

TRAINING REPORT
ON
PAYROLL MANAGEMENT
SYSTEM

Submitted to MAHARAJA RANJIT SINGH PUNJAB TECHNICAL UNIVERSITY
in partial fulfillment of the requirement for the award of the degree of

B. TECH
In
COMPUTER SCIENCE & ENGINEERING

Submitted By
YATIN
Roll. No. 190280124



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GIANI ZAIL SINGH CAMPUS COLLEGE OF ENGINEERING &
TECHNOLOGY, MRSPTU, BATHINDA-151001
DEC 2022

TRAINING REPORT

On

PAYROLL MANAGEMENT SYSTEM

Submitted to MAHARAJA RANJIT SINGH PUNJAB TECHNICAL
UNIVERSITY in partial fulfillment of the requirement for the award of the
degree of

B.TECH

in

COMPUTER SCIENCE & ENGINEERING

Submitted By

YATIN

Roll No. 190280124



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**GIANI ZAIL SINGH CAMPUS COLLEGE OF ENGINEERING &
TECHNOLOGY, MRSPTU, BATHINDA-151001**

DEC 2022

PREFACE

Training is an integral part of B.Tech and each and every student has to undergo the training for four weeks in a company.

This record is concerned about our practical training during the **7th** semester of our B.Tech. We have taken our Practical training in **INFOWIZ**. During this training, we got to learn many new things about the industry and the current requirements of companies. This training proved to be a milestone in our knowledge of present industry. Every say and every moment was an experience in itself, an experience which theoretical study can't provide.

ACKNOWLEDGEMENT

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

I express my sincere gratitude to ***Er. Jyoti Rani*** worthy HOD and ***Er. Manpreet Kaur***, Department Training & Placement In-charge for providing me an opportunity to undergo summer training at **INFOWIZ**.

I am thankful to **Er. Manpreet Kaur** for his support, cooperation, and motivation provided to me during the training for constant inspiration, presence and blessings.

Lastly, I would like to thank the almighty and my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that my quality of work.

YATIN

CANDIDATE'S DECLARATION

I, **YATIN**, Roll No. 190280124 , B.Tech (Semester- VII) of the **Gaini Zail Singh Campus College of Engineering & Technology, Maharaja Ranjit Singh Punjab Technical University, Bathinda** hereby declare that the Training Report entitled “**PAYROLL MANAGEMENT SYSTEM**” is an original work and data provided in the study is authentic to the best of my knowledge. This report has not been submitted to any other Institute for the award of any other degree.

YATIN
(190280124)



H.O.: SCO 118 - 120, Sector 34 - A, CHANDIGARH

B.O.: First Floor, Crown Tower, 100 Ft. Road, BATHINDA

Web.: www.infowiz.co.in

E-mail : info@infowiz.co.in



Certificate

No. INFOWIZ/6W2022/756This is certified that Mr./Ms. YATIN S/D/o. Sh. MUKESH KUMARof GZSCCET, Bathinda has successfully undergone Training Course PYTHONFrom 16th JULY 2022 to 16th AUG. 2022. During the tenure of the above course, we found him/her a hardworking & innovative individual.

We wish him/her a very bright and prosperous future.

Nisha
Technical Head



Managing Director

Chandigarh : 0172 4567888, 9023400888, 96460 00952

Bathinda : 0164 5007088, 90235 00888, 90236 00888

TABLE OF CONTENTS

1. INTRODUCTION.....	01
1.1 PAYROLL MANAGEMENT SYSTEM.....	01
2. LIBRARIES USED.....	03
2.1 TKINTER.....	03
2.2 RANDOM.....	07
2.3 DB.....	07
2.4 DATETIME.....	08
2.5 TIME.....	08
3. GRAPHICAL USER INTERFACE.....	09
3.1 WHAT IS GUI.....	09
3.2 WHAT IS PYTHON GUI.....	09
3.3 PYTHON GUI USES.....	09
4. PYTHON CODE FOR PMS.....	11
5. CONCLUSION.....	20

CHAPTER 1

INTRODUCTION TO PAYROLL MANAGEMENT SYSTEM

1.1 Payroll management system:

A payroll management system is a software program that assists businesses in efficiently and accurately calculating, disbursing, and reporting employee salaries.

Payroll is one of the most crucial components of any company. It has an impact on staff morale and represents the financial health and repute of a company. Because employees rely on their wages, inaccuracies or late payments can erode trust.

1.1.1 Beneficiary of Payroll Management System Using Python

You should have a payroll system in place if your company has one or more employees. An automated payroll system assists you in meeting legal and tax obligations while also simplifying the process of paying your personnel.

1.1.2 How to create a payroll management system for your small business?

1. **Choose the apt payroll software** – Firstly, choose the best payroll software that aligns with your payroll needs.
2. **Gather information for payroll** – After, you select the best payroll software, including all the information your payroll system asks for.
3. **Calculate the Gross Pay and Net Pay** – Now, you have all the information to process payroll; you can begin calculating the pay checks. Gather timesheet data, including the overtime, check whether your payroll software integrates with time tracking software or does it already have one.

4. **Include the Employer Payroll Taxes** – Employers need to pay taxes based on employee compensation, including Federal Unemployment Tax (FUTA), State Unemployment Tax (SUTA), and other local taxes.
5. **Create a payroll entry system** – The best payroll software integrates with the accounting software to facilitate payroll journal entries. This includes employee wages, payroll taxes, and payroll deductions. When you send out a paycheck, attach the paystubs no matter what mode of payment you use. This way, employees can cross-check for errors.
6. **Document the compensation terms** – After you have made payments, check for the tax payments every quarterly to ensure whether you pay state taxes on behalf of the employees living in another state. For this, your employees need a state tax ID, which you can get from the concerned state tax website. Mostly, you must make the first payment manually to facilitate automatic payment through software.
7. **Generate the payroll documents** – Once all this payroll is done, there is still one important step. Every year, you need to send the previous earning reports such as W2, W1, W4, and 1099 forms to employees and Internal Revenue Services (IRS).
8. **The Bottom-line** – We hope this roadmap helps you in creating a payroll management system for your small business effectively. Kindly let us know your experience.

The **Payroll Management System Project in Python** was created with Python programming and a Graphical User Interface (GUI). This project is simple to handle and understand by users, and it is also suitable for novices or students interested in learning Python programming

CHAPTER 2

LIBRARIES USED

2.1. Tkinter

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.

2.1.1 Tkinter widgets

There are various widgets like button, canvas, check button, entry, etc. that are used to build the python GUI application.

SN	Widget	Description
1	<u>Button</u>	The Button is used to add various kinds of buttons to the python application.
2	<u>Canvas</u>	The canvas widget is used to draw the canvas on the window.
3	<u>Checkbutton</u>	The Checkbutton is used to display the CheckButton on the window.
4	<u>Entry</u>	The entry widget is used to display the single-line text field to the user. It

		is commonly used to accept user values.
5	<u>Frame</u>	It can be defined as a container to which, another widget can be added and organized.
6	<u>Label</u>	A label is a text used to display some message or information about the other widgets.
7	<u>ListBox</u>	The ListBox widget is used to display a list of options to the user.
8	<u>Menubutton</u>	The Menubutton is used to display the menu items to the user.
9	<u>Menu</u>	It is used to add menu items to the user.
10	<u>Message</u>	The Message widget is used to display the message-box to the user.
11	<u>Radiobutton</u>	The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
12	<u>Scale</u>	It is used to provide the slider to the user.
13	<u>Scrollbar</u>	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	<u>Text</u>	It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
14	<u>Toplevel</u>	It is used to create a separate window container.
15	<u>Spinbox</u>	It is an entry widget used to select from options of values.
16	<u>PanedWindow</u>	It is like a container widget that contains horizontal or vertical panes.
17	<u>LabelFrame</u>	A LabelFrame is a container widget that acts as the container
18	<u>MessageBox</u>	This module is used to display the message-box in the desktop based

		applications.
--	--	---------------

2.1.2 Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

2.1.2.1 Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.

`widget.pack(options)`

A list of possible options that can be passed in pack() is given below.

- **expand:** If the expand is set to true, the widget expands to fill any space.
- **Fill:** By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
- **size:** it represents the side of the parent to which the widget is to be placed on the window.

2.1.2.2 Python Tkinter grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

This is a more organized way to place the widgets to the python application.

`widget.grid(options)`

A list of possible options that can be passed inside the grid() method is given below.

- **Column**

The column number in which the widget is to be placed. The leftmost column is represented by 0.

- **Columnspan**

The width of the widget. It represents the number of columns up to which, the column is expanded.

- **ipadx,ipady**

It represents the number of pixels to pad the widget inside the widget's border.

- **padx,pady**

It represents the number of pixels to pad the widget outside the widget's border.

- **row**

The row number in which the widget is to be placed. The topmost row is represented by 0.

- **rowspan**

The height of the widget, i.e. the number of the row up to which the widget is expanded.

- **Sticky**

If the cell is larger than a widget, then sticky is used to specify the position of the widget inside the cell. It may be the concatenation of the sticky letters representing the position of the widget. It may be N, E, W, S, NE, NW, NS, EW, ES.

2.1.2.3 Python Tkinter place() method

The place() geometry manager organizes the widgets to the specific x and y coordinates.

`widget.place(options)`

A list of possible options is given below.

- **Anchor:** It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)
- **bordermode:** The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.
- **height, width:** It refers to the height and width in pixels.
- **relheight, relwidth:** It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.
- **relx, rely:** It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

2.2 random

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.

2.3. db

Python supports various databases like SQLite, MySQL, Oracle, Sybase, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements. The Python standard for database interfaces is the Python DB-API.

2.4. time

The Python `time` module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.

2.5. datetime

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 6 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, and `tzinfo`.

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

`tzinfo` – It provides time zone information objects.

`timezone` – A class that implements the `tzinfo` abstract base class as a fixed offset from the UTC

CHAPTER 3

Graphical user interface

3.1 What is gui?

GUI stands for Graphical User Interface, and refers to computer programs that provide a visual means for users to interact with an underlying application or system. For example, the GUIs on our mobile phones allow us to interact with different functions through the display, which we can touch, tap, and swipe on.

3.2 What is a Python GUI?

Very simply, a Python GUI is a GUI that is written in the Python programming language.

Python is a very popular programming language thanks to its great degree of readability, widespread adoption and most importantly, its beginner friendliness. While being incredibly useful for the fields of data science and machine learning, Python is also great for developing graphical user interfaces! In fact, it has many frameworks that even beginners can use to easily get started with developing a GUI.

3.3 Python GUI Uses

You can do many things with Python GUI programming – but here are a few of the popular examples!

3.3.1 Building a Mobile Application

With interfaces for users to like, post, comment, or interact in many kinds of ways, mobile applications are some of the best examples of Python GUIs in action! Did you know that some of these popular and successful mobile applications were written in Python? How many of them do you recognise or use yourself?

3.3.2 Games (Flappy Bird, Mount & Blade)

Apart from mobile applications, Python has been used to create some of the games that we know and love! With flashy graphics and rewarding interactivity, games are one use case that heavily leverage GUIs to create value and enjoyment for users. For example, games like Flappy Bird and Mount & Blade were programmed in Python!

3.3.3 Human Machine Interfaces in Industries

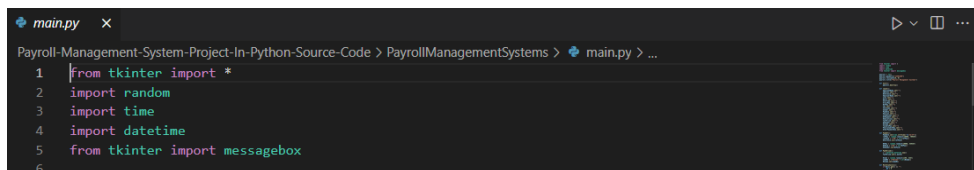
Although GUIs have great uses in entertainment and utility, they also play an important role in industries through Human Machine Interfaces (HMIs). HMIs are GUIs that provide operators with an overview of industrial monitoring and control systems, as well as the means to rectify any anomalies in operating conditions. With Python, it's easy to develop industrial applications at a lower cost, which is an extremely important factor to consider for business.

CHAPTER 4

PYTHON CODE FOR PAYROLL MANAGEMENT

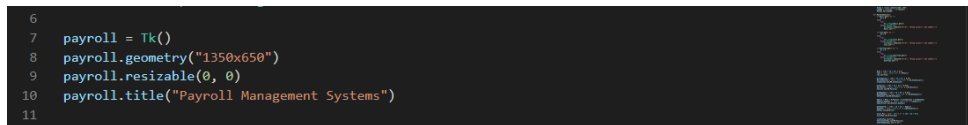
4.1 Importing libraries

First step is importing libraries. Libraries like Tkinter , random, datetime, time are imported in this project.



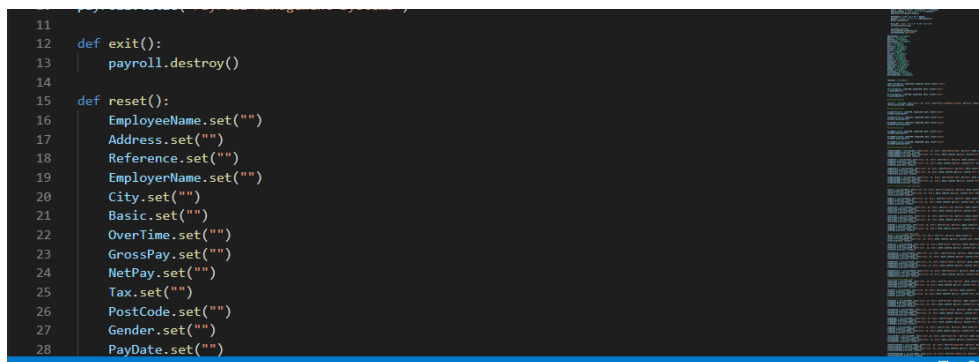
```
1 from tkinter import *
2 import random
3 import time
4 import datetime
5 from tkinter import messagebox
6
```

4.2 Creating gui for payroll management system.



```
6
7 payroll = Tk()
8 payroll.geometry("1350x650")
9 payroll.resizable(0, 0)
10 payroll.title("Payroll Management Systems")
11
```

4.3 Creating a function “exit” and “reset”.



```
11
12 def exit():
13     payroll.destroy()
14
15 def reset():
16     EmployeeName.set("")
17     Address.set("")
18     Reference.set("")
19     EmployerName.set("")
20     City.set("")
21     Basic.set("")
22     OverTime.set("")
23     GrossPay.set("")
24     NetPay.set("")
25     Tax.set("")
26     PostCode.set("")
27     Gender.set("")
28     PayDate.set("")
```

```

14
15 def reset():
16     EmployeeName.set("")
17     Address.set("")
18     Reference.set("")
19     EmployerName.set("")
20     City.set("")
21     Basic.set("")
22     OverTime.set("")
23     GrossPay.set("")
24     NetPay.set("")
25     Tax.set("")
26     PostCode.set("")
27     Gender.set("")
28     PayDate.set("")
29     Pension.set("")
30     StudeLoan.set("")
31     NIPayment.set("")
32     Deductions.set("")
33     TaxPeriod.set("")
34     NINumber.set("")
35     NICode.set("")
36     TaxablePay.set("")
37     PensionablePay.set("")
38     OtherPaymentDue.set("")

```

4.4 Create a function “PayRef”.

```

39
40 def PayRef():
41     PayDate.set(time.strftime("%d/%m/%Y"))
42     refPay = random.randint(20000, 709467)
43     refPaid = ("PR" + str(refPay))
44     Reference.set(refPaid)
45
46     NIPay = random.randint(20000, 559467)
47     NIPaid = ("NI" + str(NIPay))
48     NINumber.set(NIPaid)
49

```

4.5 Create a function “PayPeriod”.

```

49
50 def PayPeriod():
51     i = datetime.datetime.now()
52     TaxPeriod.set(i.month)
53
54     NCode = random.randint(1200, 3467)
55     CodeNI = ("NICode" + str(NCode))
56     NICode.set(CodeNI)
57

```

4.6 Create a function “MonthlySalary” to calculate monthly salary.

```
58 def MonthlySalary():
59     if Basic.get() == "":
60         BS = 0
61     else:
62         try:
63             BS = float(Basic.get())
64         except ValueError:
65             messagebox.showinfo("Error", "Wrong values!!! Use numbers.")
66             Basic.set("")
67
68     if City.get() == "":
69         CW = 0
70     else:
71         try:
72             CW = float(City.get())
73         except ValueError:
74             messagebox.showinfo("Error", "Wrong values!!! Use numbers.")
75             City.set("")
76
77     if OverTime.get() == "":
78         OT = 0
79     else:
80         try:
81             OT = float(OverTime.get())
82         except ValueError:
83             messagebox.showinfo("Error", "Wrong values!!! Use numbers.")
84             OverTime.set("")
85
```

4.7 Calculating values of gross pay, net pay, student loan , deductions, NI payments and taxable pay.

```

90     MTax = ((BS + CW + OT) * 0.3)
91     TTax = "pIn", str("%.2f" % (MTax))
92     Tax.set(TTax)
93
94     M_StudenLoan = ((BS + CW + OT) * 0.02)
95     MM_StudenLoan = "pIn", str("%.2f" % (M_StudenLoan))
96     StudenLoan.set(MM_StudenLoan)
97
98     M_Pension = ((BS + CW + OT) * 0.012)
99     MM_Pension = "pIn", str("%.2f" % (M_Pension))
100    Pension.set(MM_Pension)
101
102    M_NIPayment = ((BS + CW + OT) * 0.021)
103    MM_NIPayment = "pIn", str("%.2f" % (M_NIPayment))
104    NIPayment.set(MM_NIPayment)
105
106    Deduct = MTax + M_Pension + M_StudenLoan + M_NIPayment
107    Deducat_Payment = "pIn", str("%.2f" % (Deduct))
108    Deductions.set(Deducat_Payment)
109
110    NetPayAfter = ((BS + CW + OT) - Deduct)
111    NetAfter = "pIn", str("%.2f" % (NetPayAfter))
112    NetPay.set(NetAfter)
113
114    Gross_Pay = "pIn", str("%.2f" % (BS + CW + OT))
115    GrossPay.set(Gross_Pay)
116
117    TaxablePay.set(TTax)

```

4.8 Creating different variables.

```

120
121 EmployeeName = StringVar()
122 Address = StringVar()
123 Reference = StringVar()
124 EmployerName = StringVar()
125 City = StringVar()
126 Basic = StringVar()
127 OverTime = StringVar()
128 GrossPay = StringVar()
129 NetPay = StringVar()
130 Tax = StringVar()
131 PostCode = StringVar()
132 Gender = StringVar()
133 PayDate = StringVar()
134 Pension = StringVar()
135 StudenLoan = StringVar()
136 NIPayment = StringVar()
137 Deductions = StringVar()
138 TaxPeriod = StringVar()
139 NINumber = StringVar()
140 NICode = StringVar()
141 TaxablePay = StringVar()
142 PensionablePay = StringVar()
143 OtherPaymentDue = StringVar()
144

```

4.9 Creating frame and label for title “payroll management system”.

```

145
146     textInput = StringVar()
147
148     Tops=Frame(payroll, width=1350, height=50, bd=16, relief="raise")
149     Tops.pack(side=TOP)
150
151     LF=Frame(payroll, width=700, height=650, bd=12, relief="raise")
152     LF.pack(side=LEFT)
153
154     RF=Frame(payroll, width=600, height=650, bd=12, relief="raise")
155     RF.pack(side=RIGHT)
156
157     #=====
158
159     lblTitle = Label(Tops, font=('arial', 50, 'bold'), text="Payroll Management Systems", fg="black", bd=10,
160     lblTitle.grid(row=0, column=0)
161
162     #=====
163
164     InsideLF=Frame(LF, width=700, height=100, bd=8, relief="raise")
165     InsideLF.pack(side=TOP)
166
167     InsideLFL=Frame(LF, width=325, height=400, bd=8, relief="raise")
168     InsideLFL.pack(side=LEFT)
169
170     InsideLFR=Frame(LF, width=325, height=400, bd=8, relief="raise")
171     InsideLFR.pack(side=RIGHT)
172
173
174
175
176
177
178
179
180
181
182
183

```

```

157     #=====
158
159     lblTitle = Label(Tops, font=('arial', 50, 'bold'), text="Payroll Management Systems", fg="black", bd=10,
160     lblTitle.grid(row=0, column=0)
161
162     #=====
163
164     InsideLF=Frame(LF, width=700, height=100, bd=8, relief="raise")
165     InsideLF.pack(side=TOP)
166
167     InsideLFL=Frame(LF, width=325, height=400, bd=8, relief="raise")
168     InsideLFL.pack(side=LEFT)
169
170     InsideLFR=Frame(LF, width=325, height=400, bd=8, relief="raise")
171     InsideLFR.pack(side=RIGHT)
172
173     #=====
174
175     InsideRF=Frame(RF, width=600, height=200, bd=8, relief="raise")
176     InsideRF.pack(side=TOP)
177
178     InsideRFL=Frame(RF, width=300, height=400, bd=8, relief="raise")
179     InsideRFL.pack(side=LEFT)
180
181     InsideRFR=Frame(RF, width=300, height=400, bd=8, relief="raise")
182     InsideRFR.pack(side=RIGHT)
183

```

4.10 Creating frames and label for employee name, address, employers name and reference number on the upper left side of gui.

```

183
184 #=====Left Side
185
186 lblEmployeeName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employee Name", fg="black", bd=10, a
187 lblEmployeeName.grid(row=0, column=0)
188 txtEmployeeName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="gray", justify="left"
189 txtEmployeeName.grid(row=0, column=1)
190
191 lblAddress = Label(InsideLF, font=('arial', 12, 'bold'), text="Address", fg="black", bd=10, anchor="w")
192 lblAddress.grid(row=1, column=0)
193 txtAddress = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="gray", justify="left", tex
194 txtAddress.grid(row=1, column=1)
195
196 lblReference = Label(InsideLF, font=('arial', 12, 'bold'), text="Reference", fg="black", bd=10, anchor="
197 lblReference.grid(row=2, column=0)
198 txtReference = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="gray", justify="left", t
199 txtReference.grid(row=2, column=1)
200
201 lblEmployerName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employer Name", fg="black", bd=10, a
202 lblEmployerName.grid(row=3, column=0)
203 txtEmployerName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="gray", justify="left"
204 txtEmployerName.grid(row=3, column=1)
205
206 #-----Left Left Side
207
208 lblCity = Label(InsideLFL, font=('arial', 12, 'bold'), text="City Weighting", fg="black", bd=10, anchor=
209 lblCity.grid(row=0, column=0)
210 txtCity = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right", text

```

4.11 Creating frames and label for city, basic, overtime, gross pay, net pay on the lower left- left side of GUI.

```

206 #-----Left Left Side
207
208 lblCity = Label(InsideLFL, font=('arial', 12, 'bold'), text="City Weighting", fg="black", bd=10, anchor=
209 lblCity.grid(row=0, column=0)
210 txtCity = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right", text
211 txtCity.grid(row=0, column=1)
212
213 lblBasic = Label(InsideLFL, font=('arial', 12, 'bold'), text="Basic Salary", fg="black", bd=10, anchor="
214 lblBasic.grid(row=1, column=0)
215 txtBasic = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right", tex
216 txtBasic.grid(row=1, column=1)
217
218 lblOverTime = Label(InsideLFL, font=('arial', 12, 'bold'), text="Over Time", fg="black", bd=10, anchor="
219 lblOverTime.grid(row=2, column=0)
220 txtOverTime = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right",
221 txtOverTime.grid(row=2, column=1)
222
223 lblGrossPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Gross Pay", fg="black", bd=10, anchor="
224 lblGrossPay.grid(row=3, column=0)
225 lblGrossPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right",
226 lblGrossPay.grid(row=3, column=1)
227
228 lblNetPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Net Pay", fg="black", bd=10, anchor="w")
229 lblNetPay.grid(row=4, column=0)
230 lblNetPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right", te
231 lblNetPay.grid(row=4, column=1)
232

```

4.12 Creating frames and label for tax, pension, student loan, NI payments, deductions on the lower left-right side of gui.

```
232
233 #-----Left Right Side
234 lblTax = Label(InsideLFR, font=('arial', 12, 'bold'), text="Tax", fg="black", bd=10, anchor="w")
235 lblTax.grid(row=0, column=0)
236 txtTax = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right", textv
237 txtTax.grid(row=0, column=1)
238
239 lblPension = Label(InsideLFR, font=('arial', 12, 'bold'), text="Pension", fg="black", bd=10, anchor="w")
240 lblPension.grid(row=1, column=0)
241 txtPension = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right", t
242 txtPension.grid(row=1, column=1)
243
244 lblStudenLoan = Label(InsideLFR, font=('arial', 12, 'bold'), text="StudenLoan", fg="black", bd=10, ancho
245 lblStudenLoan.grid(row=2, column=0)
246 txtStudenLoan = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right"
247 txtStudenLoan.grid(row=2, column=1)
248
249 lblNIPayment = Label(InsideLFR, font=('arial', 12, 'bold'), text="NI Pavment", fg="black", bd=10, anchor
250 lblNIPayment.grid(row=3, column=0)
251 txtNIPayment = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right",
252 txtNIPayment.grid(row=3, column=1)
253
254 lblDeducations = Label(InsideLFR, font=('arial', 12, 'bold'), text="Deducations", fg="black", bd=10, anc
255 lblDeducations.grid(row=4, column=0)
256 txtDeducations = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="right
257 txtDeducations.grid(row=4, column=1)
258
```

4.13 Creating frames and label for post code, gender, pay date, tax period , NI number, NI code, Pensionable pay ,Taxable pay and other payments due on the right side of gui


```

259 #=====Right Side
260 lblPostCode = Label(InsideRF, font=('arial', 12, 'bold'), text="Post Code", fg="black", bd=10, anchor="w")
261 lblPostCode.grid(row=0, column=0)
262 txtPostCode = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="gray", justify="right", t
263 txtPostCode.grid(row=0, column=1)
264
265 lblGender = Label(InsideRF, font=('arial', 12, 'bold'), text="Gender", fg="black", bd=10, anchor="w")
266 lblGender.grid(row=1, column=0)
267 txtGender = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="gray", justify="right", tex
268 txtGender.grid(row=1, column=1)
269
270 #-----
271 lblPayDate = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pay Date", fg="black", bd=10, anchor="w")
272 lblPayDate.grid(row=0, column=0)
273 txtPayDate = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="left", te
274 txtPayDate.grid(row=0, column=1)
275
276 lblTaxPeriod = Label(InsideRFL, font=('arial', 12, 'bold'), text="Tax Period", fg="black", bd=10, anchor
277 lblTaxPeriod.grid(row=1, column=0)
278 txtTaxPeriod = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="left",
279 txtTaxPeriod.grid(row=1, column=1)
280
281 lblNINumber = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Number", fg="black", bd=10, anchor="
282 lblNINumber.grid(row=2, column=0)
283 txtNINumber = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="left", t
284 txtNINumber.grid(row=2, column=1)
285

```

```

Payroll-Management-System-Project-In-Python-Source-Code > PayrollManagementSystems > main.py > ...
279 txtTaxPeriod.grid(row=1, column=1)
280
281 lblNINumber = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Number", fg="black", bd=10, anchor="
282 lblNINumber.grid(row=2, column=0)
283 txtNINumber = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="left", t
284 txtNINumber.grid(row=2, column=1)
285
286 lblNICode = Label(InsideRFL, font=('arial', 12, 'bold'), text="NI Code", fg="black", bd=10, anchor="w")
287 lblNICode.grid(row=3, column=0)
288 txtNICode = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="left", tex
289 txtNICode.grid(row=3, column=1)
290
291 lblTaxablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Taxable Pay ", fg="black", bd=10, anc
292 lblTaxablePay.grid(row=4, column=0)
293 txtTaxablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="left"
294 txtTaxablePay.grid(row=4, column=1)
295
296 lblPensionablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pensionable Pay", fg="black", bd
297 lblPensionablePay.grid(row=5, column=0)
298 txtPensionablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="le
299 txtPensionablePay.grid(row=5, column=1)
300
301 lblOtherPaymentDue = Label(InsideRFL, font=('arial', 12, 'bold'), text="Other Payment Due", fg="black",
302 lblOtherPaymentDue.grid(row=6, column=0)
303 txtOtherPaymentDue = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="gray", justify="
304 txtOtherPaymentDue.grid(row=6, column=1)
305

```

4.14 Creating buttons for wage payment, reset, payref, paycode and exit and calling root window.

```

306 #-----
307 btnWagePayment = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
308 text="Wage Payment", bg="blue", command=MonthlySalary).grid(row=0, column=0)
309
310 btnReset = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
311 text="Reset System", bg="blue", command=reset).grid(row=1, column=0)
312
313 btnPayRef = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
314 text="Pay Reference", bg="blue", command=PayRef).grid(row=2, column=0)
315
316 btnPayCode = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
317 text="Pay Code", bg="blue", command=PayPeriod).grid(row=3, column=0)
318
319 btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,
320 text="Exit", bg="blue", command=exit).grid(row=4, column=0)
321
322 payroll.mainloop()

```

Payroll Management Systems

Employee Name		Post Code	
Address		Gender	
Reference		Pay Date	
Employer Name		Tax Period	
City Weighting	Tax	NI Number	<div>Wage Payment</div> <div>Reset System</div> <div>Pay Reference</div> <div>Pay Code</div> <div>Exit</div>
Basic Salary	Pension	NI Code	
Over Time	StudenLoan	Taxable Pay	
Gross Pay	NI Payment	Pensionable Pay	
Net Pay	Deducations	Other Payment Due	

CHAPTER 5

CONCLUSION

We set out to create an payroll management system where you can a Database-driven in Python that will store the employee information in the MySQL Database. We can add, delete, update and save all the employees detail.

Whenever a new employee enters in the organization, we can add their data and can delete the data whenever they leave the organization.

In payroll management system we can calculate the monthly salaries of employees. Their gross salary, net pay, deductions etc. It deals with the recording and processing Employee's payroll data so that the executives can easily manage the organizational operations.

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.