

# 1.INTRODUCTION

Payroll Management System is a simple GUI based Desktop Application in Tkinter which is user Friendly and very easy to understand. There is just a admin side in this project. This Project is very helpful for any staff to get their accurate pay data. There is no Login System in this Project. This Project is very useful for educational purpose. A **Payroll Management System Project** is a **system** used by companies to help manage the computation, disbursement, and reporting of employees' salaries efficiently and accurately. A **Payroll Management** software helps streamline and centralizes the salary payments of your organization. This is a window in which we can generate the tax payable amount, net pay to the employee, the amount payable for his loan, amount payable for pension, amount payable for NI payment.

The right side part of the window contain the gender of employee, post code, and the receipt section. The receipt section contains NI code, NI number(NI means national insurance), payment date, tax payable amount, pensionable amount. The NI code is nothing but Employers use an employee's National Insurance category letter when they run payroll to work out how much they both need to contribute. There was a functional window in right part which contains the operations reset system, wage payment, pay reference, pay code and exit.

- ❖ Reset system operation clears all the details in the window and it is ready for new employee details
- ❖ Wage payment operation will displays all the things like tax, pension, NI payment, total deductions, other payment due etc.
- ❖ Pay reference will display the payment date, NI number, payment reference number
- ❖ Pay code will generate the NI code
- ❖ Exit option is for closing the window
- ❖ Tax amount is 30% of the basic salary
- ❖ 2% of basic salary is for the loan amount
- ❖ 1.2% of basic salary is for pension
- ❖ 2.1% of basic salary is for NI payment
- ❖ Deductions is sum of tax ,pension amount, loan amount,NI payment
- ❖ Net salary=gross salary-deductions
- ❖ Overtime entry for entering amount for his overtime duty
- ❖ Gender option is for gender of employee
- ❖ Employee name and address will contain employee's details
- ❖ City option is for branch of company

## 2.IMPORTING MODULES

For this project we need to import

- tkinter module
- Random module
- Timedate module

### 2.1:tkinter module

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems. Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version. Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the `_tkinter` module for more information about supported versions. Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes, and refer to the official Tcl/Tk documentation for details that are unchanged.

#### 2.1.1: Pygame Methods

<u>pygame.init</u>	—	initialize all imported pygame modules
<u>pygame.quit</u>	—	uninitialize all pygame modules
<u>pygame.error</u>	—	standard pygame exception
<u>pygame.get_error</u>	—	get the current error message
<u>pygame.set_error</u>	—	set the current error message
<u>pygame.get_sdl_version</u>	—	get the version number of SDL
<u>pygame.get_sdl_byteorder</u>	—	get the byte order of SDL
<u>pygame.register_quit</u>	—	register a function to be called when pygame quits
<u>pygame.encode_string</u>	—	Encode a unicode or bytes object
<u>pygame.encode_file_path</u>	—	Encode a unicode or bytes object as a file system path

## 2.2:Random module

Python Random module is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random. This module can be used to perform random actions such as generating random numbers, print random a value for a list or string, etc. For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement. On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available. Almost all module functions depend on the basic function `random()`, which generates a random float uniformly in the semi-open range `[0.0, 1.0)`. Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of  $2^{19937}-1$ . The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes. The functions supplied by this module are actually bound methods of a hidden instance of the `random.Random` class. You can instantiate your own instances of `Random` to get generators that don't share state. Class `Random` can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the `random()`, `seed()`, `getstate()`, and `setstate()` methods. Optionally, a new generator can supply a `getrandbits()` method — this allows `randrange()` to produce selections over an arbitrarily large range.

## 2.3:datetime module

For applications requiring aware objects, `datetime` and `time` objects have an optional time zone information attribute, `tzinfo`, that can be set to an instance of a subclass of the abstract `tzinfo` class. These `tzinfo` objects capture information about the offset from UTC time, the time zone name, and whether daylight saving time is in effect. Only one concrete `tzinfo` class, the `timezone` class, is supplied by the `datetime` module. The `timezone` class can represent simple timezones with fixed offsets from UTC, such as UTC itself or North American EST and EDT timezones. Supporting timezones at deeper levels of detail is up to the application. The rules for time adjustment across the world are more political than rational, change frequently, and there is no standard suitable for every application aside from UTC.

### Available types

- class `datetime.date`

An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Attributes: `year`, `month`, and `day`.

- `class datetime.time`

An idealized time, independent of any particular day, assuming that every day has exactly  $24*60*60$  seconds. (There is no notion of “leap seconds” here.)

Attributes: hour, minute, second, microsecond, and tzinfo.

- `class datetime.datetime`

A combination of a date and a time.

Attributes: year, month, day, hour, minute, second, microsecond, and tzinfo.

- `class datetime.timedelta`

A duration expressing the difference between two date, time, or datetime instances to microsecond resolution.

- `class datetime.tzinfo`

An abstract base class for time zone information objects. These are used by the datetime and time classes to provide a customizable notion of time adjustment (for example, to account for time zone and/or daylight saving time).

- `class datetime.timezone`

A class that implements the tzinfo abstract base class as a fixed offset from the UTC.

## 3. WORKING

### 3.1:importing packages

```
from tkinter import *  
import random  
import time  
import datetime  
from tkinter import messagebox
```

### 3.2:creation of window

```
payroll = Tk()  
payroll.geometry("1300x650")  
payroll.resizable(0, 0)  
payroll.title("Payroll Management Systems")
```

### 3.3:creation of all objects

```
def exit():  
    payroll.destroy()  
  
def reset():  
    EmployeeName.set("")  
    Address.set("")  
    Reference.set("")  
    EmployerName.set("")  
    City.set("")  
    Basic.set("")  
    OverTime.set("")  
    GrossPay.set("")  
    NetPay.set("")  
    Tax.set("")  
    PostCode.set("")  
    Gender.set("")  
    PayDate.set("")  
    Pension.set("")  
    StudenLoan.set("")  
    NIPayment.set("")  
    Deductions.set("")  
    TaxPeriod.set("")  
    NINumber.set("")  
    NICode.set("")
```

```
TaxablePay.set("")
PensionablePay.set("")
OtherPaymentDue.set("")
```

### **3.4:main logic(calculations)**

```
MTax = ((BS + CW + OT) * 0.3)
TTax =str('% .2f' % ((MTax)))
Tax.set(TTax)
```

```
M_StudenLoan = ((BS + CW + OT) * 0.02)
MM_StudenLoan =str('% .2f' % ((M_StudenLoan)))
StudenLoan.set(MM_StudenLoan)
```

```
M_Pension = ((BS + CW + OT) * 0.012)
MM_Pension =str('% .2f' % ((M_Pension)))
Pension.set(MM_Pension)
```

```
M_NIPayment = ((BS + CW + OT) * 0.021)
MM_NIPayment =str('% .2f' % ((M_NIPayment)))
NIPayment.set(MM_NIPayment)
```

```
Deduct = MTax + M_Pension + M_StudenLoan + M_NIPayment
Deducat_Payment =str('% .2f' % ((Deduct)))
Deducations.set(Deducat_Payment)
```

```
NetPayAfter = ((BS + CW + OT) - Deduct)
NetAfter =str('% .2f' % ((NetPayAfter)))
NetPay.set(NetAfter)
```

```
Gross_Pay =str('% .2f' % (BS + CW + OT))
GrossPay.set(Gross_Pay)
```

```
TaxablePay.set(TTax)
PensionablePay.set(MM_Pension)
OtherPaymentDue.set("0.00")
```

### 3.5:creating all icons

```
lblEmployeeName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employee Name",  
fg="Steel blue", bd=10, anchor="w")  
lblEmployeeName.grid(row=0, column=0)  
txtEmployeeName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder  
blue", justify="left", textvariable = EmployeeName)  
txtEmployeeName.grid(row=0, column=1)
```

```
lblAddress = Label(InsideLF, font=('arial', 12, 'bold'), text="Address", fg="Steel blue", bd=10,  
anchor="w")  
lblAddress.grid(row=1, column=0)  
txtAddress = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue",  
justify="left", textvariable = Address)  
txtAddress.grid(row=1, column=1)
```

```
lblReference = Label(InsideLF, font=('arial', 12, 'bold'), text="Reference", fg="Steel blue",  
bd=10, anchor="w")  
lblReference.grid(row=2, column=0)  
txtReference = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue",  
justify="left", textvariable = Reference)  
txtReference.grid(row=2, column=1)  
lblCity = Label(InsideLFL, font=('arial', 12, 'bold'), text="City", fg="Steel blue", bd=10,  
anchor="w")
```

```
lblBasic = Label(InsideLFL, font=('arial', 12, 'bold'), text="Basic Salary", fg="Steel blue",  
bd=10, anchor="w")  
lblBasic.grid(row=1, column=0)  
txtBasic = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="right", textvariable = Basic)  
txtBasic.grid(row=1, column=1)
```

```
lblOverTime = Label(InsideLFL, font=('arial', 12, 'bold'), text="Over Time", fg="Steel blue",  
bd=10, anchor="w")  
lblOverTime.grid(row=2, column=0)  
txtOverTime = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="right", textvariable = OverTime)  
txtOverTime.grid(row=2, column=1)
```

```
lblGrossPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Gross Pay", fg="Steel blue",
bd=10, anchor="w")
lblGrossPay.grid(row=3, column=0)
lblGrossPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = GrossPay)
lblGrossPay.grid(row=3, column=1)
```

```
lblNetPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Net Pay", fg="Steel blue", bd=10,
anchor="w")
lblNetPay.grid(row=4, column=0)
lblNetPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = NetPay)
lblNetPay.grid(row=4, column=1)
lblTax = Label(InsideLFR, font=('arial', 12, 'bold'), text="Tax", fg="Steel blue", bd=10,
anchor="w")
lblTax.grid(row=0, column=0)
txtTax = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = Tax)
txtTax.grid(row=0, column=1)
```

```
lblPension = Label(InsideLFR, font=('arial', 12, 'bold'), text="Pension", fg="Steel blue",
bd=10, anchor="w")
lblPension.grid(row=1, column=0)
txtPension = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="right", textvariable = Pension)
txtPension.grid(row=1, column=1)
```

```
lblStudenLoan = Label(InsideLFR, font=('arial', 12, 'bold'), text="Loan", fg="Steel blue",
bd=10, anchor="w")
lblStudenLoan.grid(row=2, column=0)
txtStudenLoan = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder
blue", justify="right", textvariable = StudenLoan)
txtStudenLoan.grid(row=2, column=1)
```

```
lblNIPavment = Label(InsideLFR, font=('arial', 12, 'bold'), text="NI Payment", fg="Steel
blue", bd=10, anchor="w")
lblNIPavment.grid(row=3, column=0)
txtNIPavment = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder
blue", justify="right", textvariable = NIPayment)
txtNIPavment.grid(row=3, column=1)
```



```

lblDeducations = Label(InsideLFR, font=('arial', 12, 'bold'), text="Deducations", fg="Steel
blue", bd=10, anchor="w")
lblDeducations.grid(row=4, column=0)
txtDeducations = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder
blue", justify="right", textvariable = Deducations)
txtDeducations.grid(row=4, column=1)
lblGender = Label(InsideRF, font=('arial', 12, 'bold'), text="Gender", fg="Steel blue", bd=10,
anchor="w")
lblGender.grid(row=1, column=0)
txtGender = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="powder blue",
justify="right", textvariable = Gender)
txtGender.grid(row=1, column=1)

lblPayDate = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pay Date", fg="Steel blue",
bd=10, anchor="w")

lblPayDate.grid(row=0, column=0)

txtPayDate = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = PayDate)

txtPayDate.grid(row=0, column=1)

lblTaxPeriod = Label(InsideRFL, font=('arial', 12, 'bold'), text="Tax Period", fg="Steel blue",
bd=10, anchor="w")

lblTaxPeriod.grid(row=1, column=0)

txtTaxPeriod = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = TaxPeriod)

txtTaxPeriod.grid(row=1, column=1)

lblNINumber = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Number", fg="Steel blue",
bd=10, anchor="w")

lblNINumber.grid(row=2, column=0)

txtNINumber = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",
justify="left", textvariable = NINumber)

```

```
txtNINumber.grid(row=2, column=1)
```

```
lblNICode = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Code", fg="Steel blue", bd=10, anchor="w")
```

```
lblNICode.grid(row=3, column=0)
```

```
txtNICode = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = NICode)
```

```
txtNICode.grid(row=3, column=1)
```

```
lblTaxablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Taxable Pay ", fg="Steel blue", bd=10, anchor="w")
```

```
lblTaxablePay .grid(row=4, column=0)
```

```
txtTaxablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = TaxablePay)
```

```
txtTaxablePay .grid(row=4, column=1)
```

```
lblPensionablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pensionable Pay", fg="Steel blue", bd=10, anchor="w")
```

```
lblPensionablePay.grid(row=5, column=0)
```

```
txtPensionablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = PensionablePay)
```

```
txtPensionablePay.grid(row=5, column=1)
```

```
lblOtherPaymentDue = Label(InsideRFL, font=('arial', 12, 'bold'), text="Other Payment Due", fg="Steel blue", bd=10, anchor="w")
```

```
lblOtherPaymentDue.grid(row=6, column=0)
```

```
txtOtherPaymentDue = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = OtherPaymentDue)
```

```
txtOtherPaymentDue.grid(row=6, column=1)
```

### **3.6:creation of buttons for operations**

```
btnWagePayment = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),  
width=14,text="Wage Paymant", bg="sky blue", command=MonthlySalary).grid(row=0,  
column=0)
```

```
btnReset = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),  
width=14,text="Reset System", bg="sky blue", command=reset).grid(row=1, column=0)
```

```
btnPayRef = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),  
width=14,text="Pay Reference", bg="sky blue", command=PayRef).grid(row=2, column=0)
```

```
btnPayCode = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),  
width=14,text="Pay Code", bg="sky blue", command=PayPeriod).grid(row=3, column=0)
```

```
btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),  
width=14,text="Exit", bg="sky blue", command=exit).grid(row=4, column=0)
```

## 4. IMPLEMENTATION

### 4.1 CODING

```
from tkinter import *  
  
import random  
  
import time  
  
import datetime  
  
from tkinter import messagebox  
  
payroll = Tk()  
  
payroll.geometry("1300x650")  
  
payroll.resizable(0, 0)  
  
payroll.title("Payroll Management Systems")  
  
def exit():  
    payroll.destroy()  
  
def reset():  
    EmployeeName.set("")  
    Address.set("")  
    Reference.set("")  
    EmployerName.set("")  
    City.set("")  
    Basic.set("")  
    OverTime.set("")
```

GrossPay.set("")

NetPay.set("")

Tax.set("")

PostCode.set("")

Gender.set("")

PayDate.set("")

Pension.set("")

StudenLoan.set("")

NIPayment.set("")

Deducations.set("")

TaxPeriod.set("")

NINumber.set("")

NICode.set("")

TaxablePay.set("")

PensionablePay.set("")

OtherPaymentDue.set("")

def PayRef():

PayDate.set(time.strftime("%d/%m/%Y"))

refPay = random.randint(20000, 709467)

refPaid = ("PR" + str(refPay))

Reference.set(refPaid)

NIPay = random.randint(20000, 559467)

NIPaid = ("NI" + str(NIPay))

```

NINumber.set(NIPaid)

def PayPeriod():

    i = datetime.datetime.now()

    TaxPeriod.set(i.month)

    NCode = random.randint(1200, 3467)

    CodeNI = ("NICode" + str(NCode))

    NICode.set(CodeNI)

def MonthlySalary():

    if Basic.get() == "":

        BS = 0

    else:

        try:

            BS = float(Basic.get())

        except ValueError:

            messagebox.showinfo("Error", "Wrong values!!! Use numbers.")

            Basic.set("")

    if City.get() == "":

        CW = 0

    else:

        try:

            CW = float(City.get())

        except ValueError:

            messagebox.showinfo("Error", "Wrong values!!! Use numbers.")

```

```

        City.set("")

if OverTime.get() == "":

    OT = 0

else:

    try:

        OT = float(OverTime.get())

    except ValueError:

        messagebox.showinfo("Error", "Wrong values!!! Use numbers.")

        OverTime.set("")

MTax = ((BS + CW + OT) * 0.3)

TTax =str('%.2f' % ((MTax)))

Tax.set(TTax)

M_StudenLoan = ((BS + CW + OT) * 0.02)

MM_StudenLoan =str('%.2f' % ((M_StudenLoan)))

StudenLoan.set(MM_StudenLoan)

M_Pension = ((BS + CW + OT) * 0.012)

MM_Pension =str('%.2f' % ((M_Pension)))

Pension.set(MM_Pension)

M_NIPayment = ((BS + CW + OT) * 0.021)

MM_NIPayment =str('%.2f' % ((M_NIPayment)))

NIPayment.set(MM_NIPayment)

Deduct = MTax + M_Pension + M_StudenLoan + M_NIPayment

Deducat_Payment =str('%.2f' % ((Deduct)))

```

Deducations.set(Deducat\_Payment)

NetPayAfter = ((BS + CW + OT) - Deduct)

NetAfter =str('% .2f' % ((NetPayAfter)))

NetPay.set(NetAfter)

Gross\_Pay =str('% .2f' % (BS + CW + OT))

GrossPay.set(Gross\_Pay)

TaxablePay.set(TTax)

PensionablePay.set(MM\_Pension)

OtherPaymentDue.set("0.00")

EmployeeName = StringVar()

Address = StringVar()

Reference = StringVar()

EmployerName = StringVar()

City = StringVar()

Basic = StringVar()

OverTime = StringVar()

GrossPay = StringVar()

NetPay = StringVar()

Tax = StringVar()

PostCode = StringVar()

Gender = StringVar()

PayDate = StringVar()

Pension = StringVar()



```

StudenLoan = StringVar()

NIPayment = StringVar()

Deducations = StringVar()

TaxPeriod = StringVar()

NINumber = StringVar()

NICode = StringVar()

TaxablePay = StringVar()

PensionablePay = StringVar()

OtherPaymentDue = StringVar()

textInput = StringVar()

Tops=Frame(payroll, width=1300, height=50, bd=16, relief="raise")

Tops.pack(side=TOP)

LF=Frame(payroll, width=700, height=650, bd=12, relief="raise")

LF.pack(side=LEFT)

RF=Frame(payroll, width=600, height=650, bd=12, relief="raise")

RF.pack(side=RIGHT)

#=====

lblTitle = Label(Tops, font=('arial', 50, 'bold'), text="Payroll Management Systems", fg="Steel
blue", bd=10, anchor="w")

lblTitle.grid(row=0, column=0)

InsideLF=Frame(LF, width=700, height=100, bd=8, relief="raise")

InsideLF.pack(side=TOP)

InsideLFL=Frame(LF, width=325, height=400, bd=8, relief="raise")

InsideLFL.pack(side=LEFT)

```

```
InsideLFR=Frame(LF, width=325, height=400, bd=8, relief="raise")
```

```
InsideLFR.pack(side=RIGHT)
```

```
#=====
```

```
InsideRF=Frame(RF, width=600, height=200, bd=8, relief="raise")
```

```
InsideRF.pack(side=TOP)
```

```
InsideRFL=Frame(RF, width=300, height=400, bd=8, relief="raise")
```

```
InsideRFL.pack(side=LEFT)
```

```
InsideRFR=Frame(RF, width=300, height=400, bd=8, relief="raise")
```

```
InsideRFR.pack(side=RIGHT)
```

```
#=====Left Side
```

```
lblEmployeeName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employee Name", fg="Steel  
blue", bd=10, anchor="w")
```

```
lblEmployeeName.grid(row=0, column=0)
```

```
txtEmployeeName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder  
blue", justify="left", textvariable = EmployeeName)
```

```
txtEmployeeName.grid(row=0, column=1)
```

```
lblAddress = Label(InsideLF, font=('arial', 12, 'bold'), text="Address", fg="Steel blue", bd=10,  
anchor="w")
```

```
lblAddress.grid(row=1, column=0)
```

```
txtAddress = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue",  
justify="left", textvariable = Address)
```

```
txtAddress.grid(row=1, column=1)
```

```
lblReference = Label(InsideLF, font=('arial', 12, 'bold'), text="Reference", fg="Steel blue",  
bd=10, anchor="w")
```

```
lblReference.grid(row=2, column=0)
```

```
txtReference = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue",  
justify="left", textvariable = Reference)
```

```
txtReference.grid(row=2, column=1)
```

```
#-----Left Left Side
```

```
lblBasic = Label(InsideLFL, font=('arial', 12, 'bold'), text="Basic Salary", fg="Steel blue",  
bd=10, anchor="w")
```

```
lblBasic.grid(row=1, column=0)
```

```
txtBasic = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="right", textvariable = Basic)
```

```
txtBasic.grid(row=1, column=1)
```

```
lblOverTime = Label(InsideLFL, font=('arial', 12, 'bold'), text="Over Time", fg="Steel blue",  
bd=10, anchor="w")
```

```
lblOverTime.grid(row=2, column=0)
```

```
txtOverTime = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="right", textvariable = OverTime)
```

```
txtOverTime.grid(row=2, column=1)
```

```
lblGrossPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Gross Pay", fg="Steel blue",  
bd=10, anchor="w")
```

```
lblGrossPay.grid(row=3, column=0)
```

```
lblGrossPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="right", textvariable = GrossPay)
```

```
lblGrossPay.grid(row=3, column=1)
```

```
lblNetPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Net Pay", fg="Steel blue", bd=10,  
anchor="w")
```

```
lblNetPay.grid(row=4, column=0)
```

```
lblNetPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="right", textvariable = NetPay)
```

```
lblNetPay.grid(row=4, column=1)
```

#-----Left Right Side

```
lblTax = Label(InsideLFR, font=('arial', 12, 'bold'), text="Tax", fg="Steel blue", bd=10, anchor="w")
```

```
lblTax.grid(row=0, column=0)
```

```
txtTax = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Tax)
```

```
txtTax.grid(row=0, column=1)
```

```
lblPension = Label(InsideLFR, font=('arial', 12, 'bold'), text="Pension", fg="Steel blue", bd=10, anchor="w")
```

```
lblPension.grid(row=1, column=0)
```

```
txtPension = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Pension)
```

```
txtPension.grid(row=1, column=1)
```

```
lblStudenLoan = Label(InsideLFR, font=('arial', 12, 'bold'), text="Loan", fg="Steel blue", bd=10, anchor="w")
```

```
lblStudenLoan.grid(row=2, column=0)
```

```
txtStudenLoan = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = StudenLoan)
```

```
txtStudenLoan.grid(row=2, column=1)
```

```
lblNIPavment = Label(InsideLFR, font=('arial', 12, 'bold'), text="NI Payment", fg="Steel blue", bd=10, anchor="w")
```

```
lblNIPavment.grid(row=3, column=0)
```

```
txtNIPavment = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = NIPayment)
```

```
txtNIPavment.grid(row=3, column=1)
```

```
lblDeducations = Label(InsideLFR, font=('arial', 12, 'bold'), text="Deducations", fg="Steel blue", bd=10, anchor="w")
```

```
lblDeducations.grid(row=4, column=0)
```

```
txtDeducations = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Deducations)
```

```
txtDeducations.grid(row=4, column=1)
```

```
#=====Right Side
```

```
lblGender = Label(InsideRF, font=('arial', 12, 'bold'), text="Gender", fg="Steel blue", bd=10, anchor="w")
```

```
lblGender.grid(row=1, column=0)
```

```
txtGender = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="powder blue", justify="right", textvariable = Gender)
```

```
txtGender.grid(row=1, column=1)
```

```
#-----
```

```
lblPayDate = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pay Date", fg="Steel blue", bd=10, anchor="w")
```

```
lblPayDate.grid(row=0, column=0)
```

```
txtPayDate = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = PayDate)
```

```
txtPayDate.grid(row=0, column=1)
```

```
lblTaxPeriod = Label(InsideRFL, font=('arial', 12, 'bold'), text="Tax Period", fg="Steel blue", bd=10, anchor="w")
```

```
lblTaxPeriod.grid(row=1, column=0)
```

```
txtTaxPeriod = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = TaxPeriod)
```

```
txtTaxPeriod.grid(row=1, column=1)
```

```
lblNINNumber = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Number", fg="Steel blue", bd=10, anchor="w")
```

```
lblNINNumber.grid(row=2, column=0)
```

```
txtNINumber = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="left", textvariable = NINumber)
```

```
txtNINumber.grid(row=2, column=1)
```

```
lblNICode = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Code", fg="Steel blue", bd=10,  
anchor="w")
```

```
lblNICode.grid(row=3, column=0)
```

```
txtNICode = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue",  
justify="left", textvariable = NICode)
```

```
txtNICode.grid(row=3, column=1)
```

```
lblTaxablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Taxable Pay ", fg="Steel blue",  
bd=10, anchor="w")
```

```
lblTaxablePay .grid(row=4, column=0)
```

```
txtTaxablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder  
blue", justify="left", textvariable = TaxablePay)
```

```
txtTaxablePay .grid(row=4, column=1)
```

```
lblPensionablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pensionable Pay",  
fg="Steel blue", bd=10, anchor="w")
```

```
lblPensionablePay.grid(row=5, column=0)
```

```
txtPensionablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder  
blue", justify="left", textvariable = PensionablePay)
```

```
txtPensionablePay.grid(row=5, column=1)
```

```
lblOtherPaymentDue = Label(InsideRFL, font=('arial', 12, 'bold'), text="Other Payment Due",  
fg="Steel blue", bd=10, anchor="w")
```

```
lblOtherPaymentDue.grid(row=6, column=0)
```

```
txtOtherPaymentDue = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder  
blue", justify="left", textvariable = OtherPaymentDue)
```

```
txtOtherPaymentDue.grid(row=6, column=1)
```

```
#-----
```

```
btnWagePayment = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),
width=14,text="Wage Paymant", bg="sky blue", command=MonthlySalary).grid(row=0,
column=0)
```

```
btnReset = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),
width=14,text="Reset System", bg="sky blue", command=reset).grid(row=1, column=0)
```

```
btnPayRef = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),
width=14,text="Pay Reference", bg="sky blue", command=PayRef).grid(row=2, column=0)
```

```
btnPayCode = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),
width=14,text="Pay Code", bg="sky blue", command=PayPeriod).grid(row=3, column=0)
```

```
btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'),
width=14,text="Exit", bg="sky blue", command=exit).grid(row=4, column=0)
```

```
payroll.mainloop()
```

## 5. RESULT

### 5.1:initial window

**Payroll Management Systems**

Employee Name		Gender	
Address		Pay Date	
Reference		Tax Period	
		NI Number	
		NI Code	
		Taxable Pay	
		Pensionable Pay	
		Other Payment Due	

Basic Salary		Tax	
Over Time		Pension	
Gross Pay		Loan	
Net Pay		NI Payment	
		Deductions	

Wage Payment

Reset System

Pay Reference

Pay Code

Exit

### 5.2:after entering details

**Payroll Management Systems**

Employee Name	venkatesh	Gender	male
Address	2-167,addanki	Pay Date	
Reference		Tax Period	
		NI Number	
		NI Code	
		Taxable Pay	
		Pensionable Pay	
		Other Payment Due	

Basic Salary	100000	Tax	
Over Time	1000	Pension	
Gross Pay		Loan	
Net Pay		NI Payment	
		Deductions	

Wage Payment

Reset System

Pay Reference

Pay Code

Exit



### 5.3:after clicking on wage payment

## Payroll Management Systems

Employee Name

venkatesh

Address

2-167,addanki

Reference

Gender

male

Pay Date

Tax Period

NI Number

NI Code

Taxable Pay

30300.00

Pensionable Pay

1212.00

Other Payment Due

0.00

Wage Payment

Reset System

Pay Reference

Pay Code

Exit

Basic Salary

100000

Over Time

1000

Gross Pay

101000.00

Net Pay

65347.00

Tax

30300.00

Pension

1212.00

Loan

2020.00

NI Payment

2121.00

Deducations

35653.00

### 5.4:after selecting pay reference&pay code

## Payroll Management Systems

Employee Name

venkatesh

Address

2-167,addanki

Reference

PR37845

Gender

male

Pay Date

26/01/2022

Tax Period

1

NI Number

NI151630

NI Code

NICode2110

Taxable Pay

30300.00

Pensionable Pay

1212.00

Other Payment Due

0.00

Wage Payment

Reset System

Pay Reference

Pay Code

Exit

Basic Salary

100000

Over Time

1000

Gross Pay

101000.00

Net Pay

65347.00

Tax

30300.00

Pension

1212.00

Loan

2020.00

NI Payment

2121.00

Deducations

35653.00

### 5.5:after reset system

## Payroll Management Systems

Employee Name		Gender	
Address		Pay Date	
Reference		Tax Period	
		NI Number	
		NI Code	
		Taxable Pay	
		Pensionable Pay	
		Other Payment Due	
Basic Salary			Wage Paymant
Over Time			Reset System
Gross Pay			Pay Reference
Net Pay			Pay Code
			Exit
	Tax		
	Pension		
	Loan		
	NI Payment		
	Deducations		

## **6. CONCLUSION**

### **1. Saves Time and lowers error ratio**

Payroll management systems are efficient at handling large employee salary data. It can create and manage multiple salary structures for various seniority levels in your organization. The software automatically calculates salary heads based on the latest IT and Govt. compliance norms and records the data into its servers making it easy to retrieve data. All in all, it largely reduces the time for the entire payroll processes and eliminates the possibility of errors in salary calculation.

### **2. Simplifies the whole Payroll Method**

Calculation of salaries, deductions and incentives are not an easy process, it requires the repetition of the entire process for every employee at every month. To makes things simple a payroll management system requires you to enter the data only for the first time. It automatically calculates and repeats the process for the subsequent months.

### **3. Computerized data eliminates the hiccups of managing huge amounts of paper-based files**

Managing huge data through paper-based files can make a lot of hiccups. It is time-consuming and often makes the processes more complicated than it really is. Implementing a payroll management system replaces all the paper-based files into reliable and secure computer files.

### **4. It's Cost-effective**

A payroll management software avoids hiring more employees to handle the payroll management. It doesn't require any investment in hardware such as servers and physical software packages. Which makes the system cost-effective.

## 7. REFERENCES

- <https://eduxpert.in/payroll-management-system/>
- <https://www.geeksforgeeks.org/python-add-style-to-tkinter-button/>
- <https://www.zimyo.com/insights/functions-of-payroll-management-software/>
- <https://www.iitms.co.in/hr-management-software/>
- <https://www.ukg.com/what-is-payroll-software>