Programme: Computer Engineering — Academic Year: 2023 – 2024

Course : Programming with Python — Course Code: PWP- 22616

Semester : Sixth

MICRO PROJECT REPORT

ON

Microproject Title

Submitted by the group of 4 students,

Sr. No	Roll No	Full name of Student	Enrollment No	Exam Seat No
1	03	Badgujar Mohit Hemant	2100590050	392320
2	05	Bhatt Atharva Yogesh	2100590054	392323
3	13	Deore Samarthya Ravindra	2100590064	392332
4	32	Makhija Dhruv Harish	2100590102	392362

Under the Guidance of

Ms. Surekha H. Patil

in

Three Years Diploma Programme in Engineering and Technology of
Maharashtra State Board of Technical Education, Mumbai (Autonomous)
ISO 9001: 2008 (ISO/IEC-27001:2013)

at

0059 – Shri Shivaji Vidya Prasarak Sanstha's
Bapusaheb Shivajirao Deore Polytechnic, Vidyanagari, Deopur, Dhule-424005

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

Certificate

This is to certify that Mr. Badgujar Mohit Hemant

of **Sixth Semester** of **Diploma Programme** in **Computer Engineering** at **Shri Shivaji Vidya Prasarak Sanstha's Bapusaheb Shivajirao Deore Polytechnic, Dhule (Institute Code: 0059)** has completed the **Micro Project** satisfactorily in Subject **Programming in Python (22616)** in the academic year **2023-2024** as prescribed in the MSBTE curriculum of I Scheme.

Place: Dhule

Enrollment No: 2100590050

Date: ____ / ____/ 2024

Exam. Seat No: 392320

Project Guide **Head of the Department** **Principal**

Seal of Institute

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

Certificate

This is to certify that Mr. Bhatt Atharva Yogesh

of **Sixth Semester** of **Diploma Programme** in **Computer Engineering** at **Shri Shivaji Vidya Prasarak Sanstha's Bapusaheb Shivajirao Deore Polytechnic, Dhule (Institute Code: 0059)** has completed the **Micro Project** satisfactorily in Subject **Programming in Python (22616)** in the academic year **2023-2024** as prescribed in the MSBTE curriculum of I Scheme.

Place: Dhule

Enrollment No: 2100590054

Date: ____ / ____/ 2024

Exam. Seat No: 392323

Project Guide **Head of the Department** **Principal**

Seal of Institute

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

Certificate

This is to certify that Mr. Deore Samarthya Ravindra

of **Sixth Semester** of **Diploma Programme** in **Computer Engineering** at **Shri Shivaji Vidya Prasarak Sanstha's Bapusaheb Shivajirao Deore Polytechnic, Dhule (Institute Code: 0059)** has completed the **Micro Project** satisfactorily in Subject **Programming in Python (22616)** in the academic year **2023-2024** as prescribed in the MSBTE curriculum of I Scheme.

Place: Dhule

Enrollment No: 2100590064

Date: ____ / ____/ 2024

Exam. Seat No: 392332

Project Guide **Head of the Department** **Principal**

Seal of Institute

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

Certificate

This is to certify that Mr. Makhija Dhruv Harish

of **Sixth Semester** of **Diploma Programme** in **Computer Engineering** at **Shri Shivaji Vidya Prasarak Sanstha's Bapusaheb Shivajirao Deore Polytechnic, Dhule (Institute Code: 0059)** has completed the **Micro Project** satisfactorily in Subject **Programming in Python (22616)** in the academic year **2023-2024** as prescribed in the MSBTE curriculum of I Scheme.

Place: Dhule

Enrollment No: 2100590102

Date: ____ / ____/ 2024

Exam. Seat No: 392362

Project Guide **Head of the Department** **Principal**

Seal of Institute

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

Certificate

This is to certify that,

Sr. No	Roll No	Full name of Student	Enrollment No	Exam Seat No
1	03	Badgujar Mohit Hemant	2100590050	392320
2	05	Bhatt Atharva Yogesh	2100590054	392323
3	13	Deore Samarthya Ravindra	2100590064	392332
4	32	Makhija Dhruv Harish	2100590102	392362

students of **Sixth** Semester **Diploma Programme** in **Computer Engineering** at **Shri Shivaji Vidya Prasarak Sanstha's Bapusaheb Shivajirao Deore Polytechnic, Dhule (Institute Code : 0059),** have completed the **Micro Project** satisfactorily in Subject **Programming With Python (22616)** in the academic year **2023–2024** as prescribed in the MSBTE curriculum of I Scheme.

Place: Dhule   Date: ____ / ____/ 2024

Project Guide   **Head of the Department**   **Principal**

Seal of Institute

Part A: Micro-Project Proposal

Employee Record Management System

1. Introduction:

Our Employee Record Management System, built with Python, offers a pragmatic solution for businesses in need of an efficient and user-friendly approach to manage their employee records. Equipped with features to add, view, update, and delete employee details, our application enables organizations to uphold precise records and make well-informed HR decisions. With ongoing development, we strive to augment its capabilities and foster better practices in workforce management.

2. Aims/Benefits of the Micro-Project

To develop employee record management system using XAMPP Database, Tkinter and MySQL components which will be able to add, update, view and delete employee related information.

3. Course Outcomes Addressed:

a) Develop Python program to demonstrate use of Operators
b) Perform operations on data structures in Python.
c) Develop functions for given problem

4. Proposed Methodology:

In Employee Record Management System, we accept, display, update and delete the data from the database.

Home Page with a Welcome message, Add Employee Details Button, View Employee Details Button, Update Employee Details button and an Exit Button.

To Add an Employe click on Add Employee button, then enter the details of Employee. Then enter Employee Details, enter Employee ID, Name, Phone Number, Email Address, Age, Gender, Designation and Salary of Employee and click on Submit button. If all the details are valid then data will be saved successfully in the Database.

To View all Employee Details, click on View Employee Details. Once clicked, all the Employee Details in the Database will be shown in Table Format.

To Update Details, click on Update Employee Details. Then enter the Employee ID of Employee whose details you want to edit. If the employee with entered employee id does not exist in the database, a message box of “Employee Record does not exists” will be displayed. If the entered employee id is valid then select the field you want to update, you can select a field from Name, Phone Number, Email Address, Age, Designation and Salary. After selecting field, enter the details and click on submit button, if the details are valid then it will be updated/saved in the database else an invalid message will be displayed.

To Delete any Employee’s Details, click on Delete Employee button and enter the Employee ID of employee. If Employee with entered employee id exist in the database then it’s all details will be deleted, else an “Employee does not exist” message will be displayed.

To Exit from the Employee Management System, click on Exit button.

5. Resources Required:

Sr. no	**Name of resources**	**Specification**	**Quantity**	**Remark**
1	Computer system	**Processor:** Intel(R) Pentium(R) Dual CPU E2140@1.60GHz1.60GHz **RAM:** 512 MB	1	-
2	Operating System	**OS:** Windows 7(32bit)	1	-

6. Action Plan:

Sr. No.	Details of Activity	Planned Start Date	Planned Finish Date	Name of Responsible Team Members
1	Data Collection	08/01/2024	27/01/2024	Badgujar Mohit Hemant
2	Analysis	29/01/2024	17/02/2024	Bhatt Atharva Yogesh
3	Design	19/02/2024	09/03/2024	Deore Samarthya Ravindra
4	Development	11/03/2024	23/03/2024	Makhija Dhruv Harish
5	Report Writing	26/03/2024	28/03/2024	Deore Samarthya Ravindra

Part B Micro-Project Report

Employee Record Management System

1. Rationale:

The Employee Record Management System, built using Python, offers a pragmatic approach for businesses aiming to streamline the management of their employee data. Our application, powered by Python, facilitates effortless addition, viewing, updating and deletion of employee details, empowering organizations to uphold precise records and facilitate informed HR decisions. With ongoing enhancements to the system, we strive to augment its capabilities continuously, thereby fostering enhanced practices in workforce management.

2. Course Outcomes Achieved:

a) Develop Python program to demonstrate use of Operators

b) Perform operations on data structures in Python.

c) Design classes for given problems

3. Actual Methodology Followed:

1. Define Requirements: Understand the needs of the system. Identify what features are essential, such as adding, viewing, updating and deleting employee records. Determine if any additional features or functionalities are required.

2. Design Database Schema: Design the structure of the database to store employee records. Decide what information needs to be stored for each employee, such as name, ID, department, etc. Choose an appropriate database system like SQLite, MySQL, or PostgreSQL.

3. Set Up the Python Environment: Install Python on your system if not already installed. Decide if you want to use any frameworks or libraries such as Flask or Django for web development or Tkinter for desktop GUI applications.

4. Create the Employee Class: Define a Python class to represent an employee. Include attributes and methods to manipulate employee data, such as adding, updating, or deleting employee records.

5. Implement CRUD Operations: Implement functions or methods to perform CRUD (Create, Read, Update, Delete) operations on employee records. These functions will interact with the database to store or retrieve employee information.

6. Develop User Interface (UI): Depending on your requirements, create a user interface for interacting with the system. This could be a command-line interface (CLI), a desktop GUI using Tkinter, or a web interface using Flask or Django.

7. Implement Authentication and Authorization: If necessary, implement authentication and authorization mechanisms to control access to the system. This ensures that only authorized users can perform certain actions, such as adding or editing employee records.

8. Testing: Test the system thoroughly to ensure that it functions as expected. Test each feature and functionality to identify and fix any bugs or issues.

9. Documentation: Document the system, including its architecture, functionalities, usage instructions, and any other relevant information.

10. Deployment: Deploy the system in the desired environment, whether it's on a local machine, a server, or a cloud platform. Ensure that the system is accessible to users and meets performance requirements.

11. Maintenance and Updates: Regularly maintain and update the system to address any bugs, security vulnerabilities, or new requirements that arise over time. Consider user feedback and make improvements accordingly.

A. Algorithm

Step 1: Import necessary libraries: tkinter, messagebox, and pymysql.

Step 2: Establish a connection to the MySQL database.

Step 3: Define functions for various operations such as adding, viewing, deleting, updating, and exiting employee details.

Step 4: For adding employee details:

a. Define a function `emp_add()` which creates a new Tkinter window for adding employee details.

b. Within the `emp_add()` function, define nested functions for handling back, submit, and clearing entries.

c. Perform data validation for each input field.

d. Check if the employee ID already exists in the database.

e. If not, insert the employee details into the database.

Step 5: For viewing employee details:

a. Define a function `emp_view()` which creates a new Tkinter window for viewing employee details.

b. Retrieve all employee records from the database and display them in a tabular format.

Step 6: For deleting employee details:

a. Define a function `emp_delete()` which creates a new Tkinter window for deleting employee details.

b. Define nested functions for handling back, clearing entries, and submitting the delete operation.

c. Check if the employee ID exists in the database, if yes, delete the corresponding record.

Step 7: For updating employee details:

a. Define a function `emp_update()` which creates a new Tkinter window for updating employee details.

b. Define nested functions for handling back, clearing entries, and submitting the update operation.

c. Perform data validation for the new value to be updated.

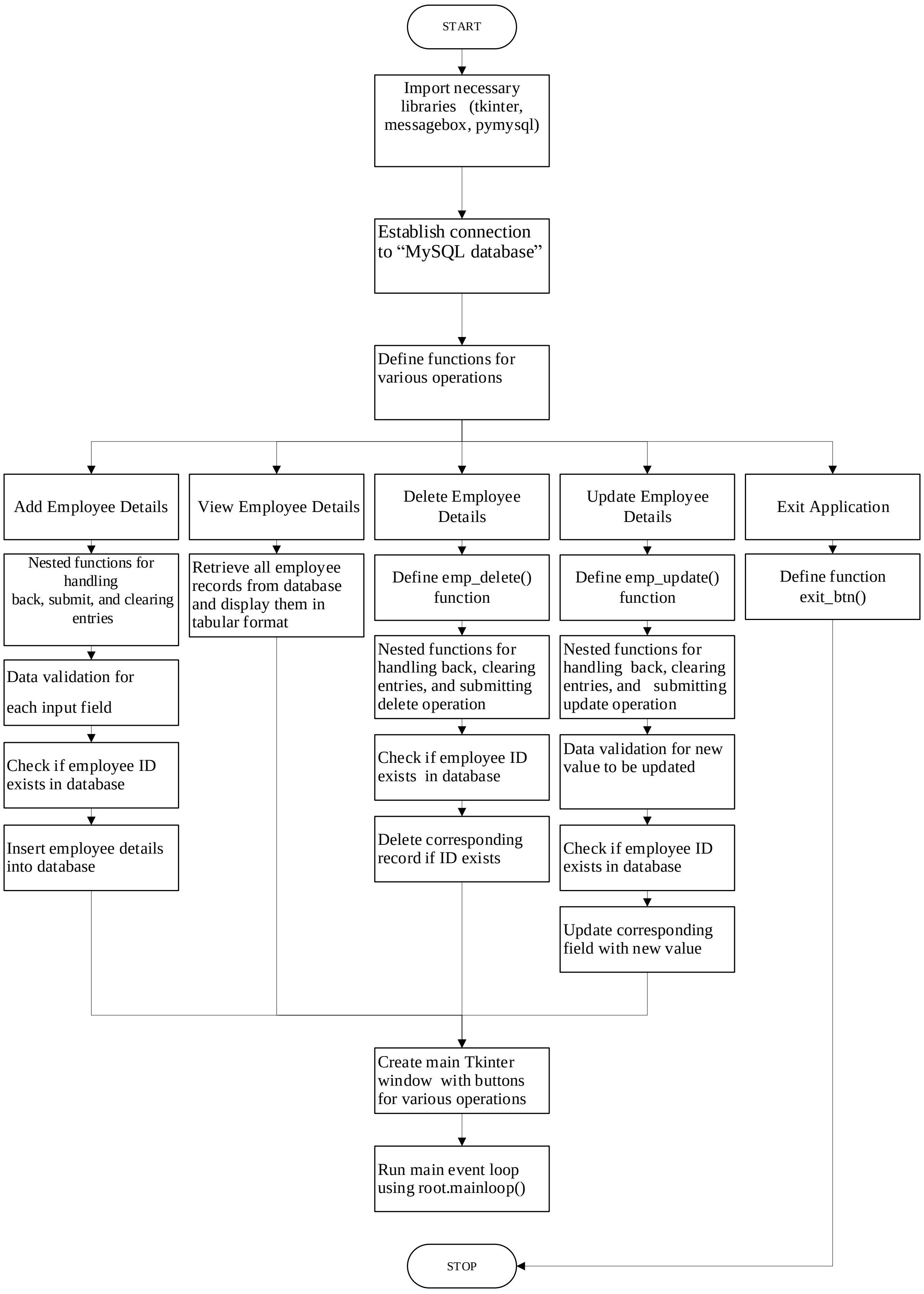
d. Check if the employee ID exists in the database, if yes, update the corresponding field with the new value.

Step 8: Define a function `exit_btn()` to exit the application.

Step 9: Create the main Tkinter window with buttons for adding, viewing, deleting, updating, and exiting employee details.

Step 10: Run the main event loop using `root.mainloop()` to display the GUI and handle user interactions.

B. Flowchart



START
Import necessary libraries (tkinter, messagebox, pymysql)
Establish connection to "MySQL database"
Define functions for various operations
Add Employee Details
View Employee Details
Delete Employee Details
Update Employee Details
Exit Application
Nested functions for handling back, submit, and clearing entries
Data validation for each input field
Check if employee ID exists in database
Insert employee details into database
Retrieve all employee records from database and display them in tabular format
Define emp_delete() function
Nested functions for handling back, clearing entries, and submitting delete operation
Check if employee ID exists in database
Delete corresponding record if ID exists
Define emp_update() function
Nested functions for handling back, clearing entries, and submitting update operation
Data validation for new value to be updated
Check if employee ID exists in database
Update corresponding field with new value
Define function exit_btn()
Create main Tkinter window with buttons for various operations
Run main event loop using root.mainloop()
STOP

C. Source Code

```
from tkinter import *
from tkinter import messagebox
import pymysql as mq
import re

mysql = mq.connect(host="localhost", user="root", password="", database="emp")
mycursor = mysql.cursor()


def emp_add():
    def back():
        add.destroy()
        root.deiconify()

    def submit():
     # Retrieve data from entry fields
        emp_id = id_entry.get()
        emp_name = name_entry.get()
        emp_age = age_entry.get()
        emp_phone = phone_entry.get()
        emp_email = email_entry.get()
        emp_gender = gender_var.get()
        emp_designation = designation_entry.get()
        emp_salary = salary_entry.get()

        # Data Validation
        errors = []

        if not emp_id.isdigit():
            errors.append("Invalid ID")
        if not emp_name.replace(" ", "").isalpha():
            errors.append("Invalid Name")
        if not emp_age.isdigit() or int(emp_age) <= 18:
            errors.append("Age should be a number greater than 18")
        if not re.match(r"[^@]+@[^@]+\.[^@]+", emp_email):
            errors.append("Invalid Email")
        if not emp_phone.isdigit() or len(emp_phone) != 10:
            errors.append("Invalid Phone Number (should be 10 digits)")
        if not emp_designation.replace(" ", "").isalpha():
            errors.append("Invalid Designation")
        if not emp_salary.isdigit():
            errors.append("Invalid Salary")

        if errors:
            messagebox.showerror("Error", "\n".join(errors))
        else:
            # Check if employee ID already exists in the database
            mycursor.execute(
                "SELECT emp_id FROM emp_details WHERE emp_id = %s", (emp_id,))
            existing_emp = mycursor.fetchone()
        if existing_emp:
            messagebox.showerror("Error", "Employee ID already exists")
        else:
            # Insert data into the database
            sql = "INSERT INTO emp_details (emp_id, name, phone, email, age, gender,
desi, salary) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
            val = (emp_id, emp_name, emp_phone, emp_email, emp_age,
                   emp_gender, emp_designation, emp_salary)
            mycursor.execute(sql, val)
            mysql.commit()
            # Clear entry fields after submission
            clear_entries()
            messagebox.showinfo(
                "Success", "Employee Record Added Successfully")
            add.destroy()
            root.deiconify()

    def clear_entries():
        id_entry.delete(0, END)
        name_entry.delete(0, END)
        age_entry.delete(0, END)
        phone_entry.delete(0, END)
        email_entry.delete(0, END)
        designation_entry.delete(0, END)
        salary_entry.delete(0, END)
        root.withdraw()

    root.withdraw()
    add = Tk()
    add.title("Employee Management System - Add Employee Details")
    add.geometry("800x800")  # Set window size here
    add.minsize(600, 400)
    add.maxsize(800, 800)
    heading_label = Label(
        add, text="Enter Employee Details", font=("Times New Roman", 30, "bold"),
padx=30, pady=30)
    heading_label.grid(row=0, column=2, columnspan=2)

    id_label = Label(add, text="ID", font=("Times New Roman", 20))
    id_label.grid(row=1, column=1, padx=10, pady=10)
    id_entry = Entry(add, font=("Times New Roman", 20))
    id_entry.grid(row=1, column=2, padx=10, pady=10)

    name_label = Label(add, text="Name", font=("Times New Roman", 20))
    name_label.grid(row=2, column=1, padx=10, pady=10)
    name_entry = Entry(add, font=("Times New Roman", 20))
    name_entry.grid(row=2, column=2, padx=10, pady=10)

    age_label = Label(add, text="Age", font=("Times New Roman", 20))
    age_label.grid(row=3, column=1, padx=10, pady=10)
    age_entry = Entry(add, font=("Times New Roman", 20))
    age_entry.grid(row=3, column=2, padx=10, pady=10)

    phone_label = Label(add, text="Phone", font=("Times New Roman", 20))
    phone_label.grid(row=4, column=1, padx=10, pady=10)
    phone_entry = Entry(add, font=("Times New Roman", 20))
    phone_entry.grid(row=4, column=2, padx=10, pady=10)

    email_label = Label(add, text="Email", font=("Times New Roman", 20))
    email_label.grid(row=5, column=1, padx=10, pady=10)
    email_entry = Entry(add, font=("Times New Roman", 20))
    email_entry.grid(row=5, column=2, padx=10, pady=10)

    gender_label = Label(add, text="Gender", font=("Times New Roman", 20))
    gender_label.grid(row=6, column=1, padx=10, pady=10)
    gender_var = StringVar()
    gender_var.set("Male")
    male_radio = Radiobutton(
        add, text="Male", variable=gender_var, value="Male", font=("Times New
Roman", 20))
    female_radio = Radiobutton(
        add, text="Female", variable=gender_var, value="Female", font=("Times New
Roman", 20))
    male_radio.grid(row=6, column=2, sticky=W)
    female_radio.grid(row=6, column=3, sticky=W)

    designation_label = Label(
        add, text="Designation", font=("Times New Roman", 20))
    designation_label.grid(row=7, column=1, padx=10, pady=10)
    designation_entry = Entry(add, font=("Times New Roman", 20))
    designation_entry.grid(row=7, column=2, padx=10, pady=10)

    salary_label = Label(add, text="Salary", font=("Times New Roman", 20))
    salary_label.grid(row=8, column=1, padx=10, pady=10)
    salary_entry = Entry(add, font=("Times New Roman", 20))
    salary_entry.grid(row=8, column=2, padx=10, pady=10)

    back_btn = Button(add, text="Back", font=("Times New Roman", 20),
                      padx=5, pady=5, command=back)
    back_btn.grid(row=9, column=1, padx=10, pady=10)

    submit_btn = Button(add, text="Submit", font=("Times New Roman", 20),
                        padx=10, pady=10, command=submit)
    submit_btn.grid(row=9, column=2, padx=10, pady=10)

    add.mainloop()


def emp_view():
    def back():
        view.destroy()
        root.deiconify()
    root.withdraw()
    view = Tk()
    view.title("Employee Management System - View Employee Details")
    view.geometry("800x800")

    # Retrieve data from the database
    mycursor.execute("SELECT * FROM emp_details")
    rows = mycursor.fetchall()

    # Create headings
    headings = ["ID", "Name", "Phone", "Email",
                "Age", "Gender", "Designation", "Salary"]
    for col, heading in enumerate(headings):
        label = Label(view, text=heading, font=("Times New Roman", 20, "bold"))
        label.grid(row=0, column=col, padx=5, pady=5)

    # Insert data into table
    for row_idx, row_data in enumerate(rows, start=1):
        for col_idx, cell_data in enumerate(row_data):
            label = Label(view, text=cell_data, font=("Times New Roman", 15))
            label.grid(row=row_idx, column=col_idx, padx=5, pady=5)

    back_btn = Button(view, text="Back", font=("Times New Roman", 15),
                      padx=5, pady=5, command=back)
    back_btn.grid(row=9, column=1, padx=10, pady=10)

    view.mainloop()


def emp_delete():
    def back():
        delete.destroy()
        root.deiconify()

    def clear_entries():
        id_entry.delete(0, END)

    def submit():
        emp_id = id_entry.get()
        mycursor.execute(
            "SELECT emp_id FROM emp_details WHERE emp_id = %s", (emp_id,))
        existing_emp = mycursor.fetchone()
        if existing_emp:
            sql = "DELETE FROM emp_details WHERE emp_id=%s"
            val = (emp_id,)
            mycursor.execute(sql, val)
            mysql.commit()
            clear_entries()
            messagebox.showinfo(
                "Success", "Employee Record Deleted Successfully")
            delete.destroy()
            root.deiconify()
        else:
            messagebox.showinfo(
                "Error", "Employee Record does not exists")
            clear_entries()

    root.withdraw()
    delete = Tk()
    delete.title("Employee Management System - Delete Employee Details")
    delete.geometry("800x800")  # Set window size here
    delete.minsize(600, 400)
    delete.maxsize(800, 800)
    heading_label = Label(
        delete, text="Delete Employee Details", font=("Times New Roman", 30, "bold"),
padx=30, pady=30)
    heading_label.grid(row=0, column=1, columnspan=2)

    id_label = Label(delete, text="Enter Employee ID to delete:",
                     font=("Times New Roman", 20))
    id_label.grid(row=4, column=1, padx=10, pady=10)
    id_entry = Entry(delete, font=("Times New Roman", 20))
    id_entry.grid(row=4, column=2, padx=10, pady=10)
    back_btn = Button(delete, text="Back", font=("Times New Roman", 20),
                      padx=5, pady=5, command=back)
    back_btn.grid(row=6, column=1, padx=10, pady=10)

    submit_btn = Button(delete, text="Submit", font=("Times New Roman", 20),
                        padx=10, pady=10, command=submit)
    submit_btn.grid(row=6, column=2, padx=10, pady=10)

    delete.mainloop()


def emp_update():
    def back():
        update.destroy()
        root.deiconify()

    def clear_entries():
        id_entry.delete(0, END)
        new_value_entry.delete(0, END)

    def submit():
        emp_id = id_entry.get()
        mycursor.execute(
            "SELECT emp_id FROM emp_details WHERE emp_id = %s", (emp_id,))
```

```
existing_emp = mycursor.fetchone()
field_to_update = update_var.get()
new_value = new_value_entry.get()
if existing_emp:
    if field_to_update == "Select Field":
        messagebox.showerror(
            "Error", "Please select a field to update")

    if field_to_update == "Name":
        if not new_value.isalpha():
            messagebox.showerror("Name Error", "Invalid Name")

        else:
            sql = "UPDATE emp_details SET name = %s WHERE emp_id = %s"
    elif field_to_update == "Phone":
        if not new_value.isdigit() or len(new_value) != 10:
            messagebox.showerror(
                "Phone Error", "Invalid Phone Number (should be 10 digits)")
        else:
            print("Updating phone number...")
            sql = "UPDATE emp_details SET phone = %s WHERE emp_id = %s"

    elif field_to_update == "Email":
        if not re.match(r"[^@]+@[^@]+\.[^@]+", new_value):
            messagebox.showerror("Email Error", "Invalid Email")
        else:
            sql = "UPDATE emp_details SET email = %s WHERE emp_id = %s"
    elif field_to_update == "Age":
        if not new_value.isdigit() or int(new_value) <= 18:
            messagebox.showerror("Age Error", "Invalid Age")
        else:
            sql = "UPDATE emp_details SET age = %s WHERE emp_id = %s"
    elif field_to_update == "Designation":
        if not new_value.isalpha():
            messagebox.showerror(
                "Designation Error", "Invalid Designation")
        else:
            sql = "UPDATE emp_details SET desi = %s WHERE emp_id = %s"
    elif field_to_update == "Salary":
        if not new_value.isdigit():
            messagebox.showerror("Salary Error", "Invalid Salary")
        else:
            sql = "UPDATE emp_details SET salary = %s WHERE emp_id = %s"

    try:
        mycursor.execute(sql, (new_value, emp_id))
        mysql.commit()
        messagebox.showinfo(
            "Success", "Employee Record Updated Successfully")
        update.destroy()
```

```
            root.deiconify()
        except mq.Error as e:
            messagebox.showerror("Error", f"Error updating record: {e}")
    else:
        messagebox.showinfo(
            "Error", "Employee Record does not exist")
        clear_entries()

root.withdraw()
update = Tk()
update.title("Employee Management System - Update Employee Details")
update.geometry("800x800")
update.minsize(600, 400)
update.maxsize(800, 800)

heading_label = Label(update, text="Update Employee Details", font=(
    "Times New Roman", 30, "bold"), padx=30, pady=30)
heading_label.grid(row=0, column=1, columnspan=2)

id_label = Label(update, text="Enter Employee ID:",
                 font=("Times New Roman", 20))
id_label.grid(row=1, column=1, padx=10, pady=10)
id_entry = Entry(update, font=("Times New Roman", 20))
id_entry.grid(row=1, column=2, padx=10, pady=10)

update_options = ["Select Field", "Name", "Phone",
                  "Email", "Age", "Designation", "Salary"]
update_var = StringVar()
update_var.set(update_options[0])  # Default value
update_label = Label(
    update, text="Select Field to Update:", font=("Times New Roman", 20))
update_label.grid(row=2, column=1, padx=10, pady=10)
update_dropdown = OptionMenu(update, update_var, *update_options)
update_dropdown.config(font=("Times New Roman", 20))
update_dropdown.grid(row=2, column=2, padx=10, pady=10)

new_value_label = Label(update, text="New Value:",
                        font=("Times New Roman", 20))
new_value_label.grid(row=3, column=1, padx=10, pady=10)
new_value_entry = Entry(update, font=("Times New Roman", 20))
new_value_entry.grid(row=3, column=2, padx=10, pady=10)

submit_btn = Button(update, text="Submit", font=(
    "Times New Roman", 20), padx=10, pady=10, command=submit)
submit_btn.grid(row=4, column=2, padx=10, pady=10)

back_btn = Button(update, text="Back", font=(
    "Times New Roman", 20), padx=5, pady=5, command=back)
back_btn.grid(row=4, column=1, padx=10, pady=10)
```

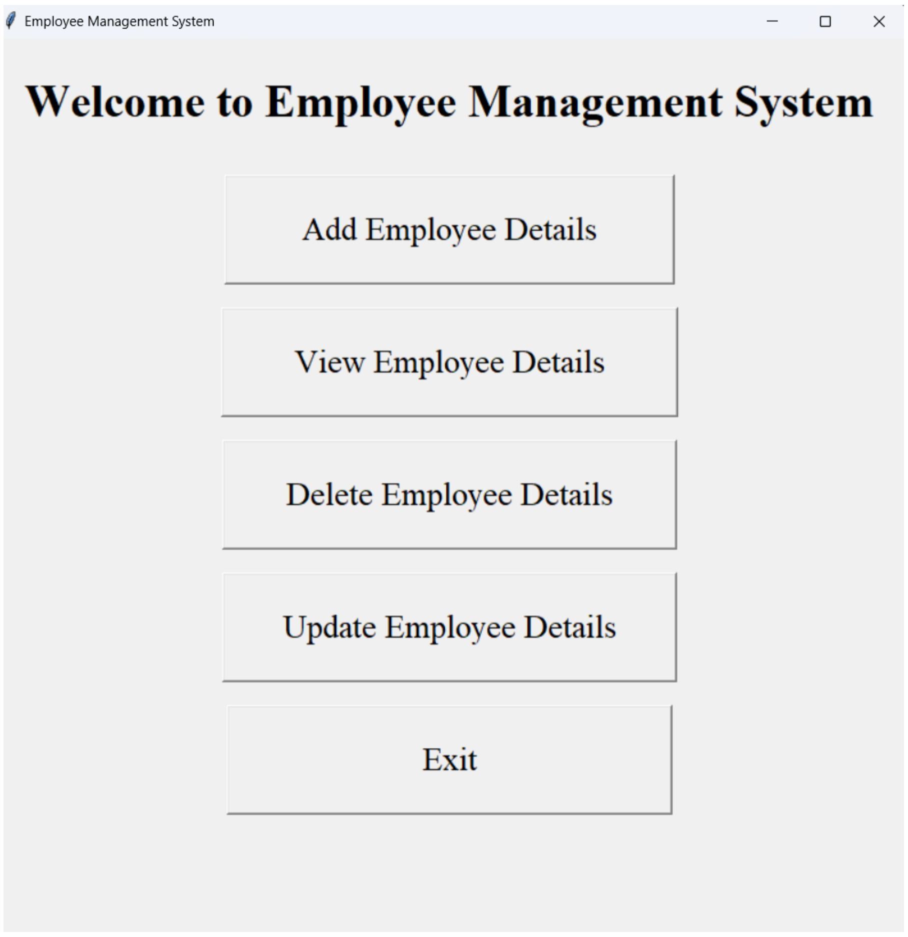
```
    update.mainloop()


def exit_btn():
    root.destroy()


root = Tk()
root.geometry("800x800")
root.minsize(600, 400)
root.maxsize(800, 800)
root.title("Employee Management System")
heading_label = Label(
    text="Welcome to Employee Management System", font=("Times New Roman", 30,
"bold"), padx=30, pady=30)
heading_label.pack(fill=X)

button_padding_y = 10

add_emp = Button(text="Add Employee Details", font=(
    "Times New Roman", 22), padx=57, pady=20, command=emp_add)
view_emp = Button(text="View Employee Details",
                  font=("Times New Roman", 22), padx=53, pady=20, command=emp_view)

delete_emp = Button(text="Delete Employee Details",
                    font=("Times New Roman", 22), padx=45, pady=20,
command=emp_delete)

update_emp = Button(text="Update Employee Details",
                    font=("Times New Roman", 22), padx=42, pady=20,
command=emp_update)

exit_btn = Button(text="Exit",
                  font=("Times New Roman", 22), padx=160, pady=20, command=exit_btn)


add_emp.pack(pady=10)
view_emp.pack(pady=10)
delete_emp.pack(pady=10)
update_emp.pack(pady=10)
exit_btn.pack(pady=10)

root.mainloop()
```

4. Actual Resources Required:

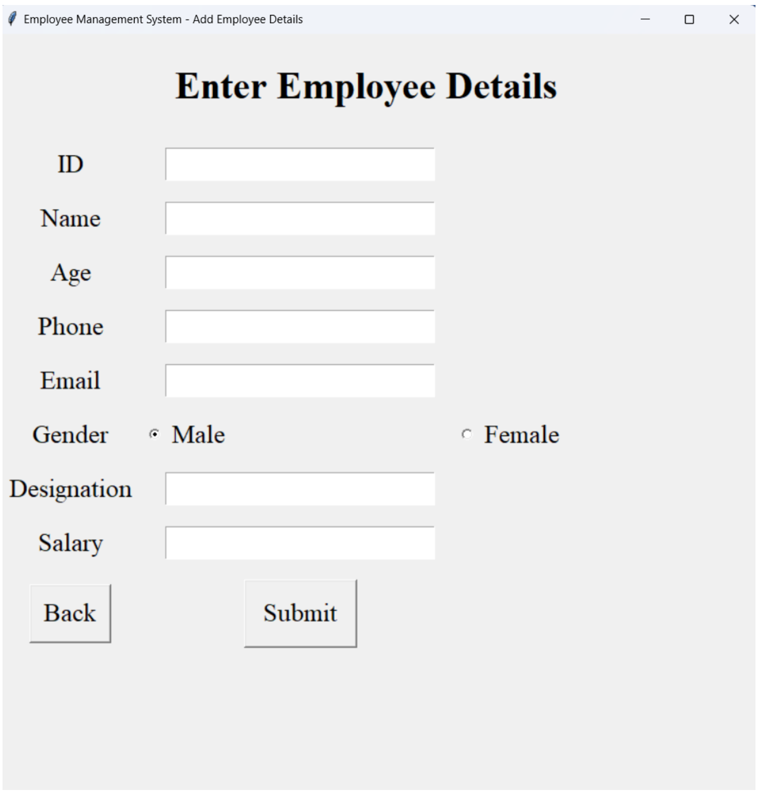
Sr. no	Name of resources	Specification	Quantity	Remark
1	Computer system	**Processor:** Intel(R) Pentium(R) Dual CPU E2140@1.60GHz1.60GHz **RAM:** 512 MB	1	-
2	Operating System	**OS:** Windows 7(32bit)	1	-

5. Output of Micro-Project:

a) **Home Page :-** Contains Welcome Message and Buttons to navigate to their specific function



b) Add Employee Details:- Adding new Employee details in the database

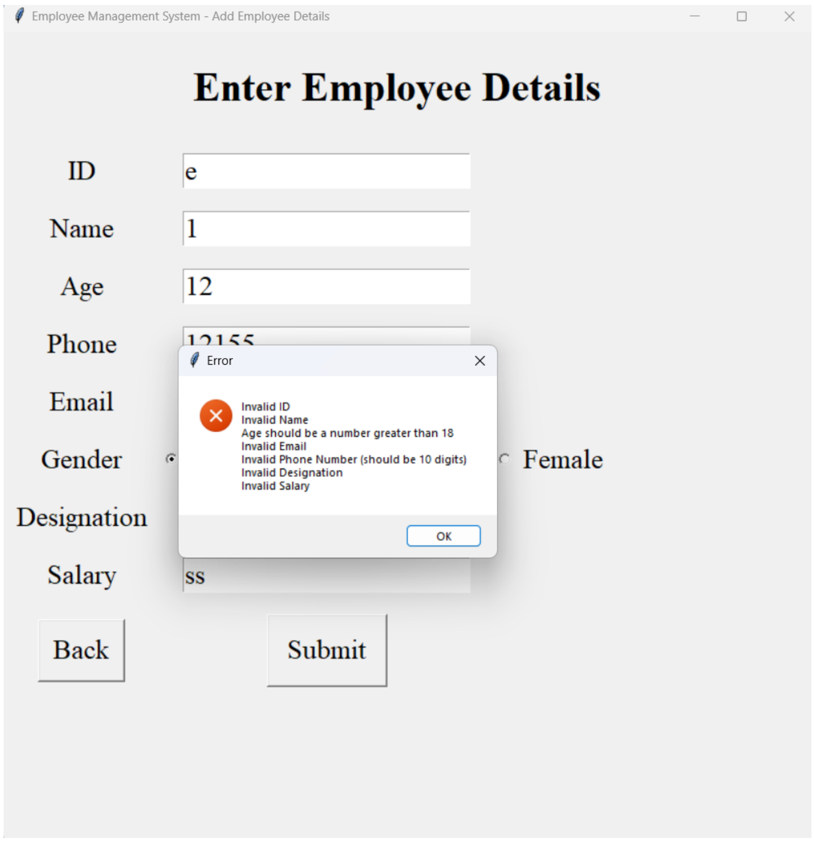


Employee Management System - Add Employee Details

Enter Employee Details

ID		
Name		
Age		
Phone		
Email		
Gender	[✓ Male]	Female
Designation		
Salary		

c) Add Employee Details – Validation:- Validating all the entry fields before inserting data into the database, if data is incorrect display error message for specific field



Employee Management System - Add Employee Details

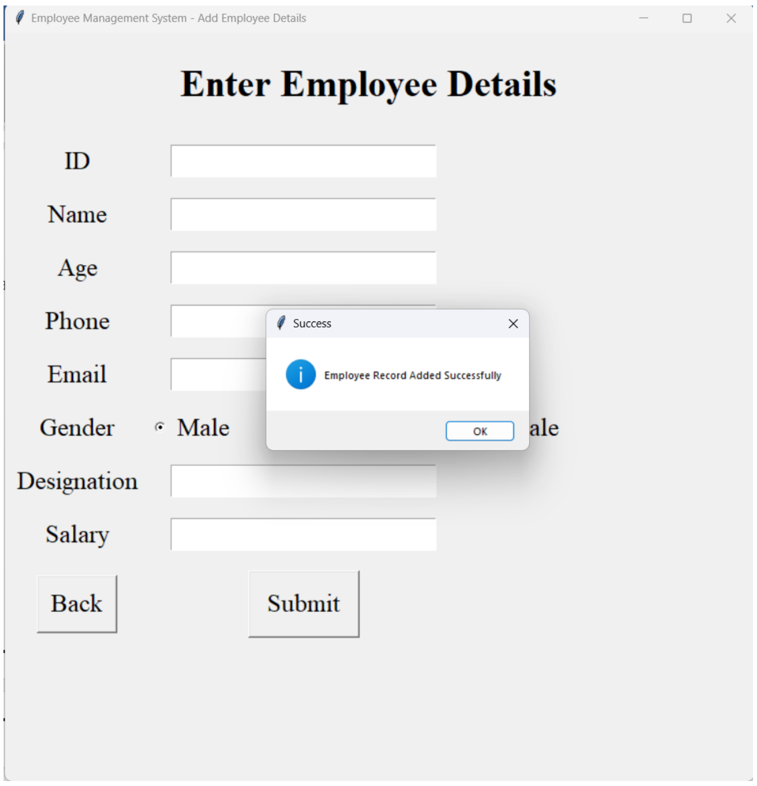
Enter Employee Details

ID	e	
Name	1	
Age	12	
Phone	12155	
Email		
Gender	[✓ Male]	Female
Designation		
Salary	ss	

Error

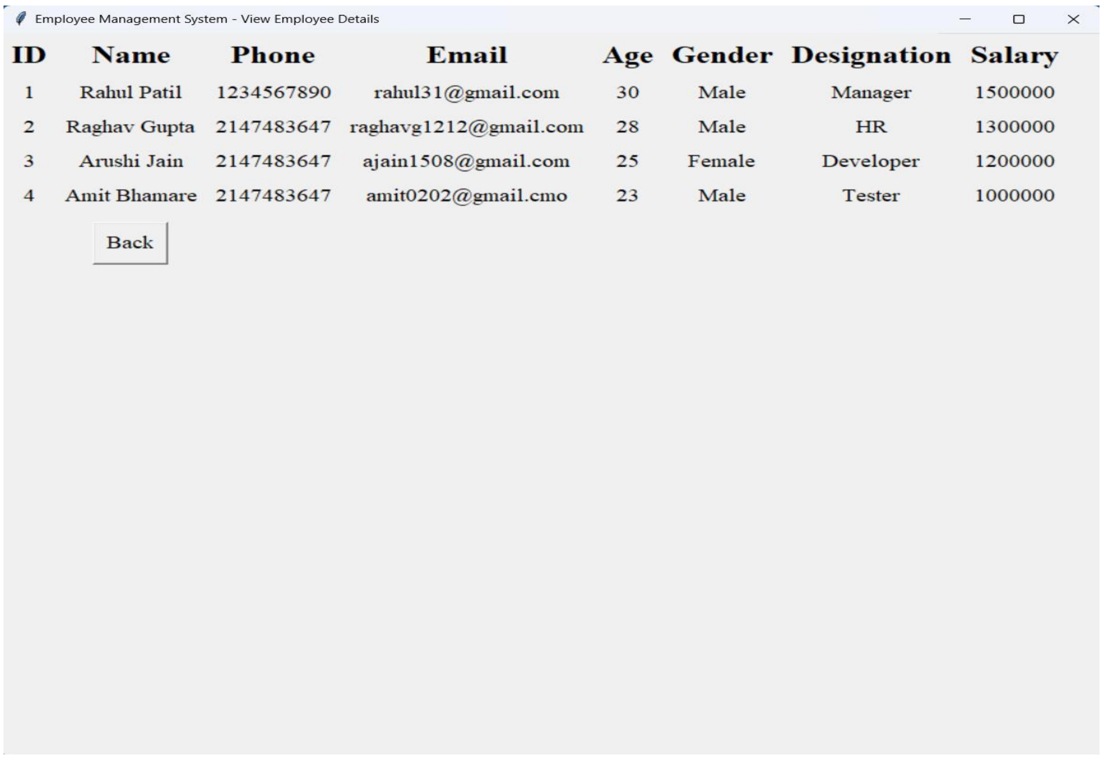
```
Invalid ID
Invalid Name
Age should be a number greater than 18
Invalid Email
Invalid Phone Number (should be 10 digits)
Invalid Designation
Invalid Salary
```

d) **Add Employee Details – Success Message:-** After validating and verifying that all data is correct insert it into the database. If data is successfully inserted into the database then show success message



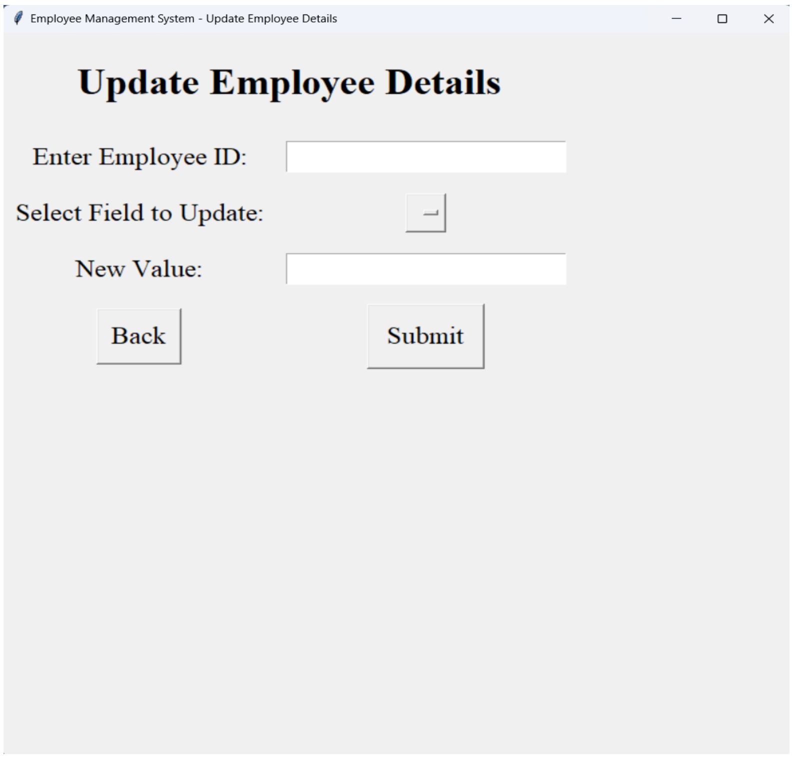
Employee Record Added Successfully

e) **View Employee Details:-** Displaying all the Employers record from the database in table format



ID	Name	Phone	Email	Age	Gender	Designation	Salary
1	Rahul Patil	1234567890	rahul31@gmail.com	30	Male	Manager	1500000
2	Raghav Gupta	2147483647	raghavg1212@gmail.com	28	Male	HR	1300000
3	Arushi Jain	2147483647	ajain1508@gmail.com	25	Female	Developer	1200000
4	Amit Bhamare	2147483647	amit0202@gmail.cmo	23	Male	Tester	1000000

f) **Update Employee Details:-** Updating details of Employee if employee with entered employee id exists in database


Employee Management System - Update Employee Details

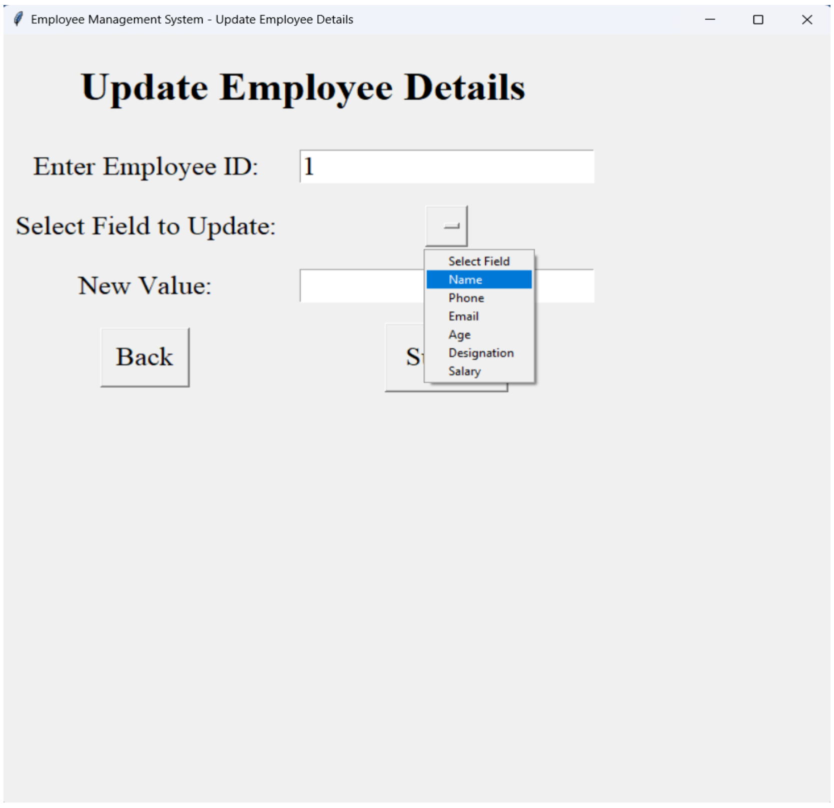
Update Employee Details

Enter Employee ID:

Select Field to Update:

New Value:

g) **Update Employee Details - Select Field:-** Selecting which field to update of employee with entered employee id


Employee Management System - Update Employee Details

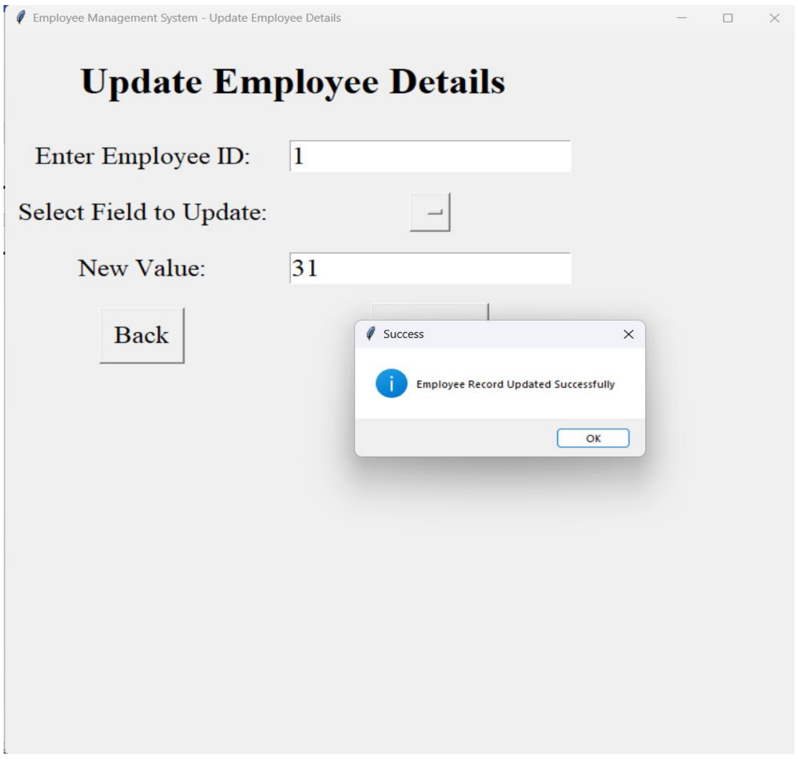
Update Employee Details

Enter Employee ID: 1

Select Field to Update:

New Value:

h) Update Employee Details – Record Updated Successfully:- Validating the entered field, if correct the update the record of employee with that employee id, else show error message


Employee Management System - Update Employee Details

Update Employee Details

Enter Employee ID: 1

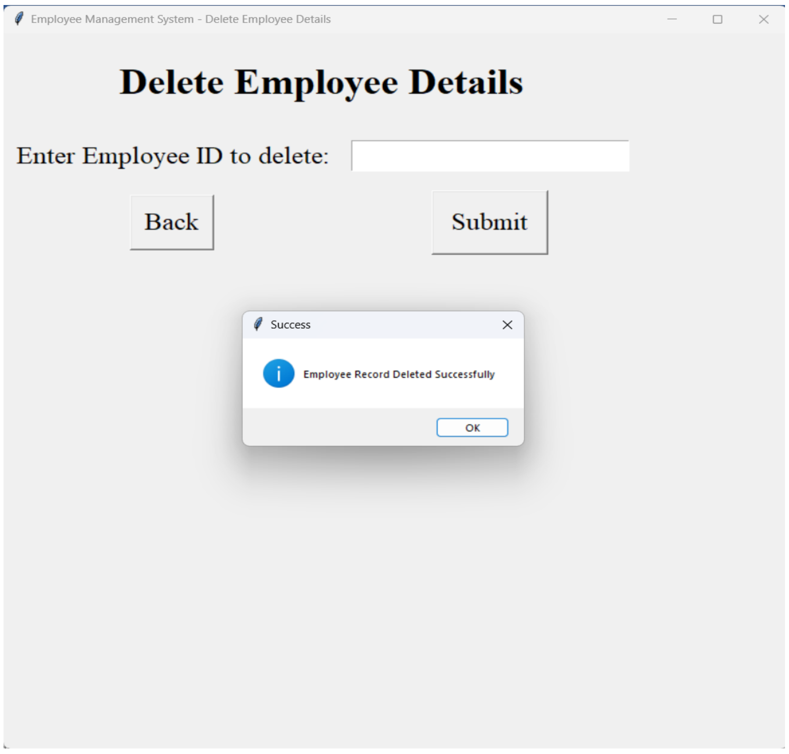
Select Field to Update:

New Value: 31

Success

Employee Record Updated Successfully

i) Delete Employee Details:- Delete record of employee if employee with entered employee id exists in database, else show error message


Employee Management System - Delete Employee Details

Delete Employee Details

Enter Employee ID to delete:

Success

Employee Record Deleted Successfully

6. Skill Developed:

1) We learned how to define a class and how to create object of class.
2) We learned how to implement the array of object.
3) We learned the concept of menu driven program.
4) We learned how to import package in the program.

7. Applications of this Micro-Project:

1) The employee record management system is use in the offices for managing the information of employee.

Evaluation Sheet for the Micro Project

Academic Year: 2023-2024	**Name of Faculty:** Ms. Surekha H. Patil
Sem: Sixth	**Program Name and Code:** CO(6I)
Course Code: 22616	**Course Name:** Programming with Python

Title of the Project: Employee Management System

COs addressed by the Micro Project:

a) Develop Python program to demonstrate use of Operators

b) Perform operations on data structures in Python.

c) Develop functions for given problem

- **Major Learning Outcomes achieved by students by doing the Project:**

  (a) Practical Outcomes
  Basic Knowledge: Apply knowledge of basic mathematics, science and basic engineering to solve the problems related to application of computers and communication services in storing, manipulating and transmitting data, often in the context of a business or other enterprise.

  (b) Unit Outcomes (in Cognitive domain)

  Unit 1(A), Unit 2(A)

  (c) Outcomes in Affective Domain

  Study the working of Employee Management System

- **Comment/Suggestions about team work/leadership/inter-personal Communication (If Any):**

  ____

- **Any Other Comment:**

  ____

- **Marks:**

  Name of Student: Mohit Hemant Badgujar

(A) Marks for Group work:	(B) Marks for Individual work:	(C) Total Marks (A+B) =

(Ms. Surekha H. Patil Lecturer in Programming with Python)

Signature with Name and Designation of the Faculty Member

Evaluation Sheet for the Micro Project

Academic Year: 2023-2024	**Name of Faculty:** Ms. Surekha H. Patil
Sem: Sixth	**Program Name and Code:** CO(6I)
Course Code: 22616	**Course Name:** Programming with Python

Title of the Project: Employee Management System

COs addressed by the Micro Project:

a) Develop Python program to demonstrate use of Operators

b) Perform operations on data structures in Python.

c) Develop functions for given problem

➢ **Major Learning Outcomes achieved by students by doing the Project:**

(a) **Practical Outcomes**
Basic Knowledge: Apply knowledge of basic mathematics, science and basic engineering to solve the problems related to application of computers and communication services in storing, manipulating and transmitting data, often in the context of a business or other enterprise.

(b) **Unit Outcomes (in Cognitive domain)**

Unit 1(A), Unit 2(A)

(c) **Outcomes in Affective Domain**

Study the working of Employee Management System

➢ **Comment/Suggestions about team work/leadership/inter-personal Communication (If Any):**

____

➢ **Any Other Comment:**

____

➢ **Marks:**

Name of Student: Atharva Yogesh Bhatt

(A) Marks for Group work:	(B) Marks for Individual work:	(C) Total Marks (A+B) =

(Ms. Surekha H. Patil Lecturer in Programming with Python)

Signature with Name and Designation of the Faculty Member

Evaluation Sheet for the Micro Project

Academic Year: 2023-2024	**Name of Faculty:** Ms. Surekha H. Patil
Sem: Sixth	**Program Name and Code:** CO(6I)
Course Code: 22616	**Course Name:** Programming with Python

Title of the Project: Employee Management System

COs addressed by the Micro Project:

a) Develop Python program to demonstrate use of Operators

b) Perform operations on data structures in Python.

c) Develop functions for given problem

- **Major Learning Outcomes achieved by students by doing the Project:**

  (a) Practical Outcomes
  Basic Knowledge: Apply knowledge of basic mathematics, science and basic engineering to solve the problems related to application of computers and communication services in storing, manipulating and transmitting data, often in the context of a business or other enterprise.

  (b) Unit Outcomes (in Cognitive domain)

  Unit 1(A), Unit 2(A)

  (c) Outcomes in Affective Domain

  Study the working of Employee Management System

- **Comment/Suggestions about team work/leadership/inter-personal Communication (If Any):**

  ____

- **Any Other Comment:**

  ____

- **Marks:**
  Name of Student: Samarthya Ravindra Deore

(A) Marks for Group work:	(B) Marks for Individual work:	(C) Total Marks (A+B) =

(Ms. Surekha H. Patil Lecturer in Programming with Python)

Signature with Name and Designation of the Faculty Member

Evaluation Sheet for the Micro Project

Academic Year: 2023-2024	**Name of Faculty:** Ms. Surekha H. Patil
Sem: Sixth	**Program Name and Code:** CO(6I)
Course Code: 22616	**Course Name:** Programming with Python

Title of the Project: Employee Management System

COs addressed by the Micro Project:

a) Develop Python program to demonstrate use of Operators

b) Perform operations on data structures in Python.

c) Develop functions for given problem

➢ **Major Learning Outcomes achieved by students by doing the Project:**

(a) **Practical Outcomes**
Basic Knowledge: Apply knowledge of basic mathematics, science and basic engineering to solve the problems related to application of computers and communication services in storing, manipulating and transmitting data, often in the context of a business or other enterprise.

(b) **Unit Outcomes (in Cognitive domain)**

Unit 1(A), Unit 2(A)

(c) **Outcomes in Affective Domain**

Study the working of Employee Management System

➢ **Comment/Suggestions about team work/leadership/inter-personal Communication (If Any):**

____

➢ **Any Other Comment:**

____

➢ **Marks:**
Name of Student: Dhruv Harish Makhija

(A) Marks for Group work:	(B) Marks for Individual work:	(C) Total Marks (A+B) =

(Ms. Surekha H. Patil Lecturer in Programming with Python)

Signature with Name and Designation of the Faculty Member