

## **APPENDIX 1**

# **ONLINE EMPLOYEE MANAGEMENT SOFTWARE**

### **PROJECT REPORT**

*By*

**FAYSAL KABIR SHANTO**

Reg No : 12111058

Section:K21YB

Roll Number:7

**B BRUCELEE**

Reg No : 12114394

Section: K21YB

Roll Number: 32

**AAYUSH KUMAR**

Reg No : 12112141

Section : K21YB

Roll Number : 66



**School of Computer Science Engineering  
Lovely Professional University, Jalandhar  
November-2022**

## **APPENDIX 2**

### **Student Declaration**

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources has been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we shall take full responsibility for it.

Name : Faysal Kabir Shanto

Roll no : 7

Reg no: 12111058

Name : B Brucelee

Roll no : 32

Reg no: 12114394

Name : Aayush

Roll no : 66

Reg no: 12112141

Date: 11/11/2022

## **APPENDIX 3**

### **TABLE OF CONTENTS**

<b>NO.</b>	<b>TITLE</b>	<b>Page</b>
<b>1.</b>	<b>APPENDIX 1</b>	<b>1</b>
<b>2.</b>	<b>APPENDIX 2</b>	<b>2</b>
<b>3.</b>	<b>APPENDIX 3</b>	<b>3</b>
<b>4.</b>	<b>INTRODUCTION TO THE PROJECT</b>	<b>4</b>
<b>5.</b>	<b>BACKGROUND AND OBJECTIVE OF THE PROJECT</b>	<b>5</b>
<b>6.</b>	<b>DESCRIPTION OF PROJECT</b>	<b>6</b>
<b>7.</b>	<b>DESCRIPTION OF WORK DIVISION IN TERMS OF ROLES</b>	<b>7</b>
<b>8.</b>	<b>IMPLEMENTATION OF SCHEDULED WORK OF PROJECT</b>	<b>8</b>
<b>9.</b>	<b>FLOW CHART</b>	<b>24</b>
<b>10.</b>	<b>TABLES USED</b>	<b>25</b>
<b>11.</b>	<b>TECHNOLOGIES AND FRAMEWORK TO BE USED</b>	<b>26</b>
<b>12.</b>	<b>RESULTS</b>	<b>27</b>
<b>13.</b>	<b>LEARNING OUTCOMES</b>	<b>28</b>
<b>14.</b>	<b>CONCLUSION</b>	<b>29</b>
<b>15.</b>	<b>REFERENCES</b>	<b>30</b>
<b>16.</b>	<b>SCREENSHOTS</b>	<b>31</b>

## **INTRODUCTION TO THE PROJECT**

Employee Management system is an application that enables users to create and store Employee Records. The application also provides facilities of a payroll system which enables user to generate Pay slips too. This application is helpful to department of the organization which maintains data of employees related to an organization.

Every organization whether government or private uses an information system to store data of their staff. It is simple to understand and can be used by anyone who is not even familiar with simple employees system. It is user friendly and just asks the user to follow step by step operations by giving him few options. It is fast and can perform many operations of a company.

This software package has been developed using the powerful coding tools of PYTHON. The software is very user friendly. The package contains different modules like Employee details. This version of the software has multi-user approach. For further enhancement or development of the package, user feedback will be considered.

## **BACKGROUND AND OBJECTIVES OF THE PROJECT:**

**Project assigned:** ONLINE EMPLOYEE MANAGEMENT SOFTWARE

**Project number:** 01

**Members worked:** Faysal Kabir Shanto, B Brucelee, Aayush Kumar

**Background:** The project was done using the programming languages Python, MySQL, Tkinter (built-in python library)

**Objectives:** To manage and manipulate the basic employee data of an organization. In this world of growing technologies everything has been computerized. With large number of work opportunities the Human workforce has increased. Thus there is a need of a system which can handle the data of such a large number of Employees in an organization. This project simplifies the task of maintain records because of its user friendly nature

**Concrete Goals:** Connecting database to the GUI programming to obtain a project like an employee management system

**Motivation:** we can do amazing software like employee management system by just knowing Tkinter, python, and MySQL commands

**Project outcomes:** In our Employee Management System project, one can manage and manipulate the data of employees like employee Id, name, number, email, address, gender, and salary of an employee. The provided data management functions are, we can add new employee into the database and can update the existing record, we can search for a record to get details in case of larger database and also we can delete a record of unwanted employees, and we can also have an overview of a database having all data by just one click on show all records button. Simply we can have all controls on the basic information taking from an employee with ease. The main and important method is that we have a login system, where one can use MySQL credentials and login to the database and use it with secured data of employees.

## **DESCRIPTION OF PROJECT:**

- Employee Management System, we should take details from the employee and store it into a particular database to have access over them
- We need a user interface application or GUI for operating the commands and functions that connecting to the database to manipulate and manage data
- We need a login system to secure the data or to get accessed by only limited members or admins
- We should add methods, which are required for managing and manipulating the employee data
- **Methods like :**
- Adding a new employee data
- Updating existing records
- Searching existing records
- Deleting a record in case of unwanted
- Fetching records from the database
- Showing the records on the console according to the function used
- Login system to connect database and to secure data
- Added time and date of a record

## **Description of Work Division in terms of Roles among Students:**

All most all the project and reports written are equally contributed by three of us Faysal Kabir Shanto, B Brucelee, Aayush Kumar by discussing in group texts and video conferences. We suggested to each other which and what will make good, what to add, and solving complex & minor errors together to get a good & expected outcome of our prior project discussions.

**Coding part:** Done by three of us by dividing work, all the codes are coordinated and arranged in a process by Aayush Kumar to get the desired output

**Testing part:** Testing was done by Faysal Kabir Shanto to check and report what are the errors in the GUI and database part. Raised errors are solved by B Brucelee and Aayush Kumar

**Report part:** Report content was done by three of us Faysal Kabir Shanto, B Brucelee, Aayush Kumar, and typed by Faysal Kabir Shanto and B Brucelee

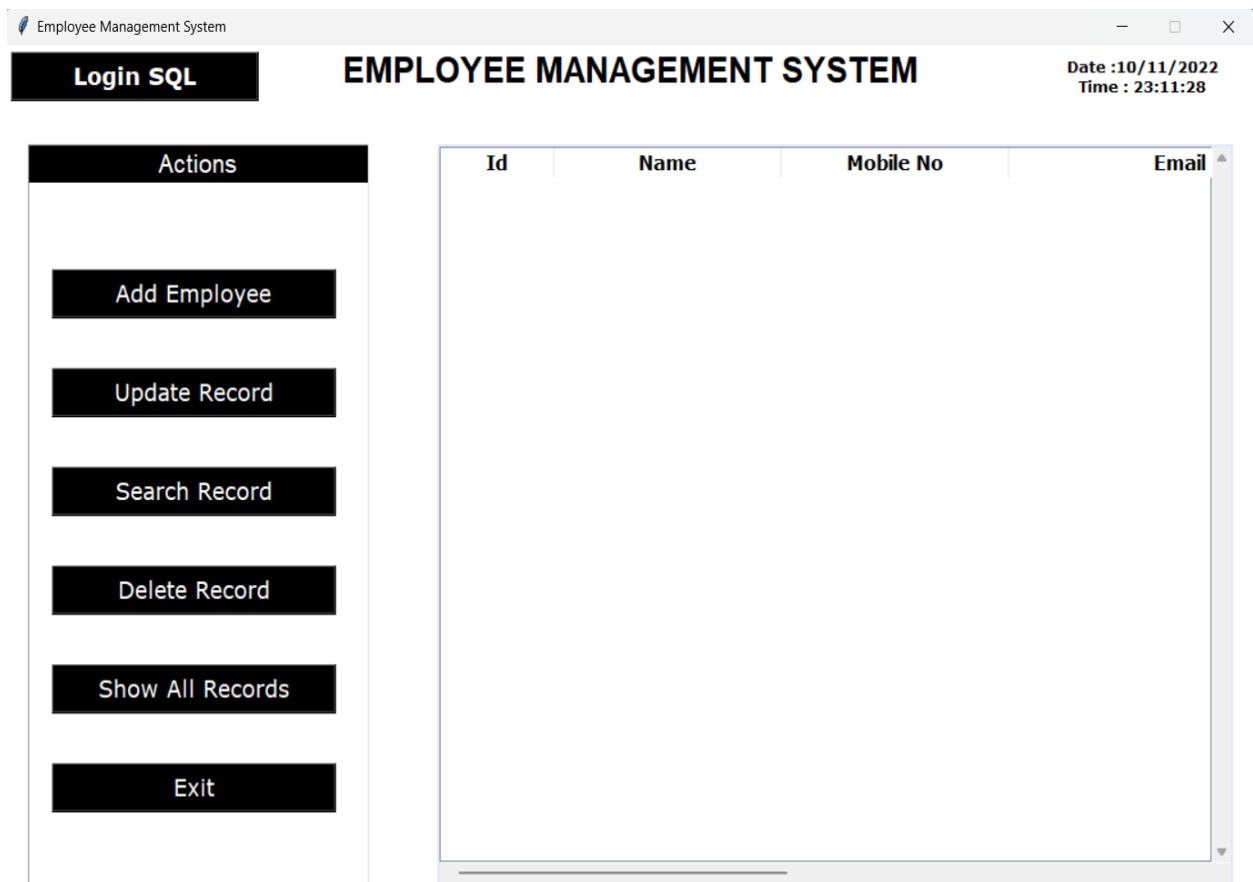
Faysal Kabir Shanto  
Roll No: 7

B Brucelee  
Roll No: 32

Aayush Kumar  
Roll No:66

# IMPLEMENTATION OF SCHEDULED WORK OF PROJECT

## Output and UI of the project:



Snap1: This is the main window and UI of the project

- About the geometry of this window, it contains one mainwindow named (root) and contains two frames named(DataEntryFrame and ShowDataFrame)



- This is the user interface of the project
- Add Employee, UpdateEmployee, Search Record, Delete record, ShowAllRecords, Exit are the buttons on DataEntryFrame and methods to manage and manipulate the data
- On the top left corner, “LogIn Database” button is on mainwindow (root)and it is the method to log in and connects to the database
- It has main label named “Employee Management System” situated in the root window
- On the top right corner , It is Time and Date, time calls it functions every 200milliseconds, so that time keeps on running until window exists
- The right part or ShowDataFrame will fetch records from the database and displays it like a treeview, it will display the data according to the MySQL query
- The root window is non-resizable & fixed in size

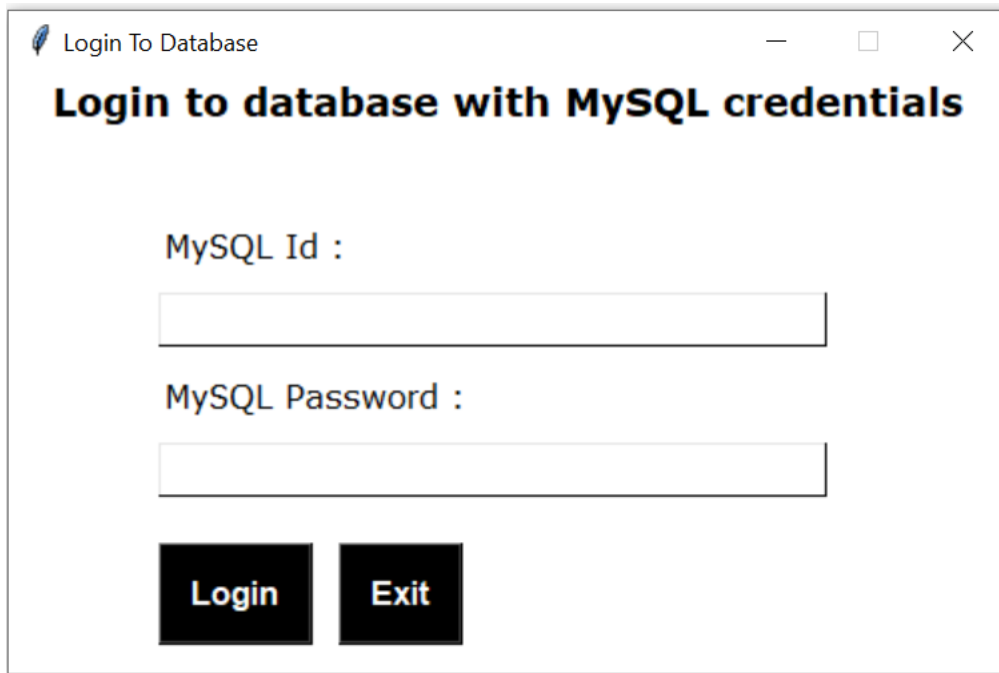
### Code of mainwindow(root)&importings of tkinter:

```
from tkinter import *
from tkinter import Toplevel, messagebox, filedialog
from tkinter.ttk import Treeview
from tkinter import ttk
import pandas
import mysql.connector
import time

#-----CREATING MAINWINDOW
root = Tk()
root.title("Employee Management System")
root.config(bg="white")
root.geometry("1100x700+200+50")
root.resizable(False, False)
```

Snap2: code of root window

## 1. LogIn Database :



Login To Database

**Login to database with MySQL credentials**

MySQL Id :

MySQL Password :

Login Exit

- When we click on the login database button this popup window will open and no other button on the root window will work until we close this login window
- To login into the database, the user should have MySQL installed in the computer and also pymysql module installed in python libraries
- User should use MySQL user id and password to login into the database Eg: MySQL user id= root MySQL password= Umamahesh@123 These are my MySQL credentials to log in to the database and the host is by default 'localhost' so that, anyone should use their MySQL credentials as LogIn Database details
- LogIn button connects to the database if credentials are correct else popup message will show that credentials are wrong
- Exit button will destroy the window and goes to mainwindow(root)

## Code of connection to database and login button:

```
##***** DEFINING
LOGINDATABASE *****##
def Connectdb():

    #-----> defining login button & making connection to database
    <-----#
    def submitdb():
        global con, mycursor
        host = "localhost"
        user = "root"
        password = "Fay@102485sal"
        try:
            con = mysql.connector.connect(host=host, user=user,
password=password)
            mycursor = con.cursor()
        except:
            messagebox.showerror('Notifications', 'Data is incorrect please try
again', parent=dbroot)
            return
        try:
            strr = 'create database employeemanagementsystem1'
            mycursor.execute(strr)
            strr = 'use employeemanagementsystem1'
            mycursor.execute(strr)
            strr = 'create table employeeedata1(id int,name varchar(20),mobile
varchar(12),email varchar(30),address varchar(100),gender varchar(50),salary
varchar(50),date varchar(50),time varchar(50))'
            mycursor.execute(strr)
            strr = 'alter table employeeedata1 modify column id int not null'
            mycursor.execute(strr)
            strr = 'alter table employeeedata1 modify column id int primary key'
            mycursor.execute(strr)
            messagebox.showinfo('Notification',
                                'database created and now you are connected
connected to the database ....',
                                parent=dbroot)

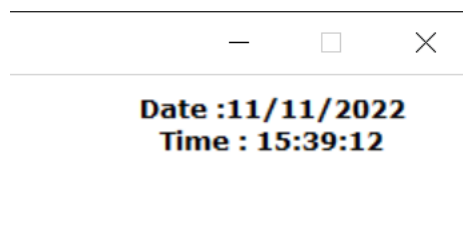
        except:
            strr = 'use employeemanagementsystem1'
            mycursor.execute(strr)
            messagebox.showinfo('Notification', 'Now you are connected to the
database ....', parent=dbroot)
            dbroot.destroy()
```

- ❖ This code will connect to the database and create a database named “employeemanagementsystem1” if it was the first time else it will use the created database
- ❖ This process is done by try and except statements to avoid errors recreating the existing database
- ❖ It also creates a table name “employeeedata1”
- ❖ This will happen when we click on the login button in LogIn Database window

### Code for date and time:

```
##***** DEFINING TIME&DATE
METHODS *****##
def tick():
    time_string = time.strftime("%H:%M:%S")
    date_string = time.strftime("%d/%m/%Y")
    clock.config(text='Date :' + date_string + "\n" + "Time : " + time_string)
    clock.after(200, tick)

##***** END OF TIME&DATE
METHODS *****##
```



- ‘clock.after(200,tick)’ this will recall ‘tick’ function every 200milliseconds so time constantly runs on the root window
- And this also useful in adding added date and added time of a record to the database

## 2. Add Employee:

- When we click on Add Employee this window will open and it is not resizable and no other button on the root window will work until we close the Add Employee window
- Here Id part can not be null and can not be any other data type except Integer if any other datatype or null value is submitted the popup error message will open and it does not affect or change anything in the database
- Any other value in add employee form can be null and of any datatype except id
- Id is defined as the primary key in the database
- When we click on the submit button the data entered in the entry labels will be stored in the database and displayed in the ShowDataFrame
- Id is a unique value otherwise it will raise a popup error message

### Code for addemployee & getting and storing entered data into database:

The screenshot displays the 'EMPLOYEE MANAGEMENT SYSTEM' application. On the left, a sidebar contains a 'Login SQL' button and an 'Actions' menu with buttons for 'Add Employee', 'Update Record', 'Search Record', 'Delete Record', 'Show All Records', and 'Exit'. The main window features a title bar with 'Employee Management System', a header with 'EMPLOYEE MANAGEMENT SYSTEM', and a status bar showing 'Date :11/11/2022' and 'Time : 15:52:32'. A modal window titled 'Enter details' is open, containing input fields for 'Enter Id :', 'Enter Name :', 'Enter Mobile :', 'Enter Email :', 'Enter Address:', 'Enter Gender :', and 'Enter Salary :', along with a 'Submit' button. The background window is partially obscured by the modal.

```

##### DEFINING
ADDEMPLOYEE #####
def addemployee():

    #<<<----- defining the submit button in addemployee form -----
->>>#
    def submitadd():
        id = idval.get()                #getting data from entries...or
entry lables
        name = nameval.get()
        mobile = mobileval.get()
        email = emailval.get()
        address = addressval.get()
        gender = genderval.get()
        salary = salaryval.get()
        addedtime = time.strftime("%H:%M:%S") #to get time
        addeddate = time.strftime("%d/%m/%Y") #to get date
        try:
            strr = 'insert into employeeedata1 values(%s,%s,%s,%s,%s,%s,%s,%s,%s)'
            mycursor.execute(strr, (id, name, mobile, email, address, gender,
salary, addeddate, addedtime))
            con.commit()
            res = messagebox.askyesnocancel('Notificatrions','Id {} Name {} Added
sucessfully.. and want to clean the form'.format(id,name),
parent=addroot)

            # clearing the entry form after adding details to database only if
res==true
            if (res == True):
                idval.set('')
                nameval.set('')
                mobileval.set('')
                emailval.set('')
                addressval.set('')
                genderval.set('')
                salaryval.set('')
            except:
                messagebox.showerror('Notifications', 'Id Already Exist try another
id...', parent=addroot)
            strr = 'select * from employeeedata1'
            mycursor.execute(strr)
            datas = mycursor.fetchall()
            employeetable.delete(*employeetable.get_children())
            for i in datas:

```

```

        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]] #making
data as list to print on treeview
        employeetable.insert(' ', END, values=vv) #to show
data,employeetable treeview ,prints from starting to end of (vv) values

        #<<<----- end of defining sumbitbutton of addemployee -----
----->>>#

```

- ❖ This is the code for adding entered data into the database
- ❖ 'strr= select \* from employeedata1' this will select all data from the database and the rest of the below code is used to show the added data in ShowDataFrame as a treeview
- ❖ 'employeetable.delete(\*employeetable.get\_children())' " this will clear the console in showdataframe and the code below it(vv=i[]) will print all the records and the newly added data also

### 3. Update record:

The screenshot displays the 'Employee Management System' interface. At the top, there is a title bar with 'Employee Management System' and window controls. Below this, a header section contains 'Login SQL' on the left, 'EMPLOYEE MANAGEMENT SYSTEM' in the center, and 'Date :11/11/2022' and 'Time : 15:54:11' on the right. A sidebar on the left lists 'Actions' with buttons for 'Add Employee', 'Update Record', 'Search Record', 'Delete Record', 'Show All Records', and 'Exit'. The 'Update Record' button is highlighted. A modal dialog box titled 'Update Record' is open in the center, containing input fields for 'Enter Id :', 'Enter Name :', 'Enter Mobile :', 'Enter Email :', 'Enter Address:', 'Enter Gender :', 'Enter Salary :', 'Enter Date :', and 'Enter Time :'. An 'Update' button is located at the bottom right of the dialog. In the background, a table with columns 'No' and 'Email' is partially visible.

- When we click on Update Employee this window will open and it is not resizable and no other button on the root window will work until we close the Update Employee window
- Here for updating the record id is mandatory and without id we cannot update the record because id is the primary key
- As similarly any other entry can be null except id
- On clicking the update button the record with provided id will be updated in database as well as the updated record will be shown in the ShowDataFrame
- Code for update record and adding the updated result to database is as same as addemployee



## 4. Search Record:

- When we click on Search Employee this window will open and it is not resizable and no other button on the root window will work until we close the Search Employee window
- Here for searching the record, we can search with any parameter
- We can search with only any one of the above parameters
- On searching the valid parameter the results will be displayed on the ShowDataFrame
- Search only with exact value do not search with half values

### Code for search employee and displaying the data in ShowDataFrame:

```
##### DEFINING
SEARCHEMPLOYEE #####
def searchemployee():

    #<<<----- defining the search button in searchemployee form -----
    >>>#
    def search():
        id = idval.get()
        name = nameval.get()
        mobile = mobileval.get()
        email = emailval.get()
        address = addressval.get()
        gender = genderval.get()
        salary = salaryval.get()
        addeddate = time.strftime("%d/%m/%Y")

        #<<<----- code for searching records ----->>>#
        if (id != ''):
            strr = 'select *from employee data1 where id=%s' #it will select
all records with the entered value
            mycursor.execute(strr, (id)) #executing that
statement
```

```

        datas = mycursor.fetchall() #it will fetch all
records and stores in datas
        employeetable.delete(*employeetable.get_children())# it will clear
the console(showdataframe)
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]#it
will print the fetched data of the searched record on the console
            employeetable.insert('', END, values=vv) # it will print fetched
record from starting to end of that value
        elif (name != ''):
            strrr = 'select *from employeedata1 where name=%s'
            mycursor.execute(strrr, (name))
            datas = mycursor.fetchall()
            employeetable.delete(*employeetable.get_children())
            for i in datas:
                vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
                employeetable.insert('', END, values=vv)
        elif (mobile != ''):
            strrr = 'select *from employeedata1 where mobile=%s'
            mycursor.execute(strrr, (mobile))
            datas = mycursor.fetchall()
            employeetable.delete(*employeetable.get_children())
            for i in datas:
                vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
                employeetable.insert('', END, values=vv)
        elif (email != ''):
            strrr = 'select *from employeedata1 where email=%s'
            mycursor.execute(strrr, (email))
            datas = mycursor.fetchall()
            employeetable.delete(*employeetable.get_children())
            for i in datas:
                vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
                employeetable.insert('', END, values=vv)
        elif (address != ''):
            strrr = 'select *from employeedata1 where address=%s'
            mycursor.execute(strrr, (address))
            datas = mycursor.fetchall()
            employeetable.delete(*employeetable.get_children())
            for i in datas:
                vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
                employeetable.insert('', END, values=vv)
        elif (gender != ''):
            strrr = 'select *from employeedata1 where gender=%s'
            mycursor.execute(strrr, (gender))
            datas = mycursor.fetchall()

```

```

        employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
        employeetable.insert('', END, values=vv)
elif (salary != ''):
    strr = 'select *from employeedata1 where salary=%s'
    mycursor.execute(strr, (salary))
    datas = mycursor.fetchall()
    employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
        employeetable.insert('', END, values=vv)

elif (addeddate != ''):
    strr = 'select *from employeedata1 where addeddate=%s'
    mycursor.execute(strr, (addeddate))
    datas = mycursor.fetchall()
    employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
        employeetable.insert('', END, values=vv)

#<<<----- end of searching records code ----->>>#

```

- This is the code used to search the records
- At first, it will select the records with the details entered in the entry box, Next, it fetches all details of that record and next prints it into the showdataframe, this is the logic behind that code for searching the record
- This will execute when we hit on the search button in search record window

## 5. Delete Record:

The screenshot displays the 'EMPLOYEE MANAGEMENT SYSTEM' interface. On the left, under the 'Actions' menu, the 'Delete Record' button is highlighted. The main area shows a table of employee records. The third record, with ID 3, name 'aayush', mobile number '9988776655', and email 'aayush@gmail.com', is selected (highlighted in blue).

Id	Name	Mobile No	Email
1	faysal	9903032977	faysal@gmail.com
2	brucelee	9098575561	brucelee@gmail.com
3	aayush	9988776655	aayush@gmail.com
4	kabir	1234567890	kabir@gmail.com
5	shanto	8796054898	shanto@gmail.com

- To delete a record we should select the record as we selected in the snap, and next we should hit the delete button, then the record is deleted from database and it will also disappear from treeview or showdataframe
- We can also delete the record by searching it first and selecting it on showdataframe and hitting on the delete button

## Code for delete employee and deleting the data from ShowDataFrame&database:

```
##### DEFINING DELETE
EMPLOYEE #####
def deleteemployee():
    cc = employeetable.focus()          #focus on the record which we click and
    gets data of that record
    content = employeetable.item(cc)
    pp = content['values'][0]           #it gives the id of the particular record
    to delete
    strr = 'delete from employeeedata1 where id=%s'
    mycursor.execute(strr, (pp,))
    con.commit()
    messagebox.showinfo('Notifications', 'Id {} deleted
    sucessfully...'.format(pp))
    strr = 'select *from employeeedata1'
    mycursor.execute(strr)
    datas = mycursor.fetchall()
    employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
        employeetable.insert('', END, values=vv)

##### END OF
DELETEEMPLOYEE #####
```

- This code will delete the data from database first and next, it clears the console and reprints the console by fetching details from database
- So the deleted record will be disappeared from the console or showdataframe

## 6.ShowAllRecords:

Employee Management System

Login SQL

EMPLOYEE MANAGEMENT SYSTEM

Date :11/11/2022  
Time : 16:23:42

Actions

Add Employee

Update Record

Search Record

Delete Record

Show All Records

Exit

Id	Name	Mobile No	Email
----	------	-----------	-------

Employee Management System

Login SQL

EMPLOYEE MANAGEMENT SYSTEM

Date :11/11/2022  
Time : 16:22:00

Actions

Add Employee

Update Record

Search Record

Delete Record

Show All Records

Exit

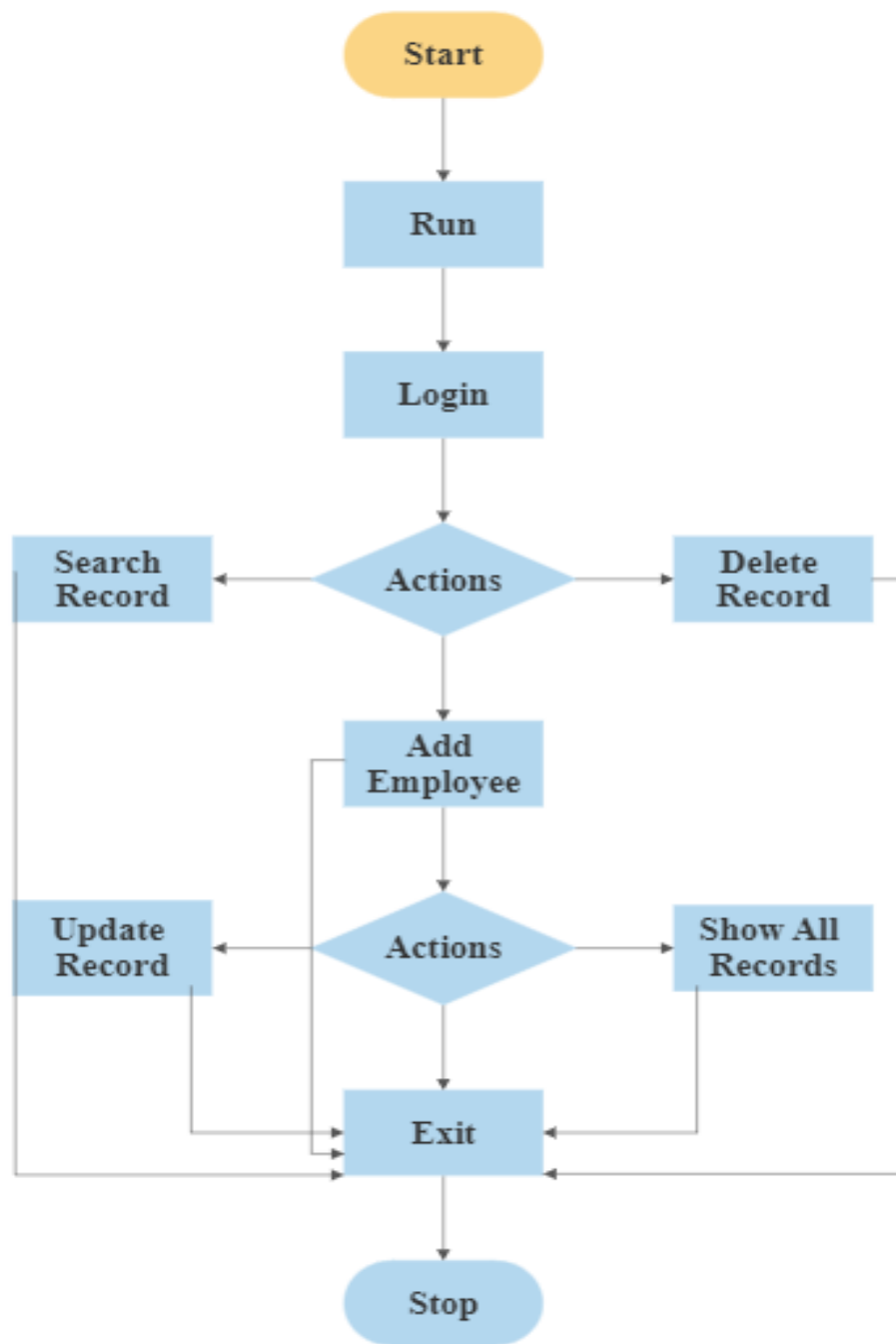
Id	Name	Mobile No	Email
1	faysal	9903032977	faysal@gmail.com
2	brucelee	9098575561	brucelee@gmail.com
3	aayush	9988776655	aayush@gmail.com
4	kabir	1234567890	kabir@gmail.com
5	shanto	8796054898	shanto@gmail.com

- The show all records button will print all data available in the database on showdataframe or console
- The code for show all records is simple it will get all records and puts on the console with just one click

## **7.Exit:**

- Exit will just destroy the mainwindow(root) and code exits when we click on it
- And when we exit the data base is automatically logged out and we need to login again when we run to fetch data and manipulate data

## FLOW CHART:





## TABLES USED:

Tables are database objects that contain all the data in a database. In tables, data is logically organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field in the record. The table used in our project is given below:

```
mysql> use employeemanagementsystem1;
Database changed
mysql> select * from employeeedata1;
```

id	name	mobile	email	address	gender	salary	date	time
1	Aayush	8969985598	aayushhpandey@gmail.com	here and there	Male	1 Cr.	10/11/2022	22:11:18
2	Faisal	99999999	a@b.com	hostel	Male	1 thousand	11/11/2022	18:36:24
3	BruceLee	8888888888	b@a.com	China	Male	1 Thousand	11/11/2022	18:37:05

```
3 rows in set (0.00 sec)
```

## **Technologies and Framework to be used :**

**IDE used :** VS Code

**Programming languages used:** python, python(Tkinter), MySQL

- Python is used for methods and connecting GUI buttons, labels, entryforms etc
- Tkinter is used to develop the graphical user interface
- MySQL is used to develop the database related quires and storing the data

### **Important:-**

- To get a connection to database installing MySQL is mandatory
- After installing MySQL to connect to python ‘import mysql.connector’ command is written in the code
- So the module mysql.connector is also important to get installed in python libraries
- If mysql.connector is not installed in the python libraries do these steps:
  - ❖ If code is running in VS Code
  - ❖ Goto command prompt-->give command ‘pip install mysql.connector’

## **RESULTS:**

- 1.** On the completion of this project we achieved a graphical user interface for our employee management system.
- 2.** We can login to our database from the login interface by inserting our username and password.
- 3.** Then we can add the records of the new employee to the database by “Add Employee” option.
- 4.** We can also update the information of a particular employee by the “Update Record” button.
- 5.** We can search the record of a particular employee by using the “Search Record” option.
- 6.** We also can delete the information of a particular employee by using the “Delete Record” option.
- 7.** We can see all the records of all the employees altogether at a time by using “Show All Records” option.
- 8.** We can also exit from any of the above mentioned options by using the “Exit” option.
- 9.** So, above all this project is a complete and sufficient platform to manage the records of the employees of an organization.

## **LEARNING OUTCOMES:**

1. Through medium of this project I learn python programming language which is used in building websites, software and also helps in performing automate tasks, and conduct data analysis.
2. The project also helped us to learn MYSQL, which is a tool used to manage databases and servers, it's widely used in relation to managing and organizing data in databases.
3. The project also helped us to acquire knowledge in field of tkinter. Tkinter is standard GUI library of python. It provides us fast and easy way to create GUI applications.
4. Through this project we learnt to make flow charts which is a type of diagram that represents a workflow or process. We retreat our algorithm through flow charts in the project.
5. This project helped us to acquire skill of completing a task in the most effective and efficient way. This process helped us to work collaboratively with a group of people in order to achieve a goal.

## **CONCLUSION:**

In this report, an information system's development has been presented. The report's content consists of the whole task solution starting from the programming environments that have been selected, going through the database, the application's analyze and construction, and finishing with the code-implementation.

An employee management system is a program to automate or computerize all employee management operations. If you operate a startup of under a dozen workers or are in control of a thousand-strong global as a multinational corporation, relying on manual systems to track and handle the staff will turn rapidly into an administrative nightmare. Also, if you opt to track the information and data manually, there are high chances of human errors as well as some compliance risks.

So, to avoid this happening, you need a system that can keep track of your work and smoothens the process.

Our project is an implementation of employee management system. We have successfully completed it. We take this opportunity to express our sense of indebtedness and gratitude to all those who helped us in completing this project and implementation.

## REFERENCES:

[1] – Begg Carolyn, Connolly Thomas, Database systems (a Practical approach to Design, Implementation, and Management), Addison-Wesley, an imprint of Pearson Education, University of Paisley (U.K.), Fourth edition 2005

[2] – Bodnar George /Duquesne University/, Hopwood William /Florida Atlantic University/, Accounting Information systems, Eighth Edition, Prentice Hall, Upper Saddle River, New Jersey .

[3] – Andersen Virginia, Access 2000: The Complete Reference, Blacklick, OH, USA:

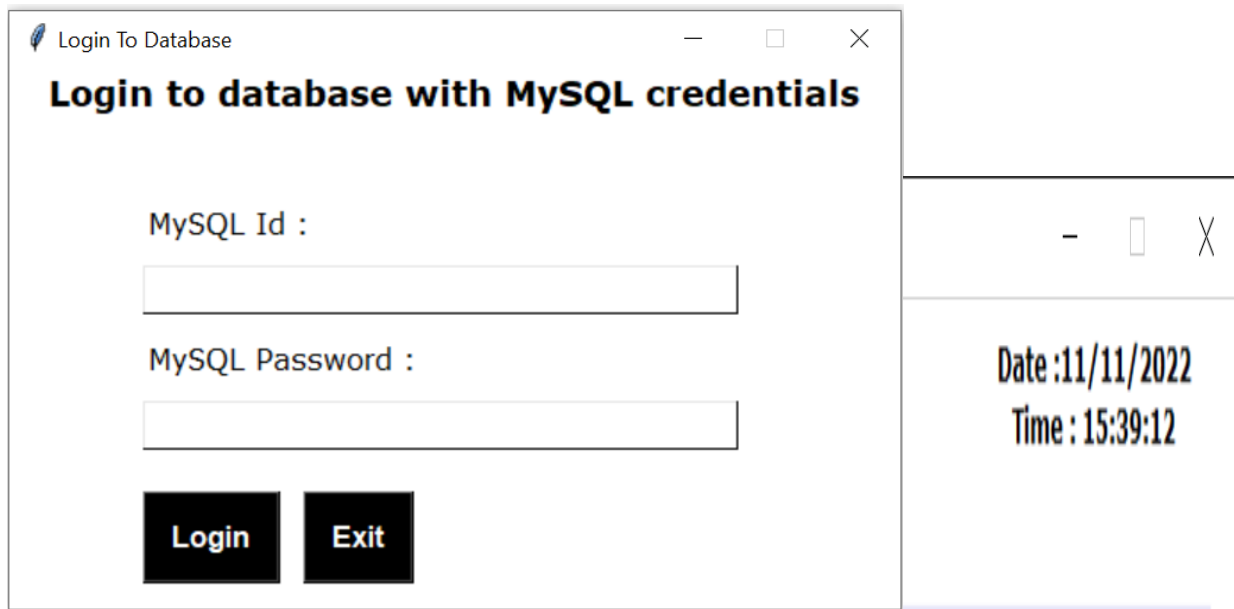
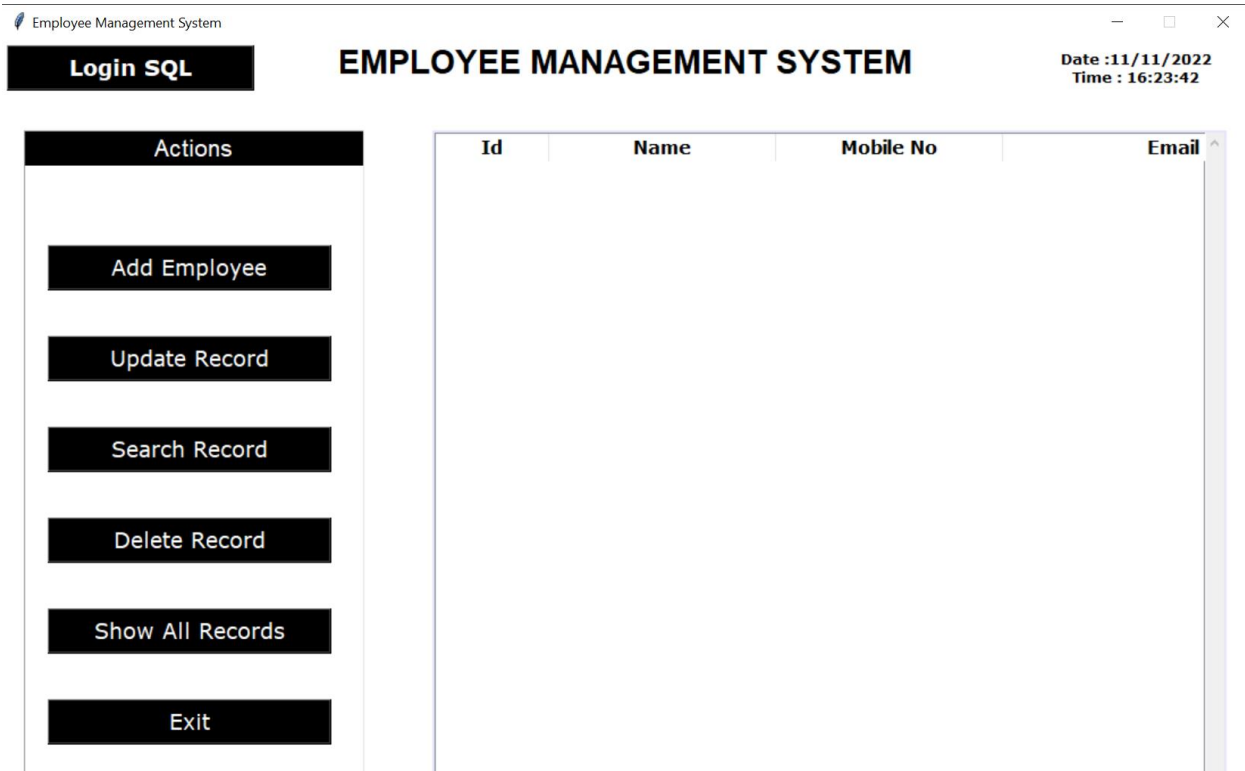
McGraw-Hill Professional Book Group, 2001,  
<http://site.ebrary.com/lib/vaxjo/Doc?id=5002842> (2006-05-25).

[4] – Andersson Tobias, [DAB744] C# Course Lectures, School of Mathematics and  
System Engineering, Växjö University.

[5] - <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vboritextboxctltasks.asp> (2006-05-25).

## SCREENSHOTS:

### GUI:



The screenshot displays the 'Employee Management System' application. On the left, a sidebar contains a 'Login SQL' button and an 'Actions' menu with options: 'Add Employee', 'Update Record', 'Search Record', 'Delete Record', 'Show All Records', and 'Exit'. The main area shows a table with columns 'Employee No' and 'Email'. An 'Update Record' dialog box is open, featuring input fields for 'Enter Id', 'Enter Name', 'Enter Mobile', 'Enter Email', 'Enter Address', 'Enter Gender', 'Enter Salary', 'Enter Date', and 'Enter Time'. An 'Update' button is located at the bottom of the dialog. The top right corner of the application window shows the date '11/11/2022' and time '20:39:54'.





Employee Management System

Login SQL

EMPLOYEE MANAGEMENT SYSTEM

Date :11/11/2022  
Time : 16:22:00

Actions

Add Employee

Update Record

Search Record

Delete Record

Show All Records

Exit

Id	Name	Mobile No	Email
1	faysal	9903032977	faysal@gmail.com
2	brucelee	9098575561	brucelee@gmail.com
3	aayush	9988776655	aayush@gmail.com
4	kabir	1234567890	kabir@gmail.com
5	shanto	8796054898	shanto@gmail.com

Employee Management System

Login SQL

EMPLOYEE MANAGEMENT SYSTEM

Date :11/11/2022  
Time : 20:43:15

Actions

Add Employee

Update Record

Search Record

Delete Record

Show All Records

Exit

Id	Name	Mobile No	Email
1	faysal	9903032977	faysal@gmail.com
2	brucelee	9098575561	brucelee@gmail.com
3	aayush	9988776655	aayush@gmail.com
4	kabir	1234567890	kabir@gmail.com
5	shanto	8796054898	shanto@gmail.com

Notification

?

Do you want to exit?

Yes
No
Cancel

## CODE:

```
# coding: utf-8

# In[ ]:

#-----> EMPLOYEE MANAGEMENT SYSTEM <-----
#-----#

##### DEFINING
ADDEMPLOYEE #####

def addemployee():

    #<<<----- defining the submit button in addemployee form -----
    ->>>#
    def submitadd():
        id = idval.get()                                #getting data from entries...or
entry labes
        name = nameval.get()
        mobile = mobileval.get()
        email = emailval.get()
        address = addressval.get()
        gender = genderval.get()
        salary = salaryval.get()
        addedtime = time.strftime("%H:%M:%S") #to get time
        addeddate = time.strftime("%d/%m/%Y") #to get date
        try:
            strr = 'insert into employee data1 values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)'
            mycursor.execute(strr, (id, name, mobile, email, address, gender,
salary, addeddate, addedtime))
            con.commit()
            res = messagebox.askyesnocancel('Notificatrions','Id {} Name {} Added
sucessfully.. and want to clean the form'.format(id,name),
parent=addroot)

            # clearing the entry form after adding details to database only if
res==true
            if (res == True):
                idval.set('')
                nameval.set('')
                mobileval.set('')
                emailval.set('')
                addressval.set('')
```

```

        genderval.set('')
        salaryval.set('')
    except:
        messagebox.showerror('Notifications', 'Id Already Exist try another
id...', parent=addroot)
    strr = 'select * from employeeedata1'
    mycursor.execute(strr)
    datas = mycursor.fetchall()
    employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]] #making
data as list to print on treeview
        employeetable.insert('', END, values=vv) #to show
data,employeetable treeview ,prints from starting to end of (vv) values

    #<<<----- end of defining sumbitbutton of addemployee -----
----->>>#

    #<<<----- creating window for employeeedata entry -----
----->>>#
    addroot = Toplevel(master=DataEntryFrame)
    addroot.grab_set()
    addroot.geometry('470x470+220+200')
    addroot.title('Enter details')
    addroot.config(bg='white')
    addroot.resizable(False,False) #making window nonresizable

    # ----- Addemployee Labels

    idlabel = Label(addroot, text='Enter Id : ', bg='#ffffff', font=('verdana',
15), width=12, anchor='w')
    idlabel.place(x=10, y=10)

    namelabel = Label(addroot, text='Enter Name : ', bg='#ffffff',
font=('verdana', 15), width=12, anchor='w')
    namelabel.place(x=10, y=70)

    mobilelabel = Label(addroot, text='Enter Mobile : ',bg='#ffffff',
font=('verdana', 15), width=12, anchor='w')
    mobilelabel.place(x=10, y=130)

    emaillabel = Label(addroot, text='Enter Email : ',bg='#ffffff',
font=('verdana', 15), width=12, anchor='w')
    emaillabel.place(x=10, y=190)

```

```

addresslabel = Label(addroot, text='Enter Address: ',bg='#ffffff',
font=('verdana', 15), width=12, anchor='w')
addresslabel.place(x=10, y=250) # DCAE96

genderlabel = Label(addroot, text='Enter Gender : ', bg='#ffffff',
font=('verdana', 15), width=12, anchor='w')
genderlabel.place(x=10, y=310)

salarylabel = Label(addroot, text='Enter Salary : ', bg='#ffffff',
font=('verdana', 15), width=12, anchor='w')
salarylabel.place(x=10, y=370)

#----- Addemployee Entrybg='#ffffff', font=('verdana', 17),
width=12, anchor='w')
idval = StringVar()
nameval = StringVar()
mobileval = StringVar()
emailval = StringVar()
addressval = StringVar()
genderval = StringVar()
salaryval = StringVar()

#----- Addemployee entry lables
identry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=idval, width=18)
identry.place(x=250, y=10)

nameentry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=nameval, width=18)
nameentry.place(x=250, y=70)

mobileentry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=mobileval, width=18)
mobileentry.place(x=250, y=130)

emailentry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=emailval, width=18)
emailentry.place(x=250, y=190)

addressentry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=addressval, width=18)
addressentry.place(x=250, y=250)

genderentry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=genderval, width=18)

```

```

genderentry.place(x=250, y=310)

salaryentry = Entry(addroot, font=('Helvetica', 15, 'bold'), bd=1,
textvariable=salaryval, width=18)
salaryentry.place(x=250, y=370)

#----- addemployee submit button
submitbtn = Button(addroot, text="Submit", bd=1, font=("Bodoni", 16),
width=10, fg="white")
submitbtn.config(relief="raised", bg="#000000", activebackground="yellow",
command=submitadd)
submitbtn.place(x=150, y=420)

addroot.mainloop()
##### END OF
ADDEMPLOYEE #####

##### DEFINING
SEARCHEMPLOYEE #####
def searchemployee():

    #<<<----- defining the search button in searchemployee form -----
    ---->>>#
    def search():
        id = idval.get()
        name = nameval.get()
        mobile = mobileval.get()
        email = emailval.get()
        address = addressval.get()
        gender = genderval.get()
        salary = salaryval.get()
        addeddate = time.strftime("%d/%m/%Y")

        #<<<----- code for searching records ----->>>#
        if (id != ''):
            strr = 'select *from employeeedata1 where id=%s' #it will select
all records with the entered value
            mycursor.execute(strr, (id,)) #executing that
statement
            datas = mycursor.fetchall() #it will fetch all
records and stores in datas
            employeetable.delete(*employeetable.get_children())# it will clear
the console(showdataframe)
            for i in datas:

```

```

        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]#it
will print the fetched data of the searched record on the console
        employeetable.insert('', END, values=vv) # it will print fetched
record from starting to end of that value
    elif (name != ''):
        strr = 'select *from employeeedata1 where name=%s'
        mycursor.execute(strr, (name,))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)
    elif (mobile != ''):
        strr = 'select *from employeeedata1 where mobile=%s'
        mycursor.execute(strr, (mobile,))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)
    elif (email != ''):
        strr = 'select *from employeeedata1 where email=%s'
        mycursor.execute(strr, (email,))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)
    elif (address != ''):
        strr = 'select *from employeeedata1 where address=%s'
        mycursor.execute(strr, (address,))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)
    elif (gender != ''):
        strr = 'select *from employeeedata1 where gender=%s'
        mycursor.execute(strr, (gender,))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)
    elif (salary != ''):

```

```

        strrr = 'select *from employeeedata1 where salary=%s'
        mycursor.execute(strrr, (salary,))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)

    elif (addeddate != ''):
        strrr = 'select *from employeeedata1 where addeddate=%s'
        mycursor.execute(strrr, (addeddate))
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)

#<<<----- end of searching records code ----->>>#

#<<<----- creating a window of searching from ----->>>#
searchroot = Toplevel(master=DataEntryFrame)
searchroot.grab_set()
searchroot.geometry('470x540+220+200')
searchroot.title('Search records')
searchroot.config(bg='white')
searchroot.resizable(False, False)

# -----search employee Labels
idlabel = Label(searchroot, text='Enter Id : ',bg='#ffffff', font=('verdana',
17), width=12, anchor='w')
idlabel.place(x=10, y=10)

namelabel = Label(searchroot, text='Enter Name : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
namelabel.place(x=10, y=70)

mobilelabel = Label(searchroot, text='Enter Mobile : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
mobilelabel.place(x=10, y=130)

emaillabel = Label(searchroot, text='Enter Email : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
emaillabel.place(x=10, y=190)

```



```

addresslabel = Label(searchroot, text='Enter Address: ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
addresslabel.place(x=10, y=250) # DCAE96

genderlabel = Label(searchroot, text='Enter Gender : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
genderlabel.place(x=10, y=310)

salarylabel = Label(searchroot, text='Enter Salary : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
salarylabel.place(x=10, y=370)

datelabel = Label(searchroot, text='Enter Date : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
datelabel.place(x=10, y=430)

#-----search employee Entry
idval = StringVar()
nameval = StringVar()
mobileval = StringVar()
emailval = StringVar()
addressval = StringVar()
genderval = StringVar()
salaryval = StringVar()
dateval = StringVar()

#-----saerch employee entrylables
identry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=idval)
identry.place(x=250, y=10)

nameentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=nameval)
nameentry.place(x=250, y=70)

mobileentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=mobileval)
mobileentry.place(x=250, y=130)

emailentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=emailval)
emailentry.place(x=250, y=190)

addressentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=addressval)

```

```

addressentry.place(x=250, y=250)

genderentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=genderval)
genderentry.place(x=250, y=310)

salaryentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=salaryval)
salaryentry.place(x=250, y=370)

dateentry = Entry(searchroot, font=('arial', 13, 'bold'), bd=1,
textvariable=dateval)
dateentry.place(x=250, y=430)

#----- search button
searchbtn = Button(searchroot, text='Search', bd=3,fg="white",
font=("Bodoni", 16), width=10, command=search)
searchbtn.config(relief="raised", bg="black", activebackground="yellow")
searchbtn.place(x=150, y=480)

searchroot.mainloop()

##### END OF DEFINING
SEARCHEMPLOYEE #####

##### DEFINING DELETE
EMPLOYEE #####
def deleteemployee():
    cc = employeetable.focus()          #focus on the record which we click and
gets data of that record
    content = employeetable.item(cc)
    pp = content['values'][0]           #it gives the id of the particular record
to delete
    strr = 'delete from employeeedata1 where id=%s'
    mycursor.execute(strr, (pp,))
    con.commit()
    messagebox.showinfo('Notifications', 'Id {} deleted
sucessfully...'.format(pp))
    strr = 'select *from employeeedata1'
    mycursor.execute(strr)
    datas = mycursor.fetchall()
    employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
        employeetable.insert('', END, values=vv)

```

```

##### END OF
DELETEEMPLOYEE #####

##### DEFINING
UPDATEEMPLOYEE #####
def updateemployee():

    #<<<----- defining updatebutton in update entry form -----
>>>#
    def update():
        id = idval.get()
        name = nameval.get()
        mobile = mobileval.get()
        email = emailval.get()
        address = addressval.get()
        gender = genderval.get()
        salary = salaryval.get()
        date = dateval.get()
        time = timeval.get()

        strr = 'update employee data1 set
name=%s,mobile=%s,email=%s,address=%s,gender=%s,salary=%s,date=%s,time=%s where
id=%s'
        mycursor.execute(strr, (name, mobile, email, address, gender, salary,
date, time, id))
        con.commit()
        messagebox.showinfo('Notifications', 'Id {} Modified
sucessfully...'.format(id), parent=updateroot)
        strr = 'select *from employee data1'
        mycursor.execute(strr)
        datas = mycursor.fetchall()
        employeetable.delete(*employeetable.get_children())
        for i in datas:
            vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
            employeetable.insert('', END, values=vv)

    #<<<----- creating a window for updateentry form -----
>>>#
    updateroot = Toplevel(master=DataEntryFrame)
    updateroot.grab_set()
    updateroot.geometry('470x585+220+160')
    updateroot.title('Update Record')
    updateroot.config(bg='white')
    updateroot.resizable(False, False)

```

```

#----- Updateemployee Labels
idlabel = Label(updateroot, text='Enter Id : ',bg='#ffffff', font=('verdana',
17), width=12, anchor='w')
idlabel.place(x=10, y=10)

namelabel = Label(updateroot, text='Enter Name : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
namelabel.place(x=10, y=70)

mobilelabel = Label(updateroot, text='Enter Mobile : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
mobilelabel.place(x=10, y=130)

emaillabel = Label(updateroot, text='Enter Email : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
emaillabel.place(x=10, y=190)

addresslabel = Label(updateroot, text='Enter Address: ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
addresslabel.place(x=10, y=250) # DCAE96

genderlabel = Label(updateroot, text='Enter Gender : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
genderlabel.place(x=10, y=310)

salarylabel = Label(updateroot, text='Enter Salary : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
salarylabel.place(x=10, y=370)

datelabel = Label(updateroot, text='Enter Date : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
datelabel.place(x=10, y=430)

timelabel = Label(updateroot, text='Enter Time : ', bg='#ffffff',
font=('verdana', 17), width=12, anchor='w')
timelabel.place(x=10, y=490)

#----- Updateemployee Entry
idval = StringVar()
nameval = StringVar()
mobileval = StringVar()
emailval = StringVar()
addressval = StringVar()
genderval = StringVar()

```

```

salaryval = StringVar()
dateval = StringVar()
timeval = StringVar()

#-----updateemployee entry labels

identry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=idval)
identry.place(x=250, y=10)

nameentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=nameval)
nameentry.place(x=250, y=70)

mobileentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=mobileval)
mobileentry.place(x=250, y=130)

emailentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=emailval)
emailentry.place(x=250, y=190)

addressentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=addressval)
addressentry.place(x=250, y=250)

genderentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=genderval)
genderentry.place(x=250, y=310)

salaryentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=salaryval)
salaryentry.place(x=250, y=370)

dateentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=dateval)
dateentry.place(x=250, y=430)

timeentry = Entry(updateroot, font=('arial', 12, 'bold'), bd=1,
textvariable=timeval)
timeentry.place(x=250, y=490)

#----- update button
updatebtn = Button(updateroot, text="Update", bd=3,fg="white",
font=("Bodoni", 16, "bold"), width=10,command=update)

```

```

updatebtn.config(relief="raised", bg="black", activebackground="yellow")
updatebtn.place(x=150, y=540)

cc = employeetable.focus()      #this will get all details of existing record
when clicked on it and fills in update entryform
content = employeetable.item(cc)
pp = content['values']
if (len(pp) != 0):
    idval.set(pp[0])
    nameval.set(pp[1])
    mobileval.set(pp[2])
    emailval.set(pp[3])
    addressval.set(pp[4])
    genderval.set(pp[5])
    salaryval.set(pp[6])
    dateval.set(pp[7])
    timeval.set(pp[8])

updateroot.mainloop()

##### END OF DEFINING
UPDATEEMPLOYEE #####

##### DEFINING SHOW ALL
RECORDS #####
def showallrecords():
    strr = 'select *from employeeedata1'
    mycursor.execute(strr)
    datas = mycursor.fetchall()
    employeetable.delete(*employeetable.get_children())
    for i in datas:
        vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8]]
        employeetable.insert('', END, values=vv)

##### END OF
SHOWALLRECORDS #####

##### DEFINING
EXITBUTTON #####
def exit_window():
    res = messagebox.askyesnocancel('Notification', 'Do you want to exit?')
    if (res == True):
        root.destroy()

```

```
##### END OF
EXITBUTTON #####

##***** DEFINING
LOGINDATABASE *****##
def Connectdb():

    #-----> defining login button & making connection to database
    <-----#
    def submitdb():
        global con, mycursor
        host = "localhost"
        user = "root"
        password = "Fay@102485sal"
        try:
            con = mysql.connector.connect(host=host, user=user,
password=password)
            mycursor = con.cursor()
        except:
            messagebox.showerror('Notifications', 'Data is incorrect please try
again', parent=dbroot)
            return
        try:
            strr = 'create database employeemanagementsystem1'
            mycursor.execute(strr)
            strr = 'use employeemanagementsystem1'
            mycursor.execute(strr)
            strr = 'create table employeeedata1(id int,name varchar(20),mobile
varchar(12),email varchar(30),address varchar(100),gender varchar(50),salary
varchar(50),date varchar(50),time varchar(50))'
            mycursor.execute(strr)
            strr = 'alter table employeeedata1 modify column id int not null'
            mycursor.execute(strr)
            strr = 'alter table employeeedata1 modify column id int primary key'
            mycursor.execute(strr)
            messagebox.showinfo('Notification',
                                'database created and now you are connected
connected to the database ....',
                                parent=dbroot)

        except:
            strr = 'use employeemanagementsystem1'
            mycursor.execute(strr)
            messagebox.showinfo('Notification', 'Now you are connected to the
database ....', parent=dbroot)
```

```

        dbroot.destroy()

        #-----> creating window for login database button <-----
        -----#
        messagebox.showinfo('Information..!', 'Please login with MySQL user id and
password to create a database as localhost...!')
        dbroot = Toplevel()
        dbroot.grab_set()
        dbroot.geometry("500x300+500+250")
        dbroot.resizable(False, False)
        dbroot.config(bg='white')
        dbroot.title("Login To Database")

        # -----Connectdb Labels
        label1 = Label(dbroot, text="Login to database with MySQL credentials",
bg="white", fg="black", font=("verdana", 14, "bold"))
        label1.pack(side="top", fill=BOTH)

        userlabel = Label(dbroot, text="MySQL Id :", font=("verdana", 12),bg="white")
        userlabel.place(x=75, y=75)

        passwordlabel = Label(dbroot, text="MySQL Password :", font=("verdana",
12),bg="white")
        passwordlabel.place(x=75, y=150)

        # -----Connectdb Entry
        userval = StringVar()
        passwordval = StringVar()

        #-----Connectdb entrylabels
        userentry = Entry(dbroot, textvariable=userval, bd=1, bg="white",
relief="raised", width=30,font=("arial",15))
        userentry.place(x=75, y=110)

        passwordentry = Entry(dbroot, textvariable=passwordval, bd=1,
bg="white",font=("arial",15), relief="raised", width=30, show="*")
        passwordentry.place(x=75, y=185)

        # -----login button & exit button
        login_button = Button(dbroot, text="Login", bg="black",font=("arial",
12,"bold"),
                                command=submitdb)
        login_button.config(activebackground="black",
activeforeground="white",fg="white",padx=10,pady=10)
        login_button.place(x=75, y=235)

```



```

        exit_button = Button(dbroot, text="Exit", bg="black", font=("arial",
12, "bold"), fg="white",
                                command=dbroot.destroy)
        exit_button.config(activebackground="black",
activeforeground="#ffffff", padx=10, pady=10)
        exit_button.place(x=165, y=235)

        dbroot.mainloop()

##### END OF LOGINDATABASE
BUTTON #####

##### DEFINING TIME&DATE
METHODS #####
def tick():
    time_string = time.strftime("%H:%M:%S")
    date_string = time.strftime("%d/%m/%Y")
    clock.config(text='Date :' + date_string + "\n" + "Time : " + time_string)
    clock.after(200, tick)

##### END OF TIME&DATE
METHODS #####

##### IMPORTING & CREATING
MAINWINDOW #####
from tkinter import *
from tkinter import Toplevel, messagebox, filedialog
from tkinter.ttk import Treeview
from tkinter import ttk
import pandas
import mysql.connector
import time

#-----CREATING MAINWINDOW
root = Tk()
root.title("Employee Management System")
root.config(bg="white")
root.geometry("1100x700+200+50")
root.resizable(False, False)

##### CREATING TWO FRAMES IN MAIN
WINDOW #####

```

```

#----- CREATING & DEVELOPING DATA
ENTRY FRAME
DataEntryFrame = Frame(root, bg="white", bd=1, relief="groove")
DataEntryFrame.place(x=20, y=80, width=300, height=600)

#-----data entry frame labels
frontlabel = Label(DataEntryFrame, text="Actions", bg="black", fg="white",
                    font=("arial", 16))
frontlabel.pack(side="top", fill=BOTH)

addbtn = Button(DataEntryFrame, text="Add Employee", relief="raised",
                bg="black", fg="white", font=("verdana", 14),
                width=20, command=addemployee)
addbtn.place(x=20, y=100)

updatebtn = Button(DataEntryFrame, text="Update Record", relief="raised",
                  bg="black", fg="white", font=("verdana", 14),
                  width=20, command=updateemployee)
updatebtn.place(x=20, y=180)

searchbtn = Button(DataEntryFrame, text="Search Record", relief="raised",
                  bg="black", fg="white", font=("verdana", 14),
                  width=20, command=searchemployee)
searchbtn.place(x=20, y=260)

delete_button = Button(DataEntryFrame, text="Delete Record", relief="raised",
                      bg="black", fg="white", font=("verdana", 14),
                      width=20, command=deleteemployee)
delete_button.place(x=20, y=340)

showall_button = Button(DataEntryFrame, text="Show All Records", relief="raised",
                       bg="black", fg="white", font=("verdana", 14),
                       width=20, command=showallrecords)
showall_button.place(x=20, y=420)

exit_button = Button(DataEntryFrame, text="Exit", relief="raised",
                    bg="black", fg="white", font=("verdana", 14),
                    width=20, command=exit_window)
exit_button.place(x=20, y=500)

#-----END OF DATA
ENTRY FRAME

```

```

#-----CRAETING AND DEVELOPING SHOW
DATA FRAME
ShowDataFrame = Frame(root, bg='Lavender', borderwidth=2)
ShowDataFrame.place(x=380, y=80, width=700, height=600)

#-----making a table in showdataframe to get tree view of data
#-----making/creating treeview
style = ttk.Style()
style.configure('Treeview.Heading', font=('verdana', 12, 'bold'),
foreground='Black')
style.configure('Treeview', font=('Helvetica', 12), foreground='black',
background='lavender')
scroll_x = Scrollbar(ShowDataFrame, orient=HORIZONTAL)
scroll_y = Scrollbar(ShowDataFrame, orient=VERTICAL)
employeetable = Treeview(ShowDataFrame, columns=('Id', 'Name', 'Mobile No',
'Email', 'Address', 'Gender', 'Salary', 'Added Date', 'Added Time'),
yscrollcommand=scroll_y.set,
xscrollcommand=scroll_x.set)
scroll_x.pack(side=BOTTOM, fill=X)
scroll_y.pack(side=RIGHT, fill=Y)
scroll_x.config(command=employeetable.xview)
scroll_y.config(command=employeetable.yview)
employeetable.heading('Id', text='Id')
employeetable.heading('Name', text='Name')
employeetable.heading('Mobile No', text='Mobile No')
employeetable.heading('Email', text='Email')
employeetable.heading('Address', text='Address')
employeetable.heading('Gender', text='Gender')
employeetable.heading('Salary', text='Salary')
employeetable.heading('Added Date', text='Added Date')
employeetable.heading('Added Time', text='Added Time')
employeetable['show'] = 'headings'
employeetable.column('Id', width=100)
employeetable.column('Name', width=200)
employeetable.column('Mobile No', width=200)
employeetable.column('Email', width=300)
employeetable.column('Address', width=200)
employeetable.column('Gender', width=100)
employeetable.column('Salary', width=150)
employeetable.column('Added Date', width=150)
employeetable.column('Added Time', width=150)
employeetable.pack(fill=BOTH, expand=1)

#-----END OF
SHOW DATA FRAME

```

```

##### END OF TWO FRAME
DEVELOPMENTS #####

##### FOOTER / TOP SIDE OF MAIN
WINDOW(ROOT) #####

#----- MAIN LABEL/HEADING
main_label = Label(root, text="EMPLOYEE MANAGEMENT SYSTEM", bg="white",
fg="black",
                    font=("arial", 22, "bold"))
main_label.pack(side="top")

#----- DATE&TIME
clock = Label(root, font=('verdana', 10, 'bold'), borderwidth=3, bg='white',
fg="black")
clock.place(x=930, y=5)
tick()

#----- LOGIN DATABASE BUTTON
connectbutton = Button(root, text='Login SQL', width=16, font=("verdana", 14,
"bold"), command=Connectdb)
connectbutton.config(bg="black", fg="white", activebackground="yellow",
activeforeground="black")
connectbutton.place(x=5, y=5)

root.mainloop()

#-----> END OF CODE <-----
-----#

#ALL CODE WAS EXPLAINED IN DETAIL IN THE REPORT
#DONE BY:-
# AAYUSH KUMAR    rollno:-66
# FAIZAL          rollno:-65
# B.BRUCELEE      rollno:-59

# In[ ]:

# In[ ]:

```

