

# Course Based Learning

(Via Projects)

Project is Based on the Language Python

**“EMPLOYEE MANAGEMENT SYSTEM(EMP)”**

Submitted By

MOHAMMED AKHIL - 2020051505

Under the Guidance of

**Dr Vishwa Kiran S**

**Mr K B Hemanth Raj**



**Future Vision**

## Future Vision BIE

Future Vision Beginners Intermediate Experts (BIE)

Kaveri Nagar, R T Nagar Post, Bengaluru – 560032

# Employee Management System

## Abstract:

**Employee Management System** is a distributed application, developed to maintain the details of employees working in any organization. It maintains the information about the personal details of their employees, also the details about the payroll system which enable to generate the payslip. The application is actually a suite of applications developed using Java.

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.

The project has been an enriching experience for me in the field of programming and Enterprise Application development. The project has been developed to fulfill the requirements of the Employees in Labour Ministry.

## Introduction:-

Employee management is the modern computer based record management system of employees of any firm. It is found that this is very efficient way to manage records as well as attendance record through this system.

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.

## Background:

The application software takes care of database and day to day operations. For the ease of the user the web-based application is developed using ASP and SQL server in the back .

## Purpose / Objective:

The main goal of this project is to make the record of employee's easier & quicker.

- It is situated for all level of peoples.
- It provides proper details about the entire employee & their posts.
- User friendly environment makes the data handling more easily.
- It easily provides an environment where the user can get information about all

the employees / worker.

### Scope of new System:

- It is user friendly, can be accessed by any one.
- It has user id and password system to maintain privacy and security.
- It is very fast and accurate.
- No need of extra manual effort.
- Just need little knowledge to operate the system.
- Doesn't require any extra hardware device.

### Software requirements:

- Operating System : Windows XP or higher.
- Front End tool : MySQL
- Back End tool : Python.

### Motivation:-

Realizing a higher need of development efforts and the investment of time, developing uniform, more user-friendly application software for implementation. With keeping use in mind, supporting existing business process of DGLW appears as a fruitful concept for adding more value through a web based application. There by increasing quality of services offered.

### Existing System:

- Existing EMS is based on the standalone system.
- It is developed on the access 95 hence it is not compatible on modern operating system.

## Limitations of Existing System:

- Existing EMS is not user friendly.
- It is not provided with the detailed project information done or to be assigned based on the application.
- It needs extra manual power also.

## Proposed System:

This system consists of different table which contains the record of employees & it is commented through VB which is the front end. In VB we have seen the information about the employee. You can also provide the different buttons like, add, delete, edit, exit etc. which helps you to edit any data & make your work easier.

## System Requirement Specifications (Functional & Non-Functional):

The aim of the system is to develop **"EMPLOYEE MANAGEMENT SYSTEM"** software, which should automate the process to create and store employee details . The system is supposed to be used as a subsystem in a large office system, which could be manual system or a computerized one. Therefore, the proposed system must be able to function under both circumstances.

**The proposed system** is not a freeware and due to the usage of swings, becomes user interactive.

- The project demand a page of employee details that include:
- Employees personal detail.
- Employees salary, allowances, deductions.

## Problem definition:

- In the old system the main task of editing is not done easily& it will also take time. But in the proposed system the main assumptions are, the system should already contains all the hardware's & software's as well as the person who use this product should familiar with window XP or any operating system.

## Proposed Methodology:

### Solution strategies:

The main solution of the problem is to provide suitable & user friendly environment to a user so that the user can maintain the detail of employee in a very easy manner.

### References:

- Sample template and guidance given by our mentor Mr K.B Hemanth Raj during our course session.
- The course textbook, "**Python for Everybody**" by Charles R. Severance.
- The textbook, "**Software Engineering**" by IAN Sommerville.

### ABSTRACT:

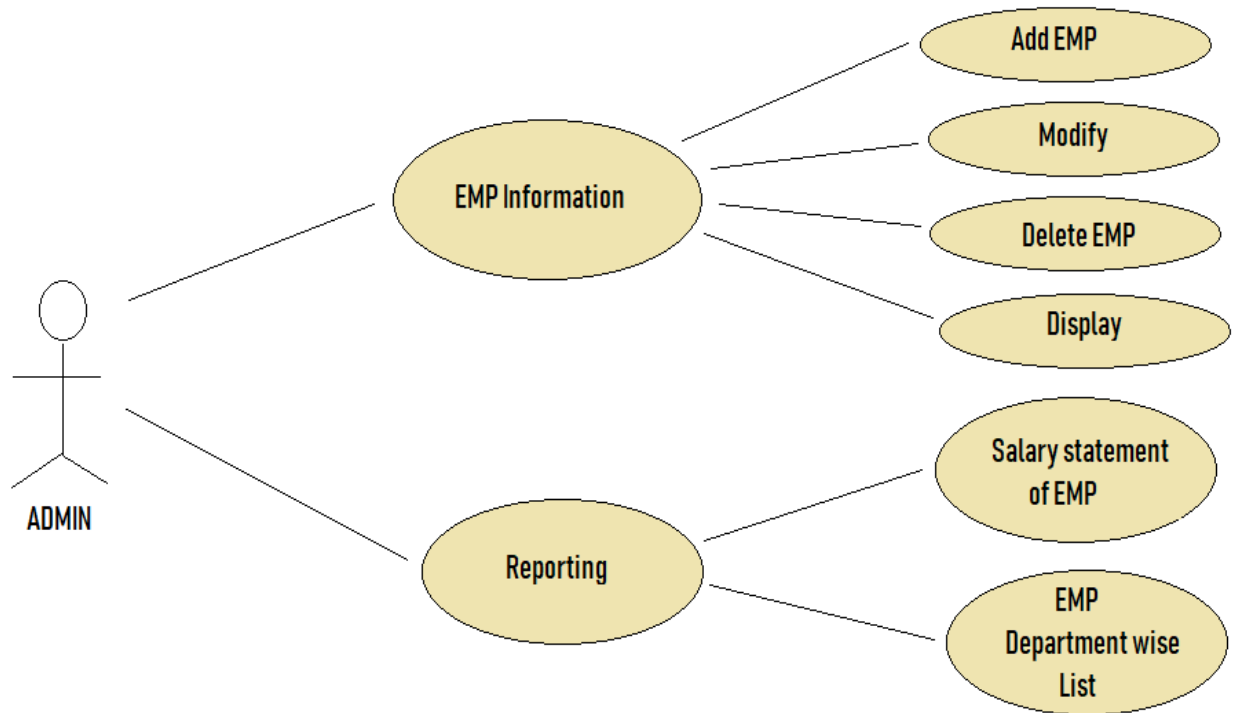
#### Defining A System:

Collections of components, which are interconnected, and work together to realize some objective, form a system. There are three major components in every system, namely input, processing and output.

### DESIGNING:

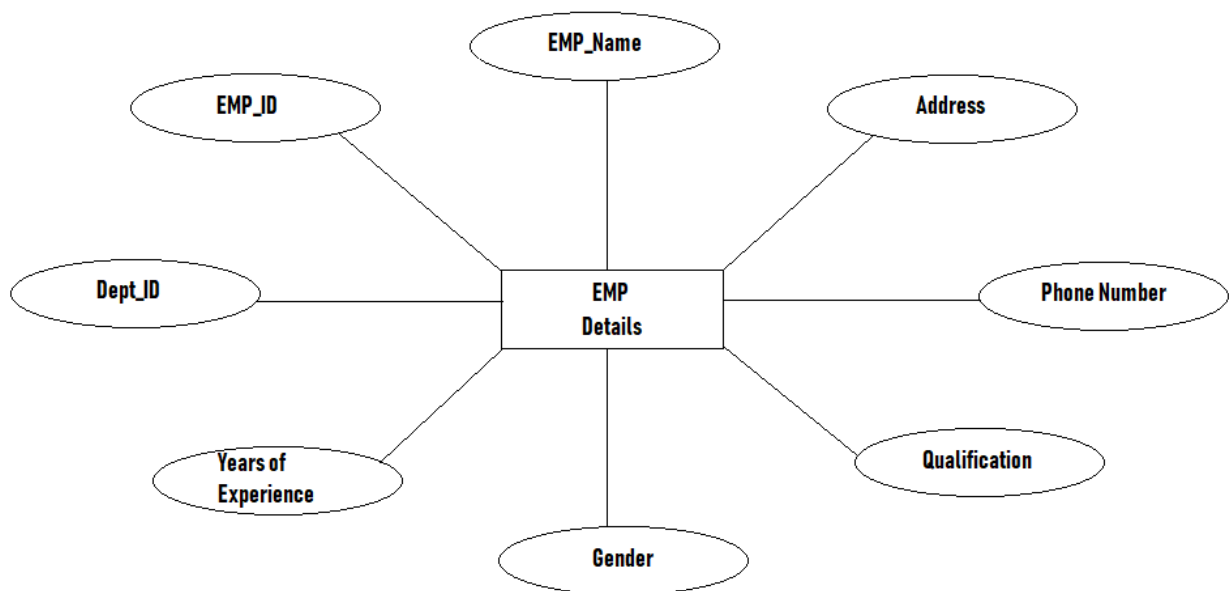
#### Use case diagram:

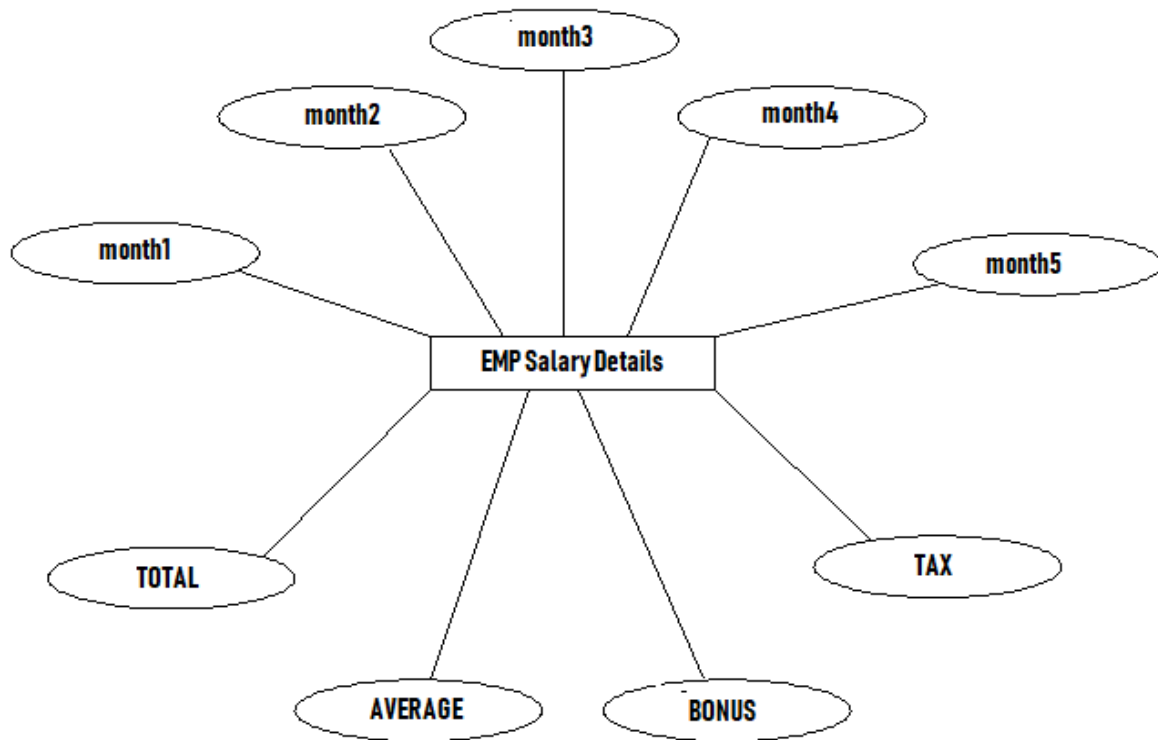
A use case diagram in the Unified Modelling Language(UML) is a type of behavioural diagram defined by and created from a use-case analysis. its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals(represented as use cases),and any dependencies between those use cases. Use case diagrams are formally included in two modelling languages defined by the OMG: the Unified Modelling Language(UML) and the systems Modelling Language(sys ML)



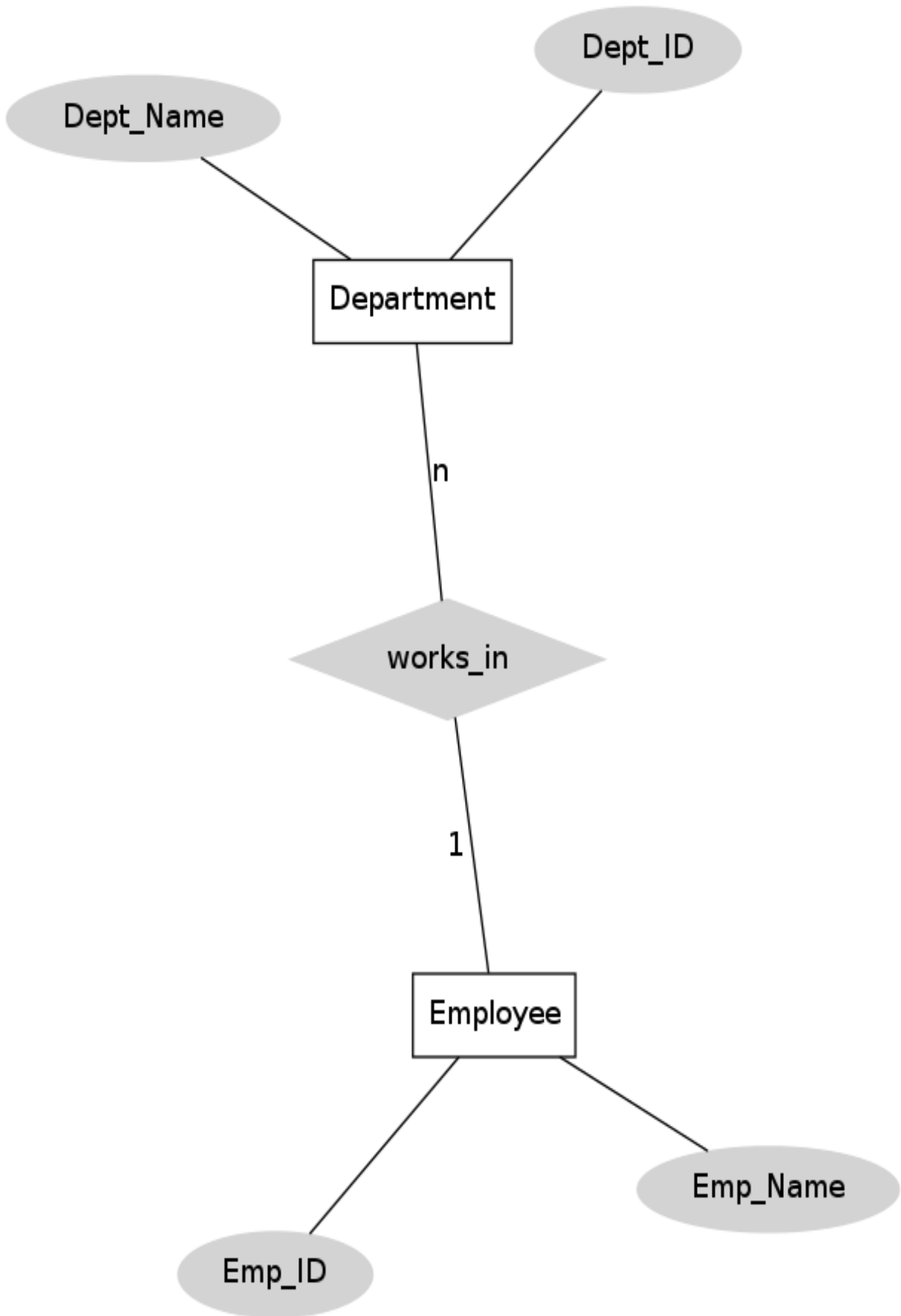
**ER Diagrams:**

Developing databases is a very important task to develop a system. Before going to form exact database tables and establishing relationships between them, we conceptually or logically can model our database using ER diagrams.





ER-DIAGRAMS:- Employee Personal and Salary Details.

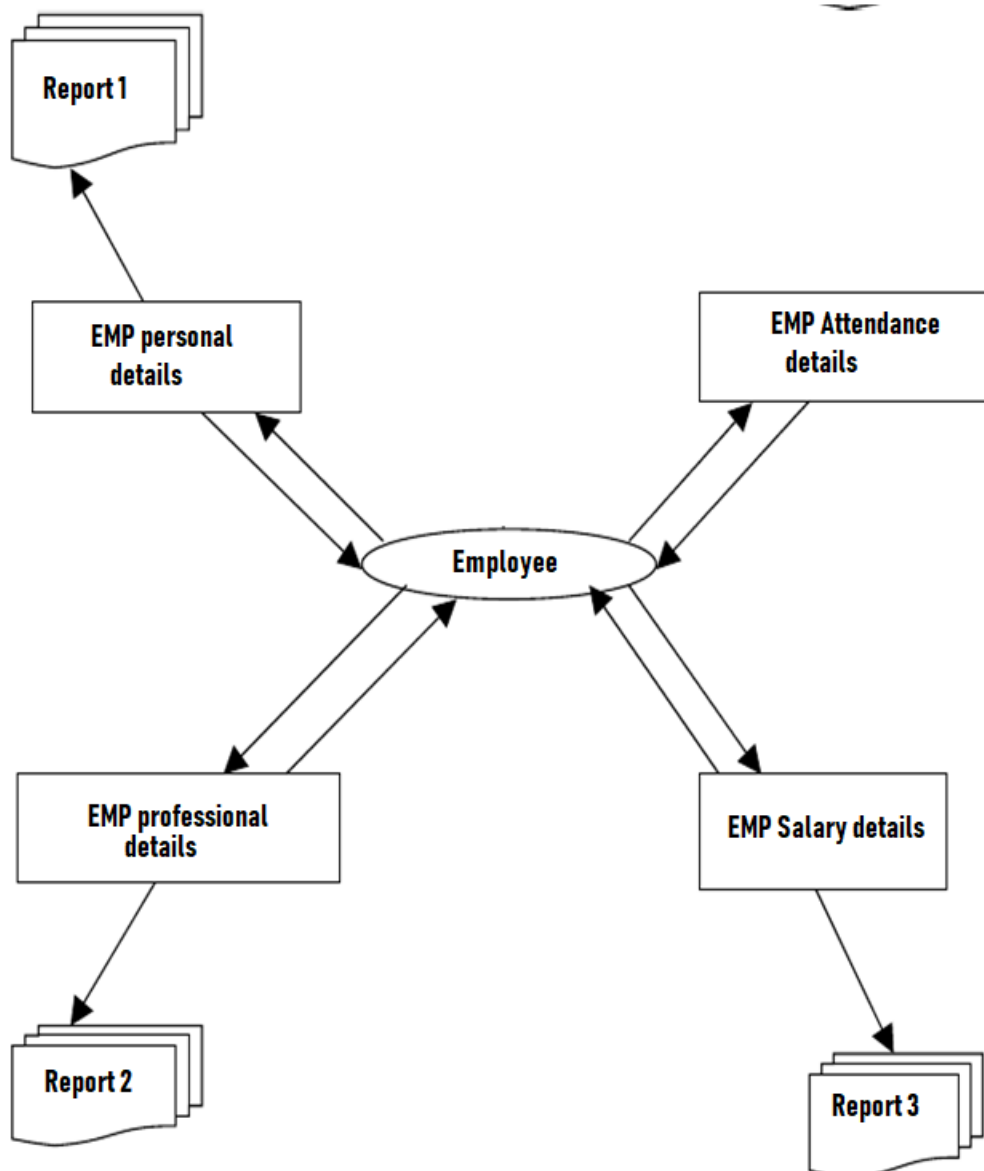




**ER- Diagram** - Showing Relation (1 to n) between Employee and Department

**Data Flow Diagram (DFD):**

A **data-flow diagram** is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops.



**DFD: Showing Employee Set of Details**

## **SYSTEM TESTING:**

Software testing is a crucial element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and the attendant "cost" associated with a software failure are motivating forces for well-planned, thorough testing. Testing is a set of activities that can be planned in advance and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it is vital success of the system.

### **Unit Testing:**

Unit testing focuses on the verification of smallest unit of software design of the module. To check whether each module in the software works properly so that it gives desired outputs to the given inputs. All validations and conditions are tested in the module level in unit test. Control paths are tested to ensure the information properly flows into and out of the program unit under test.

### **User Acceptance Testing:**

The performance of user interactive testing is actually the user show. The user gives live data and checks whether software is giving specified outputs.

## **SYSTEM IMPLEMENTATION:**

Implementation is the phase where the system goes for actual functioning. Hence in this phase one has to be cautious because all the efforts undertaken during the project will be fruitful only if the software is properly implemented according to the plans made. The implementation phase is less creative than system design. It is primarily concerned with user training, site preparation and file conversion.

## **SYSTEM MAINTENANCE:**

1. Corrective Maintenance
2. Adaptive Maintenance
3. Perfective Maintenance

## 4.Preventive Maintenance

**SOFTWARE LIFE CYCLE MODEL:**

The phases and the outputs of the waterfall model at each phase are summarized below in tabular format.

PHASE	OUTPUT
Problem Description	Feasibility Report
Analysis	SRS
Design	Detailed Design Report
Coding	Complete Source Code
Testing	Test plan, report and manual
Installation	Installation Report

**OBJECT ORIENTED CONCEPTS– JAVA:**

**Java** is a general computer programming language developed by Sun Microsystems. Originally called "Oak", by its inventor James Gosling, **Java** was designed with several innovative features. These include a language that is entirely object oriented, and the ability to write an application once and move it to virtually any platform.

**CLASSES:**

A class is the blueprint from which individual objects are created.

**OBJECTS:**

Objects are key to understanding object-oriented technology.

**Bundling code into individual software objects provides a number of benefits, including:**

- 1) Modularity
- 2) Information-hiding
- 3) Code re-use
- 4) Pluggability and debugging ease.

**INHERITANCE:**

Object-oriented programming allows classes to inherit commonly used state and behaviour from other classes. In the Java programming language, each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of subclasses.

**INTERFACE:**

An interface is a group of related methods with empty bodies.

**PACKAGE:**

A package is a namespace that organizes a set of related classes and interfaces.

**SOURCE CODE:**

```
import sqlite3

#it connects to the database
def connectDatabase():
    global conn
    conn=sqlite3.connect('EMP.db')
    print("opened database successfully")

#this would create a table
def CreateTable():

#employee details
    conn.execute("""CREATE TABLE IF NOT EXISTS EDETAILS
        (EMPID INT[20],
        EMPNAME VARCHAR[45],
        DEPTID INT[20],
        DPTNAME VARCHAR[50],
        ADDRESS VARCHAR[100],
        PHONE NUMBER INT[10],
        QUALIFICATION VARCHAR[10],
        YEARSOFEXP INT[20],
        GENDER VARCHAR[15]);""")
```

## #EMPLOYEE SALARY DETAILS

```
conn.execute("CREATE TABLE IF NOT EXISTS SDETAILS
(EMPID INT[20],
EMPNAME VARCHAR[45],
MONTH1 INT[30],
MONTH2 INT[30],
MONTH3 INT[30],
MONTH4 INT[30],
MONTH5 INT[30],
MONTH6 INT[30],
TOTAL INT[30],
AVERAGE INT[30],
BONUS INT[50],
TAX INT[50]);")
```

## #LEAVE

```
conn.execute("CREATE TABLE IF NOT EXISTS LDETAILS
(EMPID INT[20],
EMPNAME CHAR[45],
NO OF LEAVES INT[20]);")
```

## #EMPLOYEE EXP DETAILS

```
conn.execute("CREATE TABLE IF NOT EXISTS XDETAILS
(EMPID INT[20],
EMPNAME VARCHAR[45],
YEAR OF EXP INT[20],
SPECIFICATION VARCHAR[45],
DESIGNATION VARCHAR[45]);")
```

```
print("table created successfully")
```

```
#this will close a connection to database
```

```
def CloseDatabase():
```

```
conn.close()
```

```
#this would insert a new record to database
```

```
def NewAdmission():
```

```
    print("\n Employee Details:")
```

```
    empid=int(input("enter the employee id:"))
```

```
    empname=input("enter employee name:")
```

```
    dptid=int(input("enter department id:"))
```

```
    dptname=input("enter department name:")
```

```
    address=input("enter address:")
```

```
    phonenumber=int(input("enter phone number:"))
```

```
    qualification=input("enter qualification:")
```

```
    yearsofexp=int(input("enter number of years of experience:"))
```

```
    gender=input("enter gender:")
```

```
    conn.execute("INSERT INTO  
EDetails(EMPID,EMPNAME,DEPTID,DPTNAME,ADDRESS,PHONE,QUALIFICATION,YEARSOFEXP,GE  
NDER)VALUES(?,?,?,?,?,?,?,),(empid,empname,dptid,dptname,address,phonenumber,qualification,ye  
arsofexp,gender);")
```

```
#SALARY
```

```
print("\n Salary Details:")
```

```
empid=int(input("enter the employee id:"))
```

```
empname=input("enter employee name:")
```

```
month1=int(input("enter salary for month 1:"))
```

```
month2=int(input("enter salary for month 2:"))
```

```
month3=int(input("enter salary for month 3:"))
```

```
month4=int(input("enter salary for month 4:"))
```

```
month5=int(input("enter salary for month 5:"))
```

```
month6=int(input("enter salary for month 6:"))
```

```
total=month1+month2+month3+month4+month5+month6
```

```
print("Total=",total)
```

```
average=total/6
```

```
print("Average=",average)
```

```
bonus=total+1000
```

```
print("Bonus=",bonus)
```

```
tax=total/500

print("Tax=",tax)

conn.execute("INSERT INTO SDETAILS
(EMPID,EMPNAME,MONTH1,MONTH2,MONTH3,MONTH4,MONTH5,MONTH6,TOTAL,AVERAGE,BON
US,TAX)VALUES(?,?,?,?,?,?,?,?),(empid,empname,month1,month2,month3,month4,month5,mont
h6,total,average,bonus,tax));

#leave

print("\n Leave Details")

empid=int(input("enter the employee id:"))
empname=input("enter employee name:")
noofleaves=int(input("enter number of leaves:"))

conn.execute("INSERT INTO LDETAILS
(EMPID,EMPNAME,NO)VALUES(?,?,?),(empid,empname,noofleaves));

#EXPERIENCE

print("\n Experience Details:")

empid=int(input("enter the employee id:"))
empname=input("enter employee name:")
yearofexp=int(input("enter number of year of experience:"))
specialization=input("enter specialization:")
designation=input("enter designation:")

conn.execute("INSERT INTO
XDETAILS(EMPID,EMPNAME,YEAR,SPECIFICATION,DESIGNATION)VALUES(?,?,?,?,?),(empid,empnam
e,yearofexp,specialization,designation));

conn.commit()

#this would display employee details

def getEDetails():

    cursor=conn.execute("SELECT
EMPID,EMPNAME,DEPTID,DPTNAME,ADDRESS,PHONE,QUALIFICATION,YEARSOFEXP,GENDER
FROM EDETAILS")

    for row in cursor:

        print("EMP ID=",row[0])
        print("EMPNAME=",row[1])
```



```
print("DEPT ID=",row[2])
print("DEPT NAME=",row[3])
print("ADDRESS=",row[4])
print("PHONE=",row[5])
print("QUALIFICATION=",row[6])
print("YEAROFEXP=",row[7])
print("GENDER=",row[8],"\n")
```

#this would display employee's salary details

```
def getSDetails():
```

```
    cursor=conn.execute("SELECT
EMPID,EMPNAME,MONTH1,MONTH2,MONTH3,MONTH4,MONTH5,MONTH6,TOTAL,AVERAGE,BONU
S,TAX FROM SDETAILS")
```

```
    for row in cursor:
```

```
        print("EMP ID=",row[0])
        print("EMPNAME=",row[1])
        print("month1=",row[2])
        print("month2=",row[3])
        print("month3=",row[4])
        print("month4=",row[5])
        print("month5=",row[6])
        print("month6=",row[7])
        print("total=",row[8])
        print("average=",row[9])
        print("bonus=",row[10])
        print("tax=",row[11],"\n")
```

#this would display employee's leave details

```
def getLDetails():
```

```
    cursor=conn.execute("SELECT EMPID,EMPNAME,NO FROM LDETAILS")
```

```
    for row in cursor:
```

```
        print("empid=",row[0])
        print("empname=",row[1])
        print("number of leaves=",row[2],"\n")
```

```
#this would display employee's experience details
```

```
def getXDetails():
```

```
    cursor=conn.execute("SELECT EMPID,EMPNAME,YEAR,SPECIFICATION,DESIGNATION FROM  
XDETAILS")
```

```
    for row in cursor:
```

```
        print("empid=",row[0])
```

```
        print("empname=",row[1])
```

```
        print("year=",row[2])
```

```
        print("specification=",row[3])
```

```
        print("designation=",row[4],"\n")
```

```
#this would display all data
```

```
def getData():
```

```
    getEDetails()
```

```
    getSDetails()
```

```
    getLDetails()
```

```
    getXDetails()
```

```
#this would update employee details
```

```
def UpdateEDetails():
```

```
    #this would update employee details
```

```
    print("\n Employee Details:")
```

```
    empid=int(input("enter the employee id:"))
```

```
    empname=input("enter employee name:")
```

```
    dptid=int(input("enter department id:"))
```

```
    dptname=input("enter department name:")
```

```
    address=input("enter address:")
```

```
    phonenumber=int(input("enter phone number:"))
```

```
    qualification=input("enter qualification:")
```

```
    yearsofexp=int(input("enter number of years of experience:"))
```

```
    gender=input("enter gender:")
```

```
    conn.execute("UPDATE EDETAILS SET  
EMPNAME=?,DEPTID=?,DPTNAME=?,ADDRESS=?,PHONE=?,QUALIFICATION=?,YEARSOFEXP=?,GEND  
ER=? WHERE  
EMPID=?",(empname,dptid,dptname,address,phonenumber,qualification,yearsofexp,gender,empid))
```

```
print("total number of rows updated:",conn.total_changes)

#this would update employee's salary details
def UpdateSDetails():
#this would update salary details
    print("\n Salary Details:")
    empid=int(input("enter the employee id:"))
    empname=input("enter employee name:")
    month1=int(input("enter salary for month 1:"))
    month2=int(input("enter salary for month 2:"))
    month3=int(input("enter salary for month 3:"))
    month4=int(input("enter salary for month 4:"))
    month5=int(input("enter salary for month 5:"))
    month6=int(input("enter salary for month 6:"))
    total=month1+month2+month3+month4+month5+month6
    print("total=",total)
    average=total/6
    print("Average=",average)
    bonus=total+1000
    print("Bonus=",bonus)
    tax=total/500
    print("tax=",tax)

    conn.execute("UPDATE SDETAILS SET EMPNAME=?,MONTH1=?,
MONTH2=?,MONTH3=?,MONTH4=?,MONTH5=?,MONTH6=?,TOTAL=?,AVERAGE=?,BONUS=?,TAX=?
WHERE
EMPID=?",(empname,month1,month2,month3,month4,month5,month6,total,average,bonus,tax,empid))

    print("total number of rows updated:",conn.total_changes)

#this would update employee's leave details
def UpdateLDetails():
    print("Leave Details:")
    empid=int(input("enter the employee id:"))
    empname=input("enter employee name:")
    noofleaves=int(input("enter number of leaves:"))
```

```
conn.execute("UPDATE LDETAILS SET EMPNAME=?,NO=? WHERE  
EMPID=?", (empname, noofleaves, empid))
```

```
print("total number of rows updated:", conn.total_changes)
```

```
#this would update employee's experience details
```

```
def UpdateXDetails():
```

```
    print("\n Experience Details:")
```

```
    empid=int(input("enter the employee id:"))
```

```
    empname=input("enter employee name:")
```

```
    yearofexp=int(input("enter number of year of experience:"))
```

```
    specialization=input("enter specialization:")
```

```
    designation=input("enter designation:")
```

```
conn.execute("UPDATE XDETAILS SET EMPNAME=?,YEAR=?,SPECIFICATION=?,DESIGNATION=?  
WHERE EMPID=?", (empname, yearofexp, specialization, designation, empid))
```

```
print("total number of rows updated:", conn.total_changes)
```

```
conn.commit()
```

```
print("total number of rows updated:", conn.total_changes)
```

```
#this would update employee's all details
```

```
def UpdateDetails():
```

```
    UpdateEDetails()
```

```
    UpdateSDetails()
```

```
    UpdateLDetails()
```

```
    UpdateXDetails()
```

```
#this would delete employee details
```

```
def DeleteTuple():
```

```
    empid=int(input("enter the employee id to be deleted:"))
```

```
conn.execute("DELETE FROM EDETAILS WHERE EMPID=?", (empid,))
```

```
conn.execute("DELETE FROM SDETAILS WHERE EMPID=?", (empid,))
```

```
conn.execute("DELETE FROM LDETAILS WHERE EMPID=?", (empid,))

conn.execute("DELETE FROM XDETAILS WHERE EMPID=?", (empid,))

conn.commit()

print("total number of rows deleted:", conn.total_changes)
```

#this would search employee information-All

```
def SearchEmployeeInfo():
```

```
    empid=int(input("enter the ID of employee"))
```

```
    cursor=conn.execute("SELECT
ED.EMPID,ED.EMPNAME,ED.DEPTID,ED.DPTNAME,ED.ADDRESS,ED.PHONE,ED.QUALIFICATION,ED.Y
EARSOFFEXP,ED.GENDER,SD.MONTH1,SD.MONTH2,SD.MONTH3,SD.MONTH4,SD.MONTH5,SD.MONT
H6,LD.NO,XD.SPECIFICATION,XD.DESIGNATION FROM EDETAILS AS ED,SDETAILS AS SD,LDETAILS
AS LD,XDETAILS AS XD WHERE ED.EMPID=SD.EMPID=LD.EMPID=XD.EMPID AND
ED.EMPID=?", (empid,))
```

```
    for row in cursor:
```

```
        print("employee id is :",row[0])
        print(" employee name is :",row[1])
        print(" employee's department id :",row[2])
        print("employee's department name :",row[3])
        print(" employee's address :",row[4])
        print(" employee's phone number :",row[5])
        print(" employee's qualification :",row[6])
        print(" employee's years of experience :",row[6])
        print(" employee's gender :",row[7])
        print(" employee's month1 salary :",row[8])
        print(" employee's month2 salary :",row[9])
        print("employee's month3 salary :",row[10])
        print("employee's month4 salary :",row[11])
        print("employee's month5 salary :",row[12])
        print(" employee's month6 salary :",row[13])
        print(" no of leaves applied by employee :",row[14])
        print(" employee's specialization :",row[15])
```

```
print(" employee's  designation :",row[16])

#this would search only employee details
def SearchEDetails():
    cursor=conn.execute("SELECT
EMPID,EMPNAME,DEPTID,DPTNAME,ADDRESS,PHONE,QUALIFICATION,YEARSOFEXP,GENDER
FROM EDETAILS")
    for row in cursor:
        print("EMP ID=",row[0])
        print("EMPNAME=",row[1])
        print("DEPT ID=",row[2])
        print("DEPT NAME=",row[3])
        print("ADDRESS=",row[4])
        print("PHONE=",row[5])
        print("QUALIFICATION=",row[6])
        print("YEARSOFEXP=",row[7])
        print("GENDER=",row[8],"\n")

connectDatabase()
CreateTable()
while True:
    print("Choose the options:\n 1.New Admission\n 2.view Individual Data-All")
    print("3.view All Data\n 4.Update Data")
    print("5.Delete Data\n 6.Search information\n 7.Search EmployeeDetails")
    print("8.Exit")
    data=int(input())
    if data==8:
        break;
    elif data==1:
        NewAdmission()
    elif data==2:
        while True:
            print("choose the options:\n 1.view Employee Details\n 2.view Salary Details\n 3.view
LeaveDetails\n 4.view Experience Details\n 5.exit")
            data=int(input())
```

```
    if data==5:
        break;
    elif data==1:
        getEDetails()
    elif data==2:
        getSDetails()
    elif data==3:
        getLDetails()
    elif data==4:
        getXDetails()
    else:
        print("Enter valid option")
elif data==3:
    getData()
elif data==4:
    while True:
        print("choose the options:\n 1.update individual records\n 2.update All Data\n 3.exit")
        data=int(input())
        if data==3:
            break;
        elif data==1:
            print("choose the options:\n 1.update employee details\n 2.update employee's salary
            details\n 3.update employee's leave details\n 4.update employee's experience details\n 5.exit")
            data=int(input())
            if data==5:
                break;
            elif data==1:
                UpdateEDetails()
            elif data==2:
                UpdateSDetails()
            elif data==3:
                UpdateLDetails()
            elif data==4:
                UpdateXDetails()
        else:
```

```
        print("enter valid option")

    elif data==2:
        UpdateDetails()
    else:
        print("enter valid option")

elif data==5:
    DeleteTuple()
elif data==6:
    SearchEmployeeInfo()
elif data==7:
    SearchEDetails()
else:
    print("enter valid option")

CloseDatabase()
print("THANK YOU")
```

## OUTPUT

### Database Design:

#### 1)Employee database design details:



Python 3.8.3 Shell - C:/Users/prema a/Downloads/EMP output1.py (3.8.3)

DB Browser for SQLite - C:\Users\prema a\Downloads\EMP.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: EDETAILS

	EMPID	EMPNAME	DEPTID	DPTNAME	ADDRESS	PHONE	QUALIFICATION	YEARSOFEXP	GENDER
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2	kitty	21	ece	bpet	8296319522	md	5	female
2	3	kittuy	31	eee	blore	7483829631	mbbs	5	female
3	1	kittu	11	cse	kgf	7568453542	MTech	5	male

2) Salary database design details:

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: SDETAILS New Record Delete Record

EMPID	EMPNAME	MONTH1	MONTH2	MONTH3	MONTH4	MONTH5	MONTH6	TOTAL	AVERAGE	BONUS	TAX
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1 2	kitty	35000	35500	36000	34000	33000	36000	209500	34916.666666...	210500	419
2 3	kittuy	30000	30500	31000	31500	32000	32500	187500	31250	188500	375
3 1	kittu	36000	32000	30900	35500	31000	29000	194400	32400	195400	388.8

### 3)Leave database design details:

The screenshot shows the DB Browser for SQLite interface. The title bar indicates the database file is located at C:\Users\prema a\Downloads\EMP.db. The menu bar includes File, Edit, View, Tools, and Help. The toolbar contains icons for New Database, Open Database, Write Changes, and Revert Changes. The main window has tabs for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. The 'Browse Data' tab is active, showing a table named 'LDETAILS'. The table has three columns: EMPID, EMPNAME, and NO. The data is as follows:

	EMPID	EMPNAME	NO
	Filter	Filter	Filter
1	2	kitty	2
2	3	kittuy	2
3	1	kittu	2

At the bottom of the window, there are navigation icons and a status bar showing '1 - 3 of 3'.

### 4)Experience Database Design Details:

File Edit View Tools Help

New Database
 Open Database
 Write Changes
 Revert Changes
 Open Project
 Save Project

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: XDETAILS

EMPID	EMPNAME	YEAR	SPECIFICATION	DESIGNATION
Filter	Filter	Filter	Filter	Filter
1 2	kitty	5	data hacker	data analyst
2 3	kittuy	5	software applicant	software engineer
3 1	kittu	5	data analyst	team leader

**OUTPUT:**

```
EMP output1.py - C:\Users\prema a\Downloads\EMP output1.py (3.8.3)
File Edit Format Run Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [I
Type "help", "copyright", "credits" or "license()" for more
>>>
===== RESTART: C:\Users\prema a\Downloads\EMP main p
opened database successfully
table created successfully
Choose the options:
 1.New Admission
 2.view Individual Data-All
 3.view All Data|
 4.Update Data
 5.Delete Data
 6.Search information
 7.Search EmployeeDetails
 8.Exit
1

Employee Details:
enter the employee id:001
enter employee name:kittu
enter department id:11
enter department name:cse
enter address:kgf
enter phone number:7568453542
enter qualification:MTech
enter number of years of experience:5
enter gender:male

Salary Details:
enter the employee id:001
enter employee name:kittu
enter salary for month 1:36000
enter salary for month 2:32000
enter salary for month 3:30900
enter salary for month 4:35500
enter salary for month 5:31000
enter salary for month 6:29000
Total= 194400
Average= 32400.0
Bonus= 195400
Tax= 388.8
```

EMP output1.py - C:\Users\prema a\Downloads\EMP output1.py

File Edit Format Run Options Window Help

```
enter department name:000  
enter address:kgf  
enter phone number:7568453542  
enter qualification:MTech  
enter number of years of experience:5  
enter gender:male
```

#### Salary Details:

```
enter the employee id:001  
enter employee name:kittu  
enter salary for month 1:36000  
enter salary for month 2:32000  
enter salary for month 3:30900  
enter salary for month 4:35500  
enter salary for month 5:31000  
enter salary for month 6:29000  
Total= 194400  
Average= 32400.0  
Bonus= 195400  
Tax= 388.8
```

#### Leave Details

```
enter the employee id:001  
enter employee name:kittu  
enter number of leaves:2
```

#### Experience Details:

```
enter the employee id:001  
enter employee name:kittu  
enter number of year of experience:5  
enter specialization:data analysist  
enter designation:team leader  
Choose the options:  
1.New Admission  
2.view Individual Data-All  
3.view All Data  
4.Update Data  
5.Delete Data  
6.Search information  
7.Search EmployeeDetails  
8.Exit
```



Type here to search

EMP output.py - C:\Users\prema a\Downloads\EMP outp

File Edit Format Run Options Window Help

```
8.Exit
|
choose the options:
  1.view Employee Details
  2.view Salary Details
  3.view LeaveDetails
  4.view Experience Details
  5.exit
```

```
1
EMP ID= 1
EMPNAME= KITTU
DEPT ID= 11
DEPT NAME= cse
ADDRESS= kgf
PHONE= 7483975882
QUALIFICATION= MTech
YEAROFEXP= 5
GENDER= male
```

```
EMP ID= 1
EMPNAME= kittu
DEPT ID= 11
DEPT NAME= cse
ADDRESS= kgf
PHONE= 7483975882
QUALIFICATION= mbbs
YEAROFEXP= 5
GENDER= male
```

```
EMP ID= 2
EMPNAME= kitty
DEPT ID= 21
DEPT NAME= ece
ADDRESS= bpet
PHONE= 8296319522
QUALIFICATION= md
YEAROFEXP= 5
GENDER= female
```

```
EMP ID= 3
EMPNAME= kittuy
DEPT ID= 21
```



Type here to search



EMP output.py - C:\Users\prema a\Downloads\EMP output.py (3.8.3)

File Edit Format Run Options Window Help

```
EMPNAME= kittuy
DEPT ID= 31
DEPT NAME= eee
ADDRESS= blore
PHONE= 7483829631
QUALIFICATION= mbbs
YEAROFEXP= 5
GENDER= female
```

choose the options:

- 1.view Employee Details
- 2.view Salary Details
- 3.view LeaveDetails
- 4.view Experience Details
- 5.exit

2

```
EMP ID= 1
EMPNAME= kittu
month1= 36000
month2= 34000
month3= 35200
month4= 34600
month5= 34500
month6= 33600
total= 207900
average= 34650
bonus= 208900
tax= 415.8
```

```
EMP ID= 1
EMPNAME= kittu
month1= 36000
month2= 35000
month3= 32590
month4= 35500
month5= 36000
month6= 36500
total= 211590
average= 35265
bonus= 212590
tax= 423.18
```



Type here to search



 \*Python 3.8.3 Shell\*

File Edit Shell Debug Options Window Help

```
bonus= 10000  
tax= 375
```

```
EMP ID= 1  
EMPNAME= kittu  
month1= 36000  
month2= 32000  
month3= 30900  
month4= 35500  
month5= 31000  
month6= 29000  
total= 194400  
average= 32400  
bonus= 195400  
tax= 388.8
```

choose the options:

- 1.view Employee Details
- 2.view Salary Details
- 3.view LeaveDetails
- 4.view Experience Details
- 5.exit

3

```
empid= 2  
empname= kitty  
number of leaves= 2
```

```
empid= 3  
empname= kittuy  
number of leaves= 2
```


```
empid= 1  
empname= kittu  
number of leaves= 2
```

choose the options:

- 1.view Employee Details
- 2.view Salary Details
- 3.view LeaveDetails
- 4.view Experience Details
- 5.exit



Type here to search

 \*Python 3.8.3 Shell\*

File Edit Shell Debug Options Window Help

choose the options:

- 1.view Employee Details
- 2.view Salary Details
- 3.view LeaveDetails
- 4.view Experience Details
- 5.exit

4

```
empid= 2
empname= kitty
year= 5
specification= data hacker
designation= data analysist
```

```
empid= 3
empname= kittuy
year= 5
specification= software applicant
designation= software engineer
```

```
empid= 1
empname= kittu
year= 5
specification= data analysist
designation= team leader
```

choose the options:

- 1.view Employee Details
- 2.view Salary Details
- 3.view LeaveDetails
- 4.view Experience Details
- 5.exit

5

Choose the options:

- 1.New Admission
- 2.view Individual Data-All
- 3.view All Data
- 4.Update Data
- 5.Delete Data
- 6.Search information
- 7.Search EmployeeDetails
- 8.Exit

|



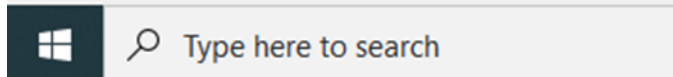
Type here to search

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
EMP ID= 2
EMPNAME= kitty
DEPT ID= 21
DEPT NAME= ece
ADDRESS= bpet
PHONE= 8296319522
QUALIFICATION= md
YEAROFEXP= 5
GENDER= female

EMP ID= 3
EMPNAME= kittuy
DEPT ID= 31
DEPT NAME= eee
ADDRESS= blore
PHONE= 7483829631
QUALIFICATION= mbbs
YEAROFEXP= 5
GENDER= female

EMP ID= 1
EMPNAME= kittu
DEPT ID= 11
DEPT NAME= cse
ADDRESS= kgf
PHONE= 7568453542
QUALIFICATION= MTech
YEAROFEXP= 5
GENDER= male

Choose the options:
1.New Admission
2.view Individual Data-All
3.view All Data
4.Update Data
5.Delete Data
6.Search information
7.Search EmployeeDetails
8.Exit
8
THANK YOU
>>> |
```



## CONCLUSION:

The Employee Management System didn't automate 100% of their work, but it is really a good start to computerize everything and entire Detail can be 100% computerized.

As far as the work done so far much care was given about the user friendliness and a very good interaction with the end users. The interface are so designed and channelled the users can never make any mistake while using the application, for an example while adding new record, user can't go out without either saving or cancelling the operation, till the time either they save or cancel the

Since this project has been designed exclusively as a project, certain complexities that do faced by any real life manual problem like total no. of employee, address redundancy etc. are considered in this project. But enhancement to the project can easily be made without changing the current design and programming structure.