"department" TEXT
doj	NUMERIC	"doj" NUMERIC
attendance	TEXT	"attendance" TEXT
Leave		CREATE TABLE "Leave" ("leave_id" INTEGER, "emp_id" INTEGER, "date1" NUMERIC, "date2" NUMERIC, "type" TEXT, "status" TEXT, "reason" TEXT, "duration" TEXT, "time" TEXT, FORE
Salary		CREATE TABLE "Salary" ("emp_id" INTEGER, "name" TEXT, "department" TEXT, "pay" NUMERIC, "ta" NUMERIC, "bonus" NUMERIC, "tax" NUMERIC, "da" NUMERIC, "leaves" NUMER
Indices (0)		
Views (0)		
Triggers (0)		

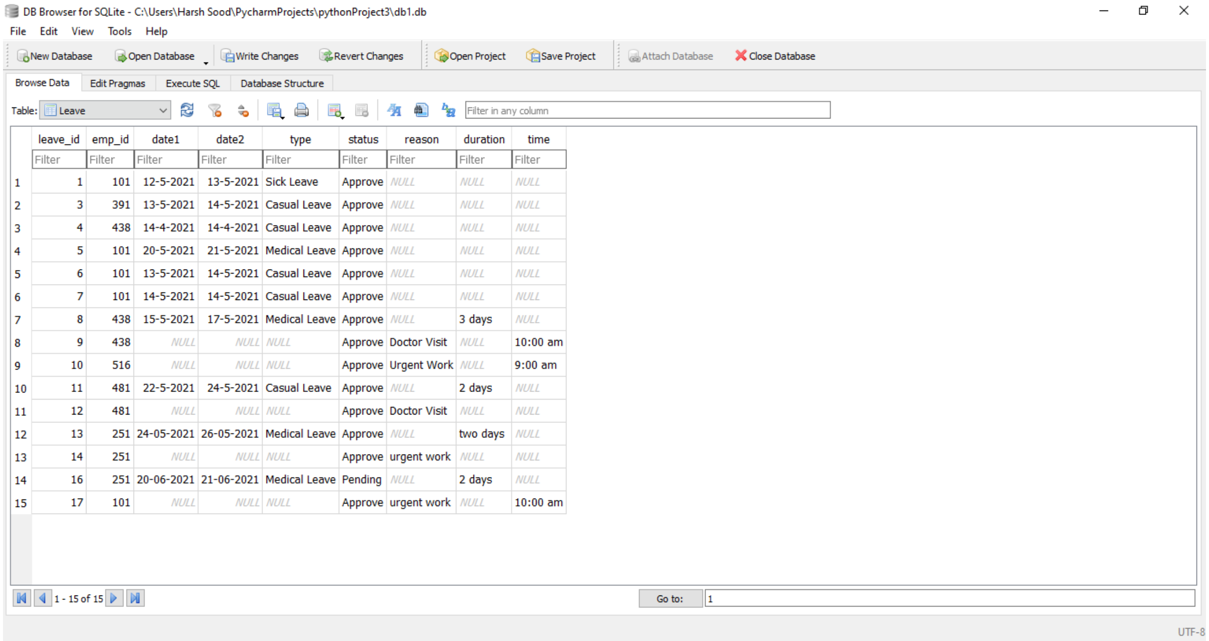


	emp_id	name	phone	password	designation	department	doj	attendance
1	101	Harsh	8837613270	harsh	employee	sales	10-11-2020	Present
2	251	Priya	8360384254	Priya	Manager	purchase	23-04--2020	Present
3	391	Aarti	8837667286	aarti	manager	sales	12-4-2011	Present
4	438	Aman	7889294156	aman	employee	sales	12-3-2020	Present
5	481	Radhika	9878859053	radhika	Manager	Purchase	21-4-2020	Present
6	516	Aayush	9988235671	aayush	Manager	Purchase	12-3-2021	Present
7	918	deepak	9216739993	deepak	employee	marketing	2-12-2000	NULL
8	951	Naman	9815240012	naman	employee	marketing	12-4-2000	NULL

The structure of ‘Leave Table’ with its data types for each of the fields are as under:

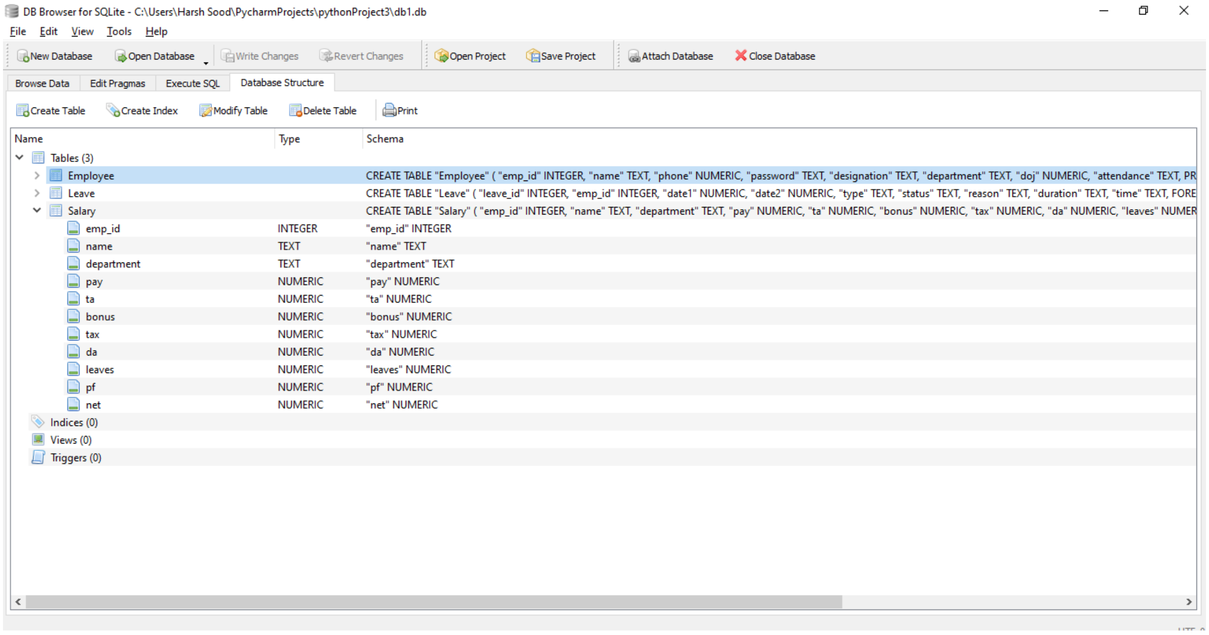


Name	Type	Schema
Tables (3)		
Employee		CREATE TABLE "Employee" ("emp_id" INTEGER, "name" TEXT, "phone" NUMERIC, "password" TEXT, "designation" TEXT, "department" TEXT, "doj" NUMERIC, "attendance" TEXT, PR
Leave		CREATE TABLE "Leave" ("leave_id" INTEGER, "emp_id" INTEGER, "date1" NUMERIC, "date2" NUMERIC, "type" TEXT, "status" TEXT, "reason" TEXT, "duration" TEXT, "time" TEXT, FORE
leave_id	INTEGER	"leave_id" INTEGER
emp_id	INTEGER	"emp_id" INTEGER
date1	NUMERIC	"date1" NUMERIC
date2	NUMERIC	"date2" NUMERIC
type	TEXT	"type" TEXT
status	TEXT	"status" TEXT
reason	TEXT	"reason" TEXT
duration	TEXT	"duration" TEXT
time	TEXT	"time" TEXT
Salary		CREATE TABLE "Salary" ("emp_id" INTEGER, "name" TEXT, "department" TEXT, "pay" NUMERIC, "ta" NUMERIC, "bonus" NUMERIC, "tax" NUMERIC, "da" NUMERIC, "leaves" NUMER
Indices (0)		
Views (0)		
Triggers (0)		



	leave_id	emp_id	date1	date2	type	status	reason	duration	time
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	101	12-5-2021	13-5-2021	Sick Leave	Approve	NULL	NULL	NULL
2	3	391	13-5-2021	14-5-2021	Casual Leave	Approve	NULL	NULL	NULL
3	4	438	14-4-2021	14-4-2021	Casual Leave	Approve	NULL	NULL	NULL
4	5	101	20-5-2021	21-5-2021	Medical Leave	Approve	NULL	NULL	NULL
5	6	101	13-5-2021	14-5-2021	Casual Leave	Approve	NULL	NULL	NULL
6	7	101	14-5-2021	14-5-2021	Casual Leave	Approve	NULL	NULL	NULL
7	8	438	15-5-2021	17-5-2021	Medical Leave	Approve	NULL	3 days	NULL
8	9	438	NULL	NULL	NULL	Approve	Doctor Visit	NULL	10:00 am
9	10	516	NULL	NULL	NULL	Approve	Urgent Work	NULL	9:00 am
10	11	481	22-5-2021	24-5-2021	Casual Leave	Approve	NULL	2 days	NULL
11	12	481	NULL	NULL	NULL	Approve	Doctor Visit	NULL	NULL
12	13	251	24-05-2021	26-05-2021	Medical Leave	Approve	NULL	two days	NULL
13	14	251	NULL	NULL	NULL	Approve	urgent work	NULL	NULL
14	16	251	20-06-2021	21-06-2021	Medical Leave	Pending	NULL	2 days	NULL
15	17	101	NULL	NULL	NULL	Approve	urgent work	NULL	10:00 am

The structure of 'Salary Table' with its data types for each of the fields are as under:



Name	Type	Schema
Tables (3)		
Employee		CREATE TABLE "Employee" ("emp_id" INTEGER, "name" TEXT, "phone" NUMERIC, "password" TEXT, "designation" TEXT, "department" TEXT, "doj" NUMERIC, "attendance" TEXT, PR
Leave		CREATE TABLE "Leave" ("leave_id" INTEGER, "emp_id" INTEGER, "date1" NUMERIC, "date2" NUMERIC, "type" TEXT, "status" TEXT, "reason" TEXT, "duration" TEXT, "time" TEXT, FORE
Salary		CREATE TABLE "Salary" ("emp_id" INTEGER, "name" TEXT, "department" TEXT, "pay" NUMERIC, "ta" NUMERIC, "bonus" NUMERIC, "tax" NUMERIC, "da" NUMERIC, "leaves" NUMER
emp_id	INTEGER	"emp_id" INTEGER
name	TEXT	"name" TEXT
department	TEXT	"department" TEXT
pay	NUMERIC	"pay" NUMERIC
ta	NUMERIC	"ta" NUMERIC
bonus	NUMERIC	"bonus" NUMERIC
tax	NUMERIC	"tax" NUMERIC
da	NUMERIC	"da" NUMERIC
leaves	NUMERIC	"leaves" NUMERIC
pf	NUMERIC	"pf" NUMERIC
net	NUMERIC	"net" NUMERIC
Indices (0)		
Views (0)		
Triggers (0)		

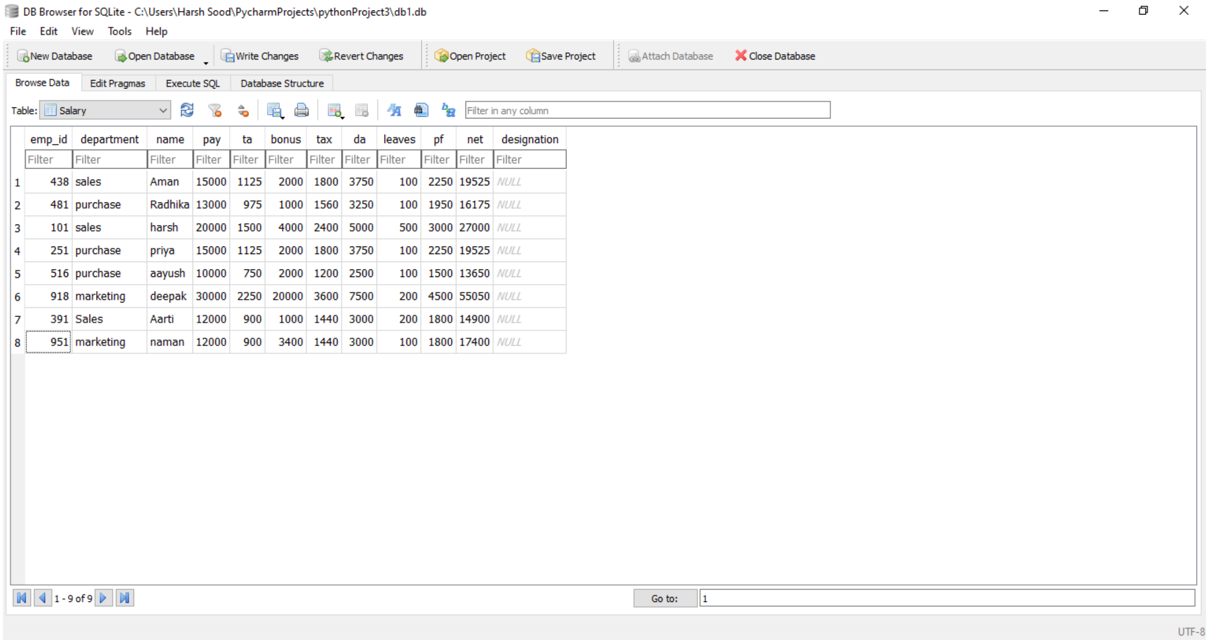


Table: Salary

	emp_id	department	name	pay	ta	bonus	tax	da	leaves	pf	net	designation
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	438	sales	Aman	15000	1125	2000	1800	3750	100	2250	19525	NULL
2	481	purchase	Radhika	13000	975	1000	1560	3250	100	1950	16175	NULL
3	101	sales	harsh	20000	1500	4000	2400	5000	500	3000	27000	NULL
4	251	purchase	priya	15000	1125	2000	1800	3750	100	2250	19525	NULL
5	516	purchase	aayush	10000	750	2000	1200	2500	100	1500	13650	NULL
6	918	marketing	deepak	30000	2250	20000	3600	7500	200	4500	55050	NULL
7	391	Sales	Aarti	12000	900	1000	1440	3000	200	1800	14900	NULL
8	951	marketing	naman	12000	900	3400	1440	3000	100	1800	17400	NULL

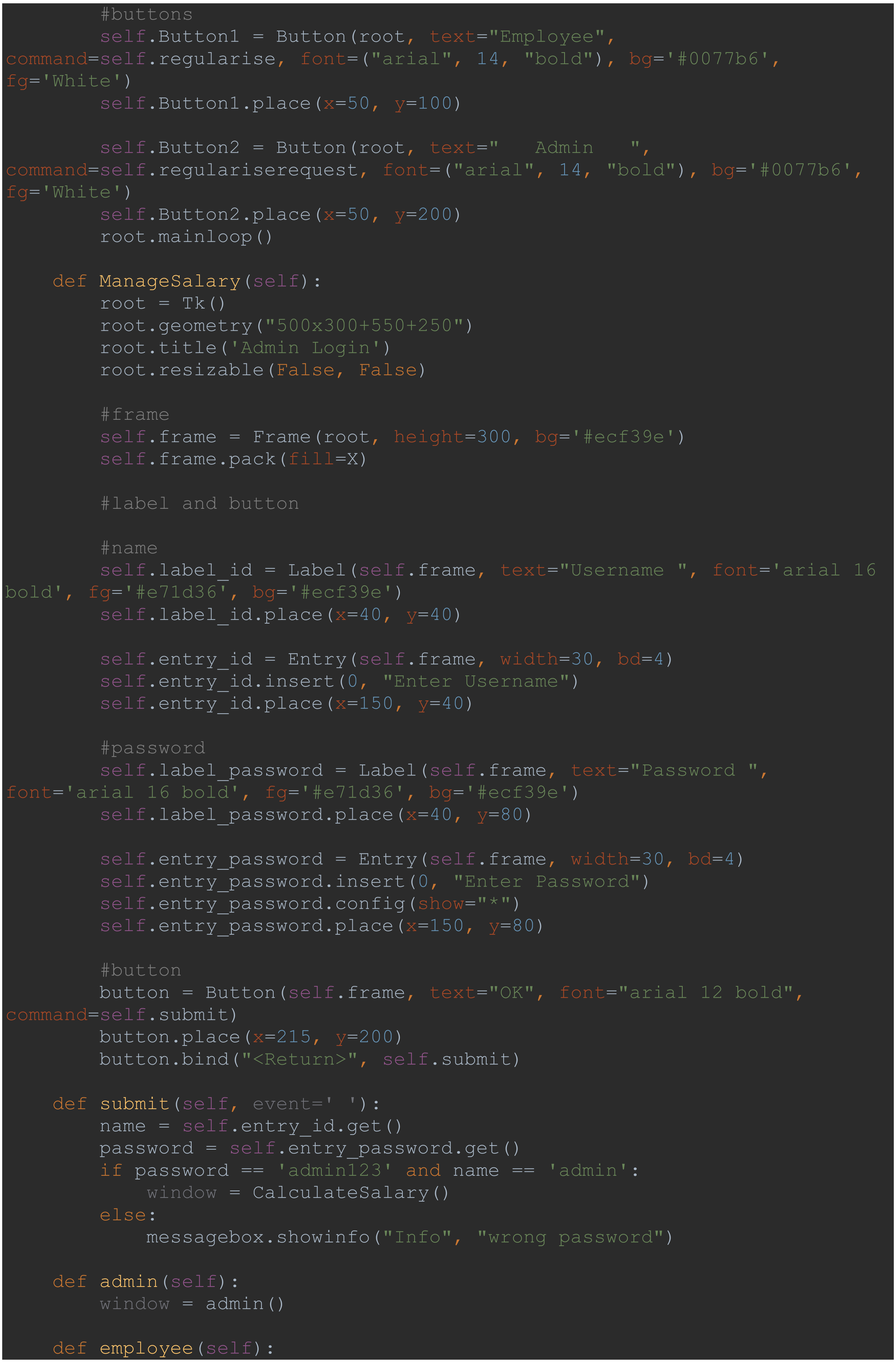
Go to: 1

The above images show the basic structure of database i.e, Name of the table, Data Type of the attributes of the table and the schema of the table, etc. In the ‘Employee’ Table, emp_id and in the ‘Leave’ table leave_id are the primary keys.

CODING: -

Main Window:

```
from tkinter import *
from Registration import Registration
from admin import admin
from employee import employee
from tkinter import messagebox
from CalculateSalary import CalculateSalary
from Regularise import form
from ApproveRequest import Approve_Request
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

class System(object):
    def __init__(self,master):
        self.master = master

        #frame
        f1 = Frame(master, width=1360, height=100, bd=8, bg="#4f772d")
        f1.place(x=110, y=0)

        lbl_information = Label(f1, font=('arial', 45, 'bold'),
text="EMPLOYEE MANAGEMENT SYSTEM", relief=GROOVE, bd=10, bg="#4f772d",
fg="White")
        lbl_information.grid(row=0, column=0)

        f2 = Frame(master, width=1360, height=50, bd=8, bg="#4f772d")
        f2.place(x=0, y=110)

        self.Button1 = Button(f2, text="Employee", command=self.employee,
font=("arial", 14, "bold"), bg='#90a955', fg='White')
        self.Button1.place(x=1080, y=0)

        self.Button2 = Button(f2, text="   Admin    ", command=self.admin,
font=("arial", 14, "bold"), bg='#90a955', fg='White')
        self.Button2.place(x=960, y=0)

        self.Button3 = Button(f2, text="Register",
command=self.Registration, font=("arial", 14, "bold"), bg='#90a955',
fg='White')
        self.Button3.place(x=100, y=0)

        self.Button4 = Button(f2, text="Home", font=("arial", 14, "bold"),
bg='#90a955', fg='White', command=master.destroy)
        self.Button4.place(x=20, y=0)

        f3 = Frame(master, width=1200, height=450, bd=8, bg="#31572c")
        f3.place(x=60, y=220)

        label1 = Label(f3, font=('arial', 20, 'bold'), text="Welcome
Admin!!",  bg="#31572c", fg="White")
        label1.place(x=40, y=30)

        self.top_image2 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\1742-490x288.png")
        self.top_image2_label2 = Label(f3, image=self.top_image2)
        self.top_image2_label2.place(x=80, y=80)

        label2 = Label(f3, font=('arial', 20, 'bold', 'underline'),
text="Features", bg="#31572c", fg="White")
        label2.place(x=700, y=30)

        self.Button2 = Button(f3, text="Manage Leaves",
command=self.ManageLeaves, width=20, relief=GROOVE, bd=10, font=("arial",
14, "bold"), bg='#4f772d', fg='White')
        self.Button2.place(x=750, y=120)

        self.Button3 = Button(f3, text="Manage Salary",
command=self.ManageSalary, width=20, relief=GROOVE, bd=10, font=("arial",
14, "bold"), bg='#4f772d', fg='White')
        self.Button3.place(x=750, y=180)

        self.Button4 = Button(f3, text="Attendance",
command=self.attendancelist, width=20, relief=GROOVE, bd=10, font=("arial",
14, "bold"), bg='#4f772d', fg='White')
        self.Button4.place(x=750, y=240)

        self.Button5 = Button(f3, text="Regularize Attendance",
command=self.Regularise, width=20, relief=GROOVE, bd=10, font=("arial", 14,
"bold"), bg='#4f772d', fg='White')
        self.Button5.place(x=750, y=300)

    def Registration(self):
        window = Registration()

    def ManageLeaves(self):
        root = Tk()
        root.title("Manage Leaves")
        root.geometry("500x300+550+250")
        root.resizable(False,False)
        root.configure(bg="#ecf39e")

        label = Label(root, text="""Select the category you belong to:""",
bg="#ecf39e", fg="#ef233c", font=('arial', 18, 'bold'))
        label.place(x=0, y=20)

        #buttons
        self.Button1 = Button(root, text="Employee", command=self.employee,
font=("arial", 14, "bold"), bg='#0077b6', fg='White')
        self.Button1.place(x=50, y=100)

        self.Button2 = Button(root, text="   Admin    ", command=self.admin,
font=("arial", 14, "bold"), bg='#0077b6', fg='White')
        self.Button2.place(x=50, y=200)
        root.mainloop()

    def Regularise(self):
        root = Tk()
        root.title("Regularise Attendance")
        root.geometry("500x300+550+250")
        root.resizable(False,False)
        root.configure(bg="#ecf39e")

        label = Label(root, text="""Select the category you belong to:""",
bg="#ecf39e", fg="#ef233c", font=('arial', 18, 'bold'))
        label.place(x=0, y=20)
```



```
        #buttons
        self.Button1 = Button(root, text="Employee",
command=self.regularise, font=("arial", 14, "bold"), bg='#0077b6',
fg='White')
        self.Button1.place(x=50, y=100)

        self.Button2 = Button(root, text="   Admin   ",
command=self.regulariserequest, font=("arial", 14, "bold"), bg='#0077b6',
fg='White')
        self.Button2.place(x=50, y=200)
        root.mainloop()

    def ManageSalary(self):
        root = Tk()
        root.geometry("500x300+550+250")
        root.title('Admin Login')
        root.resizable(False, False)

        #frame
        self.frame = Frame(root, height=300, bg='#ecf39e')
        self.frame.pack(fill=X)

        #label and button

        #name
        self.label_id = Label(self.frame, text="Username ", font='arial 16
bold', fg='#e71d36', bg='#ecf39e')
        self.label_id.place(x=40, y=40)

        self.entry_id = Entry(self.frame, width=30, bd=4)
        self.entry_id.insert(0, "Enter Username")
        self.entry_id.place(x=150, y=40)

        #password
        self.label_password = Label(self.frame, text="Password ",
font='arial 16 bold', fg='#e71d36', bg='#ecf39e')
        self.label_password.place(x=40, y=80)

        self.entry_password = Entry(self.frame, width=30, bd=4)
        self.entry_password.insert(0, "Enter Password")
        self.entry_password.config(show="*")
        self.entry_password.place(x=150, y=80)

        #button
        button = Button(self.frame, text="OK", font="arial 12 bold",
command=self.submit)
        button.place(x=215, y=200)
        button.bind("<Return>", self.submit)

    def submit(self, event=' '):
        name = self.entry_id.get()
        password = self.entry_password.get()
        if password == 'admin123' and name == 'admin':
            window = CalculateSalary()
        else:
            messagebox.showinfo("Info", "wrong password")

    def admin(self):
        window = admin()

    def employee(self):
        window = employee()

    def regularise(self):
        window = form()

    def regulariserequest(self):
        window = Approve_Request()

    def attendancelist(self):
        root = Tk()
        root.title("Attendance Record")
        root.geometry('700x500')
        root.resizable(False, False)
        root.configure(bg="#283618")

        #frames
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(root, height=400, bg='#283618')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text="Attendance Record List", font='TimesNewRoman 24 bold', bg='#606c38', fg='White')
        self.heading.place(x=150, y=40)

        #bottom frame
        rows = []
        cur.execute('SELECT emp_id,name,designation,attendance FROM Employee where attendance=="Present"')

        column_names = [description[0] for description in cur.description]

        data = [tuple(column_names)] + cur.fetchall()

        for i in range(30):

            cols = []
            for j in range(4):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[i][j])
                cols.append(e)
            rows.append(cols)

def main():
    root = Tk()
    sys = System(root)
    root.title("Employee Management System")
    root.geometry("1360x900+0+0")
    root.configure(bg='#ecf39e')
    root.resizable(False,False)
    root.mainloop()

if __name__ == '__main__':
    main()
```

Registration page: -

```
from tkinter import *
from tkinter import messagebox
import random
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

class Registration(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.title("Registration Form")
        self.geometry("1360x900+0+0")
        self.configure(bg="#8d99ae")
        self.resizable(False, False)

        # labels and buttons
        # frames
        frame = Frame(self, width=600, height=600, bg='#2b2d42')
        frame.place(x=350, y=80)

        # top frame design
        self.heading = Label(frame, text="Registration Form",
font='TimesNewRoman 20 bold', bg='#2b2d42', fg='White')
        self.heading.place(x=160, y=20)

        # id
        self.label_id = Label(frame, text="Employee ID", font='arial 12
bold', bg='#2b2d42', fg='White')
        self.label_id.place(x=130, y=80)

        self.entry_id = Entry(frame, width=30, bd=4)
        self.entry_id.insert(0, random.randint(1, 1000))
        self.entry_id.place(x=270, y=80)

        # name
        self.label_name = Label(frame, text="Name", font='arial 12 bold',
bg='#2b2d42', fg='White')
        self.label_name.place(x=130, y=120)

        self.entry_name = Entry(frame, width=30, bd=4)
        self.entry_name.insert(0, "Enter Name")
        self.entry_name.place(x=270, y=120)

        # contact number
        self.label_phone = Label(frame, text="Contact Number", font='arial
12 bold', bg='#2b2d42', fg='White')
        self.label_phone.place(x=130, y=160)

        self.entry_phone = Entry(frame, width=30, bd=4)
        self.entry_phone.insert(0, "Enter Contact Number")
        self.entry_phone.place(x=270, y=160)

        # password
        self.label_password = Label(frame, text="Password", font='arial 12
bold', bg='#2b2d42', fg='White')
        self.label_password.place(x=130, y=200)
        self.entry_password = Entry(frame, width=30, bd=4)
        self.entry_password.insert(0, "Enter Password")
        self.entry_password.config(show="*")
        self.entry_password.place(x=270, y=200)

        # designation
        self.label_desg = Label(frame, text="Designation", font='arial 12
bold', bg='#2b2d42', fg='White')
        self.label_desg.place(x=130, y=240)

        self.entry_desg = Entry(frame, width=30, bd=4)
        self.entry_desg.insert(0, "Enter Designation")
        self.entry_desg.place(x=270, y=240)

        # department
        self.label_dept = Label(frame, text="Department", font='arial 12
bold', bg='#2b2d42', fg='White')
        self.label_dept.place(x=130, y=280)

        self.entry_dept = Entry(frame, width=30, bd=4)
        self.entry_dept.insert(0, "Enter Department")
        self.entry_dept.place(x=270, y=280)

        # Date of Joining
        self.label_DOJ = Label(frame, text="Date of Joining", font='arial
12 bold', bg='#2b2d42', fg='White')
        self.label_DOJ.place(x=130, y=320)

        self.entry_DOJ = Entry(frame, width=30, bd=4)
        self.entry_DOJ.insert(0, "Enter DOJ")
        self.entry_DOJ.place(x=270, y=320)

        # button 1- Register
        button = Button(frame, text="OK", font="arial 12 bold",
bg="#06d6a0", fg="White", bd=10, width=14, relief=GROOVE,
command=self.add_people)
        button.place(x=200, y=400)

        # button 2- Cancel
        button = Button(frame, text="Cancel", font="arial 12 bold",
bg="#06d6a0", fg="White", bd=10, width=14, relief=GROOVE,
command=self.destroy)
        button.place(x=200, y=450)

    def add_people(self):
        emp_id = self.entry_id.get()
        name = self.entry_name.get()
        phone = self.entry_phone.get()
        password = self.entry_password.get()
        designation = self.entry_desg.get()
        department = self.entry_dept.get()
        doj = self.entry_DOJ.get()

        if emp_id and name and phone and password and designation and
department and doj != "":
            try:
                # add to database
                query = "Insert into 'Employee'
(emp_id,name,phone,password,designation,department,doj)
values(?,?,?,?,?,?,?)"
                cur.execute(query, (emp_id, name, phone, password,
designation, department, doj))
                con.commit()
                messagebox.showinfo("Success", "Registered Successfully")
                self.destroy()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "fill all the fields",
icon='warning')
```

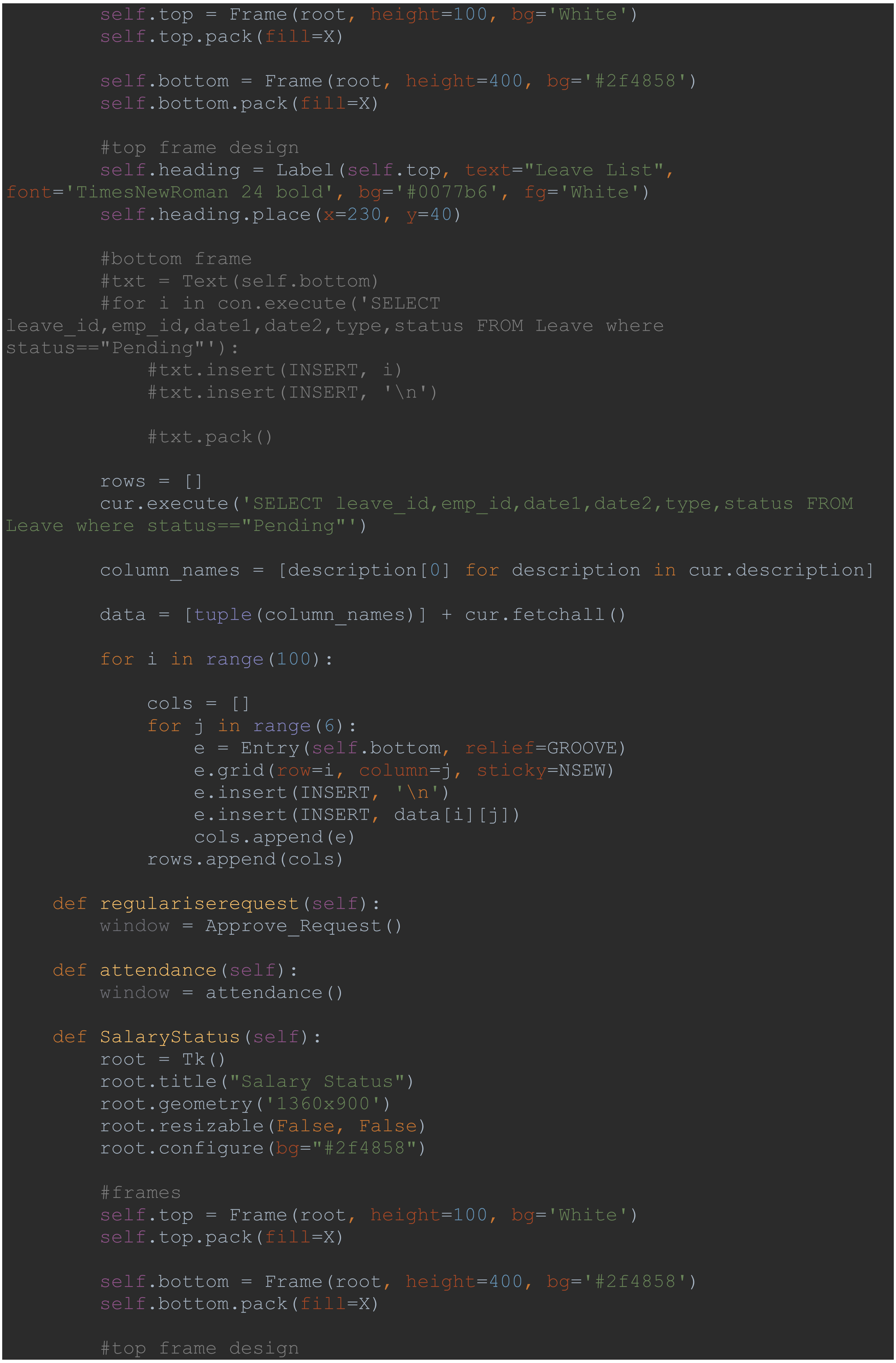
Admin Login (I)

```
from tkinter import *
from tkinter import messagebox
from AdminLogin import AdminLogin

class admin(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("500x300+550+250")
        self.title('Admin Login')
        self.resizable(False,False)

        #frame
        self.frame = Frame(self, height=300, bg='#ecf39e')
        self.frame.pack(fill=X)

        #label and button

        #name
        self.label_id = Label(self.frame, text="Username ", font='arial 16
bold', fg='#e71d36', bg='#ecf39e')
        self.label_id.place(x=40, y=40)

        self.entry_id = Entry(self.frame, width=30, bd=4)
        self.entry_id.insert(0, "Enter Username")
        self.entry_id.place(x=150, y=40)

        #password
        self.label_password = Label(self.frame, text="Password ",
font='arial 16 bold', fg='#e71d36', bg='#ecf39e')
        self.label_password.place(x=40, y=80)

        self.entry_password = Entry(self.frame, width=30, bd=4)
        self.entry_password.insert(0, "Enter Password")
        self.entry_password.config(show="*")
        self.entry_password.place(x=150, y=80)

        #button
        button = Button(self.frame, text="OK", font="arial 12 bold",
command=self.submit)
        button.place(x=215, y=200)
        button.bind("<Return>", self.submit)

    def submit(self, event=' '):
        name = self.entry_id.get()
        password = self.entry_password.get()
        if password == 'admin123' and name == 'admin':
            window = AdminLogin()
            self.destroy()
        else:
            messagebox.showinfo("Info", "wrong password")
```

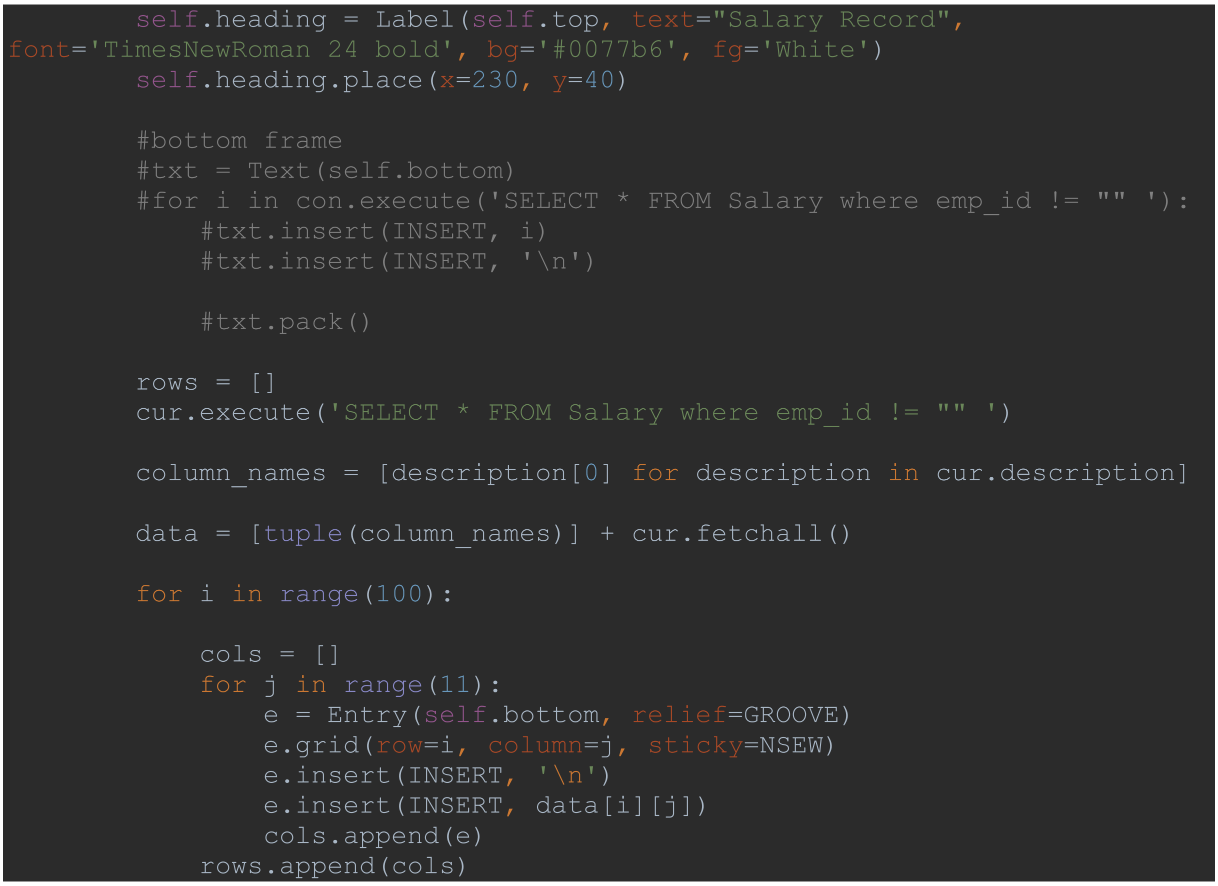
Admin Login (II)

```
from tkinter import *
from ApproveLeave import Approve_Leave
from ApproveRequest import Approve_Request
from attendance import attendance
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

class AdminLogin(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("1360x900")
        self.title("Admin Login")
        self.resizable(False, False)

        #frames
        self.top = Frame(self, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(self, height=800, bg='#33658a')
        self.bottom.pack(fill=X)

        #top frame design
        lbl_information = Label(self.top, font=('arial', 45, 'bold'),
text="ADMIN LOGIN", relief=GROOVE, bd=10, bg="#0077b6", fg="White")
        lbl_information.place(x=420, y=0)

        #button 1 = All Employee Information
        self.Button1 = Button(self.bottom, text=" All Employee Information
", command=self.AllEmployeeInformation, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button1.place(x=50, y=190)

        #image1
        self.image1_icon1 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\all_employee_information.png")
        self.label1_icon1 = Label(self.bottom, image=self.image1_icon1)
        self.label1_icon1.place(x=70, y=30)

        #button 2 = Leave Approval List
        self.Button2 = Button(self.bottom, text="      Leave Approval List
", command=self.leavelist, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button2.place(x=420, y=190)

        #image2
        self.image2_icon2 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\leave_list.png")
        self.label2_icon2 = Label(self.bottom, image=self.image2_icon2)
        self.label2_icon2.place(x=450, y=30)
        #button 3 = Approve Leave
        self.Button3 = Button(self.bottom, text="         Approve Leave
", command=self.approve, font=("TimesNewRoman", 12, "bold"), bg='#ffc300',
fg='#e71d36')
        self.Button3.place(x=770, y=190)

        #image3
        self.image3_icon3 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\approve_leave.png")
        self.label3_icon3 = Label(self.bottom, image=self.image3_icon3)
        self.label3_icon3.place(x=790, y=30)

        #button 4 = View Attendance
        self.Button4 = Button(self.bottom, text="          View Attendance
", command=self.attendance, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button4.place(x=1120, y=190)

        #image4
        self.image4_icon4 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\attendance.png")
        self.label4_icon4 = Label(self.bottom, image=self.image4_icon4)
        self.label4_icon4.place(x=1130, y=30)

        #button 5 = Regularise Attendance
        self.Button5 = Button(self.bottom, text="Regularise Attendance",
command=self.regulariserequest, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button5.place(x=620, y=460)

        #image5
        self.image5_icon5 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\regularise.png")
        self.label5_icon5 = Label(self.bottom, image=self.image5_icon5)
        self.label5_icon5.place(x=640, y=300)

        #button 6 = Salary Status
        self.Button6 = Button(self.bottom, text="    View Salary Status
", command=self.SalaryStatus, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button6.place(x=250, y=460)

        #image6
        self.image6_icon6 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\salary.png")
        self.label6_icon6 = Label(self.bottom, image=self.image6_icon6)
        self.label6_icon6.place(x=270, y=300)

        #button 7 = Logout
        self.Button7 = Button(self.bottom, text="             Logout
", font=("TimesNewRoman", 12, "bold"), bg='#ffc300', fg='#e71d36',
command=self.destroy)
        self.Button7.place(x=990, y=460)

        #image7
        self.image7_icon7 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\logout.png")
        self.label7_icon7 = Label(self.bottom, image=self.image7_icon7)
        self.label7_icon7.place(x=1000, y=300)
```

```
    def AllEmployeeInformation(self):
        root = Tk()
        root.title("All Employee's Record")
        root.geometry('700x500')
        root.resizable(False, False)
        root.configure(bg="#2f4858")

        #frames
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(root, height=400, bg='#2f4858')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text="All Employee's Record",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)

        #bottom frame
        #txt = Text(self.bottom)
        #for i in con.execute('SELECT
emp_id,name,phone,password,designation,department,doj FROM Employee where
doj != "" '):
            #txt.insert(INSERT, i)
            #txt.insert(INSERT, '\n')

            #txt.pack()

        rows = []
        cur.execute('SELECT emp_id,name,phone,designation,department,doj
FROM Employee where doj != "" ')

        column_names = [description[0] for description in cur.description]


        data = [tuple(column_names)] + cur.fetchall()


        for i in range(100):

            cols = []
            for j in range(6):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[i][j])
                cols.append(e)
            rows.append(cols)

    def approve(self):
        window = Approve_Leave()

    def leavelist(self):
        root = Tk()
        root.title("All Leaves List")
        root.geometry('700x500')
        root.resizable(False, False)
        root.configure(bg="#2f4858")

        #frames
```

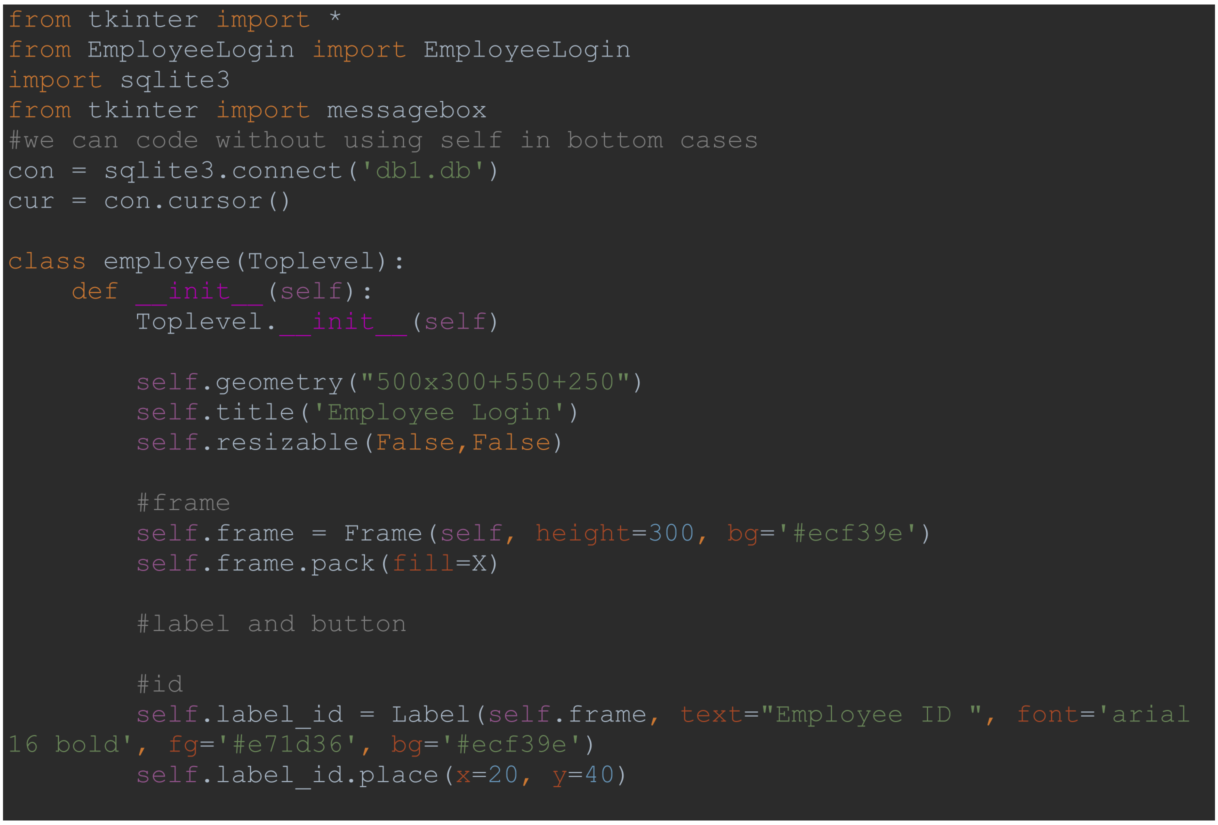


```
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(root, height=400, bg='#2f4858')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text="Leave List",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)

        #bottom frame
        #txt = Text(self.bottom)
        #for i in con.execute('SELECT
leave_id,emp_id,date1,date2,type,status FROM Leave where
status=="Pending"'):
            #txt.insert(INSERT, i)
            #txt.insert(INSERT, '\n')

            #txt.pack()

        rows = []
        cur.execute('SELECT leave_id,emp_id,date1,date2,type,status FROM
Leave where status=="Pending"')

        column_names = [description[0] for description in cur.description]

        data = [tuple(column_names)] + cur.fetchall()

        for i in range(100):

            cols = []
            for j in range(6):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[i][j])
                cols.append(e)
            rows.append(cols)

    def regulariserequest(self):
        window = Approve_Request()

    def attendance(self):
        window = attendance()

    def SalaryStatus(self):
        root = Tk()
        root.title("Salary Status")
        root.geometry('1360x900')
        root.resizable(False, False)
        root.configure(bg="#2f4858")

        #frames
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(root, height=400, bg='#2f4858')
        self.bottom.pack(fill=X)

        #top frame design
```



```
        self.heading = Label(self.top, text="Salary Record",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)

        #bottom frame
        #txt = Text(self.bottom)
        #for i in con.execute('SELECT * FROM Salary where emp_id != "" '):
            #txt.insert(INSERT, i)
            #txt.insert(INSERT, '\n')

            #txt.pack()

        rows = []
        cur.execute('SELECT * FROM Salary where emp_id != "" ')

        column_names = [description[0] for description in cur.description]

        data = [tuple(column_names)] + cur.fetchall()

        for i in range(100):

            cols = []
            for j in range(11):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[i][j])
                cols.append(e)
            rows.append(cols)
```

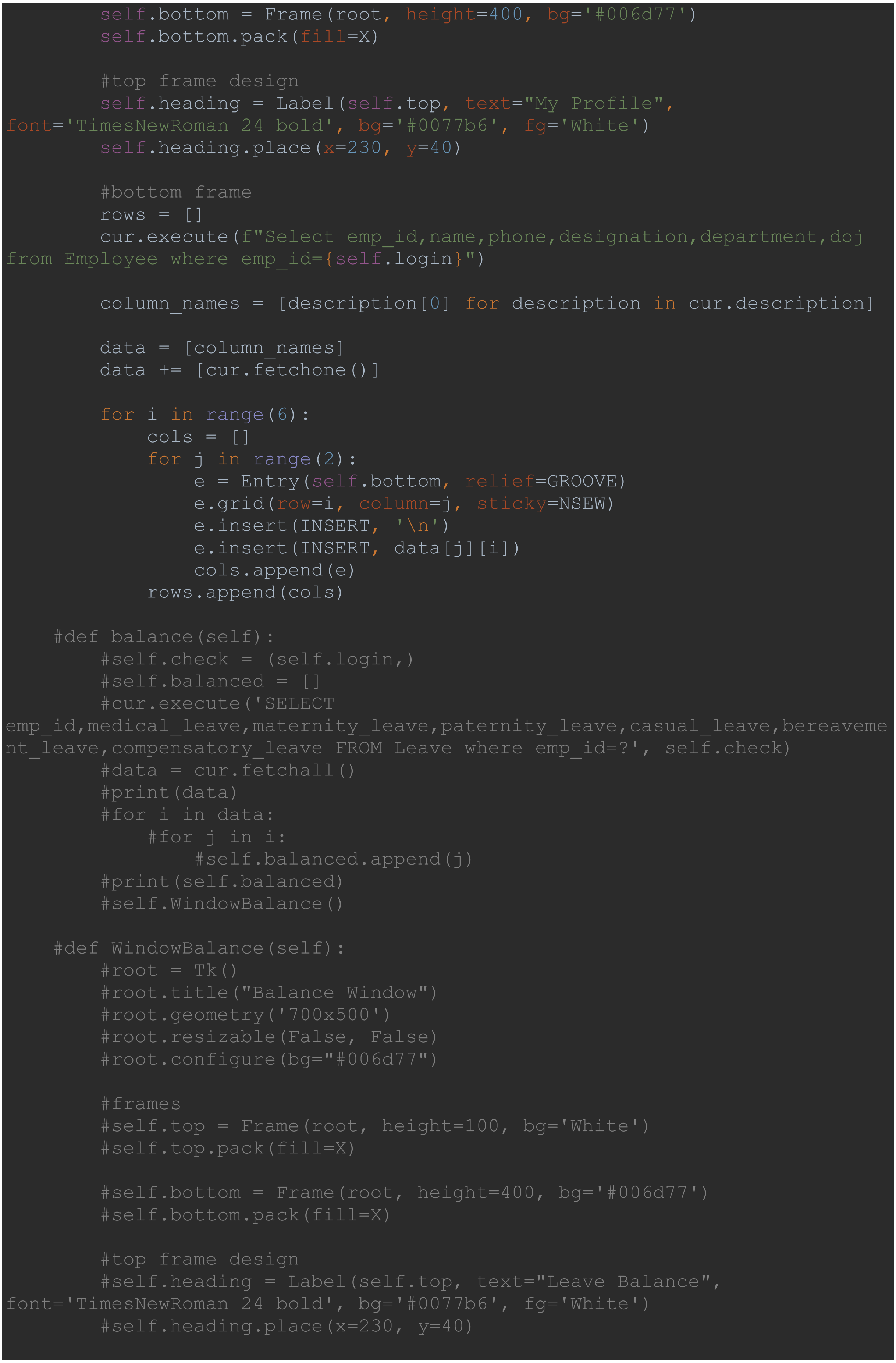
Employee Login (I)



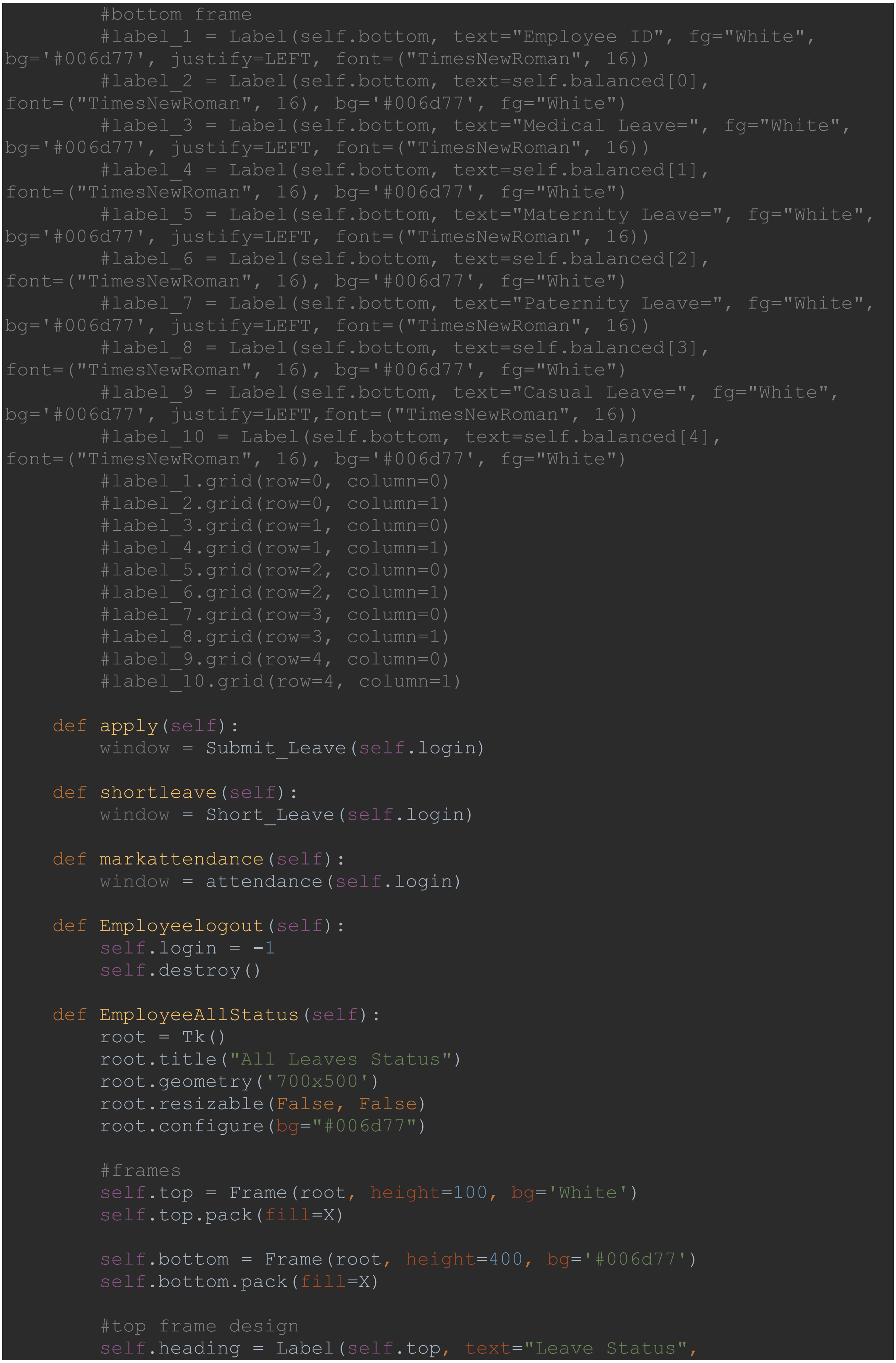
```
from tkinter import *
from EmployeeLogin import EmployeeLogin
import sqlite3
from tkinter import messagebox
#we can code without using self in bottom cases
con = sqlite3.connect('db1.db')
cur = con.cursor()

class employee(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("500x300+550+250")
        self.title('Employee Login')
        self.resizable(False,False)

        #frame
        self.frame = Frame(self, height=300, bg='#ecf39e')
        self.frame.pack(fill=X)

        #label and button

        #id
        self.label_id = Label(self.frame, text="Employee ID ", font='arial
16 bold', fg='#e71d36', bg='#ecf39e')
        self.label_id.place(x=20, y=40)
        self.entry_id = Entry(self.frame, width=30, bd=4)
        self.entry_id.insert(0, "Enter Employee ID")
        self.entry_id.place(x=150, y=40)

        #password
        self.label_password = Label(self.frame, text="Password ", font='arial 16 bold', fg='#e71d36', bg='#ecf39e')
        self.label_password.place(x=40, y=80)

        self.entry_password = Entry(self.frame, width=30, bd=4)
        self.entry_password.insert(0, "Enter Password")
        self.entry_password.config(show="*")
        self.entry_password.place(x=150, y=80)

        #button
        button = Button(self.frame, text="OK", font="arial 12 bold", command=self.submit)
        button.place(x=215, y=200)
        button.bind("<Return>", self.submit)

    def submit(self, event=' '):
        emp_id = self.entry_id.get()
        password = self.entry_password.get()
        for row in con.execute('SELECT * FROM Employee;'):
            if id == id and password == password:
                global login
                login = emp_id
                f = 1
                print("Success")
                messagebox.showinfo("Employee Login", "Login Successful")
                EmployeeLogin(emp_id)
                break
            else:
                print("Invalid")
                messagebox.showerror("Error info", "Incorrect Employee ID or Password")
```

Employee Login (II)

```
from tkinter import *
from ApplyLeave import Submit_Leave
from ShortLeave import Short_Leave
from Regularise import form
from MarkAttendance import attendance
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

class EmployeeLogin(Toplevel):
    def __init__(self,emp_id):
        Toplevel.__init__(self)

        self.geometry("1360x900")
        self.title("Employee Login")
        self.resizable(False, False)

        self.login = emp_id
        #frames
        self.top = Frame(self, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(self, height=800, bg='#83c5be')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text='EMPLOYEE LOGIN',
font=("arial", 45, "bold"), bd=10, relief=GROOVE, bg='#0077b6', fg='White')
        self.heading.place(x=370, y=0)

        #button 1 = Employee Information
        self.Button1 = Button(self.bottom, text=" Employee Information ",
command=self.EmployeeInformation, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button1.place(x=80, y=190)

        #image1
        self.image1_icon1 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\employee_information.png")
        self.label1_icon1 = Label(self.bottom, image=self.image1_icon1)
        self.label1_icon1.place(x=100, y=30)

        #button 2 = Submit Leave
        self.Button2 = Button(self.bottom, text="      Submit Leave      ",
command=self.apply, font=("TimesNewRoman", 12, "bold"), bg='#ffc300',
fg='#e71d36')
        self.Button2.place(x=600, y=190)

        #image2
        self.image2_icon2 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\submitleave.png")
        self.label2_icon2 = Label(self.bottom, image=self.image2_icon2)
        self.label2_icon2.place(x=610, y=30)

        #button 3 = Regualrization Request
        self.Button3 = Button(self.bottom, text=" Regualrization Request ",
command=self.regularise_request, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button3.place(x=1100, y=190)

        #image3
        self.image3_icon3 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\regularise.png")
        self.label3_icon3 = Label(self.bottom, image=self.image3_icon3)
        self.label3_icon3.place(x=1120, y=30)

        #button 4 = All Leaves Status
        self.Button4 = Button(self.bottom, text="    All Leaves Status
", command=self.EmployeeAllStatus, font=("TimesNewRoman", 12, "bold"),
bg='#ffc300', fg='#e71d36')
        self.Button4.place(x=340, y=360)

        #image4
        self.image4_icon4 = PhotoImage(file=r"C:\Users\Harsh
Sood\PycharmProjects\pythonProject3\all_leave_status.png")
        self.label4_icon4 = Label(self.bottom, image=self.image4_icon4)
        self.label4_icon4.place(x=350, y=200)

        #button 5 = View Salary
        self.Button5 = Button(self.bottom, text="      View Salary Slip      ", command=self.slip, font=("TimesNewRoman", 12, "bold"), bg='#ffc300', fg='#e71d36')
        self.Button5.place(x=850, y=360)

        #image5
        self.image5_icon5 = PhotoImage(file=r"C:\Users\Harsh Sood\PycharmProjects\pythonProject3\salaryslip.png")
        self.label5_icon5 = Label(self.bottom, image=self.image5_icon5)
        self.label5_icon5.place(x=860, y=200)

        #button 6 = Mark Attendance
        self.Button6 = Button(self.bottom, text="    Mark Attendance   ", command=self.markattendance, font=("TimesNewRoman", 12, "bold"), bg='#ffc300', fg='#e71d36')
        self.Button6.place(x=80, y=530)

        #image6
        self.image6_icon6 = PhotoImage(file=r"C:\Users\Harsh Sood\PycharmProjects\pythonProject3\markattendance.png")
        self.label6_icon6 = Label(self.bottom, image=self.image6_icon6)
        self.label6_icon6.place(x=90, y=370)

        #button 7 = Apply Short Leave
        self.Button7 = Button(self.bottom, text="      Apply Short Leave      ", command=self.shortleave, font=("TimesNewRoman", 12, "bold"), bg='#ffc300', fg='#e71d36')
        self.Button7.place(x=600, y=530)

        #image7
        self.image7_icon7 = PhotoImage(file=r"C:\Users\Harsh Sood\PycharmProjects\pythonProject3\shortleave.png")
        self.label7_icon7 = Label(self.bottom, image=self.image7_icon7)
        self.label7_icon7.place(x=610, y=370)

        #button 8 = Logout
        self.Button8 = Button(self.bottom, text="              Logout              ", command=self.Employeelogout, font=("TimesNewRoman", 12, "bold"), bg='#ffc300', fg='#e71d36')
        self.Button8.place(x=1100, y=530)

        #image8
        self.image8_icon8 = PhotoImage(file=r"C:\Users\Harsh Sood\PycharmProjects\pythonProject3\logout.png")
        self.label8_icon8 = Label(self.bottom, image=self.image8_icon8)
        self.label8_icon8.place(x=1110, y=370)

    def regularise_request(self):
        window = form()

    def EmployeeInformation(self):
        root = Tk()
        root.title("Employee Information")
        root.geometry('700x500')
        root.resizable(False, False)
        root.configure(bg="#006d77")

        #frames
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)
```



```
        self.bottom = Frame(root, height=400, bg='#006d77')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text="My Profile",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)

        #bottom frame
        rows = []
        cur.execute(f"Select emp_id,name,phone,designation,department,doj
from Employee where emp_id={self.login}")

        column_names = [description[0] for description in cur.description]

        data = [column_names]
        data += [cur.fetchone()]

        for i in range(6):
            cols = []
            for j in range(2):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[j][i])
                cols.append(e)
            rows.append(cols)

    #def balance(self):
        #self.check = (self.login,)
        #self.balanced = []
        #cur.execute('SELECT
emp_id,medical_leave,maternity_leave,paternity_leave,casual_leave,bereaveme
nt_leave,compensatory_leave FROM Leave where emp_id=?', self.check)
        #data = cur.fetchall()
        #print(data)
        #for i in data:
            #for j in i:
                #self.balanced.append(j)
        #print(self.balanced)
        #self.WindowBalance()

    #def WindowBalance(self):
        #root = Tk()
        #root.title("Balance Window")
        #root.geometry('700x500')
        #root.resizable(False, False)
        #root.configure(bg="#006d77")

        #frames
        #self.top = Frame(root, height=100, bg='White')
        #self.top.pack(fill=X)

        #self.bottom = Frame(root, height=400, bg='#006d77')
        #self.bottom.pack(fill=X)

        #top frame design
        #self.heading = Label(self.top, text="Leave Balance",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        #self.heading.place(x=230, y=40)
```

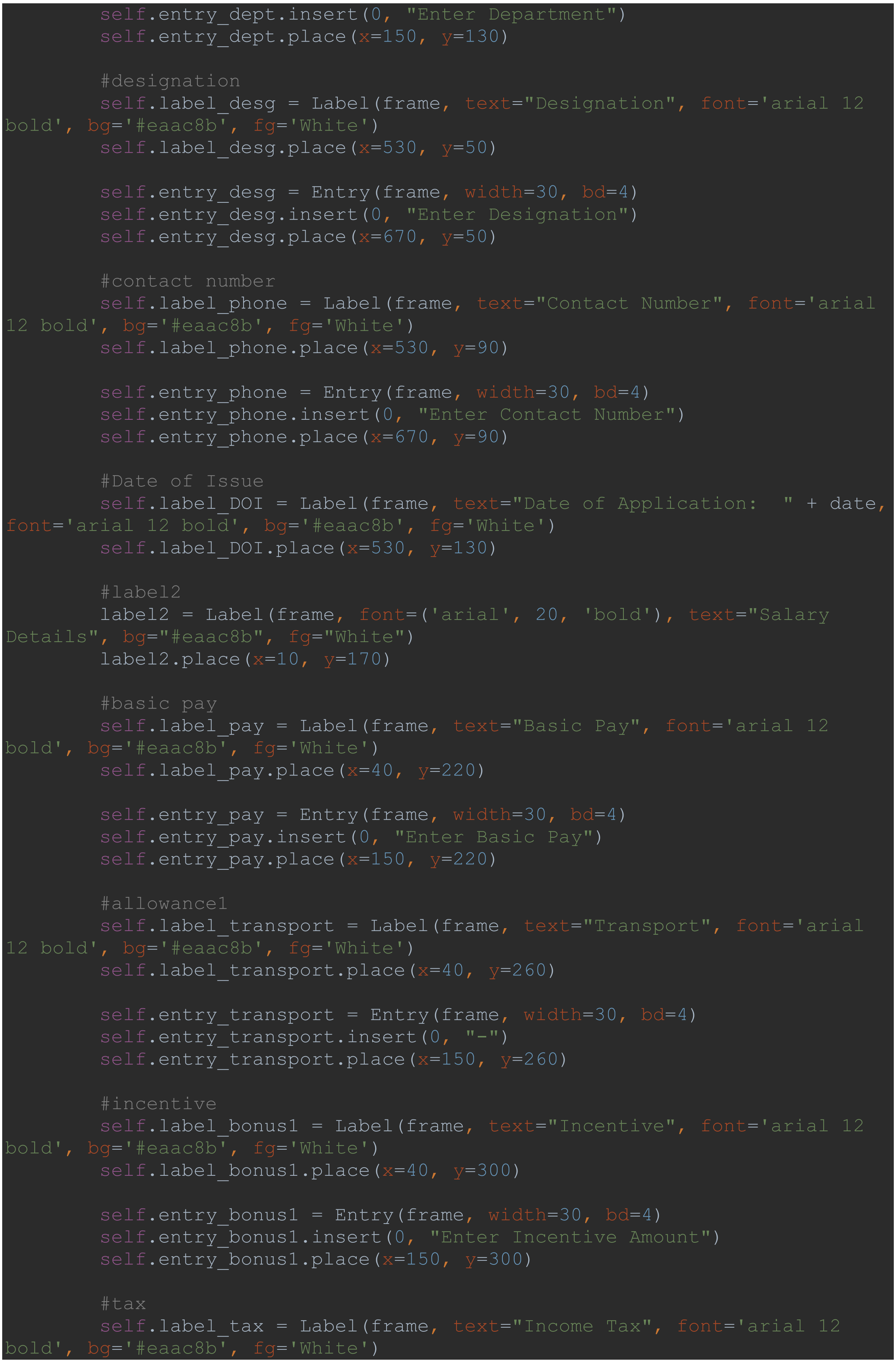


```
        #bottom frame
        #label_1 = Label(self.bottom, text="Employee ID", fg="White",
bg='#006d77', justify=LEFT, font=("TimesNewRoman", 16))
        #label_2 = Label(self.bottom, text=self.balanced[0],
font=("TimesNewRoman", 16), bg='#006d77', fg="White")
        #label_3 = Label(self.bottom, text="Medical Leave=", fg="White",
bg='#006d77', justify=LEFT, font=("TimesNewRoman", 16))
        #label_4 = Label(self.bottom, text=self.balanced[1],
font=("TimesNewRoman", 16), bg='#006d77', fg="White")
        #label_5 = Label(self.bottom, text="Maternity Leave=", fg="White",
bg='#006d77', justify=LEFT, font=("TimesNewRoman", 16))
        #label_6 = Label(self.bottom, text=self.balanced[2],
font=("TimesNewRoman", 16), bg='#006d77', fg="White")
        #label_7 = Label(self.bottom, text="Paternity Leave=", fg="White",
bg='#006d77', justify=LEFT, font=("TimesNewRoman", 16))
        #label_8 = Label(self.bottom, text=self.balanced[3],
font=("TimesNewRoman", 16), bg='#006d77', fg="White")
        #label_9 = Label(self.bottom, text="Casual Leave=", fg="White",
bg='#006d77', justify=LEFT,font=("TimesNewRoman", 16))
        #label_10 = Label(self.bottom, text=self.balanced[4],
font=("TimesNewRoman", 16), bg='#006d77', fg="White")
        #label_1.grid(row=0, column=0)
        #label_2.grid(row=0, column=1)
        #label_3.grid(row=1, column=0)
        #label_4.grid(row=1, column=1)
        #label_5.grid(row=2, column=0)
        #label_6.grid(row=2, column=1)
        #label_7.grid(row=3, column=0)
        #label_8.grid(row=3, column=1)
        #label_9.grid(row=4, column=0)
        #label_10.grid(row=4, column=1)

    def apply(self):
        window = Submit_Leave(self.login)

    def shortleave(self):
        window = Short_Leave(self.login)

    def markattendance(self):
        window = attendance(self.login)

    def Employeelogout(self):
        self.login = -1
        self.destroy()

    def EmployeeAllStatus(self):
        root = Tk()
        root.title("All Leaves Status")
        root.geometry('700x500')
        root.resizable(False, False)
        root.configure(bg="#006d77")

        #frames
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(root, height=400, bg='#006d77')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text="Leave Status",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)

        #bottom frame
        #txt = Text(self.bottom)
        #txt.pack()
        #cur.execute(f"SELECT leave_id,type,date1,date2,status FROM Leave 
where emp_id={self.login}")
        #data = cur.fetchall()
        #if any(data):
            #for i in data:
                #txt.insert(INSERT, i)
                #txt.insert(INSERT, '\n')
        #else:
            #txt.insert(INSERT, "There is no record present with given 
parameter")

        rows = []
        cur.execute(f"SELECT leave_id,type,date1,date2,status FROM Leave 
where emp_id={self.login}")

        column_names = [description[0] for description in cur.description]

        data = [tuple(column_names)] + cur.fetchall()

        for i in range(30):

            cols = []
            for j in range(5):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[i][j])
                cols.append(e)
            rows.append(cols)

    def slip(self):
        root = Tk()
        root.title("Salary Slip")
        root.geometry('700x500')
        root.resizable(False, False)
        root.configure(bg="#006d77")

        #frames
        self.top = Frame(root, height=100, bg='White')
        self.top.pack(fill=X)

        self.bottom = Frame(root, height=400, bg='#006d77')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text="Salary Slip", 
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)

        #bottom frame
        #txt = Text(self.bottom)
        #txt.pack()
        #cur.execute(f"Select * from Salary where emp_id={self.login}")
        #data = cur.fetchall()
        #if any(data):
            #for i in data:
                #txt.insert(INSERT, i)
                #txt.insert(INSERT, '\n')
        #else:
            #txt.insert(INSERT, "There is no record present with given 
parameter")

        #rows = []
        #cur.execute(f"Select * from Salary where emp_id={self.login}")

        #column_names = [description[0] for description in cur.description]

        #data = [column_names]
        #data += [cur.fetchone()]

        #for i in range(12):
            #cols = []
            #for j in range(2):
                #e = Entry(self.bottom, relief=GROOVE)
                #e.grid(row=i, column=j, sticky=NSEW)
                #e.insert(INSERT, '\n')
                #e.insert(INSERT, data[j][i])
                #cols.append(e)
            #rows.append(cols)

        rows = []
        cur.execute(f"Select * from Salary where emp_id={self.login}")

        column_names = [description[0] for description in cur.description]

        data = [column_names]
        data += [cur.fetchone()]

        for i in range(11):
            cols = []
            for j in range(2):
                e = Entry(self.bottom, relief=GROOVE)
                e.grid(row=i, column=j, sticky=NSEW)
                e.insert(INSERT, '\n')
                e.insert(INSERT, data[j][i])
                cols.append(e)
            rows.append(cols)
```

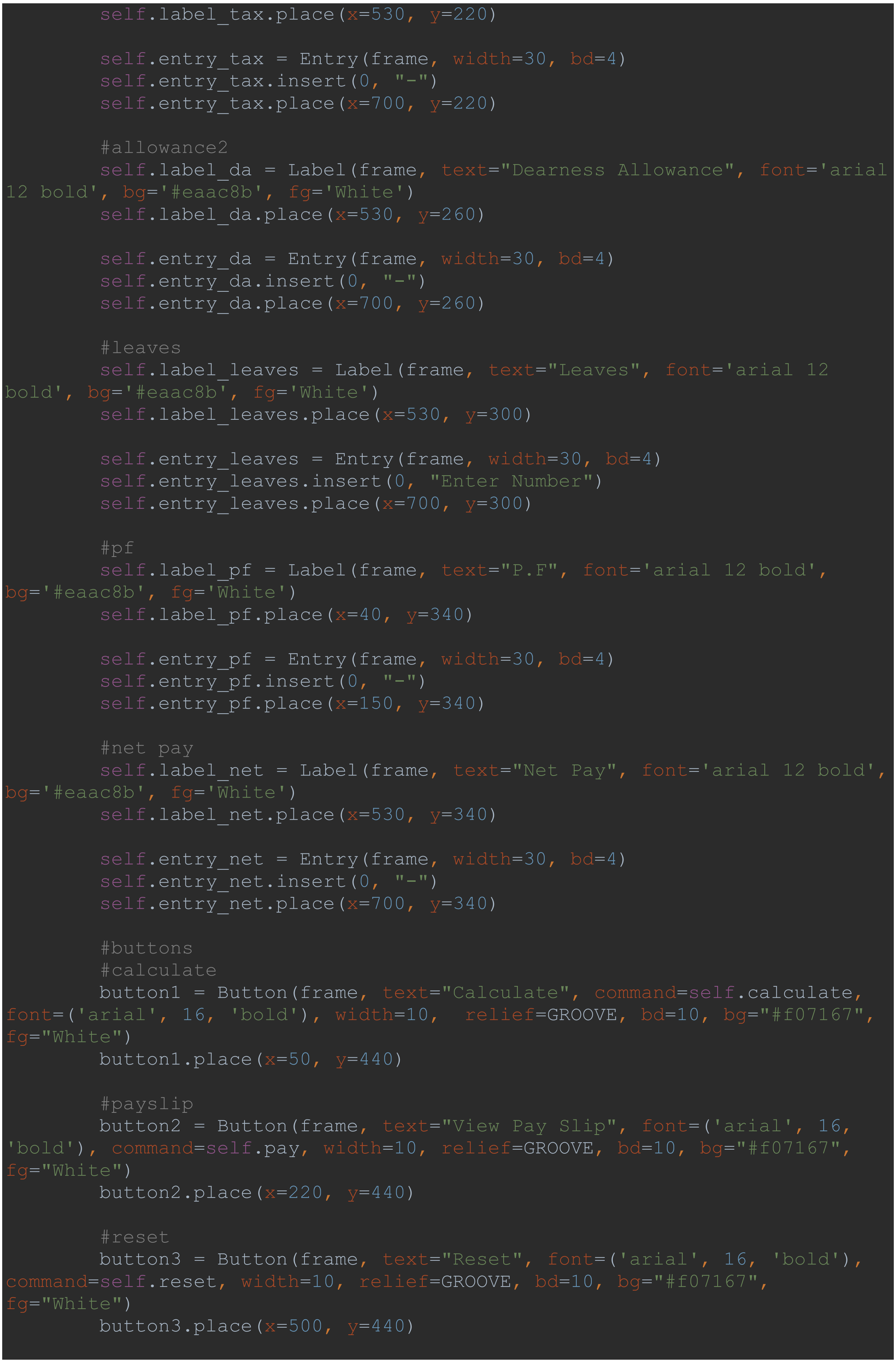
Calculate Salary

```
from tkinter import *
import datetime
date = datetime.datetime.now().date()
date = str(date)
from tkinter import messagebox
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

def change_text(ui_object, new_value):
   ui_object.delete(0,END)
   ui_object.insert(0,new_value)
```

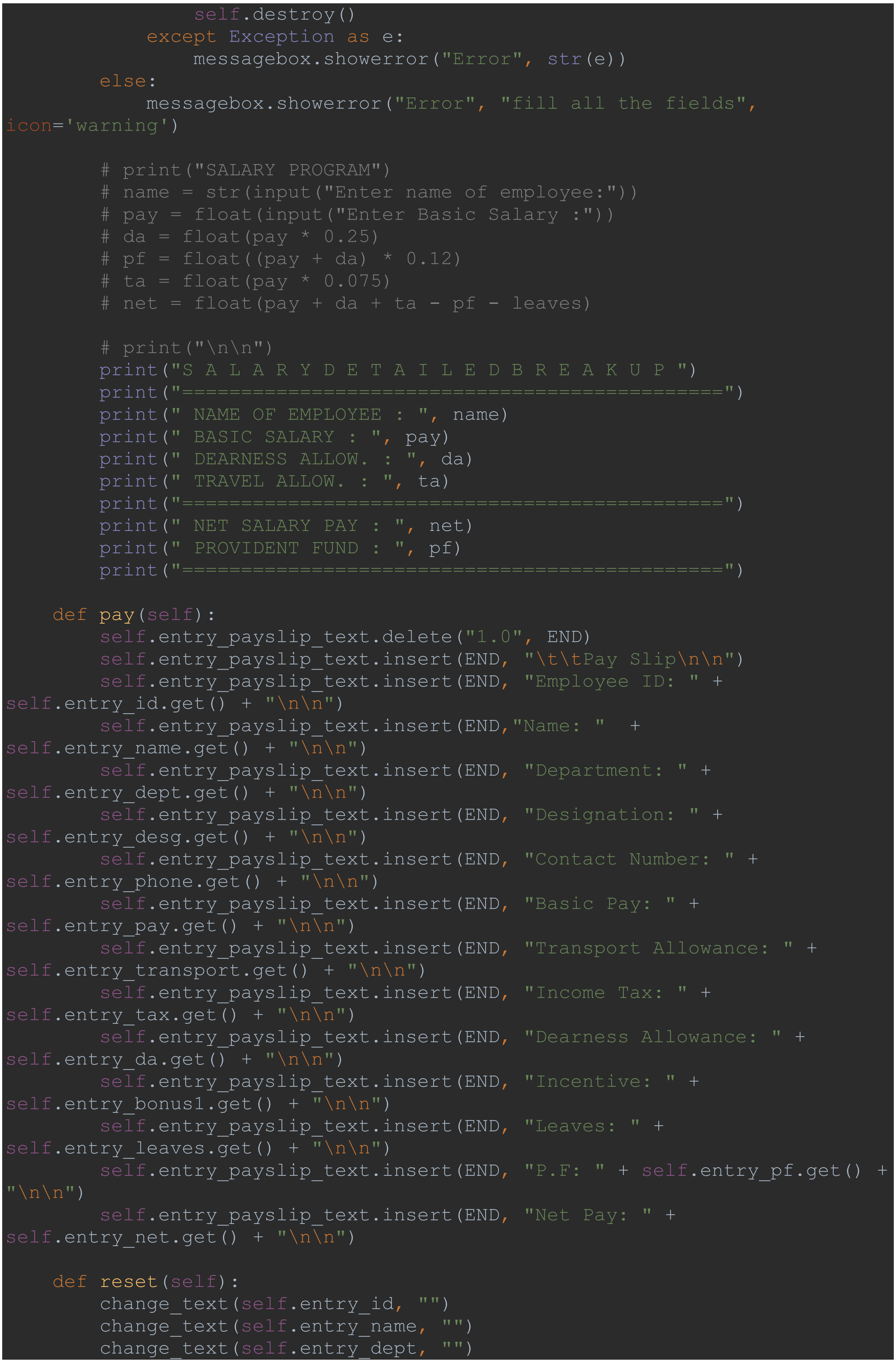
```
class CalculateSalary(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("1360x900")
        self.title("Calculate Salary")
        self.configure(bg='#fed9b7')
        self.resizable(False, False)

        #frame
        frame1 = Frame(self, width=1360, height=100, bd=8, bg="#f07167")
        frame1.place(x=310, y=0)

        lbl_information = Label(frame1, font=('arial', 45, 'bold'),
text="SALARY CALCULATION", relief=GROOVE, bd=10, bg="#f07167", fg="White")
        lbl_information.grid(row=0, column=0)

        f1 = Frame(self, width=1000, height=600, bd=8, bg="#fff1e6")
        f1.pack(side=LEFT)
        f2 = Frame(self, width=400, height=600, bd=8, bg="#f19c79")
        f2.pack(side=RIGHT)

        frame = Frame(f1, width=960, height=550, bd=8, bg="#eaac8b")
        frame.place(x=10,y=10)

        self.frame2 = Frame(f2, width=320, height=550, bd=8, bg="White")
        self.frame2.place(x=10,y=10)
        self.entry_payslip_text = Text(self.frame2, width=200, height=480,
bg="White", bd=8, relief=GROOVE)
        self.entry_payslip_text.place(x=15,y=15)


        #labels
        label1 = Label(frame, font=('arial', 20, 'bold'), text="Personal
Details", bg="#eaac8b", fg="White")
        label1.place(x=10, y=10)

        #id
        self.label_id = Label(frame, text="Employee ID", font='arial 12
bold', bg='#eaac8b', fg='White')
        self.label_id.place(x=40, y=50)

        self.entry_id = Entry(frame, width=30, bd=4)
        self.entry_id.insert(0, "Enter Employee ID")
        self.entry_id.place(x=150, y=50)

        #name
        self.label_name = Label(frame, text="Name", font='arial 12 bold',
bg='#eaac8b', fg='White')
        self.label_name.place(x=40, y=90)

        self.entry_name = Entry(frame, width=30, bd=4)
        self.entry_name.insert(0, "Enter Name")
        self.entry_name.place(x=150, y=90)

        #department
        self.label_dept = Label(frame, text="Department", font='arial 12
bold', bg='#eaac8b', fg='White')
        self.label_dept.place(x=40, y=130)

        self.entry_dept = Entry(frame, width=30, bd=4)
```



```
        self.entry_dept.insert(0, "Enter Department")
        self.entry_dept.place(x=150, y=130)

        #designation
        self.label_desg = Label(frame, text="Designation", font='arial 12
bold', bg='#eaac8b', fg='White')
        self.label_desg.place(x=530, y=50)

        self.entry_desg = Entry(frame, width=30, bd=4)
        self.entry_desg.insert(0, "Enter Designation")
        self.entry_desg.place(x=670, y=50)

        #contact number
        self.label_phone = Label(frame, text="Contact Number", font='arial
12 bold', bg='#eaac8b', fg='White')
        self.label_phone.place(x=530, y=90)

        self.entry_phone = Entry(frame, width=30, bd=4)
        self.entry_phone.insert(0, "Enter Contact Number")
        self.entry_phone.place(x=670, y=90)

        #Date of Issue
        self.label_DOI = Label(frame, text="Date of Application:  " + date,
font='arial 12 bold', bg='#eaac8b', fg='White')
        self.label_DOI.place(x=530, y=130)

        #label2
        label2 = Label(frame, font=('arial', 20, 'bold'), text="Salary
Details", bg="#eaac8b", fg="White")
        label2.place(x=10, y=170)

        #basic pay
        self.label_pay = Label(frame, text="Basic Pay", font='arial 12
bold', bg='#eaac8b', fg='White')
        self.label_pay.place(x=40, y=220)

        self.entry_pay = Entry(frame, width=30, bd=4)
        self.entry_pay.insert(0, "Enter Basic Pay")
        self.entry_pay.place(x=150, y=220)

        #allowance1
        self.label_transport = Label(frame, text="Transport", font='arial
12 bold', bg='#eaac8b', fg='White')
        self.label_transport.place(x=40, y=260)

        self.entry_transport = Entry(frame, width=30, bd=4)
        self.entry_transport.insert(0, "-")
        self.entry_transport.place(x=150, y=260)

        #incentive
        self.label_bonus1 = Label(frame, text="Incentive", font='arial 12
bold', bg='#eaac8b', fg='White')
        self.label_bonus1.place(x=40, y=300)

        self.entry_bonus1 = Entry(frame, width=30, bd=4)
        self.entry_bonus1.insert(0, "Enter Incentive Amount")
        self.entry_bonus1.place(x=150, y=300)

        #tax
        self.label_tax = Label(frame, text="Income Tax", font='arial 12
bold', bg='#eaac8b', fg='White')
```



```
        self.label_tax.place(x=530, y=220)

        self.entry_tax = Entry(frame, width=30, bd=4)
        self.entry_tax.insert(0, "-")
        self.entry_tax.place(x=700, y=220)

        #allowance2
        self.label_da = Label(frame, text="Dearness Allowance", font='arial
12 bold', bg='#eaac8b', fg='White')
        self.label_da.place(x=530, y=260)

        self.entry_da = Entry(frame, width=30, bd=4)
        self.entry_da.insert(0, "-")
        self.entry_da.place(x=700, y=260)

        #leaves
        self.label_leaves = Label(frame, text="Leaves", font='arial 12
bold', bg='#eaac8b', fg='White')
        self.label_leaves.place(x=530, y=300)

        self.entry_leaves = Entry(frame, width=30, bd=4)
        self.entry_leaves.insert(0, "Enter Number")
        self.entry_leaves.place(x=700, y=300)

        #pf
        self.label_pf = Label(frame, text="P.F", font='arial 12 bold',
bg='#eaac8b', fg='White')
        self.label_pf.place(x=40, y=340)

        self.entry_pf = Entry(frame, width=30, bd=4)
        self.entry_pf.insert(0, "-")
        self.entry_pf.place(x=150, y=340)

        #net pay
        self.label_net = Label(frame, text="Net Pay", font='arial 12 bold',
bg='#eaac8b', fg='White')
        self.label_net.place(x=530, y=340)

        self.entry_net = Entry(frame, width=30, bd=4)
        self.entry_net.insert(0, "-")
        self.entry_net.place(x=700, y=340)

        #buttons
        #calculate
        button1 = Button(frame, text="Calculate", command=self.calculate,
font=('arial', 16, 'bold'), width=10,  relief=GROOVE, bd=10, bg="#f07167",
fg="White")
        button1.place(x=50, y=440)

        #payslip
        button2 = Button(frame, text="View Pay Slip", font=('arial', 16,
'bold'), command=self.pay, width=10, relief=GROOVE, bd=10, bg="#f07167",
fg="White")
        button2.place(x=220, y=440)

        #reset
        button3 = Button(frame, text="Reset", font=('arial', 16, 'bold'),
command=self.reset, width=10, relief=GROOVE, bd=10, bg="#f07167",
fg="White")
        button3.place(x=500, y=440)
        #exit
        button4 = Button(frame, text="Exit", font=('arial', 16, 'bold'),
command=self.Exit, width=10, relief=GROOVE, bd=10, bg="#f07167",
fg="White")
        button4.place(x=670, y=440)

    def Exit(self):
        wayOut = messagebox.askyesno("Salary Calculation Window", "Do you
want to Exit the page?")
        if 'yes':
            self.destroy()
            return

    def calculate(self):
        emp_id = self.entry_id.get()
        name = self.entry_name.get()
        department = self.entry_dept.get()
        designation = self.entry_desg.get()
        pay = float(self.entry_pay.get())
        ta = float(pay * 0.075)
        bonus = float(self.entry_bonus1.get())
        tax = float(pay * 0.12)
        da = float(pay * 0.25)
        leaves = float(self.entry_leaves.get()) * 100
        pf = float((pay + da) * 0.12)
        net = float(pay + da + bonus + ta - pf - leaves)

        # transport (self.entry_transport)
        # p.f (self.entry_pf)
        # income tax (self.entry_tax)
        # dearness allowance (self.entry_da)
        # net pay (self.entry_net)

        change_text(self.entry_pf, pf)
        change_text(self.entry_tax, tax)
        change_text(self.entry_da, da)
        change_text(self.entry_net, net)
        change_text(self.entry_transport, ta)

        print(pay, emp_id, name, department, designation, ta, bonus, tax,
da, leaves, pf, net)

        '''Calculate the Gross Salary of an employee for following
allowance & deduction.
        Get Basic Salary of Employee,
        da = 25% of Basic,
        pf = 12% of Basic,
        ta = 7.50% of Basic.
        net = pay + da + ta + bonus - tax - leaves - pf
        '''
        if emp_id and name and department and designation and pay and ta
and bonus and tax and da and leaves and pf and net != "":
            try:
                # add to database
                query = "Insert into 'Salary'
(emp_id,name,department,designation,pay,ta,bonus,tax,da,leaves,pf,net)
values(?,?,?,?,?,?,?,?,?,?,?,?)"
                cur.execute(query, (emp_id, name, department, designation,
pay, ta, bonus, tax, da, leaves, pf, net))
                con.commit()
                messagebox.showinfo("Success", "Recorded Successfully")
```



```
                self.destroy()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "fill all the fields",
icon='warning')

        # print("SALARY PROGRAM")
        # name = str(input("Enter name of employee:"))
        # pay = float(input("Enter Basic Salary :"))
        # da = float(pay * 0.25)
        # pf = float((pay + da) * 0.12)
        # ta = float(pay * 0.075)
        # net = float(pay + da + ta - pf - leaves)

        # print("\n\n")
        print("S A L A R Y D E T A I L E D B R E A K U P ")
        print("==============================================")
        print(" NAME OF EMPLOYEE : ", name)
        print(" BASIC SALARY : ", pay)
        print(" DEARNESS ALLOW. : ", da)
        print(" TRAVEL ALLOW. : ", ta)
        print("==============================================")
        print(" NET SALARY PAY : ", net)
        print(" PROVIDENT FUND : ", pf)
        print("==============================================")

    def pay(self):
        self.entry_payslip_text.delete("1.0", END)
        self.entry_payslip_text.insert(END, "\t\tPay Slip\n\n")
        self.entry_payslip_text.insert(END, "Employee ID: " +
self.entry_id.get() + "\n\n")
        self.entry_payslip_text.insert(END,"Name: "  +
self.entry_name.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Department: " +
self.entry_dept.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Designation: " +
self.entry_desg.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Contact Number: " +
self.entry_phone.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Basic Pay: " +
self.entry_pay.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Transport Allowance: " +
self.entry_transport.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Income Tax: " +
self.entry_tax.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Dearness Allowance: " +
self.entry_da.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Incentive: " +
self.entry_bonus1.get() + "\n\n")
        self.entry_payslip_text.insert(END, "Leaves: " +
self.entry_leaves.get() + "\n\n")
        self.entry_payslip_text.insert(END, "P.F: " + self.entry_pf.get() +
"\n\n")
        self.entry_payslip_text.insert(END, "Net Pay: " +
self.entry_net.get() + "\n\n")

    def reset(self):
        change_text(self.entry_id, "")
        change_text(self.entry_name, "")
        change_text(self.entry_dept, "")
        change_text(self.entry_desg, "")
        change_text(self.entry_phone, "")
        change_text(self.entry_pay, "")
        change_text(self.entry_bonus1, "")
        change_text(self.entry_transport, "")
        change_text(self.entry_da, "")
        change_text(self.entry_pf, "")
        change_text(self.entry_leaves, "")
        change_text(self.entry_tax, "")
        change_text(self.entry_net, "")
        #change_text(self.entry_payslip_text, "")
        self.entry_payslip_text.delete("1.0", END)
```

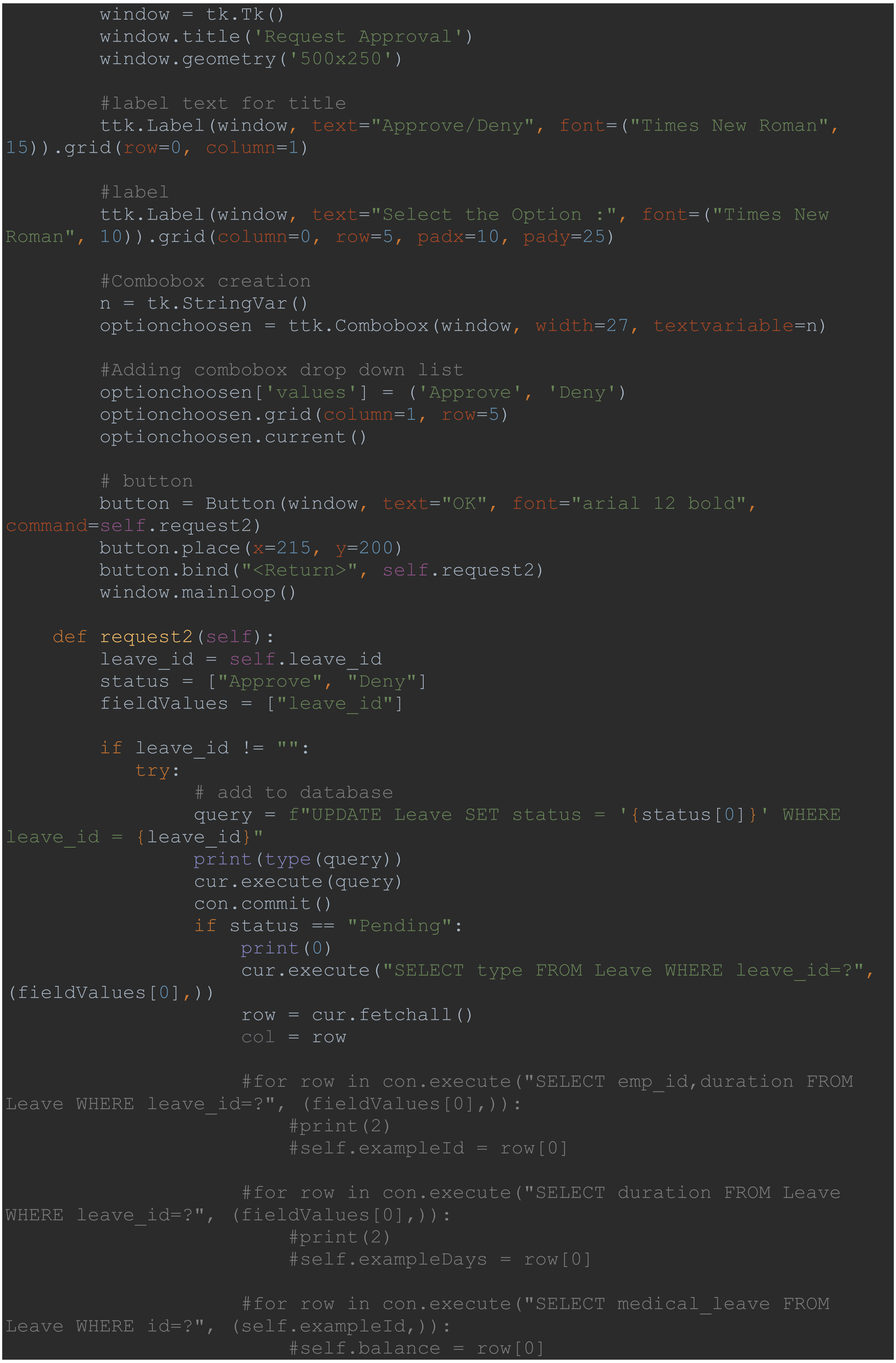
Regularise Attendance

```
from tkinter import *
from tkinter import messagebox
import datetime
date = datetime.datetime.now().date()
date = str(date)
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

class form(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("700x600")
        self.title("Regularise Attendance")
        self.resizable(False, False)

        #labels and buttons

        #frames
        self.top = Frame(self, height=100, bg='White')
        self.top.pack(fill=X)
        self.bottom = Frame(self, height=600, bg='#283618')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text=" Regularisation Request ", font='TimesNewRoman 24 bold', bg='#606c38', fg='White')
        self.heading.place(x=140, y=40)

        #id
        self.label_id = Label(self.bottom, text="Employee ID", font='arial 12 bold', bg='#283618', fg='White')
        self.label_id.place(x=170, y=40)

        self.entry_id = Entry(self.bottom, width=30, bd=4)
        self.entry_id.insert(0, "Enter Employee ID")
        self.entry_id.place(x=310, y=40)

        #name
        self.label_name = Label(self.bottom, text="Name", font='arial 12 bold', bg='#283618', fg='White')
        self.label_name.place(x=170, y=80)
        self.entry_name = Entry(self.bottom, width=30, bd=4)
        self.entry_name.insert(0, "Enter Name")
        self.entry_name.place(x=310, y=80)

        #department
        self.label_dept = Label(self.bottom, text="Department", font='arial
12 bold', bg='#283618', fg='White')
        self.label_dept.place(x=170, y=120)

        self.entry_dept = Entry(self.bottom, width=30, bd=4)
        self.entry_dept.insert(0, "Enter Department")
        self.entry_dept.place(x=310, y=120)

        #Date of Application
        self.label_DOA = Label(self.bottom, text="Date of Application:  " +
date, font='arial 12 bold', bg='#283618', fg='White')
        self.label_DOA.place(x=170, y=160)

        #reason
        self.label_reason = Label(self.bottom, text="Reason", font='arial
12 bold', bg='#283618', fg='White')
        self.label_reason.place(x=170, y=200)

        self.entry_reason = Entry(self.bottom, width=30, bd=4)
        self.entry_reason.insert(0, "Enter Reason")
        self.entry_reason.place(x=310, y=200)

        #Regularise Time
        self.label_time = Label(self.bottom, text="Time", font='arial 12
bold', bg='#283618', fg='White')
        self.label_time.place(x=170, y=240)

        self.entry_time = Entry(self.bottom, width=30, bd=4)
        self.entry_time.insert(0, "Enter Timings")
        self.entry_time.place(x=310, y=240)

        #button 1- Submit
        button = Button(self.bottom, text="Submit", font="arial 14 bold",
bg="#1b4332", fg="White", command=self.request, bd=8, relief=GROOVE,
width=14)
        button.place(x=220, y=350)
        #button 2- Cancel
        button = Button(self.bottom, text="Cancel", font="arial 14 bold",
command=self.destroy, bg="#1b4332", fg="White", bd=8, relief=GROOVE,
width=14)
        button.place(x=220, y=400)

    def request(self):
        emp_id = self.entry_id.get()
        name = self.entry_name.get()
        department = self.entry_dept.get()
        reason = self.entry_reason.get()
        status = "Pending"
        time = self.entry_time.get()

        if emp_id and name and department and reason and status and time !=
"":
            try:
                #add to database
                    query = "Insert into 'Leave'
```

```
(emp_id,reason,status,time) values(?,?,?,?)"
                    cur.execute(query, (emp_id,reason,status,time))
                    con.commit()
                    messagebox.showinfo("Regularisation Form", "Request
Applied")
                    self.destroy()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "Fill all the fields",
icon='warning')
```

Approve Request

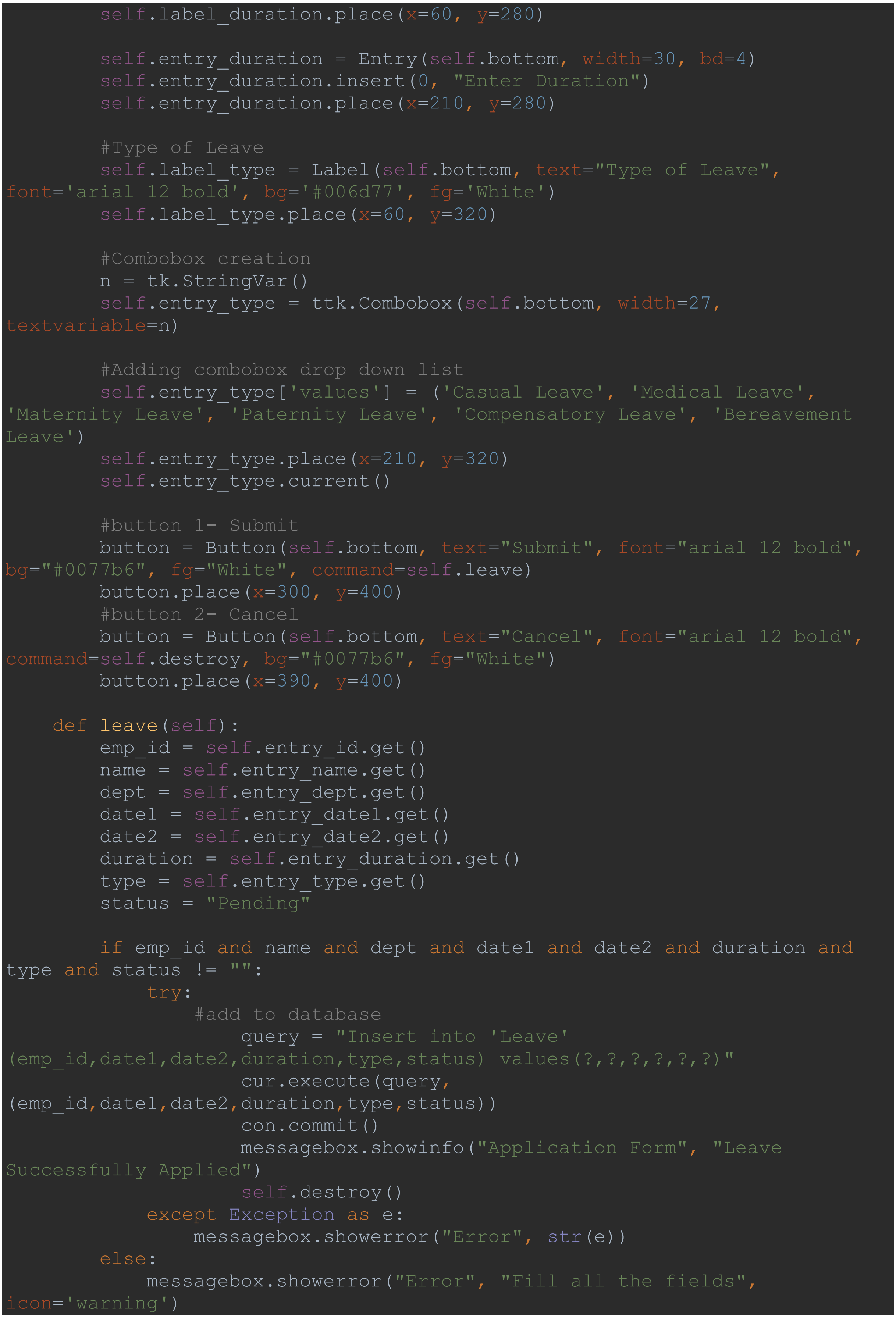
```
from tkinter import *
import tkinter as tk
from tkinter import ttk
import sqlite3
from tkinter import messagebox
#we can code without using self in bottom cases
con = sqlite3.connect('db1.db')
cur = con.cursor()

class Approve_Request(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        # exampleID is the id of the Employee who is requesting the leave
        self.exampleID = 101

        self.geometry("500x300+550+250")
        self.title('Approve Request')
        self.resizable(False,False)

        #frame
        self.frame = Frame(self, height=300, bg='#ecf39e')
        self.frame.pack(fill=X)

        #label and button

        #leave id
        self.label_leave_id = Label(self.frame, text="Leave ID",
font='arial 16 bold', fg='#e71d36', bg='#ecf39e')
        self.label_leave_id.place(x=40, y=40)

        self.entry_leave_id = Entry(self.frame, width=30, bd=4)
        self.entry_leave_id.insert(0, "Enter Leave ID")
        self.entry_leave_id.place(x=150, y=40)

        #button
        button = Button(self.frame, text="Next", font="arial 12 bold",
command=self.request1)
        button.place(x=215, y=200)
        button.bind("<Return>", self.request1)

    def request1(self, event=' '):

        self.leave_id = self.entry_leave_id.get()
```



```
        window = tk.Tk()
        window.title('Request Approval')
        window.geometry('500x250')

        #label text for title
        ttk.Label(window, text="Approve/Deny", font=("Times New Roman",
15)).grid(row=0, column=1)

        #label
        ttk.Label(window, text="Select the Option :", font=("Times New
Roman", 10)).grid(column=0, row=5, padx=10, pady=25)

        #Combobox creation
        n = tk.StringVar()
        optionchoosen = ttk.Combobox(window, width=27, textvariable=n)

        #Adding combobox drop down list
        optionchoosen['values'] = ('Approve', 'Deny')
        optionchoosen.grid(column=1, row=5)
        optionchoosen.current()

        # button
        button = Button(window, text="OK", font="arial 12 bold",
command=self.request2)
        button.place(x=215, y=200)
        button.bind("<Return>", self.request2)
        window.mainloop()

    def request2(self):
        leave_id = self.leave_id
        status = ["Approve", "Deny"]
        fieldValues = ["leave_id"]

        if leave_id != "":
           try:
                # add to database
                query = f"UPDATE Leave SET status = '{status[0]}' WHERE
leave_id = {leave_id}"
                print(type(query))
                cur.execute(query)
                con.commit()
                if status == "Pending":
                    print(0)
                    cur.execute("SELECT type FROM Leave WHERE leave_id=?",
(fieldValues[0],))
                    row = cur.fetchall()
                    col = row

                    #for row in con.execute("SELECT emp_id,duration FROM
Leave WHERE leave_id=?", (fieldValues[0],)):
                        #print(2)
                        #self.exampleId = row[0]

                    #for row in con.execute("SELECT duration FROM Leave
WHERE leave_id=?", (fieldValues[0],)):
                        #print(2)
                        #self.exampleDays = row[0]

                    #for row in con.execute("SELECT medical_leave FROM
Leave WHERE id=?", (self.exampleId,)):
                        #self.balance = row[0]
                        #print(self.balance)

                    #for row in con.execute("SELECT casual_leave FROM 
Employee WHERE id=?", (self.exampleId,)):
                        #balance1 = row[0]
                        #print(balance1)

                    #if (col[0] == ('medicalleave',)):
                        #print(3)
                        #con.execute("UPDATE Leave SET medical_leave =? 
WHERE emp_id= ?", ((self.balance - self.exampleDays), (self.exampleId)))

                    #if (col[0] == ('casualleave',)):
                        #print(3)
                        #con.execute("UPDATE Leave SET casual_leave =? 
WHERE id= ?", ((self.balance - self.exampleDays), (self.exampleId)))

                messagebox.showinfo("Success", "Regularisation Request 
approved and updated")
                self.destroy()
           except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "Wrong Record", icon='warning')
```

Apply Leave

```
from tkinter import *
from tkinter import messagebox
import tkinter as tk
from tkinter import ttk
import datetime
date = datetime.datetime.now().date()
date = str(date)
import sqlite3

con = sqlite3.connect('db1.db')
cur = con.cursor()

class Submit_Leave(Toplevel):
    def __init__(self, emp_id):
        Toplevel.__init__(self)

        self.geometry("700x600")
        self.title("Leave Application")
        self.resizable(False, False)

        self.login = emp_id

        #labels and buttons

        #frames
        self.top = Frame(self, height=100, bg='White')
        self.top.pack(fill=X)
        self.bottom = Frame(self, height=600, bg='#006d77')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text=" Application Form ",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)


        query = cur.execute("SELECT emp_id, name, department FROM Employee
WHERE emp_id = ? ",(self.login,))
        data = query.fetchone()

        #id
        self.label_id = Label(self.bottom, text="Employee ID", font='arial
12 bold', bg='#006d77', fg='White')
        self.label_id.place(x=60, y=40)

        self.entry_id = Entry(self.bottom, width=30, bd=4)
        self.entry_id.insert(0, data[0])
        self.entry_id.place(x=210, y=40)

        #name
        self.label_name = Label(self.bottom, text="Name", font='arial 12
bold', bg='#006d77', fg='White')
        self.label_name.place(x=60, y=80)

        self.entry_name = Entry(self.bottom, width=30, bd=4)
        self.entry_name.insert(0, data[1])
        self.entry_name.place(x=210, y=80)

        #department
        self.label_dept = Label(self.bottom, text="Department", font='arial
12 bold', bg='#006d77', fg='White')
        self.label_dept.place(x=60, y=120)

        self.entry_dept = Entry(self.bottom, width=30, bd=4)
        self.entry_dept.insert(0, data[2])
        self.entry_dept.place(x=210, y=120)

        #Date of Application
        self.label_DOA = Label(self.bottom, text="Date of Application:  " +
date, font='arial 12 bold', bg='#006d77', fg='White')
        self.label_DOA.place(x=60, y=160)

        #date1
        self.label_date1 = Label(self.bottom, text="Date1(Start)",
font='arial 12 bold', bg='#006d77', fg='White')
        self.label_date1.place(x=60, y=200)

        self.entry_date1 = Entry(self.bottom, width=30, bd=4)
        self.entry_date1.insert(0, "Enter Date")
        self.entry_date1.place(x=210, y=200)

        #date2
        self.label_date2 = Label(self.bottom, text="Date2(End)",
font='arial 12 bold', bg='#006d77', fg='White')
        self.label_date2.place(x=60, y=240)

        self.entry_date2 = Entry(self.bottom, width=30, bd=4)
        self.entry_date2.insert(0, "Enter Date")
        self.entry_date2.place(x=210, y=240)

        # Duration of Leave
        self.label_duration = Label(self.bottom, text="Duration of Leave",
font='arial 12 bold', bg='#006d77', fg='White')
```



```
        self.label_duration.place(x=60, y=280)

        self.entry_duration = Entry(self.bottom, width=30, bd=4)
        self.entry_duration.insert(0, "Enter Duration")
        self.entry_duration.place(x=210, y=280)

        #Type of Leave
        self.label_type = Label(self.bottom, text="Type of Leave",
font='arial 12 bold', bg='#006d77', fg='White')
        self.label_type.place(x=60, y=320)

        #Combobox creation
        n = tk.StringVar()
        self.entry_type = ttk.Combobox(self.bottom, width=27,
textvariable=n)

        #Adding combobox drop down list
        self.entry_type['values'] = ('Casual Leave', 'Medical Leave',
'Maternity Leave', 'Paternity Leave', 'Compensatory Leave', 'Bereavement
Leave')
        self.entry_type.place(x=210, y=320)
        self.entry_type.current()

        #button 1- Submit
        button = Button(self.bottom, text="Submit", font="arial 12 bold",
bg="#0077b6", fg="White", command=self.leave)
        button.place(x=300, y=400)
        #button 2- Cancel
        button = Button(self.bottom, text="Cancel", font="arial 12 bold",
command=self.destroy, bg="#0077b6", fg="White")
        button.place(x=390, y=400)

    def leave(self):
        emp_id = self.entry_id.get()
        name = self.entry_name.get()
        dept = self.entry_dept.get()
        date1 = self.entry_date1.get()
        date2 = self.entry_date2.get()
        duration = self.entry_duration.get()
        type = self.entry_type.get()
        status = "Pending"

        if emp_id and name and dept and date1 and date2 and duration and
type and status != "":
            try:
                #add to database
                    query = "Insert into 'Leave'
(emp_id,date1,date2,duration,type,status) values(?,?,?,?,?,?)"
                    cur.execute(query,
(emp_id,date1,date2,duration,type,status))
                    con.commit()
                    messagebox.showinfo("Application Form", "Leave
Successfully Applied")
                    self.destroy()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "Fill all the fields",
icon='warning')
```

Approve Leave

```
from tkinter import *
import tkinter as tk
from tkinter import ttk
import sqlite3
from tkinter import messagebox
#we can code without using self in bottom cases
con = sqlite3.connect('db1.db')
cur = con.cursor()

class Approve_Leave(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        # exampleID is the id of the Employee who is requesting the leave
        self.exampleID = 101

        self.geometry("500x300+550+250")
        self.title('Approve Leave')
        self.resizable(False,False)

        #frame
        self.frame = Frame(self, height=300, bg='#ccdbfd')
        self.frame.pack(fill=X)

        #label and button

        #leave id
        self.label_leave_id = Label(self.frame, text="Leave ID",
font='arial 16 bold', fg='#e71d36', bg='#ccdbfd')
        self.label_leave_id.place(x=40, y=40)

        self.entry_leave_id = Entry(self.frame, width=30, bd=4)
        self.entry_leave_id.insert(0, "Enter Leave ID")
        self.entry_leave_id.place(x=150, y=40)

        #button
        button = Button(self.frame, text="Next", font="arial 12 bold",
command=self.approval)
        button.place(x=215, y=200)
        button.bind("<Return>", self.approval)

    def approval(self, event=' '):

        self.leave_id = self.entry_leave_id.get()

        window = tk.Tk()
        window.title('Leave Approval')
        window.geometry('500x250')

        #label text for title
        ttk.Label(window, text="Approve/Deny", font=("Times New Roman",
15)).grid(row=0, column=1)

        #label
        ttk.Label(window, text="Select the Option :", font=("Times New
Roman", 10)).grid(column=0, row=5, padx=10, pady=25)

        #Combobox creation
        n = tk.StringVar()
        optionchoosen = ttk.Combobox(window, width=27, textvariable=n)

        #Adding combobox drop down list
        optionchoosen['values'] = ('Approve', 'Deny')
        optionchoosen.grid(column=1, row=5)
        optionchoosen.current()

        #button
        button = Button(window, text="OK", font="arial 12 bold",
command=self.appt)
        button.place(x=215, y=200)
        button.bind("<Return>", self.appt)
        window.mainloop()

    def appt(self):
        leave_id = self.leave_id
        status = ["Approve", "Deny"]
        fieldValues = ["leave_id"]

        if leave_id != "":
           try:
                #add to database
                query = f"UPDATE Leave SET status = '{status[0]}' WHERE
leave_id = {leave_id}"
                print(type(query))
                cur.execute(query)
                con.commit()
                if status == "Pending":
                    print(0)
                    cur.execute("SELECT type FROM Leave WHERE leave_id=?",
(fieldValues[0],))
                    row = cur.fetchall()
                    col = row

                    for row in con.execute("SELECT emp_id,duration FROM
Leave WHERE leave_id=?", (fieldValues[0],)):
                        print(2)
                        self.exampleId = row[0]

                    for row in con.execute("SELECT duration FROM Leave
WHERE leave_id=?", (fieldValues[0],)):
                        print(2)
                        self.exampleDays = row[0]

                    for row in con.execute("SELECT medical_leave FROM Leave
WHERE emp_id=?", (self.exampleId,)):
                        self.balance = row[0]
                        print(self.balance)

                    for row in con.execute("SELECT casual_leave FROM Leave
WHERE emp_id=?", (self.exampleId,)):
                        balance1 = row[0]
                        print(balance1)

                    if (col[0] == ('medicalleave',)):
                        print(3)
                        con.execute("UPDATE Leave SET medical_leave =?
WHERE emp_id= ?", ((self.balance - self.exampleDays), (self.exampleId)))

                    if (col[0] == ('casualleave',)):
                        print(3)
                        con.execute("UPDATE Leave SET casual_leave =? WHERE
emp_id= ?", ((self.balance - self.exampleDays), (self.exampleId)))

                messagebox.showinfo("Success", "Leave approved and
updated")
                self.destroy()
           except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "Wrong Record", icon='warning')
```

Attendance

```
from tkinter import *
from AttendanceStatus import Attendance_Status
import sqlite3
#we can code without using self in bottom cases
con = sqlite3.connect('db1.db')
cur = con.cursor()

class attendance(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("500x300+550+250")
        self.title('View Attendance')
        self.resizable(False,False)

        #frame
        self.frame = Frame(self, height=300, bg='#ccdbfd')
        self.frame.pack(fill=X)

        #label and button

        #id
        self.label_id = Label(self.frame, text="Employee ID ", font='arial
16 bold', fg='#e71d36', bg='#ccdbfd')
        self.label_id.place(x=20, y=40)

        self.entry_id = Entry(self.frame, width=30, bd=4)
        self.entry_id.insert(0, "Enter Employee ID")
        self.entry_id.place(x=150, y=40)

        #password
        self.label_password = Label(self.frame, text="Password ",
font='arial 16 bold', fg='#e71d36', bg='#ccdbfd')
        self.label_password.place(x=40, y=80)

        self.entry_password = Entry(self.frame, width=30, bd=4)
        self.entry_password.insert(0, "Enter Password")
        self.entry_password.config(show="*")
        self.entry_password.place(x=150, y=80)

        #button
        button = Button(self.frame, text="OK", font="arial 12 bold",
command=self.attendancestatus)
        button.place(x=215, y=200)
    def attendancestatus(self):
        window = Attendance_Status()
```

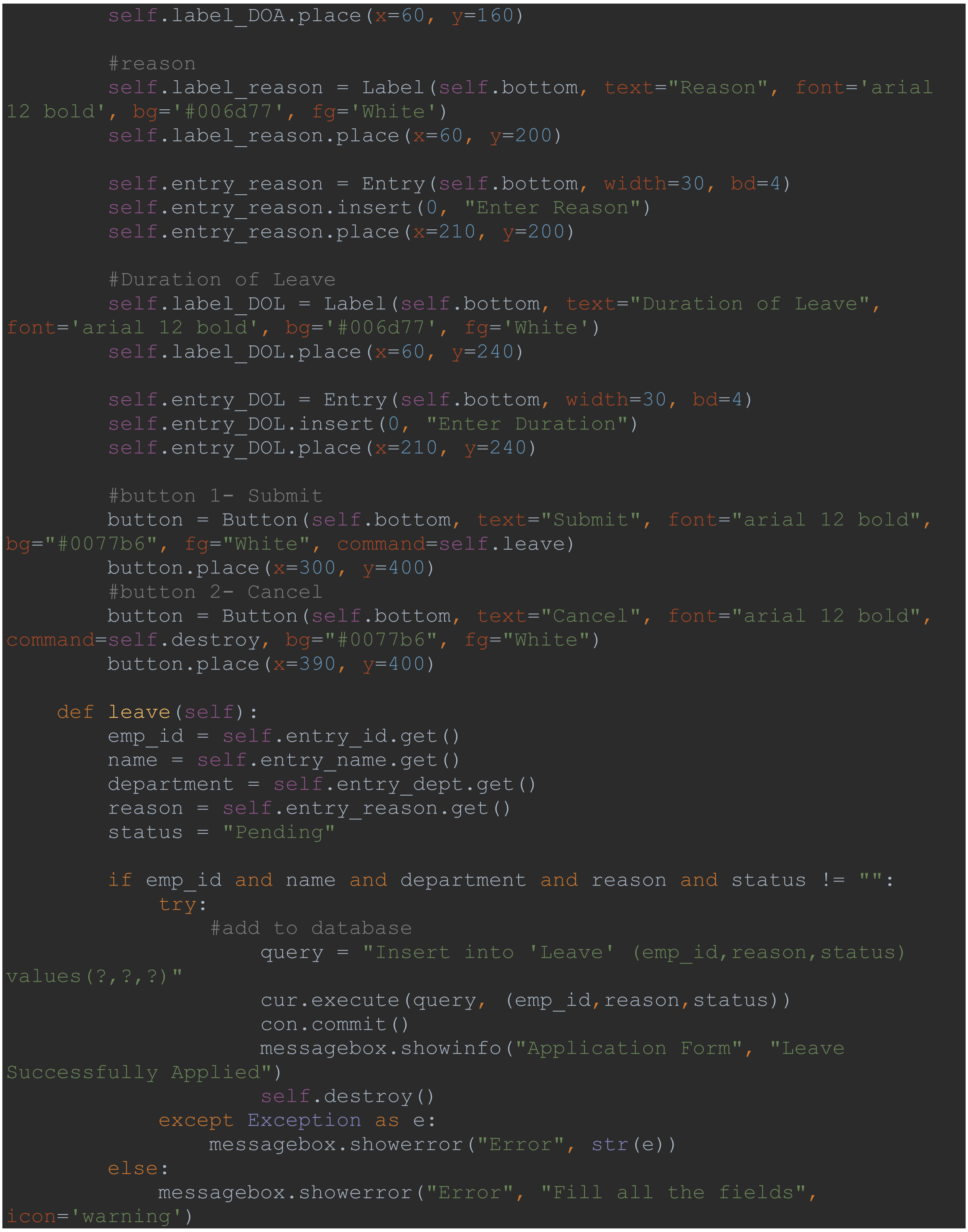
Attendance Status

```
from tkinter import *
import datetime
date = datetime.datetime.now().date()
date = str(date)
import sqlite3
#we can code without using self in bottom cases
con = sqlite3.connect('db1.db')
cur = con.cursor()

class Attendance_Status(Toplevel):
    def __init__(self):
        Toplevel.__init__(self)

        self.geometry("500x300+550+250")
        self.title('Attendance Status')
        self.resizable(False,False)

        self.login = 101

        #frame
        self.frame = Frame(self, height=300, bg='#ccdbfd')
        self.frame.pack(fill=X)

        #label and button

        query = cur.execute("SELECT attendance FROM Employee WHERE emp_id =
? ",(self.login,))
        data = query.fetchone()

        #date
        self.label_date = Label(self.frame, text="Date      " + date,
font='arial 16 bold', fg='#e71d36', bg='#ccdbfd')
        self.label_date.place(x=40, y=40)

        #password
        self.label_record = Label(self.frame, text="Record", font='arial 16
bold', fg='#e71d36', bg='#ccdbfd')
        self.label_record.place(x=40, y=80)

        self.entry_record = Entry(self.frame, width=30, bd=4)
        self.entry_record.insert(0, data[0])
        self.entry_record.place(x=150, y=80)
```

Short Leave

```
from tkinter import *
from tkinter import messagebox
import datetime
date = datetime.datetime.now().date()
date = str(date)
import sqlite3
```

```
con = sqlite3.connect('db1.db')
cur = con.cursor()

class Short_Leave(Toplevel):
    def __init__(self, emp_id):
        Toplevel.__init__(self)

        self.geometry("700x600")
        self.title("Leave Application")
        self.resizable(False, False)

        self.login = emp_id

        #labels and buttons

        #frames
        self.top = Frame(self, height=100, bg='White')
        self.top.pack(fill=X)
        self.bottom = Frame(self, height=600, bg='#006d77')
        self.bottom.pack(fill=X)

        #top frame design
        self.heading = Label(self.top, text=" Apply Short Leave ",
font='TimesNewRoman 24 bold', bg='#0077b6', fg='White')
        self.heading.place(x=230, y=40)


        query = cur.execute("SELECT emp_id, name, department FROM Employee
WHERE emp_id = ? ",(self.login,))
        data = query.fetchone()

        #id
        self.label_id = Label(self.bottom, text="Employee ID", font='arial
12 bold', bg='#006d77', fg='White')
        self.label_id.place(x=60, y=40)

        self.entry_id = Entry(self.bottom, width=30, bd=4)
        self.entry_id.insert(0, data[0])
        self.entry_id.place(x=210, y=40)

        #name
        self.label_name = Label(self.bottom, text="Name", font='arial 12
bold', bg='#006d77', fg='White')
        self.label_name.place(x=60, y=80)

        self.entry_name = Entry(self.bottom, width=30, bd=4)
        self.entry_name.insert(0, data[1])
        self.entry_name.place(x=210, y=80)

        #department
        self.label_dept = Label(self.bottom, text="Department", font='arial
12 bold', bg='#006d77', fg='White')
        self.label_dept.place(x=60, y=120)

        self.entry_dept = Entry(self.bottom, width=30, bd=4)
        self.entry_dept.insert(0, data[2])
        self.entry_dept.place(x=210, y=120)

        #Date of Application
        self.label_DOA = Label(self.bottom, text="Date of Application:  " +
date, font='arial 12 bold', bg='#006d77', fg='White')
```

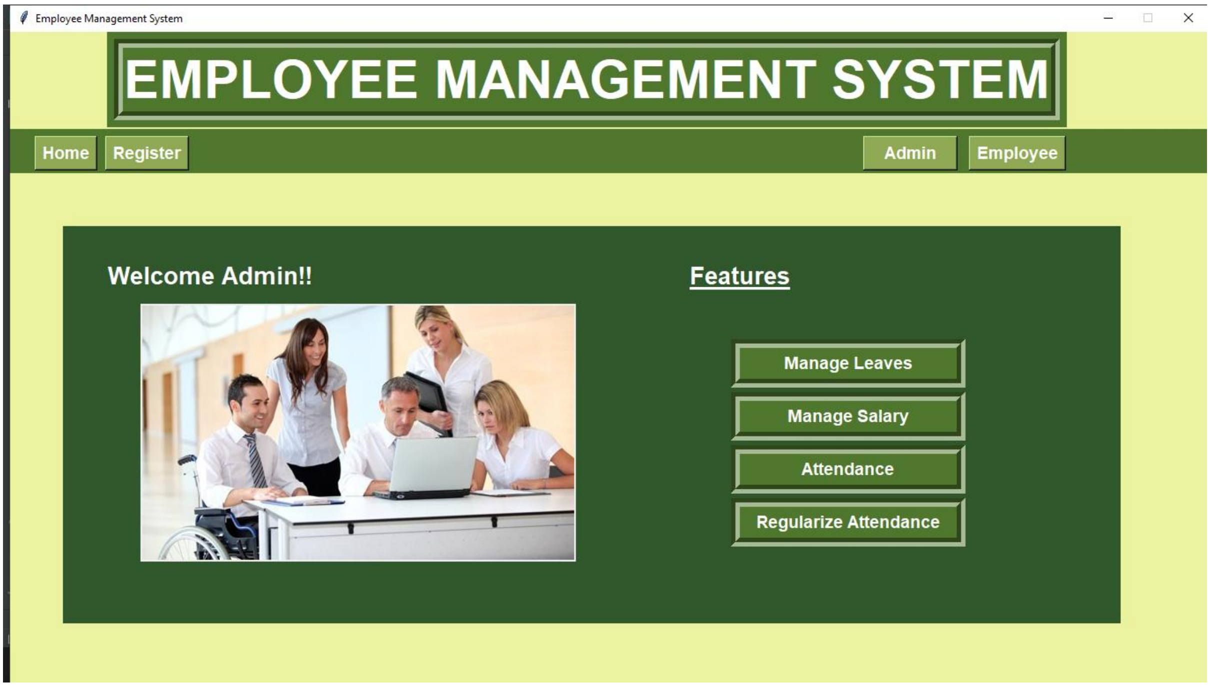


```
        self.label_DOA.place(x=60, y=160)

        #reason
        self.label_reason = Label(self.bottom, text="Reason", font='arial
12 bold', bg='#006d77', fg='White')
        self.label_reason.place(x=60, y=200)

        self.entry_reason = Entry(self.bottom, width=30, bd=4)
        self.entry_reason.insert(0, "Enter Reason")
        self.entry_reason.place(x=210, y=200)

        #Duration of Leave
        self.label_DOL = Label(self.bottom, text="Duration of Leave",
font='arial 12 bold', bg='#006d77', fg='White')
        self.label_DOL.place(x=60, y=240)

        self.entry_DOL = Entry(self.bottom, width=30, bd=4)
        self.entry_DOL.insert(0, "Enter Duration")
        self.entry_DOL.place(x=210, y=240)

        #button 1- Submit
        button = Button(self.bottom, text="Submit", font="arial 12 bold",
bg="#0077b6", fg="White", command=self.leave)
        button.place(x=300, y=400)
        #button 2- Cancel
        button = Button(self.bottom, text="Cancel", font="arial 12 bold",
command=self.destroy, bg="#0077b6", fg="White")
        button.place(x=390, y=400)

    def leave(self):
        emp_id = self.entry_id.get()
        name = self.entry_name.get()
        department = self.entry_dept.get()
        reason = self.entry_reason.get()
        status = "Pending"

        if emp_id and name and department and reason and status != "":
            try:
                #add to database
                    query = "Insert into 'Leave' (emp_id,reason,status)
values(?,?,?)"
                    cur.execute(query, (emp_id,reason,status))
                    con.commit()
                    messagebox.showinfo("Application Form", "Leave
Successfully Applied")
                    self.destroy()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showerror("Error", "Fill all the fields",
icon='warning')
```

FEATURES OF THE PROJECT

The ‘Employee Management System’ is a project that is designed using the Python and SQLite (as the front end and backend technologies). has the four different modules named as: -

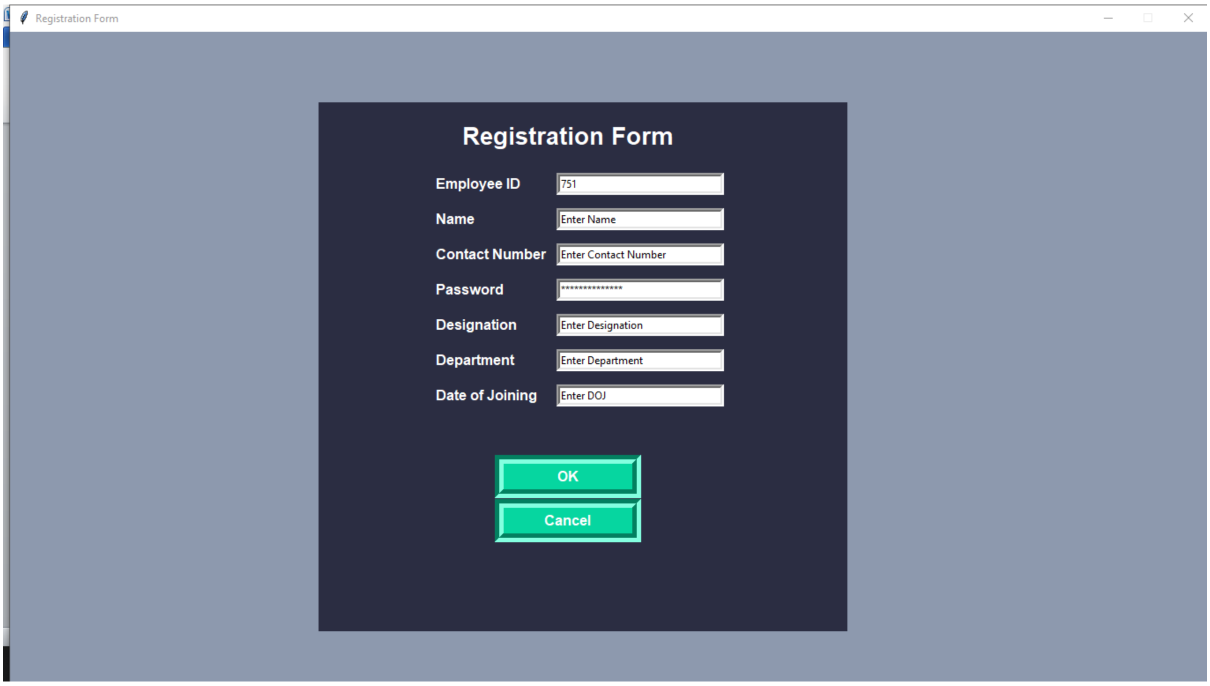
- Manage Leaves
- Manage Salary
- Manage Attendance
- Regularise Attendance



So, when the user wants to access the system and he/she clicks on the icon then, the Main window named as the ‘Employee Management System’ will open in front him/her. From there he/she can have the access to the system according to his/her requirement of data to work on.

REGISTRATION FORM: -

The user can access the system only after he/she has got himself/herself registered by filling up the details in the registration form which is as under:



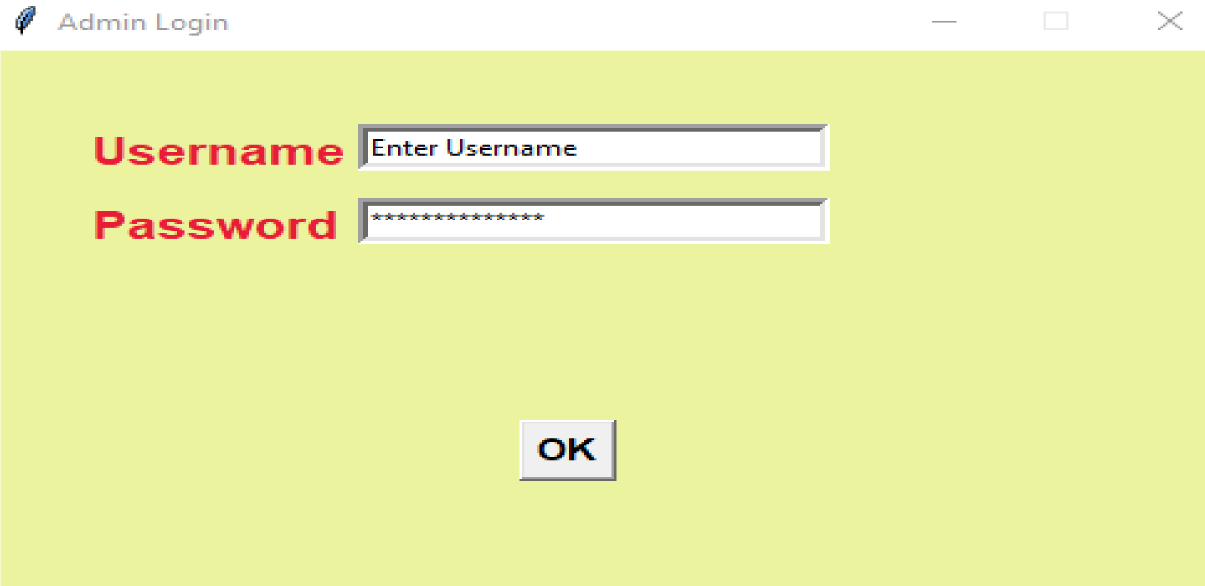
Registration Form

Employee ID	751
Name	Enter Name
Contact Number	Enter Contact Number
Password	*************
Designation	Enter Designation
Department	Enter Department
Date of Joining	Enter DOJ

After the successful registration with the particular credentials the user/employee can use that particular ID and Password in order to access their respective panel.

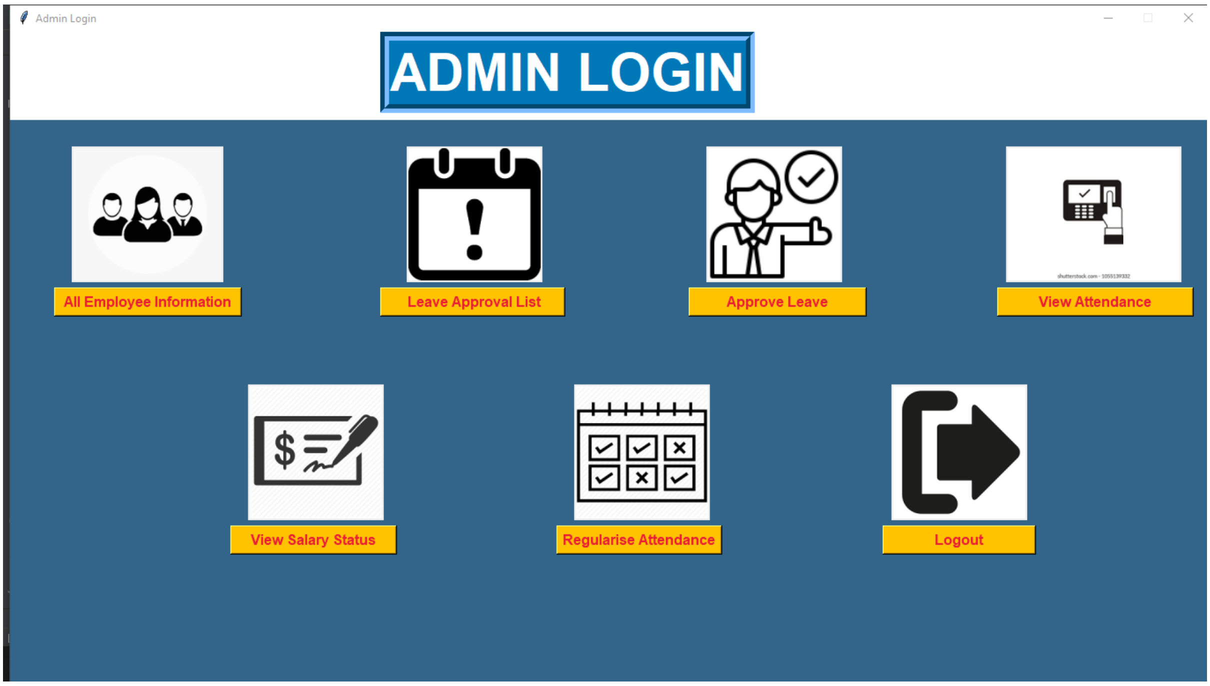
ADMIN LOGIN: -

Herein, the system consists of the separate Login Panel for the Admin Staff. They can have access to their panel once their Login Credentials have been matched and validated.



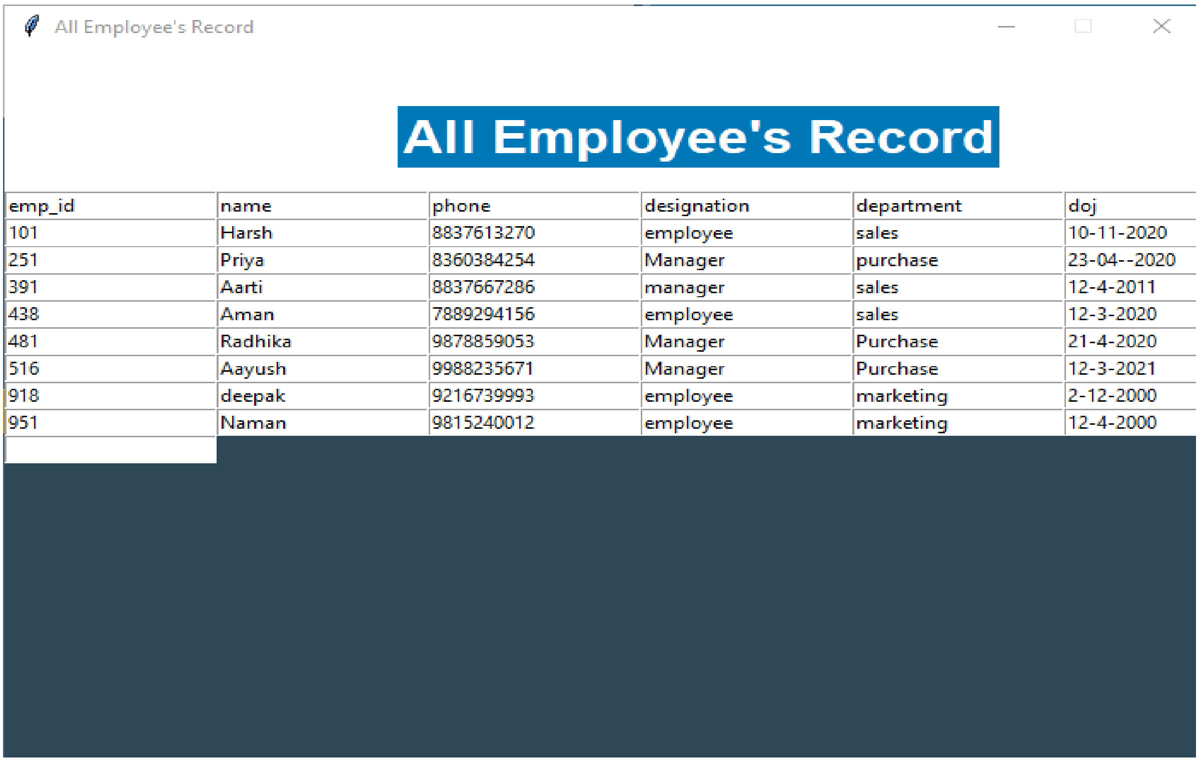
Username	Enter Username
Password	***************

This is the window that will appear where in the user from the Admin Staff will be required to fill in the login details and if the same gets validated then he/she can get the access to the panel which is shown as under:



The admin can access through the following options and make the necessary computations as per the requirement.

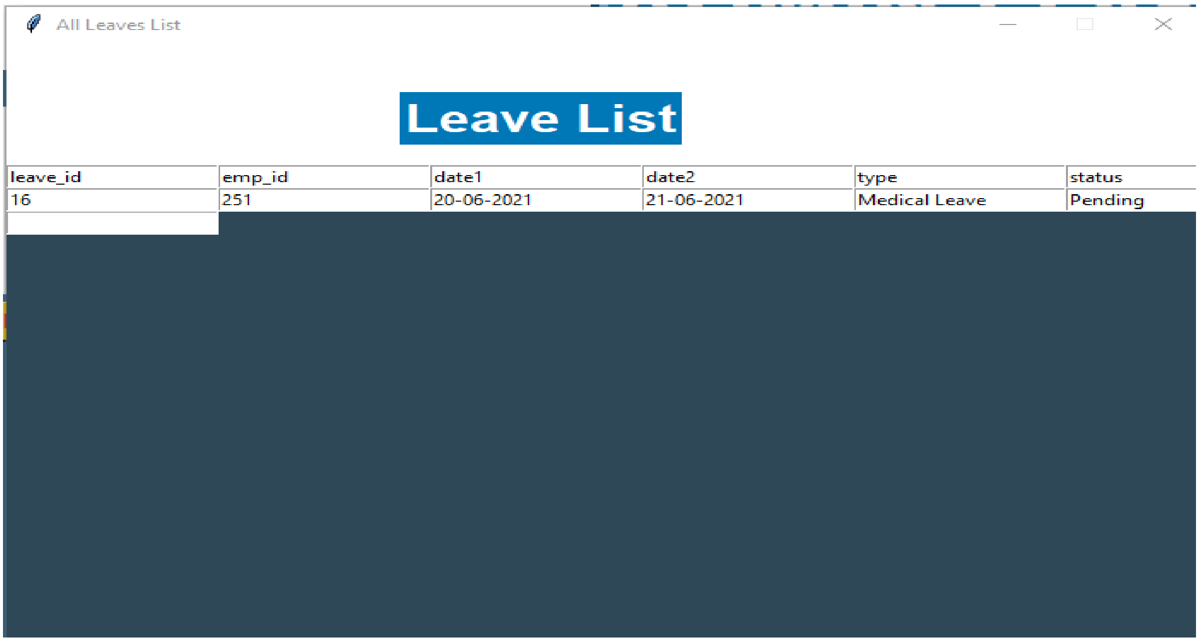
When he/she clicks on the ‘All Employee Information’ button he/she can get the details of all the employees who are working in the organisation.



All Employee's Record

emp_id	name	phone	designation	department	doj
101	Harsh	8837613270	employee	sales	10-11-2020
251	Priya	8360384254	Manager	purchase	23-04--2020
391	Aarti	8837667286	manager	sales	12-4-2011
438	Aman	7889294156	employee	sales	12-3-2020
481	Radhika	9878859053	Manager	Purchase	21-4-2020
516	Aayush	9988235671	Manager	Purchase	12-3-2021
918	deepak	9216739993	employee	marketing	2-12-2000
951	Naman	9815240012	employee	marketing	12-4-2000

The second that he/she can access through the panel is of the ‘All Leaves List’ which will help him to get the details about all the employees who have applied for the leaves and want to get them approved.



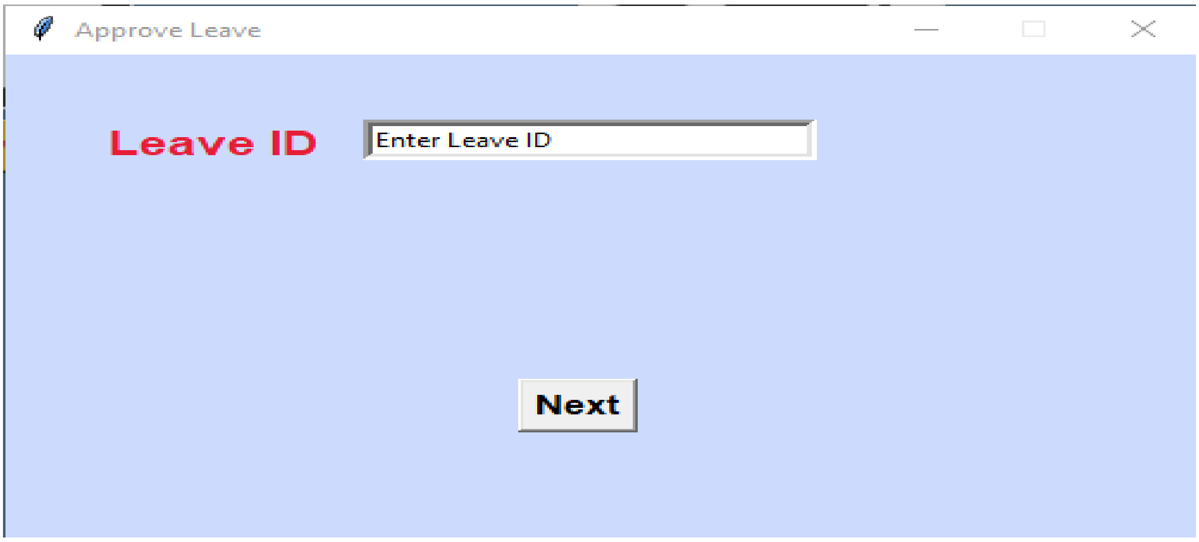
Leave List

leave_id	emp_id	date1	date2	type	status
16	251	20-06-2021	21-06-2021	Medical Leave	Pending

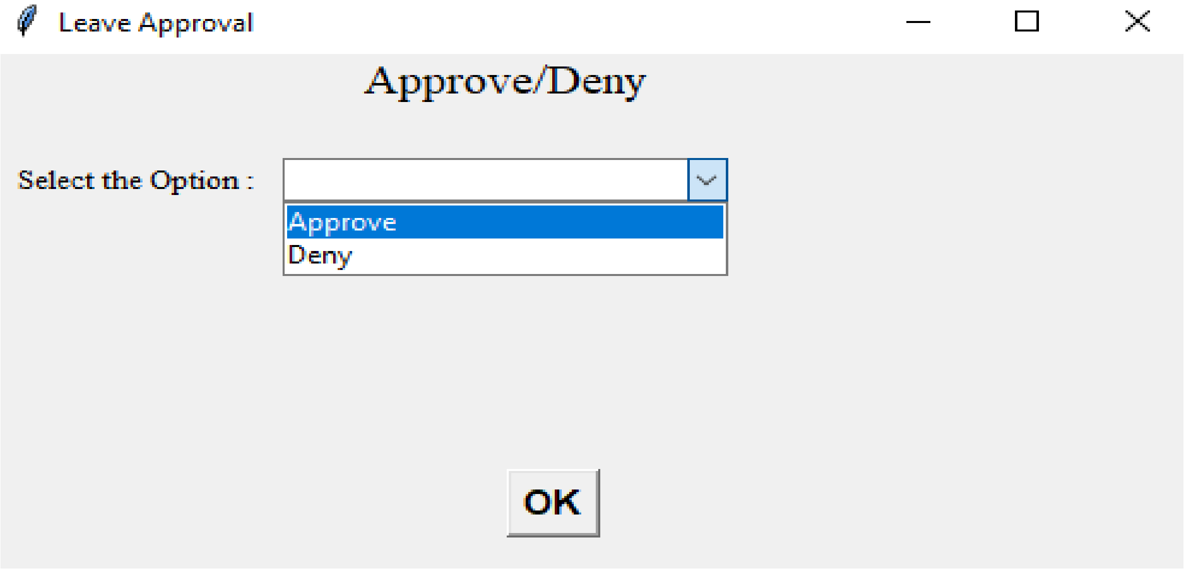
Once the leave for the particular ID has been approved, it will be removed from this list.

Further, he/she has the access to approve the leaves that are being applied by the employees.

In this case, the admin need sto click on the ‘Approve Leave’ button where they will have one dialog box open, which is as under:

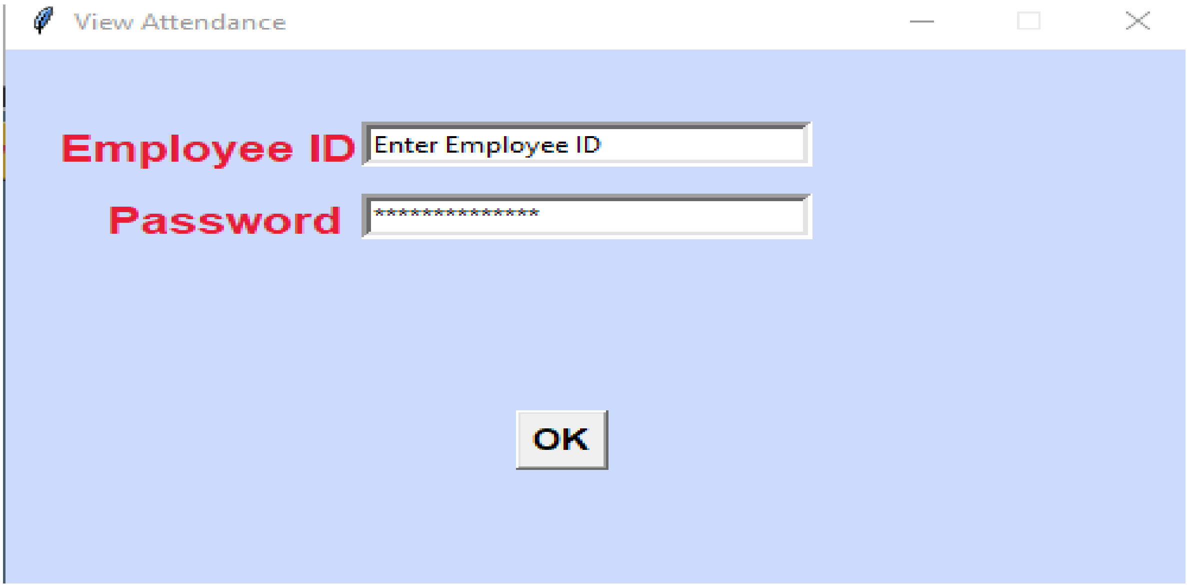


The admin will need to fill the leave_id that has been assigned to the particular leave that has been applied by the employee. After filling up the leave_id he/she needs to click on the ‘Next’ button which would direct him to the next dialog box, shown as under. Here, he/she can select the option to either approve/deny the leave for the particular leave_id that is being entered.



The same approval/denial of the leave will be updated in the 'Employee Login' of that particular employee.

Next, he/she can have the view of the attendance of the employee that he/she wishes to see for that particular date/day. This can be done by clicking on the 'View Attendance' button. After clicking on this button the following dialog box will appear which would ask for the Employee ID and Password for whom the detail of attendance needs to be checked.

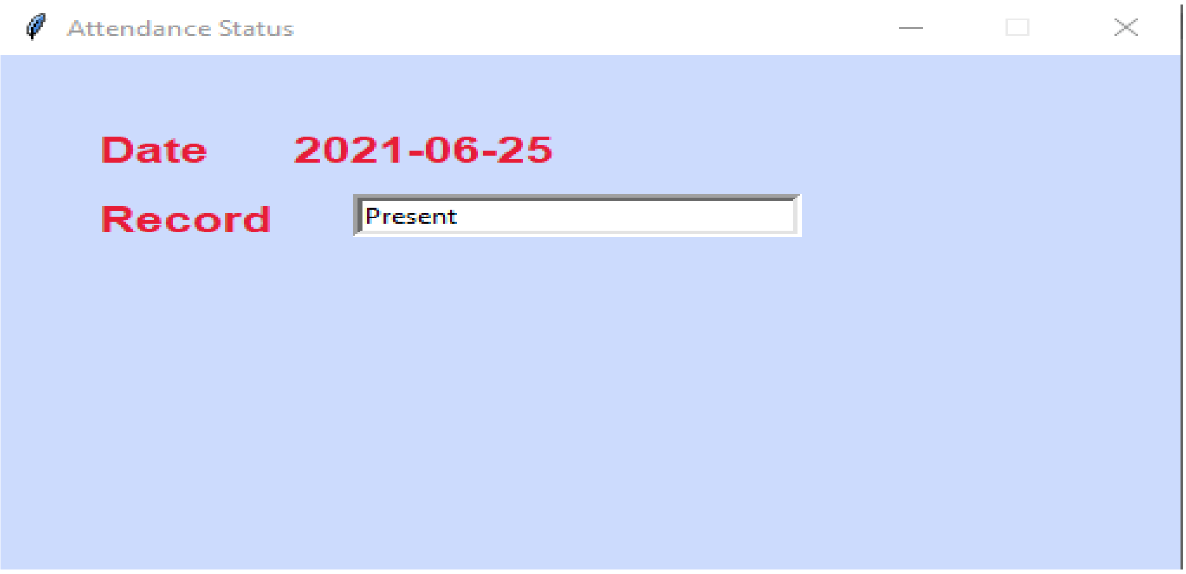


View Attendance

Employee ID: Enter Employee ID

Password: ***************

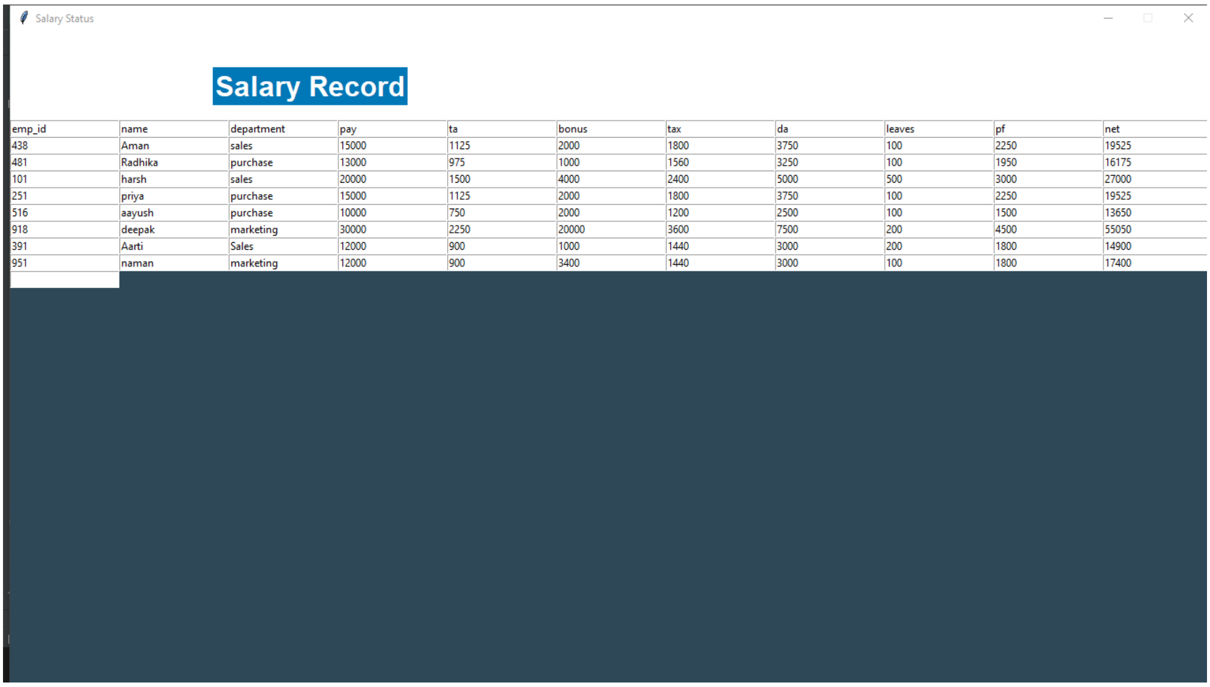
After the details are filled by the admin, a new dialog box will appear which would show the attendance status, whether he is present/absent on that day.



Date 2021-06-25

Record Present

Next he/she can have the access to the Salary Status of the employees by clicking on the 'View Salary Status' button.

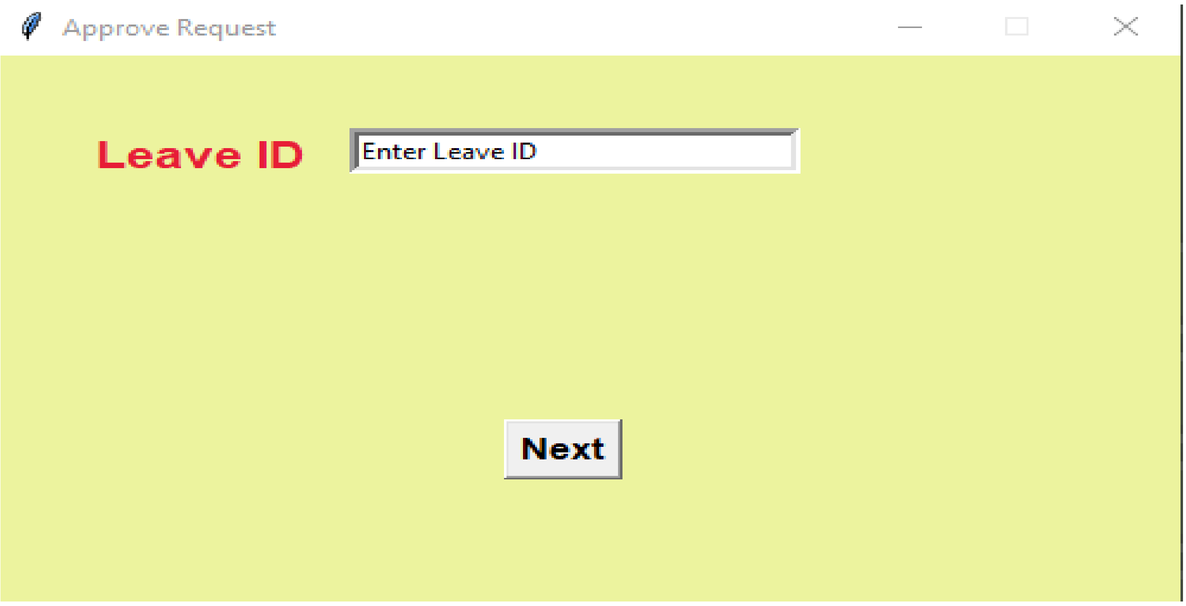


Salary Record

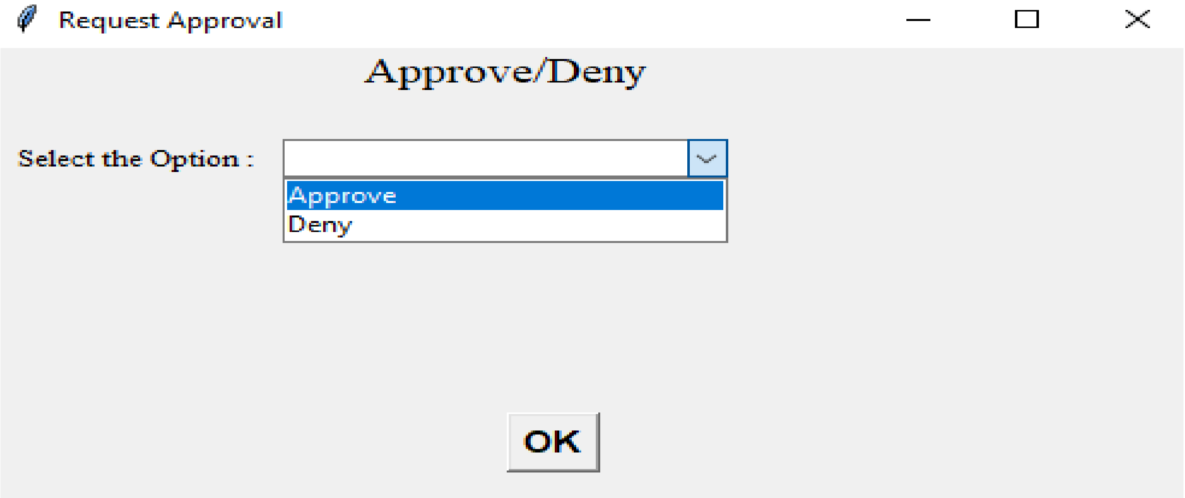
emp_id	name	department	pay	ta	bonus	tax	da	leaves	pf	net
438	Aman	sales	15000	1125	2000	1800	3750	100	2250	19525
481	Radhika	purchase	13000	975	1000	1560	3250	100	1950	16175
101	harsh	sales	20000	1500	4000	2400	5000	500	3000	27000
251	priya	purchase	15000	1125	2000	1800	3750	100	2250	19525
516	aayush	purchase	10000	750	2000	1200	2500	100	1500	13650
918	deepak	marketing	30000	2250	20000	3600	7500	200	4500	55050
391	Aarti	Sales	12000	900	1000	1440	3000	200	1800	14900
951	naman	marketing	12000	900	3400	1440	3000	100	1800	17400

Herein, the admin can get the view for those employees to whom the salary has been provided with the details of amounts and bifurcations.

Also, the admin has the access to approve the Regularization Request that has been applied by the employee. All he/she needs to do is to fill up the leave_id which would act here as the request_id.



After the id is filled he/she needs to click on the ‘Next’ button, where they can get the option of the approval/denial of the request. But the condition for the approval of the Regularization Request is that the reason should be a genuine one only then it will be answered.

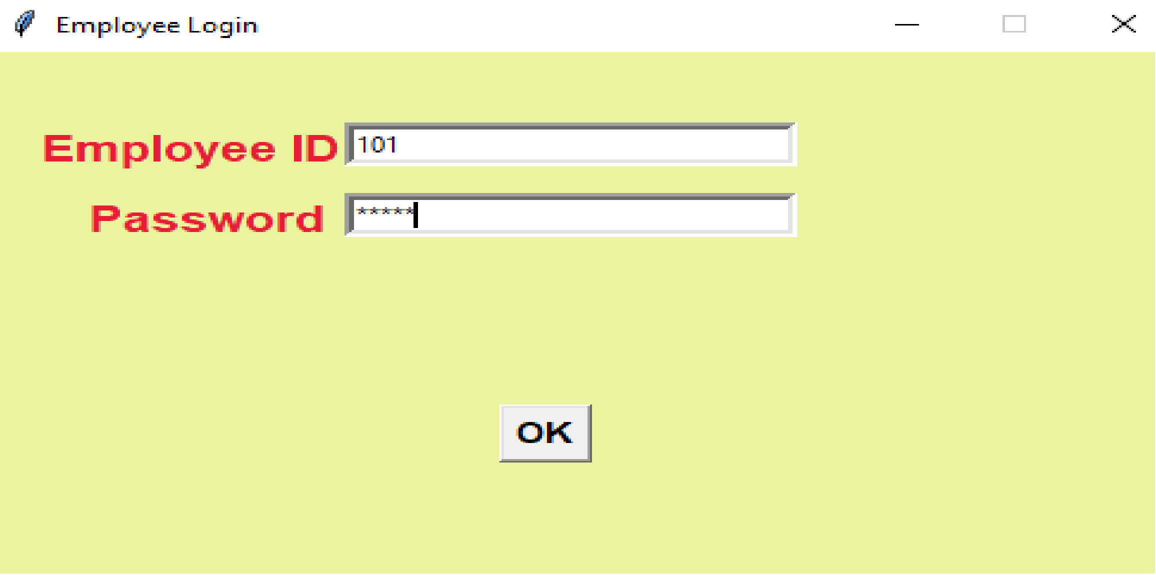


The status of either the approval/denial of the request will be further updated in the ‘Employee Login’ who has applied for thr Regulatization Request.

Also, the ‘Logout’ button will help to logout from the panel and return back to the Homepage.

EMPLOYEE LOGIN: -

After the employee has registered himself/herself into the system, they can access their panels by filling in the correct credentials like: - Employee ID and Password.

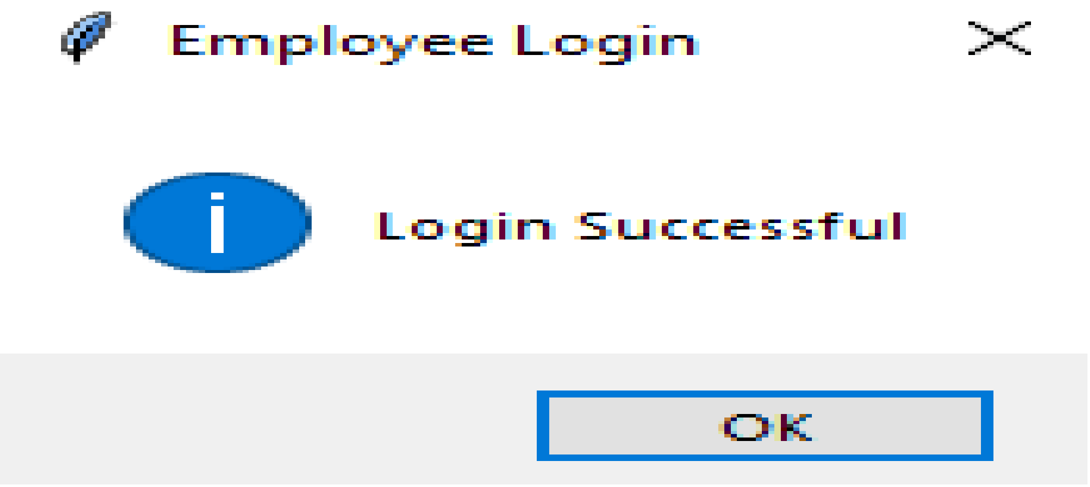


Employee Login

Employee ID: 101

Password: *****

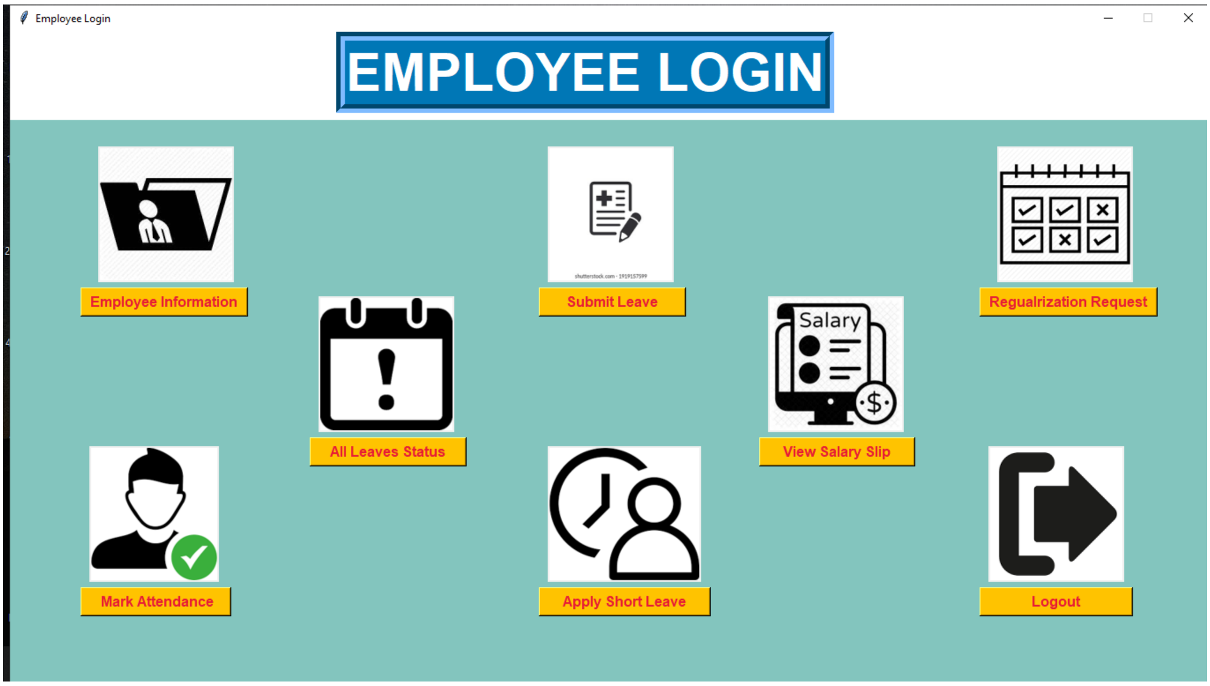
After the credentials are validated, a message box will appear with a message 'Login Successful' which is shown as under:



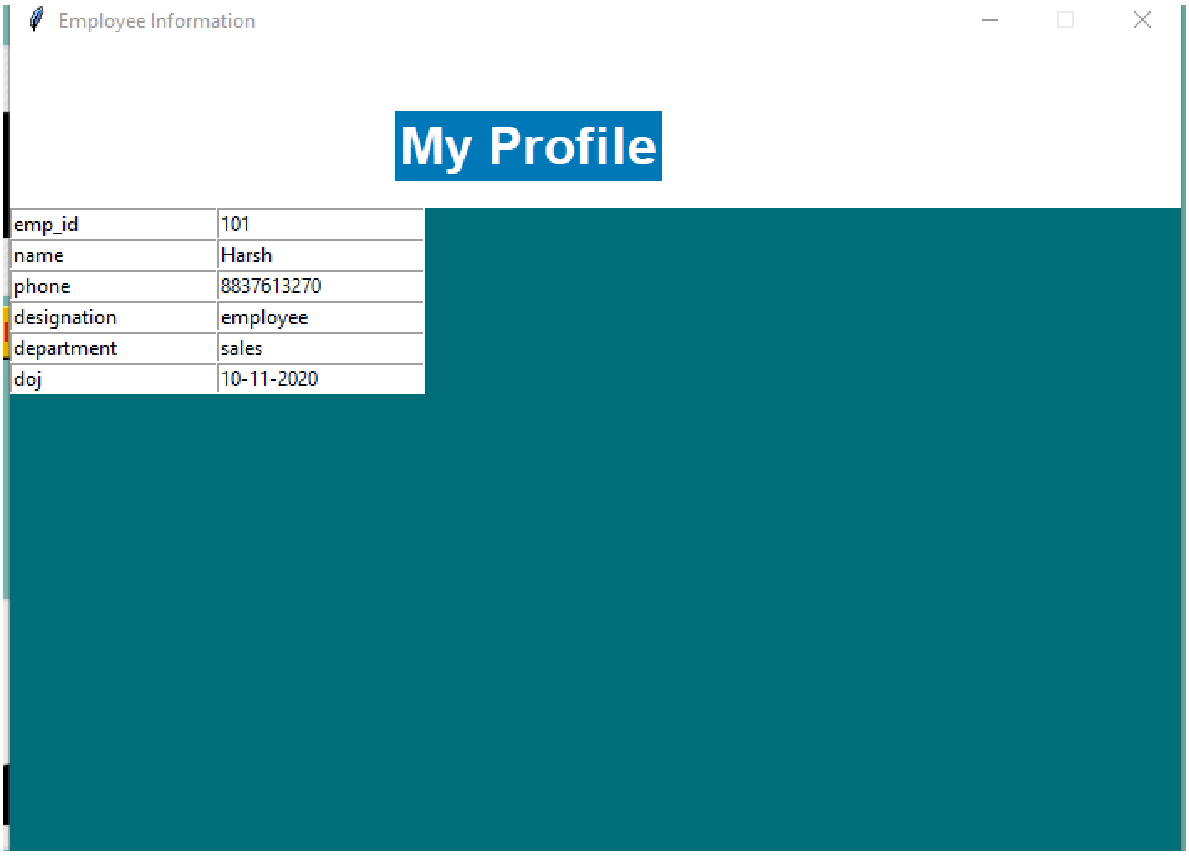
Employee Login

Login Successful

Also, the new panel of 'Employee Login' will open with the following options as shown in the image below.



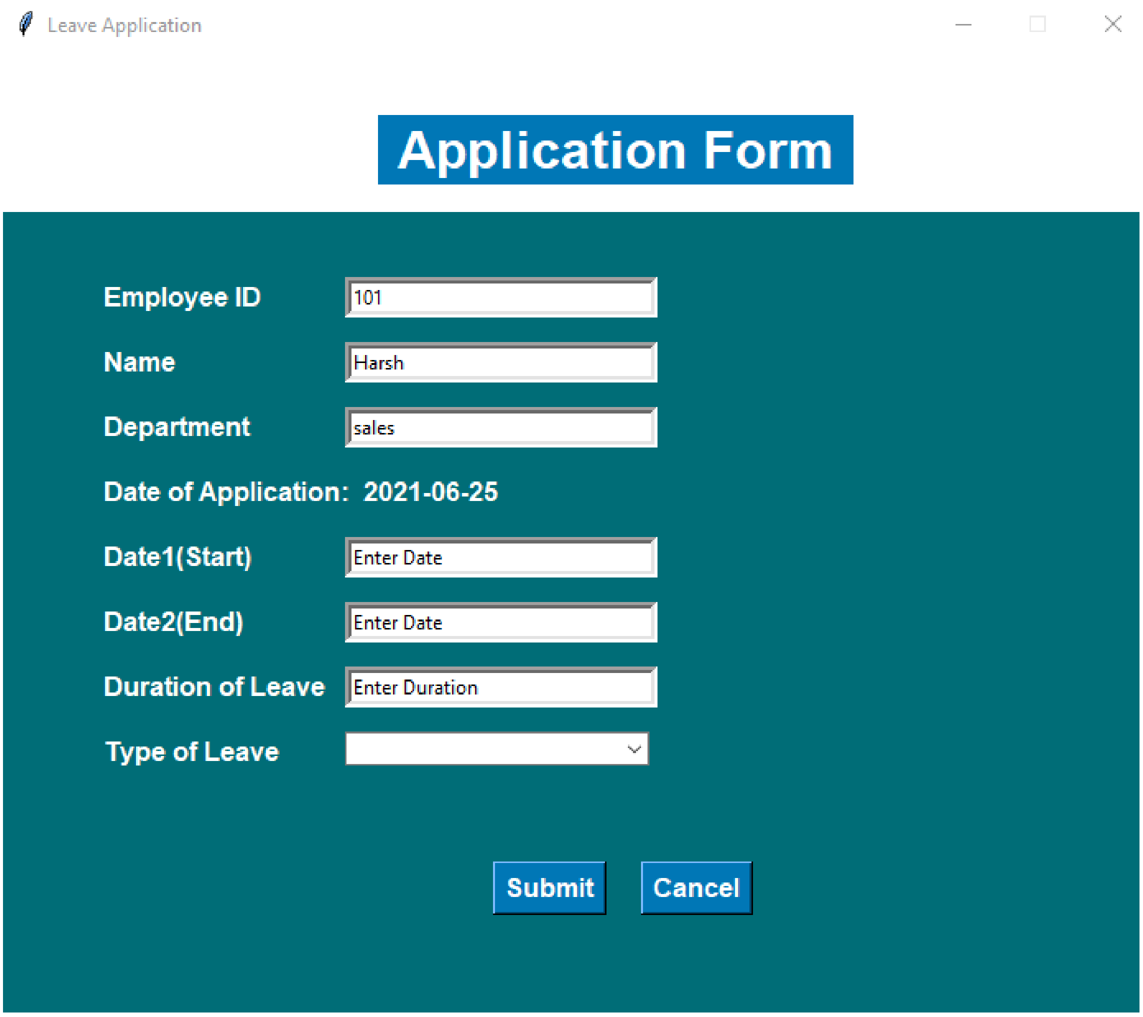
The employee can have the access to his personal information by clicking on the 'Employee Information' button. This information would be the same as filled in by him/her at the time of registration.



My Profile

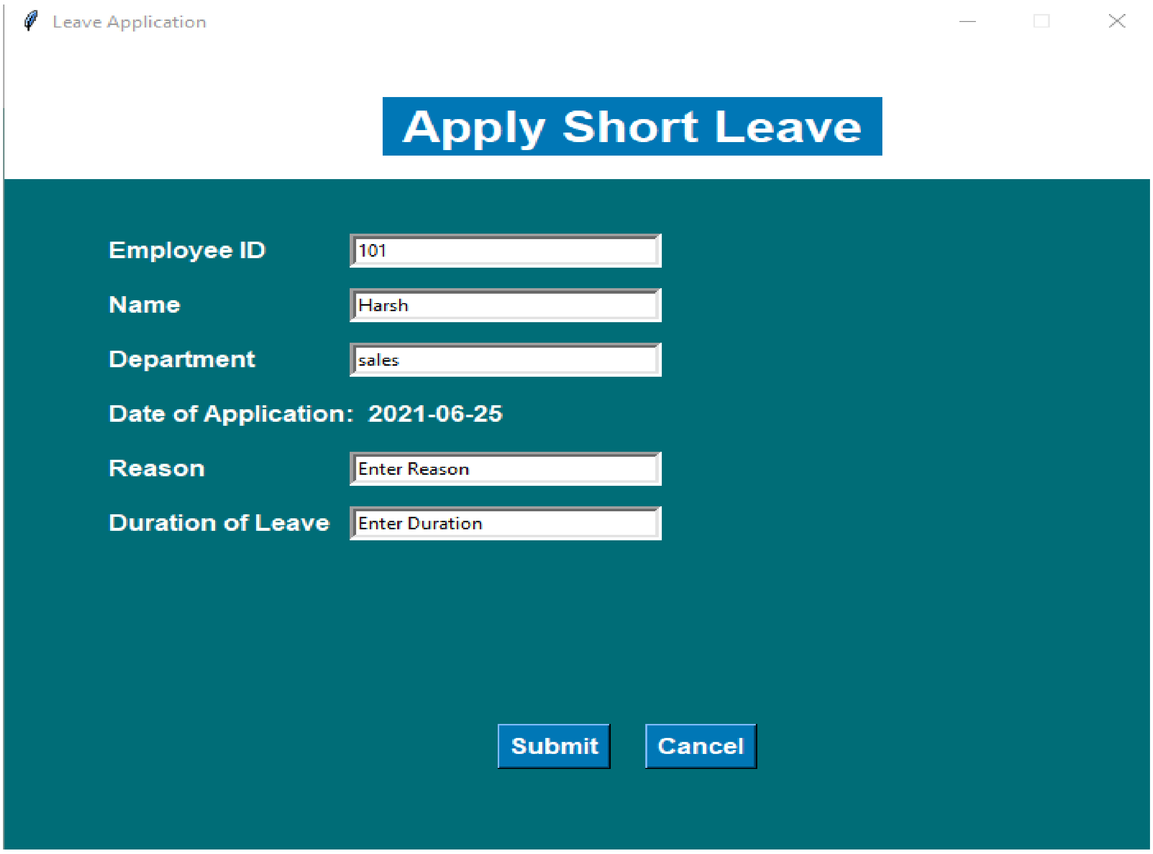
emp_id	101
name	Harsh
phone	8837613270
designation	employee
department	sales
doj	10-11-2020

Next he/she can fill up the application form in case of applying for the leave by clicking on the ‘Submit Leave’ button.



Employee ID	101
Name	Harsh
Department	sales
Date of Application:	2021-06-25
Date1(Start)	Enter Date
Date2(End)	Enter Date
Duration of Leave	Enter Duration
Type of Leave	

The employee can also apply for the short leaves i.e., leaves for a shorter period of time by clicking on the ‘Apply Short Leave’ button.

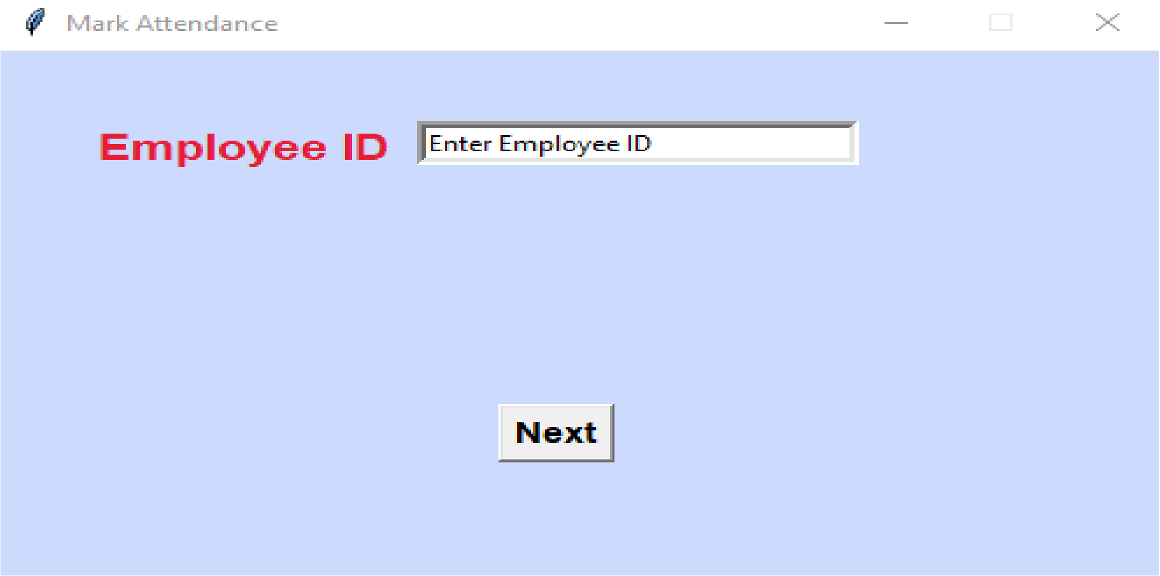


Leave Application

Apply Short Leave

Employee ID	101
Name	Harsh
Department	sales
Date of Application:	2021-06-25
Reason	Enter Reason
Duration of Leave	Enter Duration

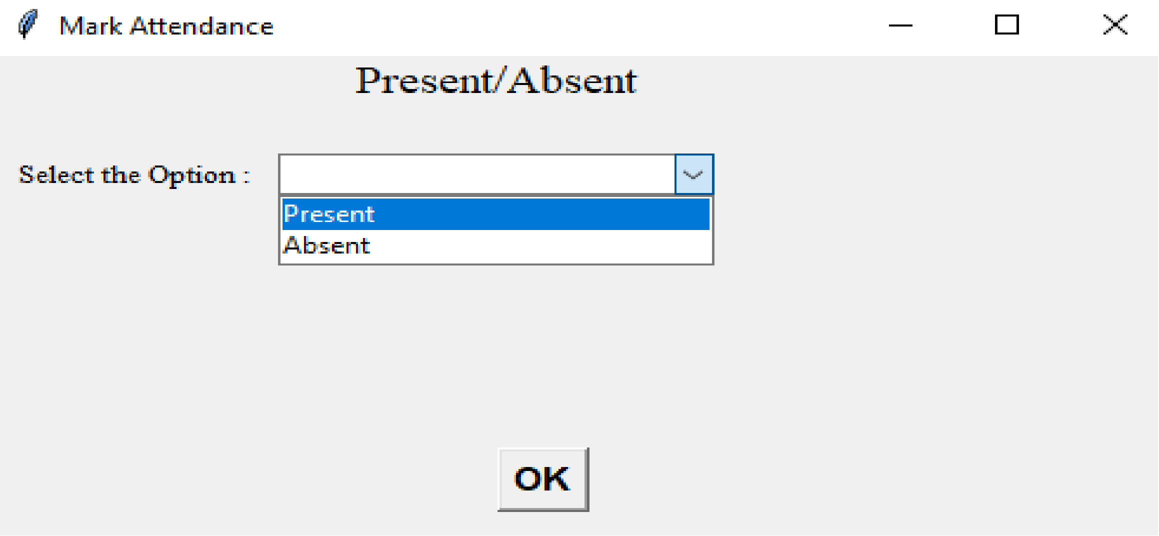
The employee can also mark his/her attendance by clicking on the ‘Mark Attendance’ button.



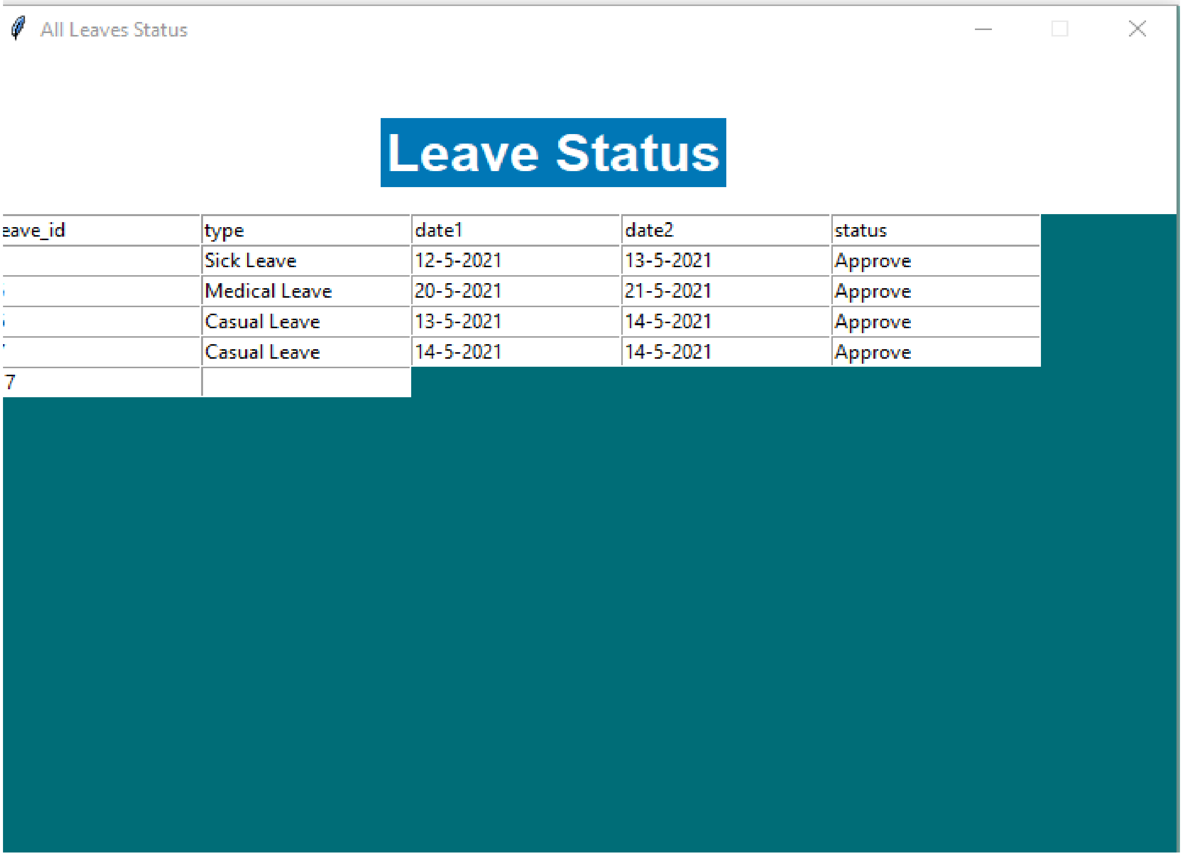
Mark Attendance

Employee ID: Enter Employee ID

After clicking on this button, a dialog box will appear asking for the Employee ID of the employee and then finally he/she needs to select the ‘Present’ so as to get it marked.



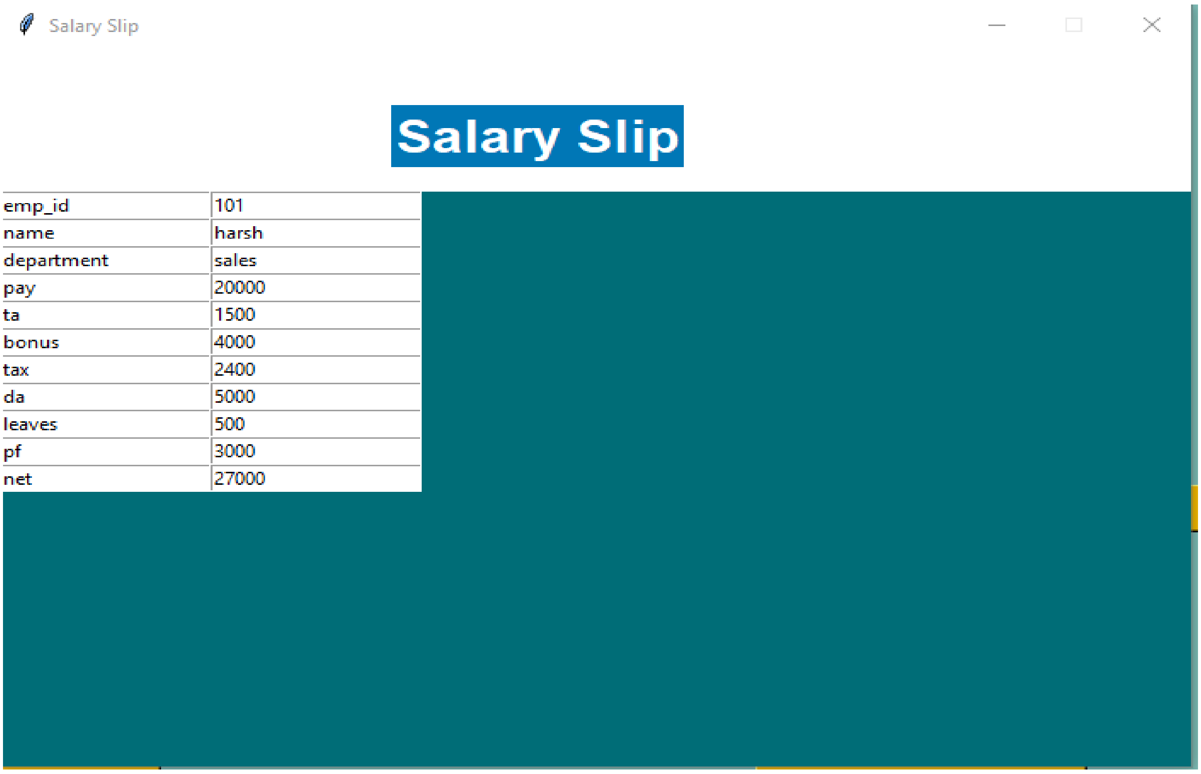
The employee can get to view about his/her leaves status by clicking on the ‘All Leaves Status’ button.



eave_id	type	date1	date2	status
	Sick Leave	12-5-2021	13-5-2021	Approve
	Medical Leave	20-5-2021	21-5-2021	Approve
	Casual Leave	13-5-2021	14-5-2021	Approve
	Casual Leave	14-5-2021	14-5-2021	Approve
7				

Here, they will be able to get the view of the status of all the leaves that have been applied by him/her that whether the leaves have been approved/denied or are still pending to be reviewed by the admin.

The employee can view his/her salary by clicking on the 'View Salary Slip' button.

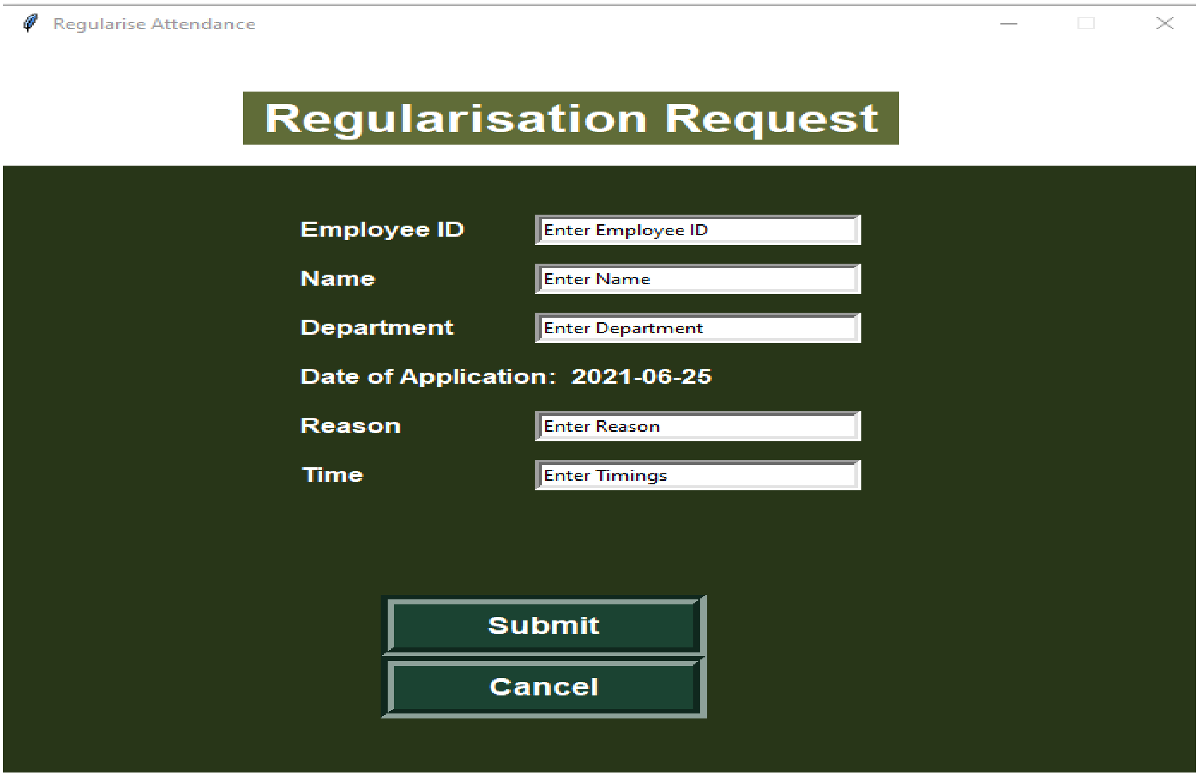


Salary Slip

emp_id	101
name	harsh
department	sales
pay	20000
ta	1500
bonus	4000
tax	2400
da	5000
leaves	500
pf	3000
net	27000

Here, he/she can get the detail about how much is the actual salary of that employee after the additions and deductions in the basic pay amount.

In order to get the attendance regularized, he/she needs to fill up the Regularization Request form by clicking on the 'Regularization Request' button.



Regularise Attendance

Regularisation Request

Employee ID: Enter Employee ID

Name: Enter Name

Department: Enter Department

Date of Application: 2021-06-25

Reason: Enter Reason

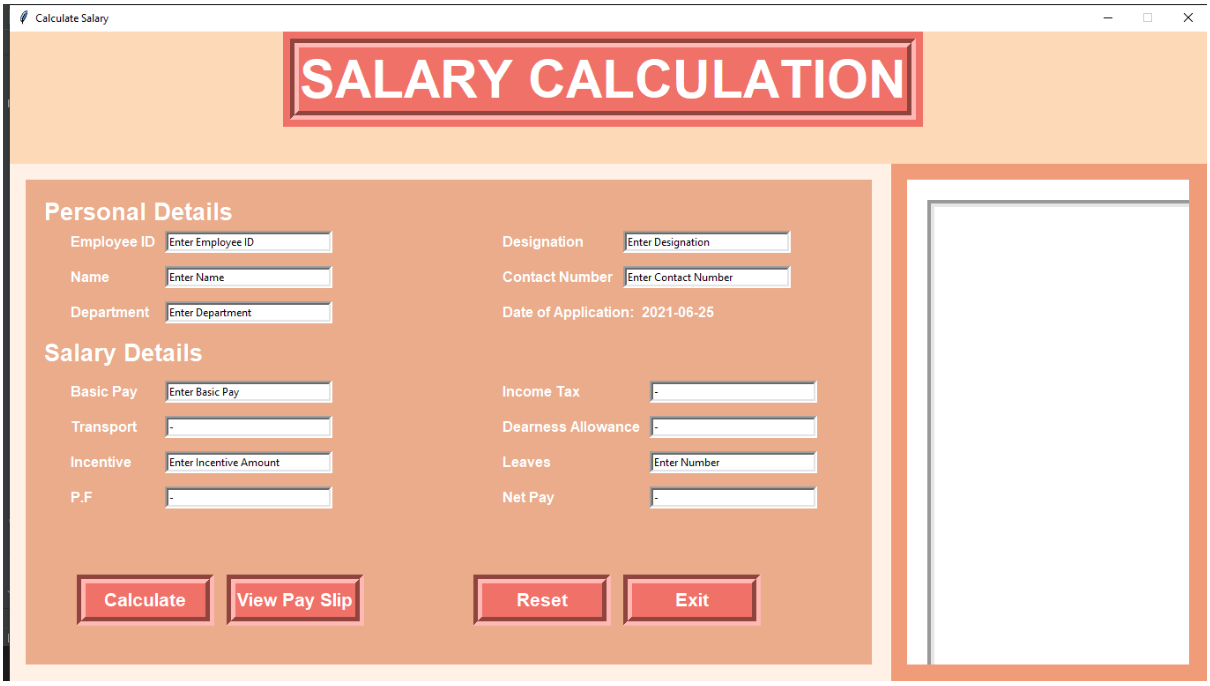
Time: Enter Timings

In this, the request will be approved by the admin only if the reason is the genuine one.

Also, the ‘Logout’ button will help to logout from the panel and return back to the Homepage.

MANAGE SALARY: -

This is the panel which is only accessible by the admin as all the Salary Computations are done by the Admin Staff itself. So, firstly the admin will need to login with the correct credentials so as to ensure operational working.



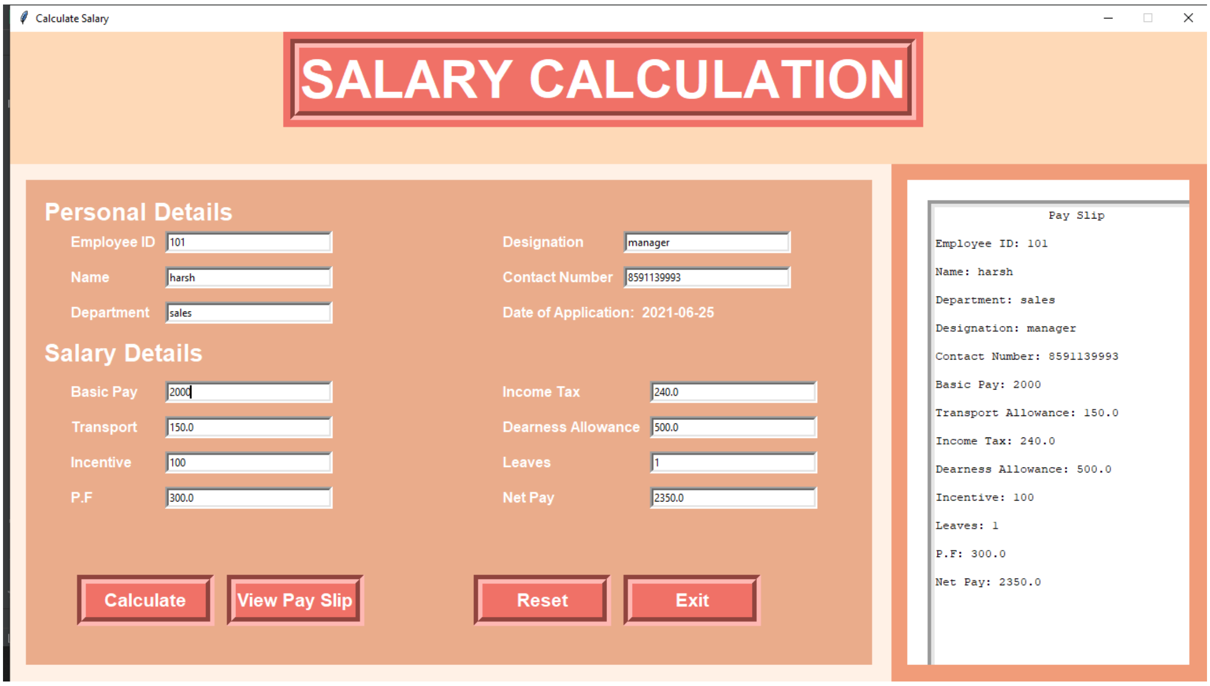
Personal Details

Employee ID: Enter Employee ID
Name: Enter Name
Department: Enter Department
Designation: Enter Designation
Contact Number: Enter Contact Number
Date of Application: 2021-06-25

Salary Details

Basic Pay: Enter Basic Pay
Transport: -
Incentive: Enter Incentive Amount
P.F: -
Income Tax: -
Dearness Allowance: -
Leaves: Enter Number
Net Pay: -

The above window will appear only when the login is done successfully. And in this panel the admin firstly requires to fill up the Personal Details and the Salary Details in the entry fields as shown in the image.



Personal Details

Employee ID: 101
Name: harsh
Department: sales
Designation: manager
Contact Number: 8591139993
Date of Application: 2021-06-25

Salary Details

Basic Pay: 2000
Transport: 150.0
Incentive: 100
P.F: 300.0
Income Tax: 240.0
Dearness Allowance: 500.0
Leaves: 1
Net Pay: 2350.0

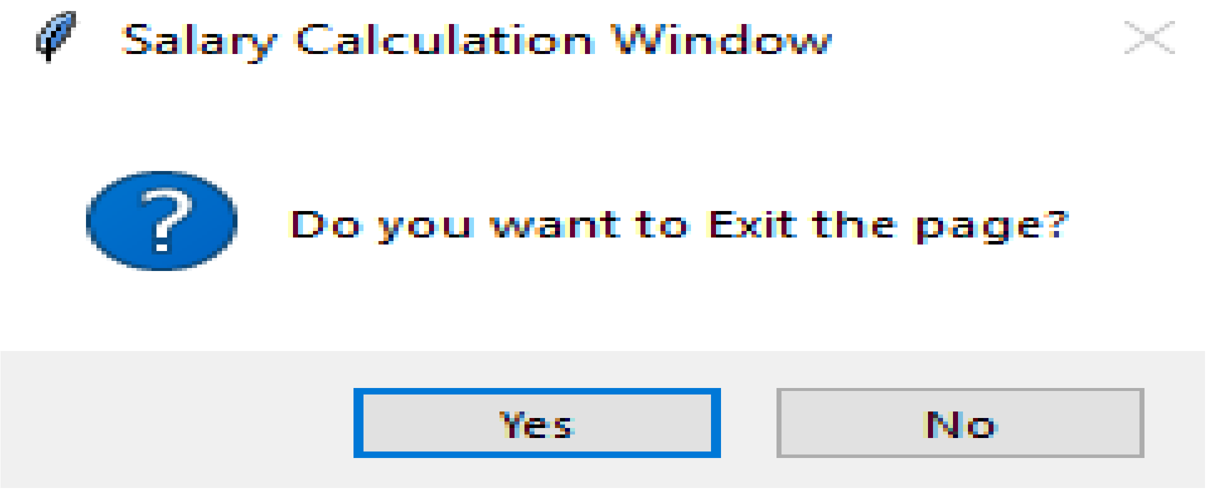
```
              Pay Slip
Employee ID: 101
Name: harsh
Department: sales
Designation: manager
Contact Number: 8591139993
Basic Pay: 2000
Transport Allowance: 150.0
Income Tax: 240.0
Dearness Allowance: 500.0
Incentive: 100
Leaves: 1
P.F: 300.0
Net Pay: 2350.0
```

After filling the details, when the user clicks on the ‘Calculate’ button therein all the rest of the calculations are performed and then viewed in the rest of the fields.

Thereafter, he/she can click on the 'View Pay Slip' button in order to view the payslip for that particular employee.

Hence, if they want to proceed with subsequent more computations for the salary then the user can click on the 'Reset' button in order to clear the data of the previous entry.

And hence, click on the 'Exit' button in order to exit from the panel. And once after clicking on the 'Exit' button the following question/answer box will appear asking if the user wants to exit or stay on the same page.



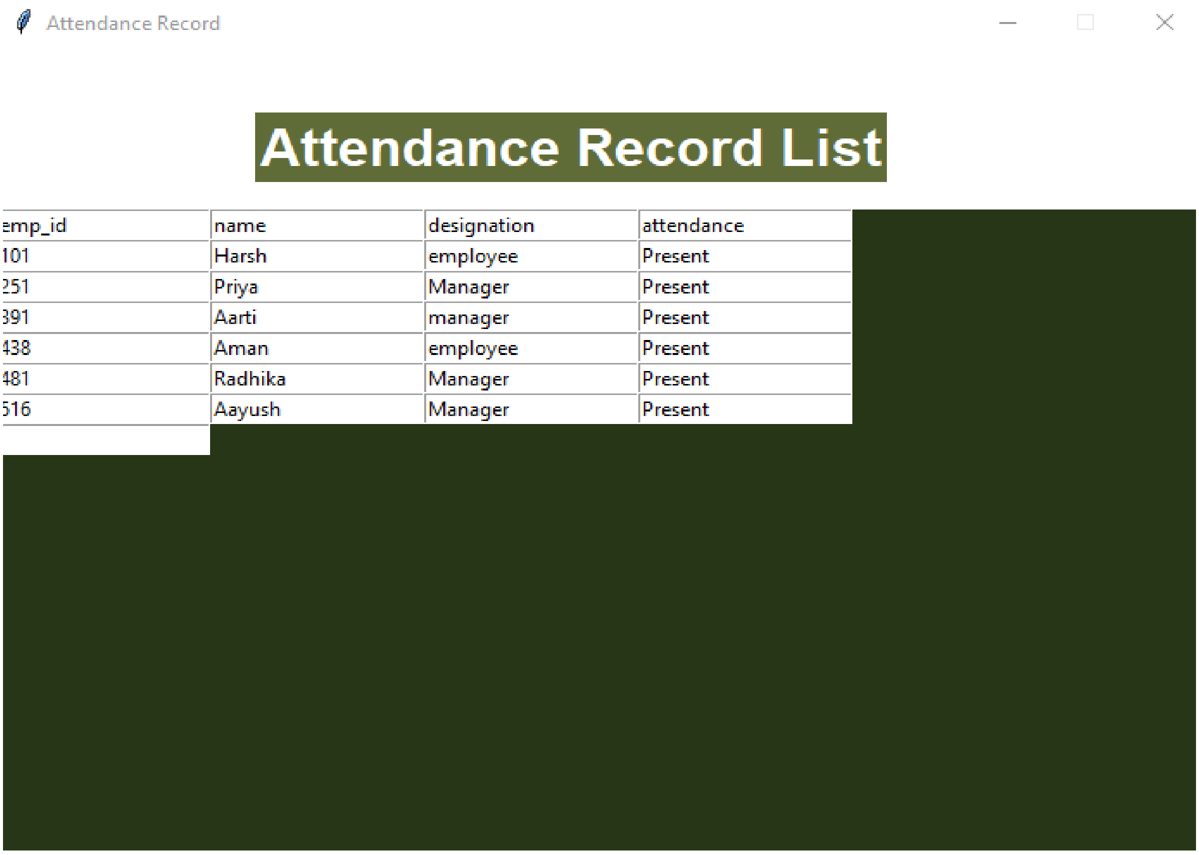
Salary Calculation Window

Do you want to Exit the page?

And if the user clicks on 'Yes ' then the user may return back to the Homepage.

ATTENDANCE RECORD: -

This module can help the admin to come to know about how many employees are present on that particular day in the workspace area.



Attendance Record List

emp_id	name	designation	attendance
101	Harsh	employee	Present
251	Priya	Manager	Present
391	Aarti	manager	Present
438	Aman	employee	Present
481	Radhika	Manager	Present
516	Aayush	Manager	Present

MANAGE LEAVES: -

This can be accessed by the employees as well as the admin as on the employee's side he/she will need to apply for the leaves (any category of leaves) and then might also require to check the status of leaves, leave balance, etc.

Whlie on the admin side, they may need to view the list for all the employees who have applied for the leaves, approve the leaves, etc.

REGULARIZE ATTENDANCE: -

This may also work on both the ends i.e., for the admin as well as for the employee. The employee may need to fill the regularization request form in case of some genuine problems and the admin may require to view the list and approve the requests as per the applicability of the situations.

HOME: -

This is the button which is placed on the Home Page/Main Window in order to exit of switch off from the system.

LIMITATIONS

Although the system prepared is a computerised one but there are some limitations too, which are as follows:

- It does not provide the facility of generating the reports related to the employees for the concrete analysis.
- The employees who are registered in the system only they can operate the system and no other outsider can operate it.
- The system is somewhat a desktop application so it will not be accessible through the web sources.
- The system does not provide any kind of guidelines for usage and ways for the management of records for the employees or for the admin staff.
- The testing of the system is not possible in this stage of the development of the system.

FUTURE SCOPE

In order to enhance the system at a later stage, following things can be taken care of:

- The generation of reports will be made possible.
- The system can be hosted with the platform on online servers to make it accessible worldwide.
- Testing of the software can be checked.
- The guidelines for usage may be added to make the usage process clear and understandable to the employees as well as the admins.

CONCLUSION:

In the end, I would like to conclude that the ‘EMPLOYEE MANAGEMENT SYSTEM’ which is just a small system consisting of four different modules is helpful for the admin as well as the employees as this can help in maintaining the records for all the computations of the employees which can help later in report generations and decision making for the bigger aspects that is done by the top management.

BIBLIOGRAPHY

The various links from where the references were taken are: -

- Channel Name – Python, ML and Data Science (Telegram)
- Channel Name – Python and Information (Telegram)
- Python Documentations
- SQL Documentations
- Interfaceideas.org
- Tutorials of Python and SQL
- www.geeksforgeeks.org
- www.stackoverflow.com