EMPLOYEE MANAGEMENT SYSTEM

FINAL PROJECT - SAT4650

BY VISHWAJEETH BALAJI

Michigan Tech
1885

APPLICATIONS

- In many organizations, employment management systems are used to support staff in providing daily efforts at their best to meet organizational objectives.

- Employee efforts are managed and guided in the appropriate direction by it. It handles and securely keeps your employee personal information as well as information linked to their employment. It is then simpler to store the data and get access to it when needed.

- Administrative tasks may be managed more quickly and easily with the staff management system. The work that employees do eventually affects the company's bottom line, making them a crucial component of your organization. Additionally, it boosts productivity and lowers expenses by assisting with staff engagement and performance management.

Michigan Tech
1885

PROBLEMS TO BE SOLVED

- Aim of developing this system is to deter all forms of fraud and aid the HR department in an organization's accurate compensation calculations. Conflicts between employees and employers frequently arise as a result of calculation errors, which have a negative impact on relationships and employee performance as a whole, both of which contribute to the success of a business. Therefore, this system seeks to provide easy and comfortable working conditions for employees so that organizations can develop and achieve their objectives.

- Designing a employee management system is to provide a straight forward, reasonably priced, and dependable system to meet the objective of making employees attendance and wage calculations simple and accurate. In order to ensure that employees receive full compensation for their labour, we want to create a system that prohibits employee fraud. This system was created primarily to address the issue of small-scale enterprises and manufacturers still keeping employee data on paper and pen.

Michigan Tech
1885

An employee management system can also help to solve the following other significant issues:

- Employer management
- Keeping thorough performance records for all staff.
- Making sure that everyone is paid accurately and on schedule.
- Tracking staff output and performance to help managers allocate resources wisely.
- Ensuring a positive staff experience and engagement.
- It can take a lot of effort to make sure there are no issues with hiring, terminating, or training new personnel.

Michigan Tech
1885

BASIC IDEA

Idea of designing employment management system (Using SQLite database and GUI) is for getting all the data of the employees at fingertips and to save lot of productive time without crawling through the entire employee database.

The task is to create a Database-driven Employee Management System in Python that will store the information in the SQLite Database. The script will contain the following operations like (Add, Update, Delete and Clear)

Michigan Tech

IMPLEMENTATION PLAN

MODULES : Functions , Database and GUI

PACKAGES: Tkinter , ttk from Tkinter (Themed Tkinter) , messagebox from Tkinter and SQLite3

DATAFLOW AND MAJOR FUNCTIONALITIES:

- **First I have decided to define a Python class called Database that represents a SQLite database connection and provides methods to interact with the "employees" table in the database.**

Michigan Tech
1885

- The __init__ method is the constructor of the Database class, which initializes the database connection and creates the "employees" table if it does not exist. The table has eight columns: id (integer primary key), name (text), age (text), doj (text), email (text), gender (text), contact (text), and address.

- The insert method allows inserting a new record into the "employees" table with the provided values for each column. It uses a parameterized SQL query to prevent SQL injection.

- The fetch method retrieves all records from the "employees" table and returns them as a list of rows. Each row is a tuple containingthe values for each column.

- The remove method allows deleting a record from the "employees" table based on the provided id.

- The update method allows updating a record in the "employees" table based on the provided id. It updates the values of all columns for the specified record.

Michigan Tech
1885

- Then the other code uses the tkinter module to create the GUI elements such as labels, entry fields, and buttons. It also uses the ttk module from tkinter for creating a combobox with dropdown values for gender selection. The messagebox module is used for displaying error and success messages.

- The Database class from the db module is used for performing database operations such as inserting, updating, deleting, and fetching employee records from the SQLite database.

- The add_employee(), update_employee(), and delete_employee() functions are called when the respective buttons are clicked, and these functions interact with the Database class to perform the corresponding database operations.

- The getData() function is called when a row is selected in the treeview widget (tv). It fetches the data from the selected row and populates the input fields with the data, allowing users to update or delete the selected employee record.

Michigan Tech
1885

- The add_employee() function is called when the "Add" button (btnAdd) is clicked. It first checks if all the required fields (Name, Age, Date of Joining, Email, Gender, Contact, and Address) are filled. If any of the fields are empty, it displays an error message using messagebox.showerror(). Otherwise, it inserts the employee record into a database using the db.insert() function, displays a success message, clears all the input fields using the clearAll() function, and updates the display using the displayAll() function.

- The update_employee() function is called when the "Update" button (btnEdit) is clicked. It performs similar validation checks as the add_employee() function, but instead of inserting a new record, it updates the selected employee record in the database using the db.update() function. It also displays a success message, clears all the input fields, and updates the display.

- The delete_employee() function is called when the "Delete" button (btnDelete) is clicked. It removes the selected employee record from the database using the db.remove() function. It then clears all the input fields and updates the display.

Michigan Tech

- The clearAll() function is called to clear all the input fields (Name, Age, Date of Joining, Email, Gender, Contact, and Address) when the "Clear" button (btnClear) is clicked.

- The code also defines the layout of the GUI using Frame, Button, and Treeview widgets. The Frame and Button widgets are used to create a row of buttons for adding, updating, deleting, and clearing employee records. The Treeview widget is used to display the employee records in a tabular format with columns for ID, Name, Age, Date of Joining, Email, Gender, Contact, and Address. The style variable is used to configure the appearance of the Treeview widget, such as font size and background color.

- The displayAll() function is not defined in the code snippet, so it is likely defined elsewhere in the larger application. Based on its name, it is likely used to fetch all the employee records from the database and display them in the Treeview widget. The function is called to initially populate the Treeview widget with data when the application starts.

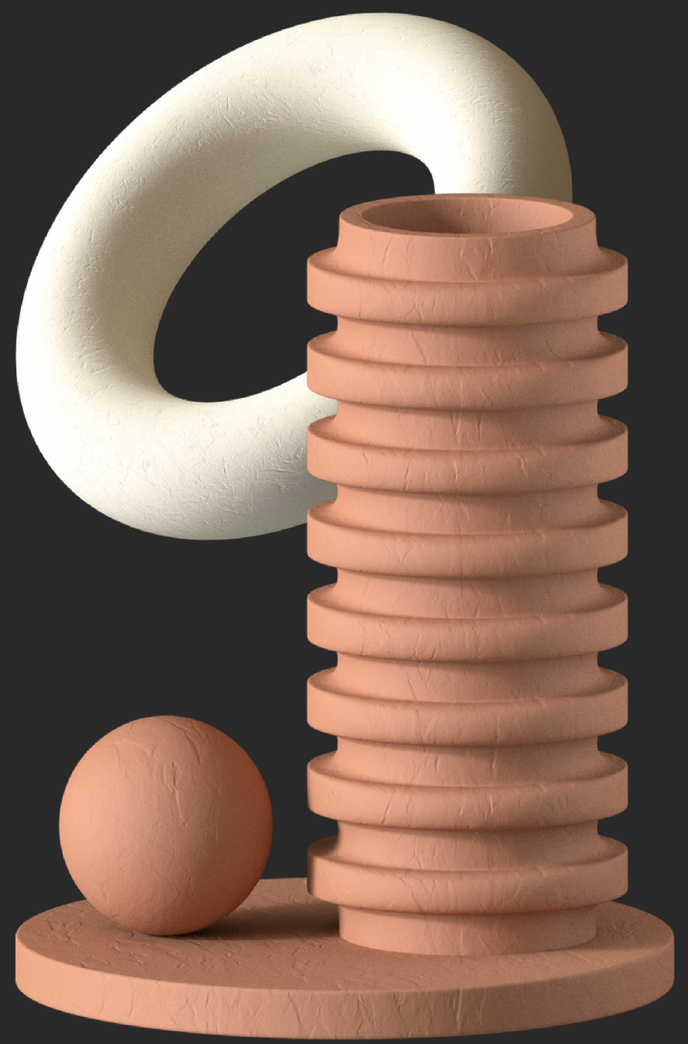
Michigan Tech
1885

INPUT CODE - SQLITE DATABASE

File Edit Format Run Options Window Help

```
import sqlite3


class Database:
    def __init__(self, db):
        self.con = sqlite3.connect(db)
        self.cur = self.con.cursor()
        sql = """
        CREATE TABLE IF NOT EXISTS employees(
            id Integer Primary Key,
            name text,
            age text,
            doj text,
            email text,
            gender text,
            contact text,
            address text
        )
        """
        self.cur.execute(sql)
        self.con.commit()

    # Insert Function
    def insert(self, name, age, doj, email, gender, contact, address):
        self.cur.execute("insert into employees values (NULL,?,?,?,?,?,?,?)",
                         (name, age, doj, email, gender, contact, address))
        self.con.commit()

    # Fetch All Data from DB
    def fetch(self):
        self.cur.execute("SELECT * from employees")
        rows = self.cur.fetchall()
        # print(rows)
        return rows

    # Delete a Record in DB
    def remove(self, id):
        self.cur.execute("delete from employees where id=?", (id,))
        self.con.commit()
```

```
    # Update a Record in DB
    def update(self, id, name, age, doj, email, gender, contact, address):
        self.cur.execute(
            "update employees set name=?, age=?, doj=?, email=?, gender=?, contact=?, address=? where id=
            (name, age, doj, email, gender, contact, address, id))
        self.con.commit()
```



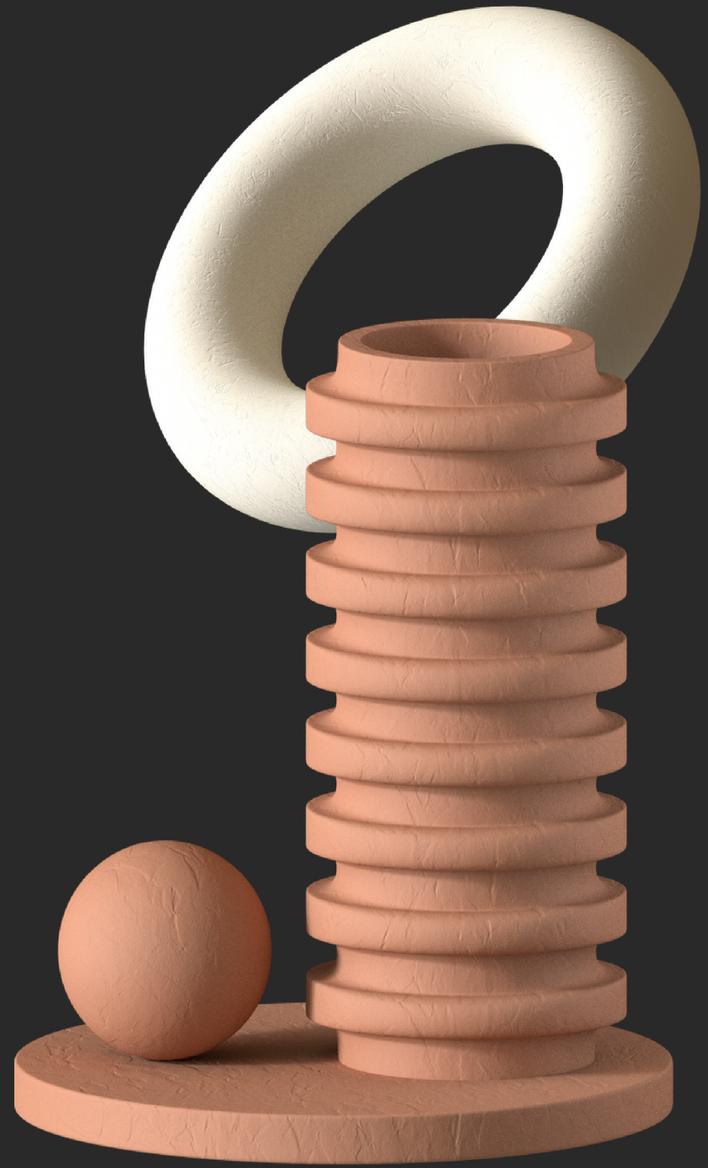
Michigan Tech
1885

INPUT CODE - EMS CREATION

final.py - C:\Users\Admin\Desktop\EMPLOYEE\final.py (3.11.1)

File Edit Format Run Options Window Help

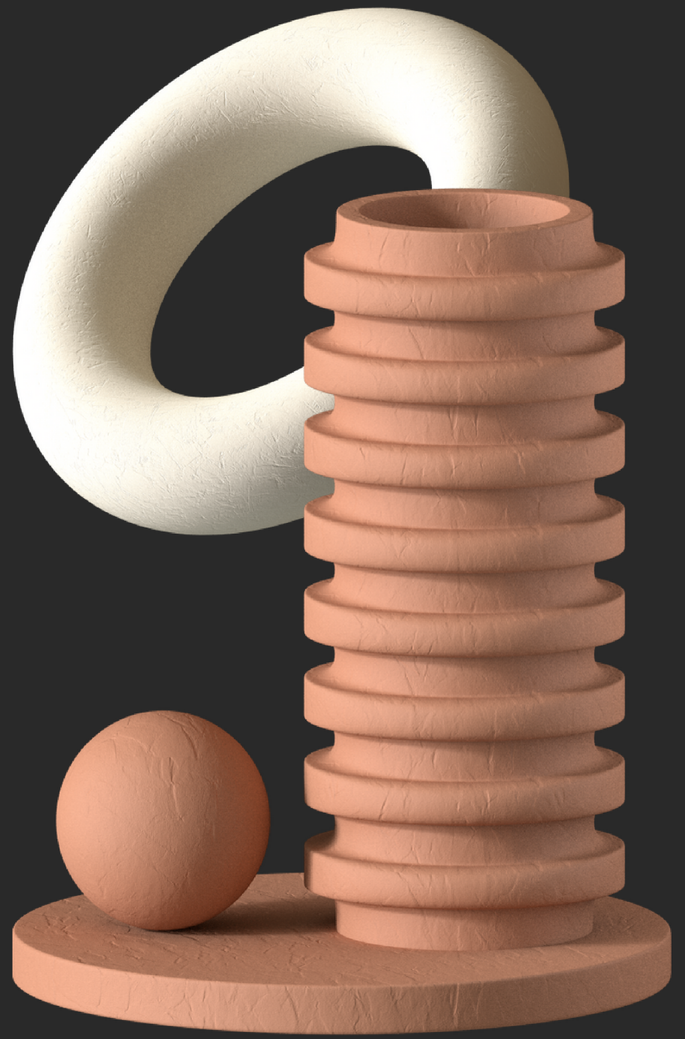
```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from db import Database

db = Database("Employee.db")
root = Tk()
root.title("Employement Management System")
root.geometry("1920x1080+0+0")
root.config(bg="#2c3e50")
root.state("zoomed")

name = StringVar()
age = StringVar()
doj = StringVar()
gender = StringVar()
email = StringVar()
contact = StringVar()

# Entries Frame
entries_frame = Frame(root, bg="#000000")
entries_frame.pack(side=TOP, fill=X)
title = Label(entries_frame, text="EMPLOYEE MANAGEMENT SYSTEM", font=("Times new roman", 18, "bold"), bg="#000000", fg="white")
title.grid(row=0, columnspan=2, padx=10, pady=20, sticky="w")

lblName = Label(entries_frame, text="NAME", font=("Times new roman", 16), bg="#000000", fg="White")
lblName.grid(row=1, column=0, padx=10, pady=10, sticky="w")
txtName = Entry(entries_frame, textvariable=name, font=("Calibri", 16), width=30)
txtName.grid(row=1, column=1, padx=10, pady=10, sticky="w")

lblAge = Label(entries_frame, text="AGE", font=("Times new roman", 16), bg="#000000", fg="white")
lblAge.grid(row=1, column=2, padx=10, pady=10, sticky="w")
txtAge = Entry(entries_frame, textvariable=age, font=("Calibri", 16), width=30)
txtAge.grid(row=1, column=3, padx=10, pady=10, sticky="w")

lbldoj = Label(entries_frame, text="DATE OF JOINING", font=("Times new roman", 16), bg="#000000", fg="white")
lbldoj.grid(row=2, column=0, padx=10, pady=10, sticky="w")
txtDoj = Entry(entries_frame, textvariable=doj, font=("Calibri", 16), width=30)
txtDoj.grid(row=2, column=1, padx=10, pady=10, sticky="w")

lblEmail = Label(entries_frame, text="EMAIL ID", font=("Times new roman", 16), bg="#000000", fg="white")
```



Michigan Tech
1885

INPUT CODE - EMS CREATION

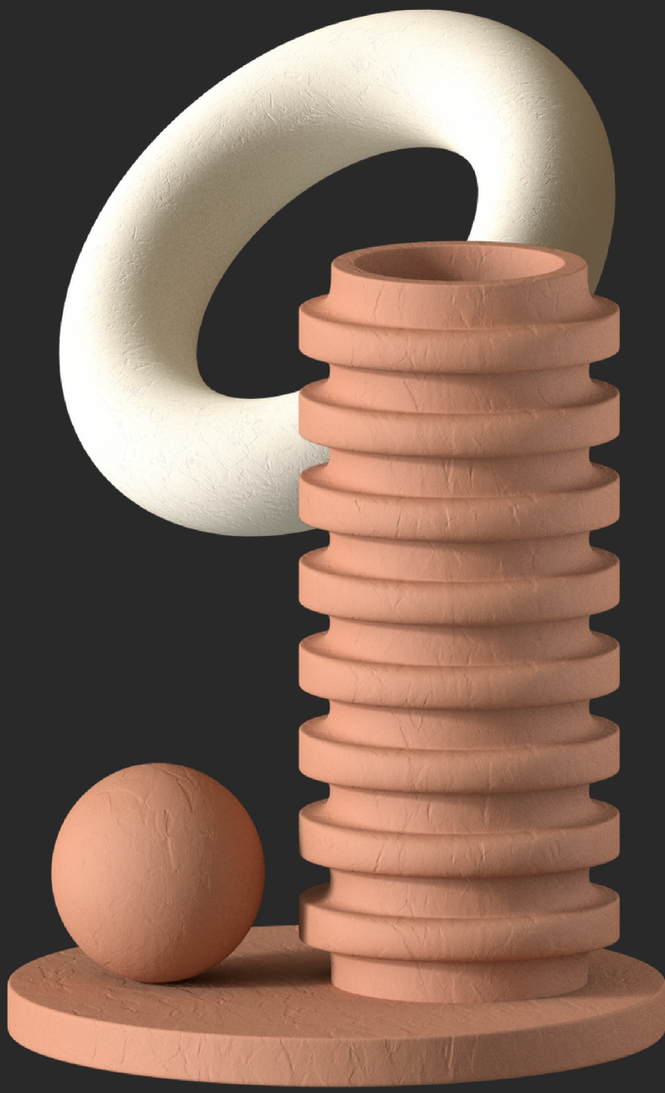
```
lblEmail.grid(row=2, column=2, padx=10, pady=10, sticky="w")
txtEmail = Entry(entries_frame, textvariable=email, font=("Calibri", 16), width=30)
txtEmail.grid(row=2, column=3, padx=10, pady=10, sticky="w")

lblGender = Label(entries_frame, text="GENDER", font=("Times new roman", 16), bg="#000000", fg="white")
lblGender.grid(row=3, column=0, padx=10, pady=10, sticky="w")
comboGender = ttk.Combobox(entries_frame, font=("Times new roman", 16), width=28, textvariable=gender, state="readonly")
comboGender['values'] = ("Male", "Female")
comboGender.grid(row=3, column=1, padx=10, sticky="w")

lblContact = Label(entries_frame, text="CONTACT NUMBER", font=("Times new roman", 16), bg="#000000", fg="white")
lblContact.grid(row=3, column=2, padx=10, pady=10, sticky="w")
txtContact = Entry(entries_frame, textvariable=contact, font=("Times new roman", 16), width=30)
txtContact.grid(row=3, column=3, padx=10, sticky="w")

lblAddress = Label(entries_frame, text="ADDRESS", font=("Times new roman", 16), bg="#000000", fg="white")
lblAddress.grid(row=4, column=0, padx=10, pady=28, sticky="w")

txtAddress = Text(entries_frame, width=85, height=5, font=("Times new roman", 16))
txtAddress.grid(row=5, column=0, columnspan=4, padx=10, sticky="w")

def getData(event):
    selected_row = tv.focus()
    data = tv.item(selected_row)
    global row
    row = data["values"]
    #print(row)
    name.set(row[1])
    age.set(row[2])
    doj.set(row[3])
    email.set(row[4])
    gender.set(row[5])
    contact.set(row[6])
    txtAddress.delete(1.0, END)
    txtAddress.insert(END, row[7])

def displayAll():
    tv.delete(*tv.get_children())
    for row in db.fetch():
        tv.insert("", END, values=row)
```



Michigan Tech

INPUT CODE - EMS CREATION

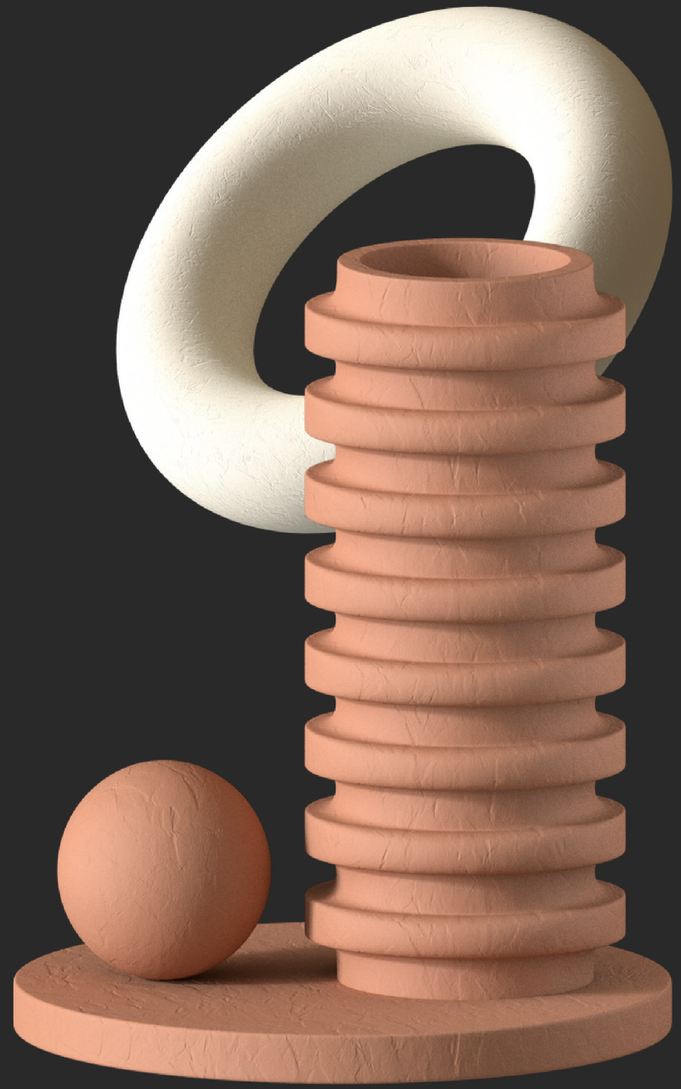
```
def add_employee():
    if txtName.get() == "" or txtAge.get() == "" or txtDoj.get() == "" or txtEmail.get() == "" or comboGender.get() == "" or txtContact.get() == "" or txtAddress.get(
            1.0, END) == "":
        messagebox.showerror("Error in Input", "Please Fill All the Details")
        return
    db.insert(txtName.get(),txtAge.get(), txtDoj.get() , txtEmail.get() ,comboGender.get(), txtContact.get(), txtAddress.get(
            1.0, END))
    messagebox.showinfo("Success", "Record Inserted")
    clearAll()
    dispalyAll()


def update_employee():
    if txtName.get() == "" or txtAge.get() == "" or txtDoj.get() == "" or txtEmail.get() == "" or comboGender.get() == "" or txtContact.get() == "" or txtAddress.get(
            1.0, END) == "":
        messagebox.showerror("Error in Input", "Please Fill All the Details")
        return
    db.update(row[0],txtName.get(), txtAge.get(), txtDoj.get(), txtEmail.get(), comboGender.get(), txtContact.get(),
              txtAddress.get(
                  1.0, END))
    messagebox.showinfo("Success", "Record Update")
    clearAll()
    dispalyAll()


def delete_employee():
    db.remove(row[0])
    clearAll()
    dispalyAll()


def clearAll():
    name.set("")
    age.set("")
    doj.set("")
    gender.set("")
    email.set("")
    contact.set("")
    txtAddress.delete(1.0, END)
```



Michigan Tech
1885

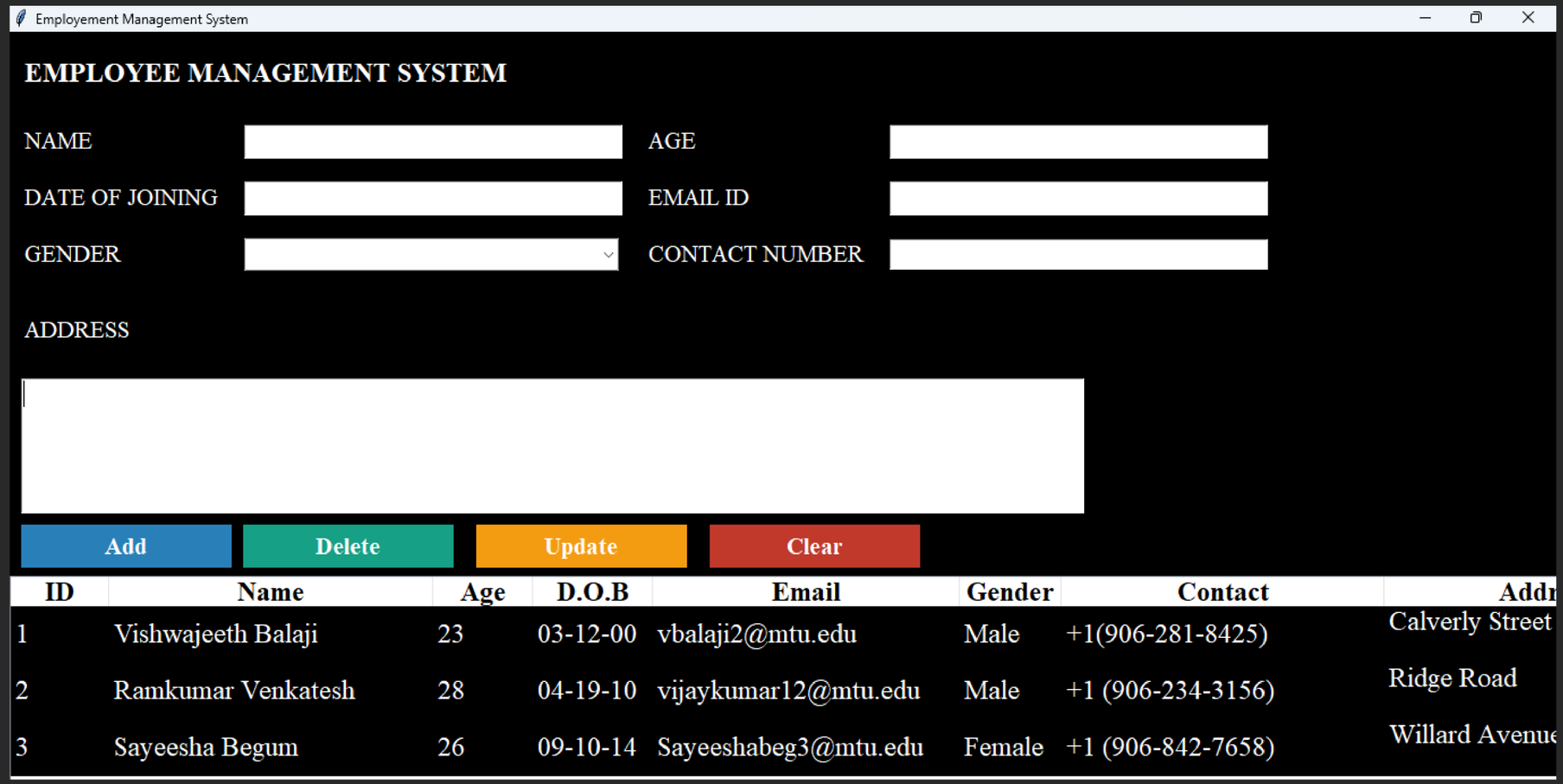
INPUT CODE - EMS CREATION

```
btn_frame = Frame(entries_frame, bg="#000000")
btn_frame.grid(row=6, column=0, columnspan=4, padx=10, pady=10, sticky="w")
btnAdd = Button(btn_frame, command=add_employee, text="Add", width=15, font=("Times new roman", 16, "bold"), fg="white",
                bg="#2980b9", bd=0).grid(row=0, column=0)
btnDelete = Button(btn_frame, command=delete_employee, text="Delete", width=15, font=("Times new roman", 16, "bold"),
                   fg="white", bg="#16a085",
                   bd=0).grid(row=0, column=1, padx=10)
btnEdit = Button(btn_frame, command=update_employee, text="Update", width=15, font=("Times new roman", 16, "bold"),
                    fg="white", bg="#f39c12",
                    bd=0).grid(row=0, column=2, padx=10)
btnClear = Button(btn_frame, command=clearAll, text="Clear", width=15, font=("Times new roman", 16, "bold"), fg="white",
                  bg="#c0392b",
                  bd=0).grid(row=0, column=3, padx=10)

# Table Frame
tree_frame = Frame(root, bg="#000000")
tree_frame.place(x=0, y=480, width=1500, height=600)
style = ttk.Style()
style.configure("mystyle.Treeview", font=('Times new roman', 18),
                rowheight=50, background='black', foreground='white')  # Modify the font of the body
style.configure("mystyle.Treeview.Heading", font=('Times new roman', 18, 'bold'), background='black')  # Modify the font of the headings
tv = ttk.Treeview(tree_frame, columns=(1, 2, 3, 4, 5, 6, 7, 8), style="mystyle.Treeview")
tv.heading("1", text="ID")
tv.column("1", width=2)
tv.heading("2", text="Name")
tv.heading("3", text="Age")
tv.column("3", width=2)
tv.heading("4", text="D.O.B")
tv.column("4", width=5)
tv.heading("5", text="Email")
tv.heading("6", text="Gender")
tv.column("6", width=5)
tv.heading("7", text="Contact")
tv.heading("8", text="Address")
tv['show'] = 'headings'
tv.bind("<ButtonRelease-1>", getData)
tv.pack(fill=X)

displayAll()
root.mainloop()
```

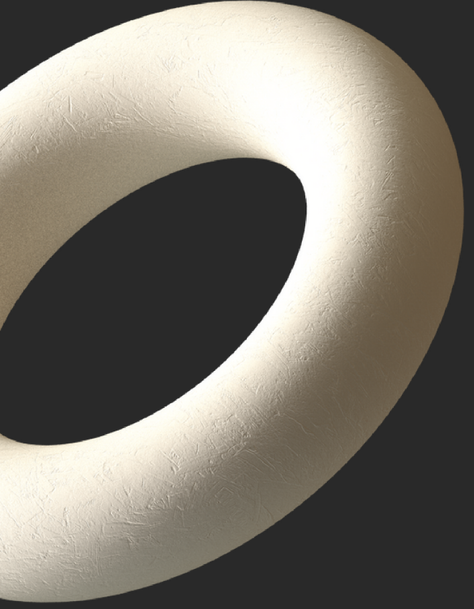


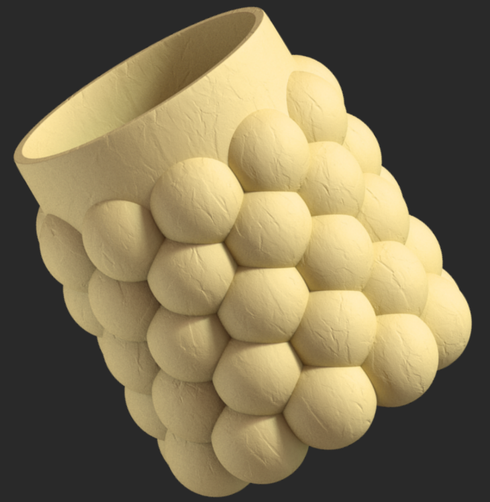
Michigan Tech
1885

OUTPUT - EMS

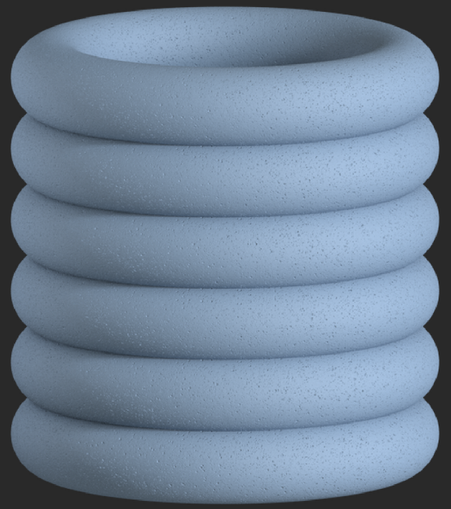


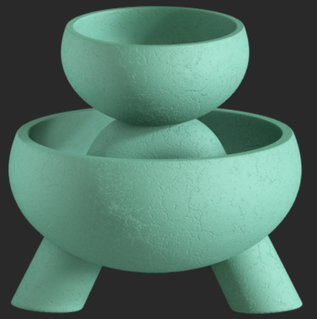
Michigan Tech





THANK YOU





Michigan Tech