Disruptive Technologies-1

[23ECH-102]

Project Report

on

Company Database Management system

**Submitted By**

**Students of 23BCS-112 Group A**

1. **SHUBH RAI-23BCS12916**
2. **HARSHIT SINGH-23BC13863**
3. **HARSHITA-23BCS**
4. **LAKSHYA GHAI-23BCSXXXX**

**Academic Unit-1**

**Department of Computer Science Engineering  
University Institute of Engineering  
Chandigarh University, Gharuan, Mohali, Punjab, India-140413  
2022**

1. **Introduction:** ***AS we all know Management of data,and it’s user interference(UI) is really important for any big organisation.It should be easy accesible,time effiicent and user friendly interface.Managing a data manually or without any dataframe is very time taking and not efficient,But by making a program for data management can do all that in no time***
2. **Brief overview of the project:** *It will be used to manage and collect the data of employees for eg. Qualification,postion,salary,no of leaves,scheduled meeting etc.It can also be used to show the vacancies in company,so that employees can refer their friends or any known.*
3. **Project Description:** It can be made with package as shown in previous slide.By entering the data, employee id it can be logged in by user.It will contain a section suppose ‘vacancies’,it will show total vacancies,cost to company,posting etc.It will contain different sections like-leaves,scheduled meeting,group task,events,income section etc.

**Project Code:** import tkinter as tk  
from tkinter import ttk, messagebox  
  
import timing  
  
# Global variables  
member\_name = ""  
member\_position = ""  
job\_vacancy = ""  
daily\_tasks = {} # Dictionary to store daily tasks for each team  
  
# Function to save data  
def save\_data():  
 global member\_name, member\_position, job\_vacancy  
 member\_name = member\_name\_entry.get()  
 member\_position = member\_position\_var.get()  
 job\_vacancy = job\_vacancy\_entry.get()  
  
 if member\_name and job\_vacancy: # Check if name and job vacancy are not empty  
 data = [member\_name, member\_position, job\_vacancy]  
 tree.insert('', 'end', values=data)  
 daily\_tasks.setdefault(member\_position, {}).setdefault(member\_name, []) # Initialize daily tasks for the new employee  
 with open('company\_database.txt', 'a') as file:  
 file.write(f"Name: {member\_name}, Position: {member\_position}, Job Vacancy: {job\_vacancy}\n")  
 member\_name\_entry.delete(0, tk.END)  
 job\_vacancy\_entry.delete(0, tk.END)  
 messagebox.showinfo("Success", "Data saved successfully!")  
 else:  
 messagebox.showwarning("Incomplete Data", "Please provide both name and job vacancy.")  
  
# Function to delete selected data  
def delete\_selected\_data():  
 selected\_item = tree.selection()  
 if selected\_item:  
 member\_name = tree.item(selected\_item, 'values')[0]  
 member\_position = tree.item(selected\_item, 'values')[1]  
 tree.delete(selected\_item)  
 del daily\_tasks[member\_position][member\_name] # Remove daily tasks for the deleted employee  
 if not daily\_tasks[member\_position]: # If there are no more employees in the team, delete the team  
 del daily\_tasks[member\_position]  
 with open('company\_database.txt', 'w') as file:  
 for item in tree.get\_children():  
 data = tree.item(item, 'values')  
 file.write(f"Name: {data[0]}, Position: {data[1]}, Job Vacancy: {data[2]}\n")  
 messagebox.showinfo("Data Deleted", "Selected data has been deleted!")  
 else:  
 messagebox.showwarning("No Selection", "Please select a row to delete.")  
  
# Function to view saved data  
def view\_data():  
 try:  
 with open('company\_database.txt', 'r') as file:  
 data = file.readlines()  
 if data:  
 for item in tree.get\_children():  
 tree.delete(item)  
 for line in data:  
 line\_data = line.strip().split(', ')  
 tree.insert('', 'end', values=line\_data)  
 messagebox.showinfo("Company Database", "Data loaded successfully!")  
 else:  
 messagebox.showinfo("No Data", "No data available.")  
 except FileNotFoundError:  
 messagebox.showinfo("No Data", "No data available.")  
  
# Function to view daily tasks for the selected employee  
def view\_daily\_tasks():  
 selected\_item = tree.selection()  
 if selected\_item:  
 member\_name = tree.item(selected\_item, 'values')[0]  
 member\_position = tree.item(selected\_item, 'values')[1]  
 tasks = daily\_tasks.get(member\_position, {}).get(member\_name, [])  
 tasks\_str = "\n".join(tasks)  
 messagebox.showinfo(f"Daily Tasks for {member\_name} - {member\_position}", tasks\_str)  
 else:  
 messagebox.showwarning("No Selection", "Please select an employee to view daily tasks.")  
  
# Function to add daily tasks for the selected employee  
def add\_daily\_task():  
 selected\_item = tree.selection()  
 if selected\_item:  
 member\_name = tree.item(selected\_item, 'values')[0]  
 member\_position = tree.item(selected\_item, 'values')[1]  
 task = task\_entry.get()  
 daily\_tasks.setdefault(member\_position, {}).setdefault(member\_name, []).append(task)  
 task\_entry.delete(0, tk.END)  
 messagebox.showinfo("Task Added", "Daily task added successfully!")  
  
 else:  
 messagebox.showwarning("No Selection", "Please select an employee to add a daily task.")  
  
# Create a Tkinter window  
window = tk.Tk()  
window.title("Company Database Management")  
  
# Create a frame for input fields  
input\_frame = tk.Frame(window)  
input\_frame.pack(pady=10)  
  
member\_name\_label = tk.Label(input\_frame, text="Name:")  
member\_name\_label.grid(row=0, column=0, padx=5, pady=5)  
member\_name\_entry = tk.Entry(input\_frame)  
member\_name\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
member\_position\_label = tk.Label(input\_frame, text="Position:")  
member\_position\_label.grid(row=1, column=0, padx=5, pady=5)  
positions = ["CEO", "CFO", "Head Developer", "Head Designer", "Manager", "Employee"]  
member\_position\_var = tk.StringVar(window)  
member\_position\_var.set(positions[0])  
position\_dropdown = tk.OptionMenu(input\_frame, member\_position\_var, \*positions)  
position\_dropdown.grid(row=1, column=1, padx=5, pady=5)  
  
member\_qualification\_label = tk.Label(input\_frame, text="qualification:")  
member\_position\_label.grid(row=2, column=0, padx=5, pady=5)  
positions = ["12th", "under-grad", "upper grad", "P.H.D", "Diploma", "matric pass"]  
member\_qualification\_var = tk.StringVar(window)  
member\_qualification\_var.set(positions[0])  
qualification\_dropdown = tk.OptionMenu(input\_frame, member\_qualification\_var, \*positions)  
qualification\_dropdown.grid(row=3, column=1, padx=5, pady=5)  
  
  
job\_vacancy\_label = tk.Label(input\_frame, text="Job Vacancy:")  
job\_vacancy\_label.grid(row=2, column=0, padx=5, pady=5)  
job\_vacancy\_entry = tk.Entry(input\_frame)  
job\_vacancy\_entry.grid(row=2, column=1, padx=5, pady=5)  
  
# Create a frame for buttons  
button\_frame = tk.Frame(window)  
button\_frame.pack(pady=10)  
  
save\_button = tk.Button(button\_frame, text="Save Data", command=save\_data)  
save\_button.grid(row=0, column=0, padx=5, pady=5)  
  
view\_button = tk.Button(button\_frame, text="View Data", command=view\_data)  
view\_button.grid(row=0, column=1, padx=5, pady=5)  
  
delete\_data\_button = tk.Button(button\_frame, text="Delete Selected Data", command=delete\_selected\_data)  
delete\_data\_button.grid(row=0, column=2, padx=5, pady=5)  
  
view\_tasks\_button = tk.Button(button\_frame, text="View Daily Tasks", command=view\_daily\_tasks)  
view\_tasks\_button.grid(row=0, column=3, padx=5, pady=5)  
  
# Create a frame for daily tasks  
daily\_tasks\_frame = tk.Frame(window)  
daily\_tasks\_frame.pack(pady=10)  
  
task\_label = tk.Label(daily\_tasks\_frame, text="Add Daily Task:")  
task\_label.grid(row=0, column=0, padx=5, pady=5)  
task\_entry = tk.Entry(daily\_tasks\_frame)  
task\_entry.grid(row=0, column=1, padx=5, pady=5)  
add\_task\_button = tk.Button(daily\_tasks\_frame, text="Add Task", command=add\_daily\_task)  
add\_task\_button.grid(row=0, column=2, padx=5, pady=5)  
  
# Create a frame for the treeview  
tree\_frame = tk.Frame(window)  
tree\_frame.pack(pady=10)  
  
# Create a treeview for data display  
columns = ("Name", "Position", "Job Vacancy")  
tree = ttk.Treeview(tree\_frame, columns=columns, show='headings', selectmode="browse")  
  
# Configure column headings  
for col in columns:  
 tree.heading(col, text=col)  
 tree.column(col, width=150, anchor="center")  
  
tree.pack(expand=True, fill='both')  
  
# Start the Tkinter main loop  
window.mainloop()

1. **Results and Key Findings: we created a database where we can store data of employee and can edit it whenever we want. we learned about the new python package and that is tkinter it was very helpful in making it.**
2. **Conclusion: Hence we created a database management system in a short period of time and as expected we faced a lot of problem while making and we could not implement other things cause of lack of experience and knowledge.**
3. **References:pycharm,github etc.**