PAYROLL MANAGEMENT SYSTEM using PYTHON

A report submitted to highlight the project

BY

PRABIN KUMAR BEHERA (12108952)

KOTTURTU SAI KIRAN (12107859)

Course Code: INT-213
Submitted To： CHERRY KHOSLA (13436)
Date of creation: 11/11/2022

Table of Contents

1. Introduction

1.1 Document Conventions

Font: Arial, Size: 15

1.2 Purpose

The purpose of this document is to give a detailed report of a Payroll Management System using Python. It will illustrate the GUI front end, the code backend as well as the complete development of the project. This document is primarily intended for anyone who wants to get an overview of how the Payroll Management System works and how it is implemented using various libraries available with Python.

1.3 Scope

This Python program allows us to use a Payroll Management System with its usual functions like taking input from user, calculating the salary, display it to the user, displaying the slip, etc. Each and every detail about employee's payment is displayed which includes: Name with employee id, tax, and net pay. This system makes easier to the user for managing payroll system as it is not time-consuming. This project is not difficult to operate and understood by the users.

1.4 Objective

The goals of a payroll system are fairly simple. In essence, they're designed to account for how much an employee earns, how much they're paid, and what taxes they pay. These three items are computed based on the number of hours worked and the wages being paid. Typically, the money being earned by the employee is calculated by adding up all the time in the given pay period and multiplying that figure by their gross pay rate per hour. The tax burden is calculated as a percentage of their salary before deductions are applied.

2. Requirement Specification

2.1 Hardware Requirements

This application is not very hardware demanding. It can be accessed with any PC/Laptop with Python installed on it. Python GUI is also required to be working for this application to execute properly.

2.2 Software Requirements

This application requires Python's Tkinter library and Python's date/time library to be working, it also requires a python compiler to compile and execute the code.

3. Modules Used

3.1 Tkinter Library

Tkinter toolkit is a way in python to create Graphical User Interfaces (GUIs) and is included in the Python standard library. Several "widgets" are included in this library which can be collectively used to build interfaces for various applications. This framework gives Python users a quick and easy way to build GUI elements with Tk toolkit's widgets. In a Python application, Tk widgets can be used to create buttons, menus, data fields, etc. These graphic elements can be connected to or interact with features, functionality, processes, data, or even other widgets once they have been established.

The widgets used in this application are:

- Textbox: The general typing area in which the user and input data which can be later stored in the directory.
- Scrollbar: The text file also contains a scrollbar which can be used to scroll between pages if the file becomes too large to fit in one screen.
- Label: Label is a widget that is used to implement display boxes where you can place text or images.

- Entry: The Entry Widget is a Tkinter Widget used to Enter or display a single line of text.
- Frame: The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way.

3.2 Python Datetime module

In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time. Python Datetime module comes built into Python, so there is no need to install it externally. Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

The DateTime module is categorized into some main classes :

- date– An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.
- time– An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.
- datetime – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond.
- timezone – A class that implements the tzinfo abstract base class as a fixed offset from the UTC

3.3 Python OS module

The OS module in Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system-dependent functionality.

4. Front-End GUI

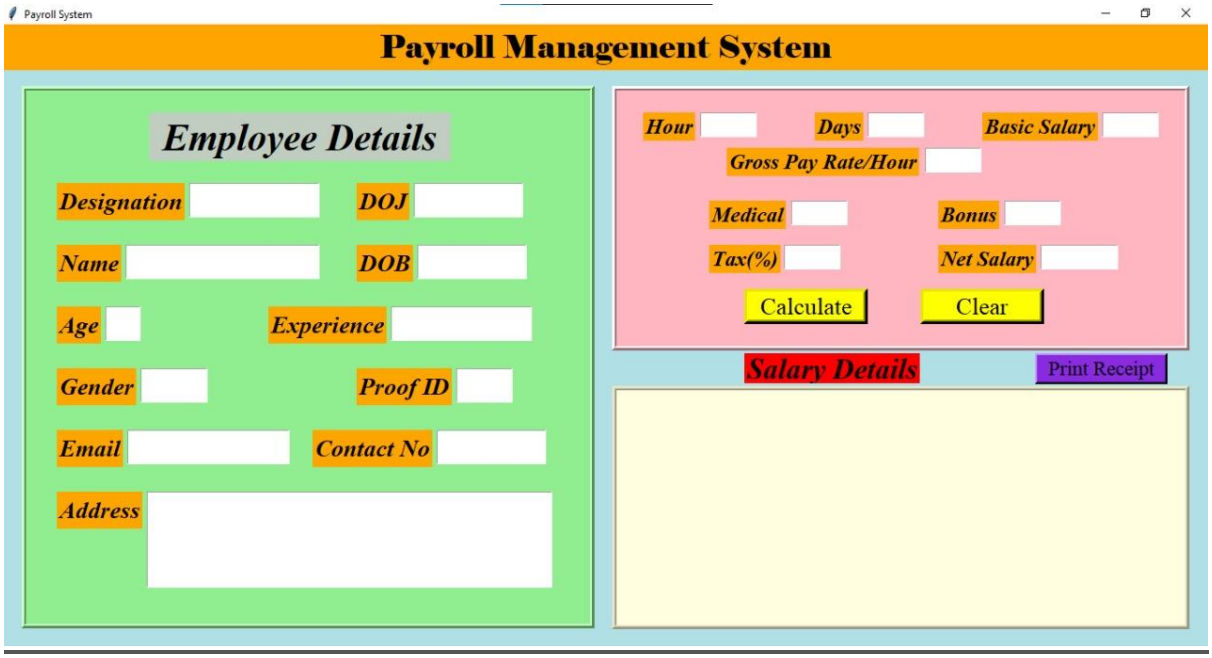


Figure 1: First Screen when fields are empty.

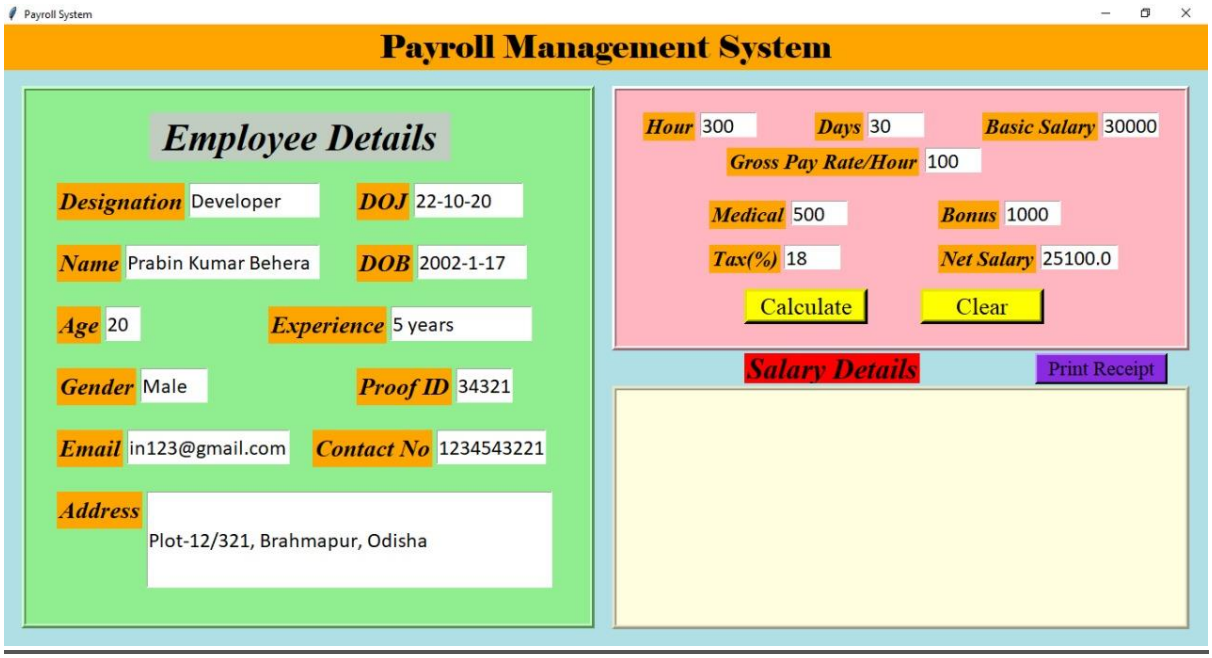


Figure 2: After filling the details salary is calculate.

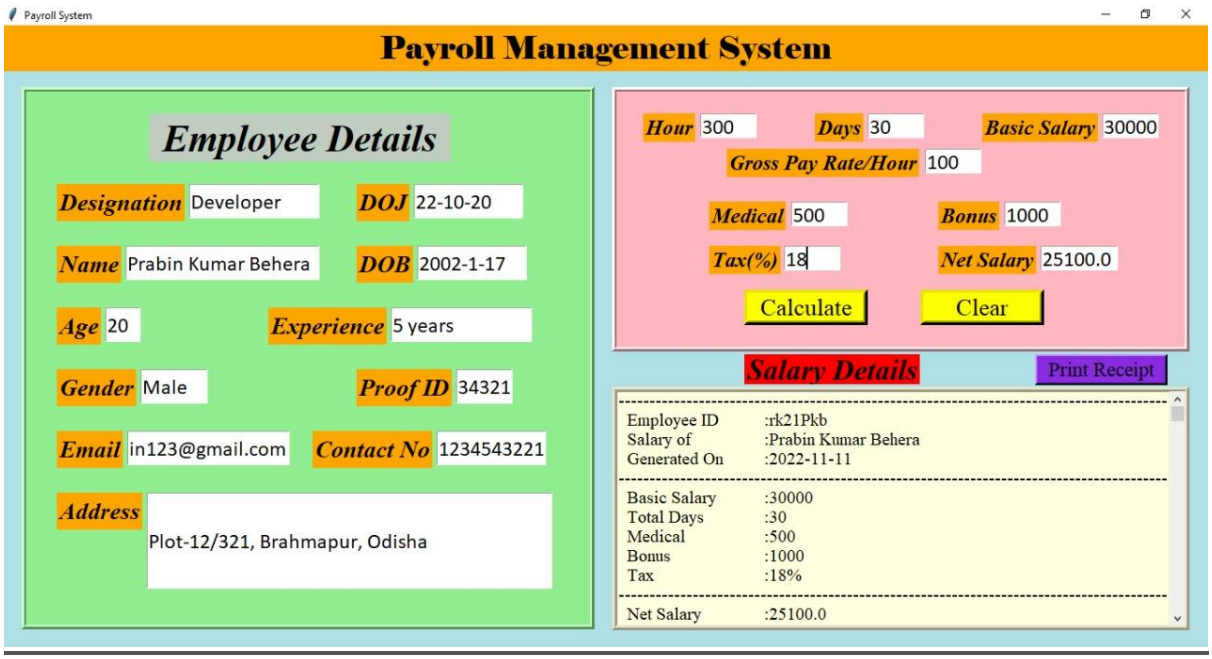
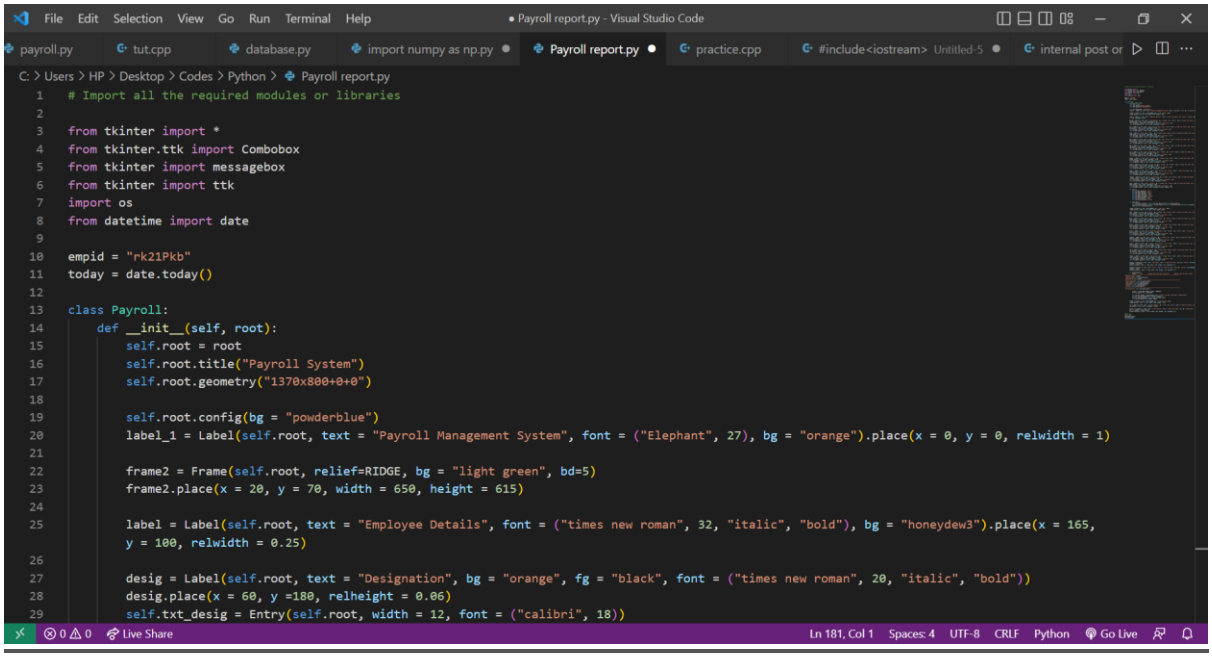
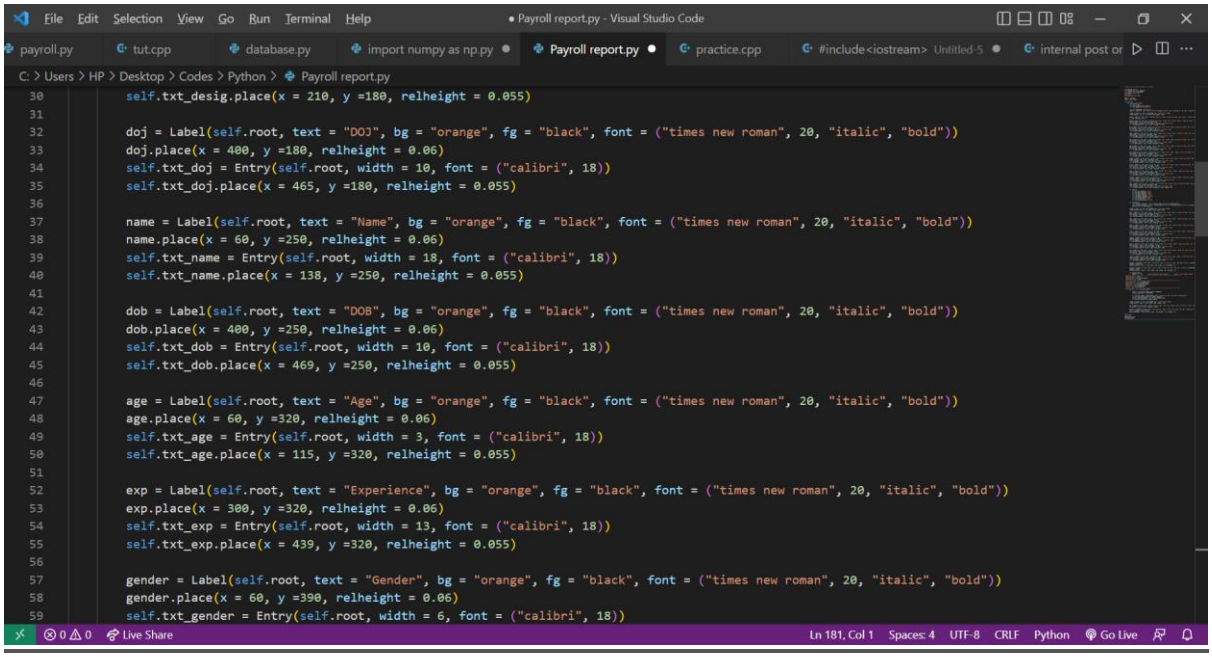


Figure 3: Display of Receipt.

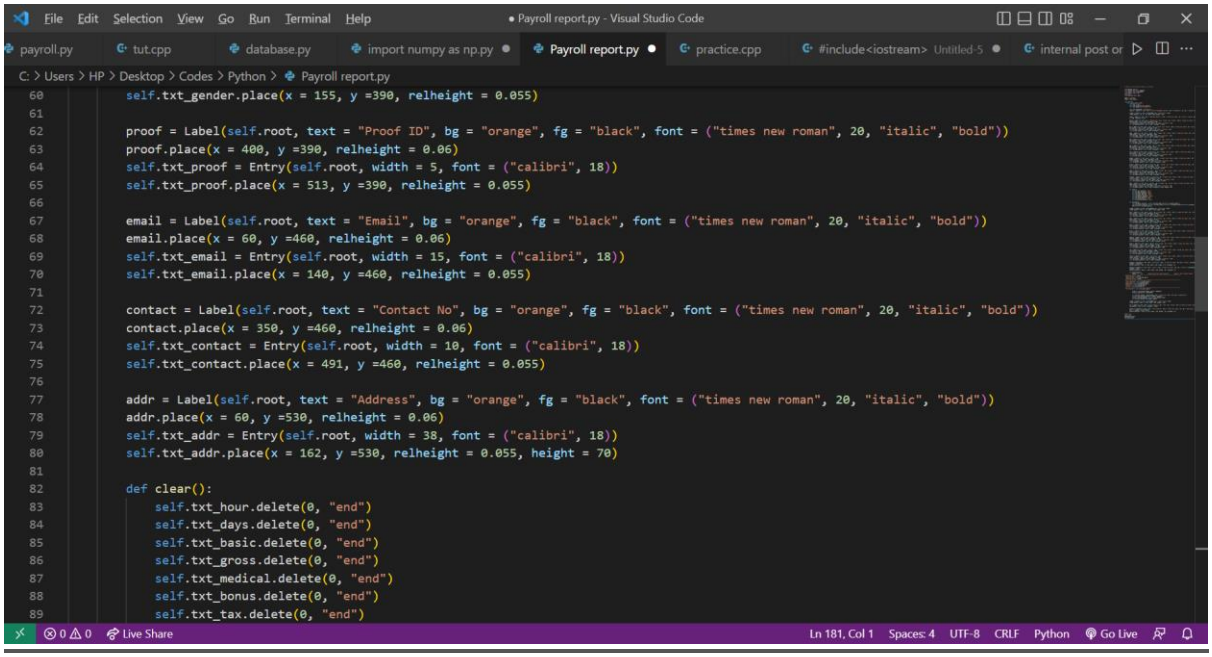
5. Back-End Code



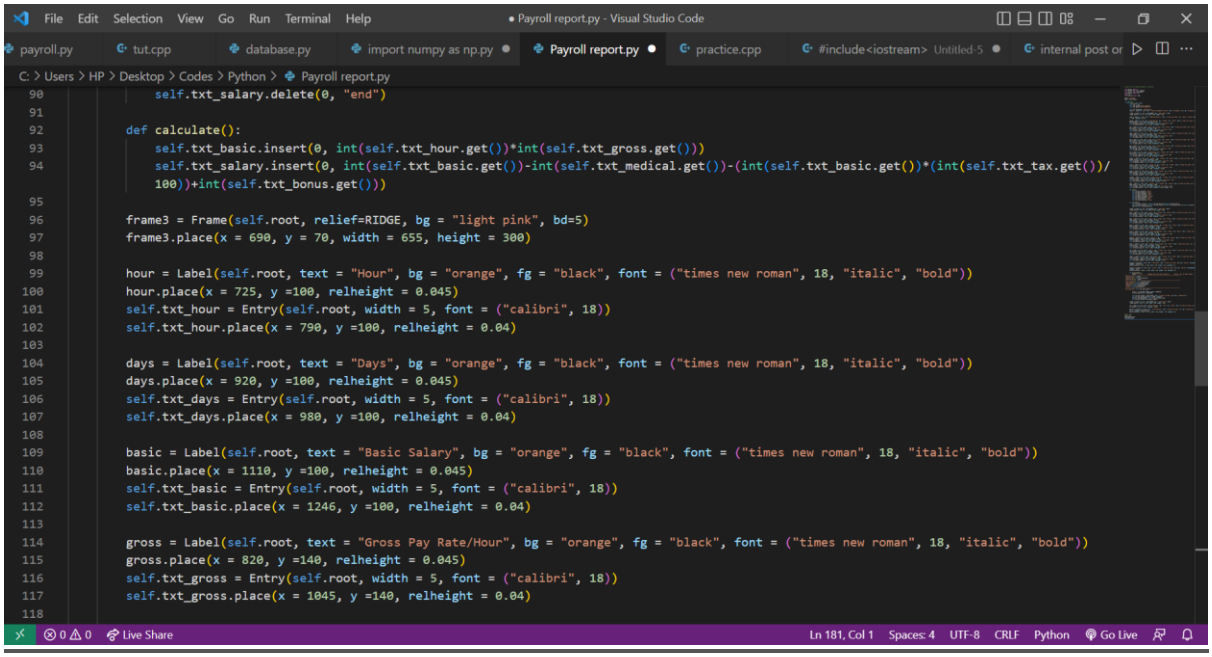
```
# Import all the required modules or libraries

from tkinter import *
from tkinter.ttk import Combobox
from tkinter import messagebox
from tkinter import ttk
import os
from datetime import date

empid = "rk21Pkb"
today = date.today()

class Payroll:
    def __init__(self, root):
        self.root = root
        self.root.title("Payroll System")
        self.root.geometry("1370x800+0+0")

        self.root.config(bg = "powderblue")
        label_1 = Label(self.root, text = "Payroll Management System", font = ("Elephant", 27), bg = "orange").place(x = 0, y = 0, relwidth = 1)

        frame2 = Frame(self.root, relief=RIDGE, bg = "light green", bd=5)
        frame2.place(x = 20, y = 70, width = 650, height = 615)

        label = Label(self.root, text = "Employee Details", font = ("times new roman", 32, "italic", "bold"), bg = "honeydew3").place(x = 165,
        y = 100, relwidth = 0.25)

        desig = Label(self.root, text = "Designation", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        desig.place(x = 60, y =180, relheight = 0.06)
        self.txt_desig = Entry(self.root, width = 12, font = ("calibri", 18))
```



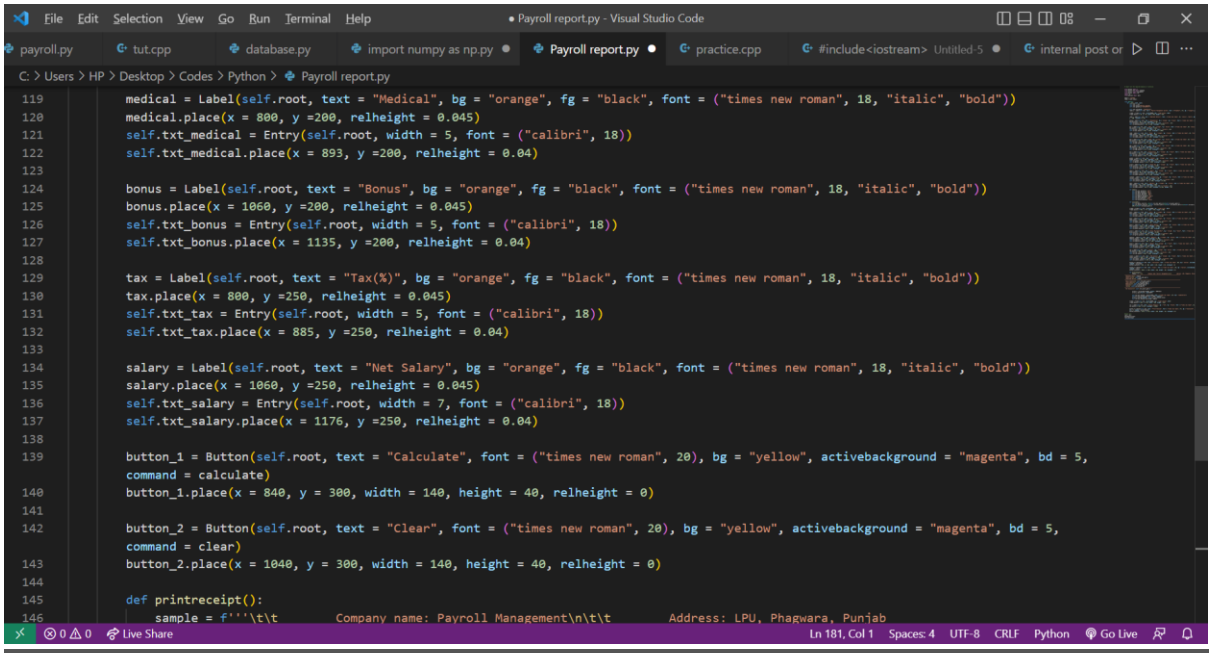
```
        self.txt_desig.place(x = 210, y =180, relheight = 0.055)

        doj = Label(self.root, text = "DOJ", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        doj.place(x = 400, y =180, relheight = 0.06)
        self.txt_doj = Entry(self.root, width = 10, font = ("calibri", 18))
        self.txt_doj.place(x = 465, y =180, relheight = 0.055)

        name = Label(self.root, text = "Name", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        name.place(x = 60, y =250, relheight = 0.06)
        self.txt_name = Entry(self.root, width = 18, font = ("calibri", 18))
        self.txt_name.place(x = 138, y =250, relheight = 0.055)

        dob = Label(self.root, text = "DOB", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        dob.place(x = 400, y =250, relheight = 0.06)
        self.txt_dob = Entry(self.root, width = 10, font = ("calibri", 18))
        self.txt_dob.place(x = 469, y =250, relheight = 0.055)

        age = Label(self.root, text = "Age", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        age.place(x = 60, y =320, relheight = 0.06)
        self.txt_age = Entry(self.root, width = 3, font = ("calibri", 18))
        self.txt_age.place(x = 115, y =320, relheight = 0.055)

        exp = Label(self.root, text = "Experience", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        exp.place(x = 300, y =320, relheight = 0.06)
        self.txt_exp = Entry(self.root, width = 13, font = ("calibri", 18))
        self.txt_exp.place(x = 439, y =320, relheight = 0.055)

        gender = Label(self.root, text = "Gender", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        gender.place(x = 60, y =390, relheight = 0.06)
        self.txt_gender = Entry(self.root, width = 6, font = ("calibri", 18))
```



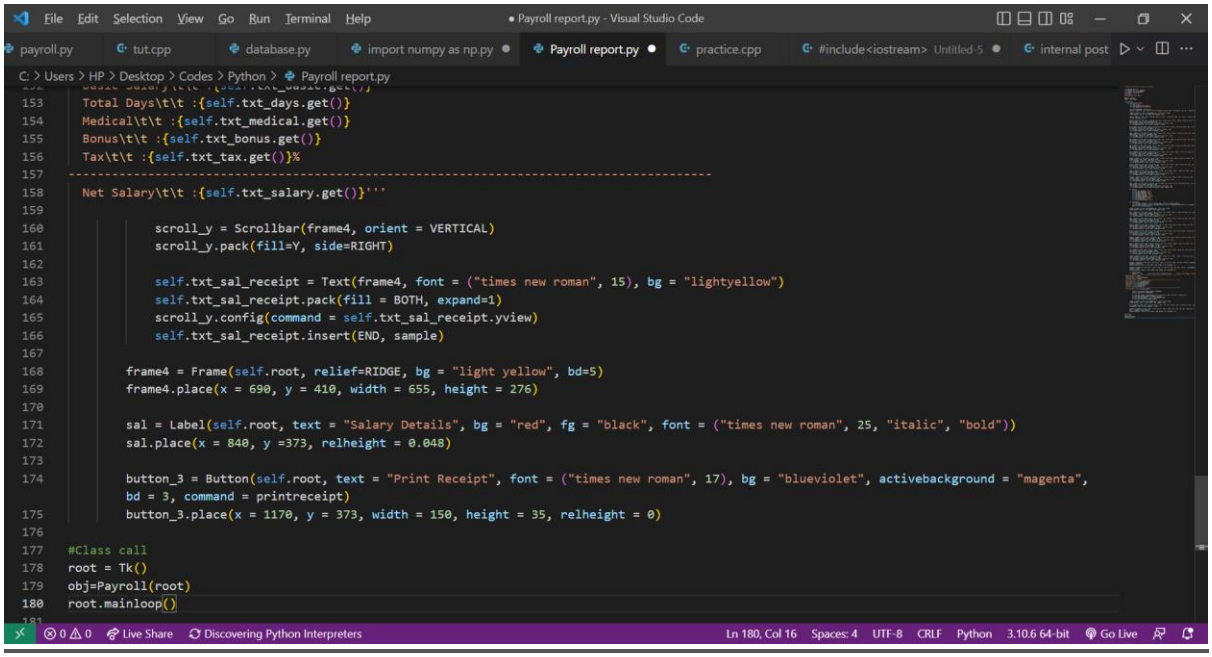
```
        self.txt_gender.place(x = 155, y =390, relheight = 0.055)

        proof = Label(self.root, text = "Proof ID", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        proof.place(x = 400, y =390, relheight = 0.06)
        self.txt_proof = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_proof.place(x = 513, y =390, relheight = 0.055)

        email = Label(self.root, text = "Email", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        email.place(x = 60, y =460, relheight = 0.06)
        self.txt_email = Entry(self.root, width = 15, font = ("calibri", 18))
        self.txt_email.place(x = 140, y =460, relheight = 0.055)

        contact = Label(self.root, text = "Contact No", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        contact.place(x = 350, y =460, relheight = 0.06)
        self.txt_contact = Entry(self.root, width = 10, font = ("calibri", 18))
        self.txt_contact.place(x = 491, y =460, relheight = 0.055)

        addr = Label(self.root, text = "Address", bg = "orange", fg = "black", font = ("times new roman", 20, "italic", "bold"))
        addr.place(x = 60, y =530, relheight = 0.06)
        self.txt_addr = Entry(self.root, width = 38, font = ("calibri", 18))
        self.txt_addr.place(x = 162, y =530, relheight = 0.055, height = 70)

        def clear():
            self.txt_hour.delete(0, "end")
            self.txt_days.delete(0, "end")
            self.txt_basic.delete(0, "end")
            self.txt_gross.delete(0, "end")
            self.txt_medical.delete(0, "end")
            self.txt_bonus.delete(0, "end")
            self.txt_tax.delete(0, "end")
```



```
            self.txt_salary.delete(0, "end")

        def calculate():
            self.txt_basic.insert(0, int(self.txt_hour.get())*int(self.txt_gross.get()))
            self.txt_salary.insert(0, int(self.txt_basic.get())-int(self.txt_medical.get())-(int(self.txt_basic.get())*(int(self.txt_tax.get())/
            100))+int(self.txt_bonus.get()))

        frame3 = Frame(self.root, relief=RIDGE, bg = "light pink", bd=5)
        frame3.place(x = 690, y = 70, width = 655, height = 300)

        hour = Label(self.root, text = "Hour", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        hour.place(x = 725, y =100, relheight = 0.045)
        self.txt_hour = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_hour.place(x = 790, y =100, relheight = 0.04)

        days = Label(self.root, text = "Days", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        days.place(x = 920, y =100, relheight = 0.045)
        self.txt_days = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_days.place(x = 980, y =100, relheight = 0.04)

        basic = Label(self.root, text = "Basic Salary", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        basic.place(x = 1110, y =100, relheight = 0.045)
        self.txt_basic = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_basic.place(x = 1246, y =100, relheight = 0.04)

        gross = Label(self.root, text = "Gross Pay Rate/Hour", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        gross.place(x = 820, y =140, relheight = 0.045)
        self.txt_gross = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_gross.place(x = 1045, y =140, relheight = 0.04)
```



```
        medical = Label(self.root, text = "Medical", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        medical.place(x = 800, y =200, relheight = 0.045)
        self.txt_medical = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_medical.place(x = 893, y =200, relheight = 0.04)

        bonus = Label(self.root, text = "Bonus", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        bonus.place(x = 1060, y =200, relheight = 0.045)
        self.txt_bonus = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_bonus.place(x = 1135, y =200, relheight = 0.04)

        tax = Label(self.root, text = "Tax(%)", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        tax.place(x = 800, y =250, relheight = 0.045)
        self.txt_tax = Entry(self.root, width = 5, font = ("calibri", 18))
        self.txt_tax.place(x = 885, y =250, relheight = 0.04)

        salary = Label(self.root, text = "Net Salary", bg = "orange", fg = "black", font = ("times new roman", 18, "italic", "bold"))
        salary.place(x = 1060, y =250, relheight = 0.045)
        self.txt_salary = Entry(self.root, width = 7, font = ("calibri", 18))
        self.txt_salary.place(x = 1176, y =250, relheight = 0.04)

        button_1 = Button(self.root, text = "Calculate", font = ("times new roman", 20), bg = "yellow", activebackground = "magenta", bd = 5,
        command = calculate)
        button_1.place(x = 840, y = 300, width = 140, height = 40, relheight = 0)

        button_2 = Button(self.root, text = "Clear", font = ("times new roman", 20), bg = "yellow", activebackground = "magenta", bd = 5,
        command = clear)
        button_2.place(x = 1040, y = 300, width = 140, height = 40, relheight = 0)

        def printreceipt():
            sample = f'''\t\t        Company name: Payroll Management\n\t\t        Address: LPU, Phagwara, Punjab
    ----------------------------------------------------------------------------------
      Employee ID\t\t :{empid}
      Salary of\t\t :{self.txt_name.get()}
      Generated On\t\t :{today}
    ----------------------------------------------------------------------------------
      Basic Salary\t\t :{self.txt_basic.get()}
      Total Days\t\t :{self.txt_days.get()}
      Medical\t\t :{self.txt_medical.get()}
      Bonus\t\t :{self.txt_bonus.get()}
      Tax\t\t :{self.txt_tax.get()}%
    ----------------------------------------------------------------------------------
      Net Salary\t\t :{self.txt_salary.get()}'''

            scroll_y = Scrollbar(frame4, orient = VERTICAL)
            scroll_y.pack(fill=Y, side=RIGHT)

            self.txt_sal_receipt = Text(frame4, font = ("times new roman", 15), bg = "lightyellow")
            self.txt_sal_receipt.pack(fill = BOTH, expand=1)
            scroll_y.config(command = self.txt_sal_receipt.yview)
            self.txt_sal_receipt.insert(END, sample)

        frame4 = Frame(self.root, relief=RIDGE, bg = "light yellow", bd=5)
        frame4.place(x = 690, y = 410, width = 655, height = 276)

        sal = Label(self.root, text = "Salary Details", bg = "red", fg = "black", font = ("times new roman", 25, "italic", "bold"))
        sal.place(x = 840, y =373, relheight = 0.048)

        button_3 = Button(self.root, text = "Print Receipt", font = ("times new roman", 17), bg = "blueviolet", activebackground = "magenta",
        bd = 3, command = printreceipt)
```



6. Conclusion

We learned how to use Python's inbuilt Tkinter and Datetime modules properly using this application. We also learned about the OS module and making our application access the PC directory for saving and retrieving files. This management system helps us to display the salary details of employee. Using the Tkinter module, we also learned how to work with Python's GUI and use it to design a user-friendly application to simplify tasks.

7. References

- https://www.geeksforgeeks.org/os-module-python-examples/
- https://www.geeksforgeeks.org/python-datetime-module/
- https://www.geeksforgeeks.org/python-tkinter-frame-widget/
- https://www.geeksforgeeks.org/python-tkinter-entry-widget/
- https://www.tutorialspoint.com/python/tk_label.htm
- https://docs.python.org/3/library/os.html