Godavari Institute Of Engineering & Technology

(Department of Computer Engineering)

Project Report

On

EMPLOYEE PAYMENT MANAGEMENT SYSTEM

Subject-: python

Presented By

Roll No	Name
16551A0538	K.MUKTA

ABSTRACT

“EMPLOYEE PAYMENT MANAGEMENT SYSTEM” project is a project implemented it for Employee weekly salary payment with overtime bonus calculate application. and also generate employee payment slip and also include tax.it is very useful to run the business and easily calculate the salaries.

The project “Employee payment Management System” is implemented to reduce the manual work of hour department and give the employee the full details of his working hours.

EMPLOYEE PAYMENT MANAGEMENT SYSTEM is Desktop Application to management. Hence the project is developed proficiently to help companies t easily update their employee data, working hours ,bonus salary and finally provide a slip so that employee can easily understand.

This system entirely reduces the HR department time as well as it can be opened anywhere and check and also add the extra hours worked by the employee.

Payslips are provided to the employees to record their monthly pay and allowances. It is a document that provides the employees with details of their incomes and deductions as it would differ from employee to employee. Manual salary slips are outdated, now as most of them work in technology using companies and the pdf file of a salary slip is received on salary day. Issuing manual salary slip and HRMS system might be time-consuming and might consume a lot of paper. So, proposing an online human resource and payroll management system project which helps in generating salary of the employees every month without worries. This HR payroll software will help in distribution of payslips to employees and also helps to manage employee information efficiently. This payroll system has one HR module. HR can add all the employees by filling their personal details, their salary details and leave bifurcation. HR can upload Monthly Attendance excel File of each Employee. Then salary calculation and leave management will be done automatically by this leave management system on the basis of salary details and attendance details provided by the HR. Then the salary slip will be generated and it will auto mail the payslips to all the employees. HR can also manage employee details like updating new details of existing employee, add new employees and deleting the employee details who has left the company.

EMPLOYEE PAYMENT MANAGEMENT SYSTEM project is developed in python language which is developed using **Tkinter,python3.6** and Database no database. This is a simple and basic small level project .

INDEX

CHAPTER 1-INTRODUCTION

PYTHON

Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − we can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python graphical user interfaces (GUIs)

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWindows http://wxpython.org.
- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

There are many other interfaces available, which you can find them on the net.

PYTHON TKINTER GUI

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

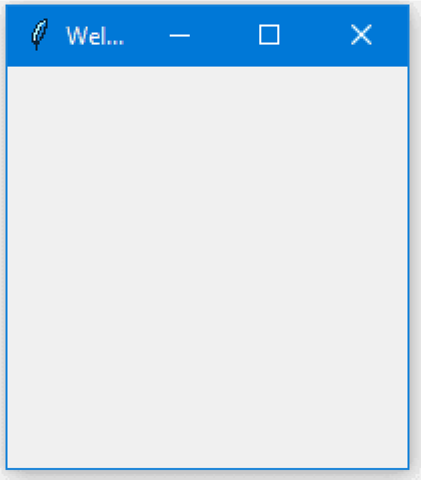
Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python
  import  tkinter
Window=tk()
Window.title(“welcome to python tkinter”)
Window.loop()
```

This would create a following window −



Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr.No.	Operator & Description
1	**Button** The Button widget is used to display buttons in your application.
2	**Canvas** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	**Checkbutton** The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	**Entry** The Entry widget is used to display a single-line text field for accepting values from a user.
5	**Frame** The Frame widget is used as a container widget to organize other widgets.
6	**Label** The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	**Listbox**

	The Listbox widget is used to provide a list of options to a user.
8	**Menubutton** The Menubutton widget is used to display menus in your application.
9	**Menu** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	**Message** The Message widget is used to display multiline text fields for accepting values from a user.
11	**Radiobutton** The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	**Scale** The Scale widget is used to provide a slider widget.
13	**Scrollbar** The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	**Text** The Text widget is used to display text in multiple lines.
15	**Toplevel** The Toplevel widget is used to provide a separate window container.
16	**Spinbox**

	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	**PanedWindow** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	**LabelFrame** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	**tkMessageBox** This module is used to display message boxes in your applications.

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

CHAPTER-2 IMPLEMENTATION

Technologies used - Python 3.6.
Python Tkinter
database

Language used - Python

CODE OF PROJECT

```
import time

import datetime

from tkinter import *

import tkinter.messagebox


root=Tk()

root.title("Employee payroll system")

root.geometry('1350x650+0+0')

root.configure(background="powder blue")


Tops=Frame(root,width=1350,height=50,bd=8,bg="powder blue")

Tops.pack(side=TOP)


f1=Frame(root,width=600,height=600,bd=8,bg="powder blue")
```

```
f1.pack(side=LEFT)
f2=Frame(root,width=300,height=700,bd=8,bg="powder blue")
f2.pack(side=RIGHT)

fla=Frame(f1,width=600,height=200,bd=8,bg="powder blue")
fla.pack(side=TOP)
flb=Frame(f1,width=300,height=600,bd=8,bg="powder blue")
flb.pack(side=TOP)

lblinfo=Label(Tops,font=('arial',45,'bold'),text="Employee Payment Management system ",bd=10,fg="green")
lblinfo.grid(row=0,column=0)

def exit():
  exit=tkinter.messagebox.askyesno("Employee system","Do you want to exit the system")
  if exit>0:
    root.destroy()
    return

def reset():
  Name.set("")
  Address.set("")
  HoursWorked.set("")
  wageshour.set("")
  Payable.set("")
  Taxable.set("")
  NetPayable.set("")
  GrossPayable.set("")
  OverTimeBonus.set("")
  Employer.set("")
  NINumber.set("")
  txtpayslip.delete("1.0",END)
def enterinfo():
  txtpayslip.delete("1.0",END)
  txtpayslip.insert(END,"\t\tPay Slip\n\n")
  txtpayslip.insert(END,"Name :\t\t"+Name.get()+"\n\n")
  txtpayslip.insert(END,"Address :\t\t"+Address.get()+"\n\n")
  txtpayslip.insert(END,"Employer :\t\t"+Employer.get()+"\n\n")
  txtpayslip.insert(END,"NI Number :\t\t"+NINumber.get()+"\n\n")
  txtpayslip.insert(END,"Hours Worked
:\t\t"+HoursWorked.get()+"\n\n")
```

```
 txtpayslip.insert(END,"Net Payable :\t\t"+NetPayable.get()+"\n\n")
 txtpayslip.insert(END,"Wages per hour :\t\t"+wageshour.get()+"\n\n")
 txtpayslip.insert(END,"Tax Paid :\t\t"+Taxable.get()+"\n\n")
 txtpayslip.insert(END,"Payable :\t\t"+Payable.get()+"\n\n")
def weeklywages():
 txtpayslip.delete("1.0",END)
 hoursworkedperweek=float(HoursWorked.get())
 wagesperhours=float(wageshour.get())

 paydue=wagesperhours*hoursworkedperweek
 paymentdue="INR",str('%.2f'%(paydue))
 Payable.set(paymentdue)

 tax=paydue*0.2
 taxable="INR",str('%.2f'%(tax))
 Taxable.set(taxable)

 netpay=paydue-tax
 netpays="INR",str('%.2f'%(netpay))
 NetPayable.set(netpays)
 if hoursworkedperweek > 40:
  overtimehours=(hoursworkedperweek-40)+wagesperhours*1.5
  overtime="INR",str('%.2f'%(overtimehours))
  OverTimeBonus.set(overtime)
 elif hoursworkedperweek<=40:
  overtimepay=(hoursworkedperweek-40)+wagesperhours*1.5
  overtimehrs="INR",str('%.2f'%(overtimepay))
  OverTimeBonus.set(overtimehrs)
 return

#================================Variables=====================
Name=StringVar()
Address=StringVar()
HoursWorked=StringVar()
wageshour=StringVar()
Payable=StringVar()
Taxable=StringVar()
NetPayable=StringVar()
GrossPayable=StringVar()
OverTimeBonus=StringVar()
Employer=StringVar()
```

```
NINumber=StringVar()
TimeOfOrder=StringVar()
DateOfOrder=StringVar()

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

#================================LabelWidget===============

lblName=Label(fla,text="Name",font=('arial',16,'bold'),bd=20,fg="red",b
g="powder blue").grid(row=0,column=0)
lblAddress=Label(fla,text="Address",font=('arial',16,'bold'),bd=20,fg="re
d",bg="powder blue").grid(row=0,column=2)
lblEmployer=Label(fla,text="Employer",font=('arial',16,'bold'),bd=20,fg=
"red",bg="powder blue").grid(row=1,column=0)
lblNINumber=Label(fla,text="NI
Number",font=('arial',16,'bold'),bd=20,fg="red",bg="powder
blue").grid(row=1,column=2)
lblHoursWorked=Label(fla,text="Hours
Worked",font=('arial',16,'bold'),bd=20,fg="red",bg="powder
blue").grid(row=2,column=0)
```

```
lblHourlyRate=Label(fla,text="Hourly Rate",font=('arial',16,'bold'),bd=20,fg="red",bg="powder blue").grid(row=2,column=2)
lblTax=Label(fla,text="Tax",font=('arial',16,'bold'),bd=20,anchor='w',fg="red",bg="powder blue").grid(row=3,column=0)
lblOverTime=Label(fla,text="OverTime",font=('arial',16,'bold'),bd=20,fg="red",bg="powder blue").grid(row=3,column=2)
lblGrossPay=Label(fla,text="GrossPay",font=('arial',16,'bold'),bd=20,fg="red",bg="powder blue").grid(row=4,column=0)
lblNetPay=Label(fla,text="Net Pay",font=('arial',16,'bold'),bd=20,fg="red",bg="powder blue").grid(row=4,column=2)

#============================EntryWidget================

etxname=Entry(fla,textvariable=Name,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxname.grid(row=0,column=1)

etxaddress=Entry(fla,textvariable=Address,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxaddress.grid(row=0,column=3)
```

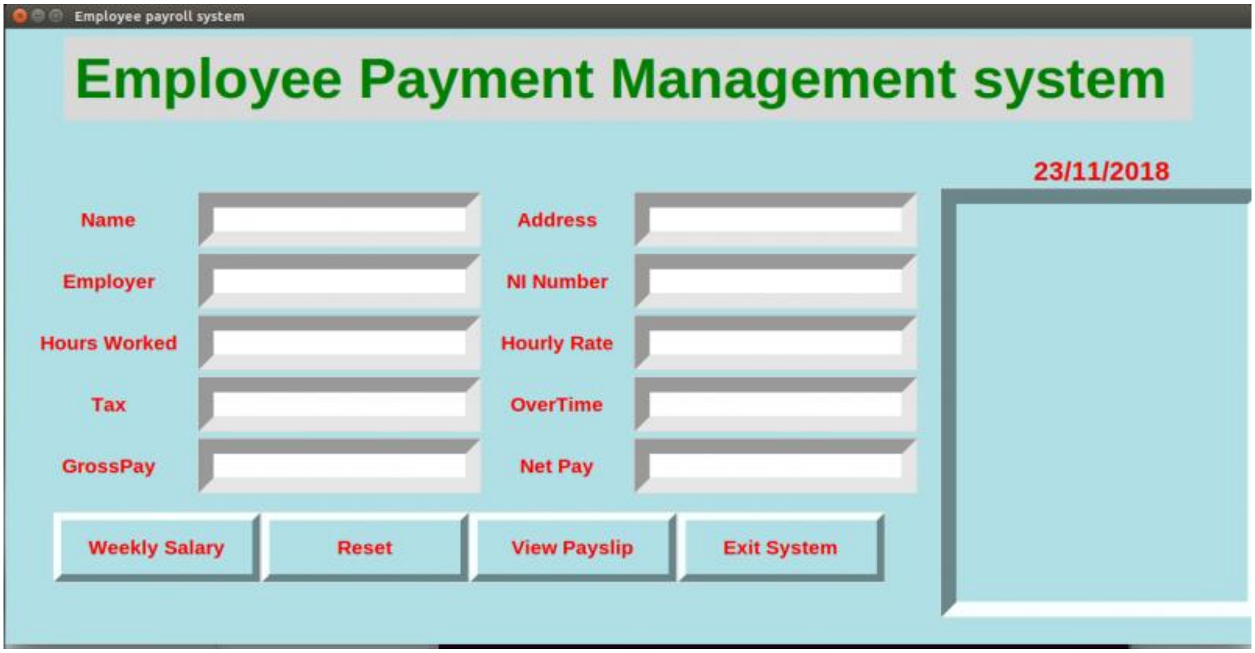
```
etxemployer=Entry(fla,textvariable=Employer,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxemployer.grid(row=1,column=1)

etxhoursworked=Entry(fla,textvariable=HoursWorked,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxhoursworked.grid(row=2,column=1)

etxwagesperhours=Entry(fla,textvariable=wageshour,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxwagesperhours.grid(row=2,column=3)

etxnin=Entry(fla,textvariable=NINumber,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxnin.grid(row=1,column=3)

etxgrosspay=Entry(fla,textvariable=Payable,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxgrosspay.grid(row=4,column=1)
```

```
etxnetpay=Entry(fla,textvariable=NetPayable,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxnetpay.grid(row=4,column=3)

etxtax=Entry(fla,textvariable=Taxable,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxtax.grid(row=3,column=1)

etxovertime=Entry(fla,textvariable=OverTimeBonus,font=('arial',16,'bold'),bd=16,width=22,justify='left')
etxovertime.grid(row=3,column=3)

#===================TextWidget=============================

payslip=Label(f2,textvariable=DateOfOrder,font=('arial',21,'bold'),fg="red",bg="powder blue").grid(row=0,column=0)
txtpayslip=Text(f2,height=22,width=34,bd=16,font=('arial',13,'bold'),fg="green",bg="powder blue")
txtpayslip.grid(row=1,column=0)
```

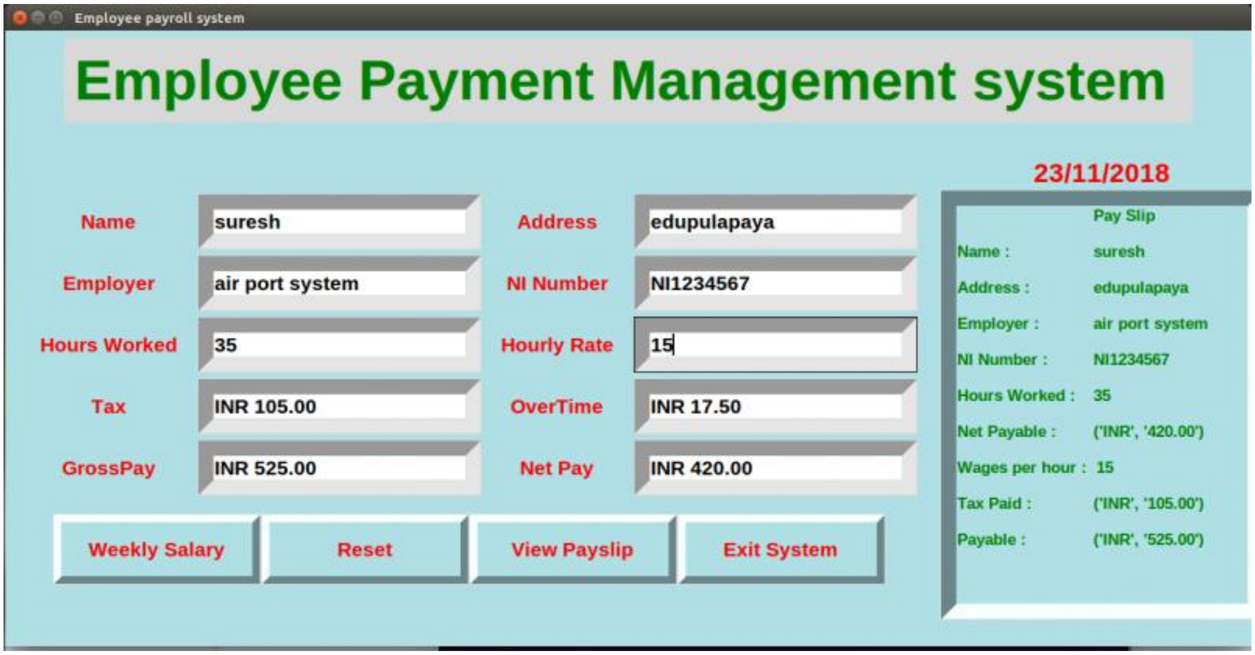
```
#==buttons====================

btnsalary=Button(flb,text='Weekly
Salary',padx=16,pady=16,bd=8,font=('arial',16,'bold'),width=14,fg="red"
,bg="powder blue",command=weeklywages).grid(row=0,column=0)

btnreset=Button(flb,text='Reset',padx=16,pady=16,bd=8,font=('arial',16
,'bold'),width=14,command=reset,fg="red",bg="powder
blue").grid(row=0,column=1)

btnpayslip=Button(flb,text='View
Payslip',padx=16,pady=16,bd=8,font=('arial',16,'bold'),width=14,comma
nd=enterinfo,fg="red",bg="powder blue").grid(row=0,column=2)

btnexit=Button(flb,text='Exit
System',padx=16,pady=16,bd=8,font=('arial',16,'bold'),width=14,comm
and=exit,fg="red",bg="powder blue").grid(row=0,column=3)

root.mainloop()
```

CHAPTER-3 SCREENSHOTS

GUI – Main display window with name of employee payment management system name current time and date with view pay slip



We have entered details:

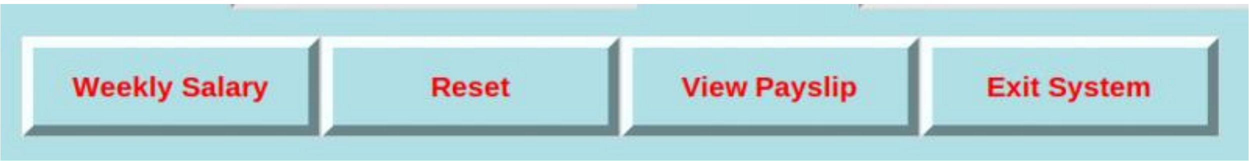


Name	suresh	Address	edupulapaya
Employer	air port system	NI Number	NI1234567
Hours Worked	35	Hourly Rate	15
Tax	INR 105.00	OverTime	INR 17.50
GrossPay	INR 525.00	Net Pay	INR 420.00

Pay Slip

```
Name :            suresh
Address :         edupulapaya
Employer :        air port system
NI Number :       NI1234567
Hours Worked :    35
Net Payable :     ('INR', '420.00')
Wages per hour :  15
Tax Paid :        ('INR', '105.00')
Payable :         ('INR', '525.00')
```

Buttons –

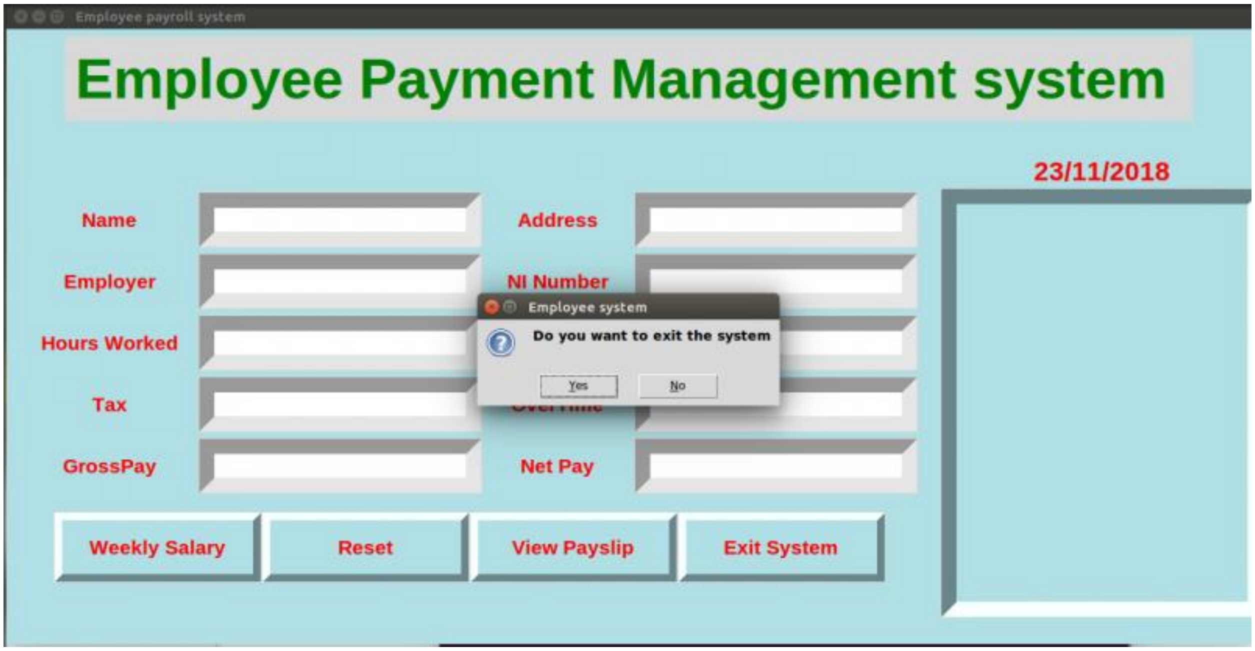


Given weekly salary

Reset button resets all the values and pay slips to view

Exit button closes the window

It also asks weather you want to exit or not:



Do you want to exit the system

CHAPTER-4 CONCLUSION

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a user friendly application.

The fast growing use of internet confirms the good future and scope of the proposed project.

Finally it has taught us a valuable lifelong lesson about the improvements and working and interacting in a group.