HIRE.ME
JOB APPLICATION MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

NISHAL I P	**220701187**
NANDEESHWARAN P	**220701179**

In partial fulfilment for the award of the degree of

BACHELOR Of

ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM CHENNAI-602105

2023- 24

BONAFIDE CERTIFICATE

Certified that this project report "**JOB APPLICATION MANAGEMENT SYSTEM**" is the bonafide work of **"NISHAL I P (220701187) AND NANDEESHWARAN P (220701179)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on ________________

SIGNATURE	**SIGNATURE**
Dr.R.SABITHA	**Mrs.Kalpana D**
Professor and II Year Academic Head	**Assistant Professor (SG),**
Computer Science and Engineering,	**Computer Science and Engineering,**
Rajalakshmi Engineering College	**Rajalakshmi Engineering**
(Autonomous),	**College (Autonomous),**
Thandalam, Chennai - 602 105	**Thandalam, Chennai - 602 105**

ABSTRACT

The proposed project is a GUI-based Database Management System (DBMS) for managing job postings and applications, designed using Python's Tkinter library. This system will cover the essential CRUD (Create, Read, Update, Delete) and search operations, providing an intuitive interface for both job seekers and employers.

Objectives

1. User-Friendly Interface: Develop a graphical user interface that simplifies the interaction with the underlying database.
2. Comprehensive Job Management: Enable employers to create, read, update, and delete job postings easily.
3. Efficient Applicant Tracking: Allow job seekers to apply for jobs, and employers to manage and track these applications.

Features

1. Job Postings Management:
- Create Employers can add new job postings by filling out a form with job details such as title, description, requirements, location, and salary.
- Read: Display a list of all job postings with detailed views.
- Update: Employers can modify existing job postings.
- Delete: Employers can remove job postings that are no longer relevant.

2. Applicant Management:
- Create: Job seekers can apply for jobs by submitting their information and attaching relevant documents.
- Read: Employers can view all applications for a particular job posting.
- Update: Applicants can update their submitted information if necessary.
- Delete: Applicants can withdraw their applications.

3. Search Functionality:
- Job seekers can search for job postings based on keywords, location, job type, and other filters.
- Employers can search for applicants based on keywords, experience, and other relevant criteria.

Technical Approach

1. Backend:
- Use SQLite for the database to store job postings, applications, and user information.
- Implement CRUD operations using SQL queries.

2. Frontend:
- Use Tkinter to create the GUI components, including forms for input, buttons for actions, and tables for displaying data.
- Ensure the UI is intuitive and responsive.

3. Integration:
- Connect the Tkinter interface with the SQLite database to handle user interactions and data manipulation seamlessly.

Expected Outcomes

- A fully functional desktop application that allows employers and job seekers to manage job postings and applications efficiently.
- A reliable and efficient database system that supports all necessary CRUD operations and advanced search functionality.
- An intuitive and user-friendly interface that enhances the user experience.

Conclusion

This project aims to streamline the job posting and application process by providing a robust and user-friendly GUI-based DBMS. By covering all essential CRUD and search operations, this system will serve as an effective tool for both employers and job seekers, facilitating better job matching and application tracking.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Introduction

This project aims to develop a Graphical User Interface (GUI)-based Database Management System (DBMS) for managing job postings and applications. The system will facilitate CRUD (Create, Read, Update, Delete) operations and provide search functionalities to streamline job management and application processes for both employers and job seekers.

1.2 Objectives

- Develop a user-friendly interface for job management.
- Implement comprehensive CRUD operations for job postings and applications.
- Enable efficient applicant tracking and management.
- Provide advanced search functionalities to filter job postings and applications.

1.3 Modules

1. User Interface Module: Design and develop the GUI using Tkinter.
2. Job Management Module: Implement CRUD operations for job postings.
3. Applicant Management Module: Implement CRUD operations for job applications.
4. Search Module: Develop search functionalities for job postings and applications.
5. Database Module: Design and manage the SQLite database.

---

2. SURVEY OF TECHNOLOGIES

2.1 Software Description

The project utilizes the following software technologies:

-Tkinter: Python's standard GUI toolkit for creating desktop applications.

- SQLite: A lightweight, disk-based database management system.

- Python: The programming language used to tie together the interface and database operations.

2.2 Languages

2.2.1 SQL

Structured Query Language (SQL) is used to interact with the database. SQL commands are utilized to perform CRUD operations and search functionalities.

2.2.2 Python

Python is the primary language used for developing the application. It provides libraries such as Tkinter for GUI development and sqlite3 for database interactions.

3. REQUIREMENTS AND ANALYSIS

3.1 Requirement Specification

Functional Requirements:

- User authentication for employers and job seekers.

- Employers can create, read, update, and delete job postings.

- Job seekers can apply for jobs and manage their applications.

- Search functionalities for job postings and applications.

Non-Functional Requirements:

- The system should be user-friendly and responsive.

- It should ensure data integrity and security.

3.2 Hardware and Software Requirements

Hardware Requirements:

- Processor: Intel Core i3 or higher

- RAM: 4 GB or more

- Hard Disk: 500 GB or more

Software Requirements:

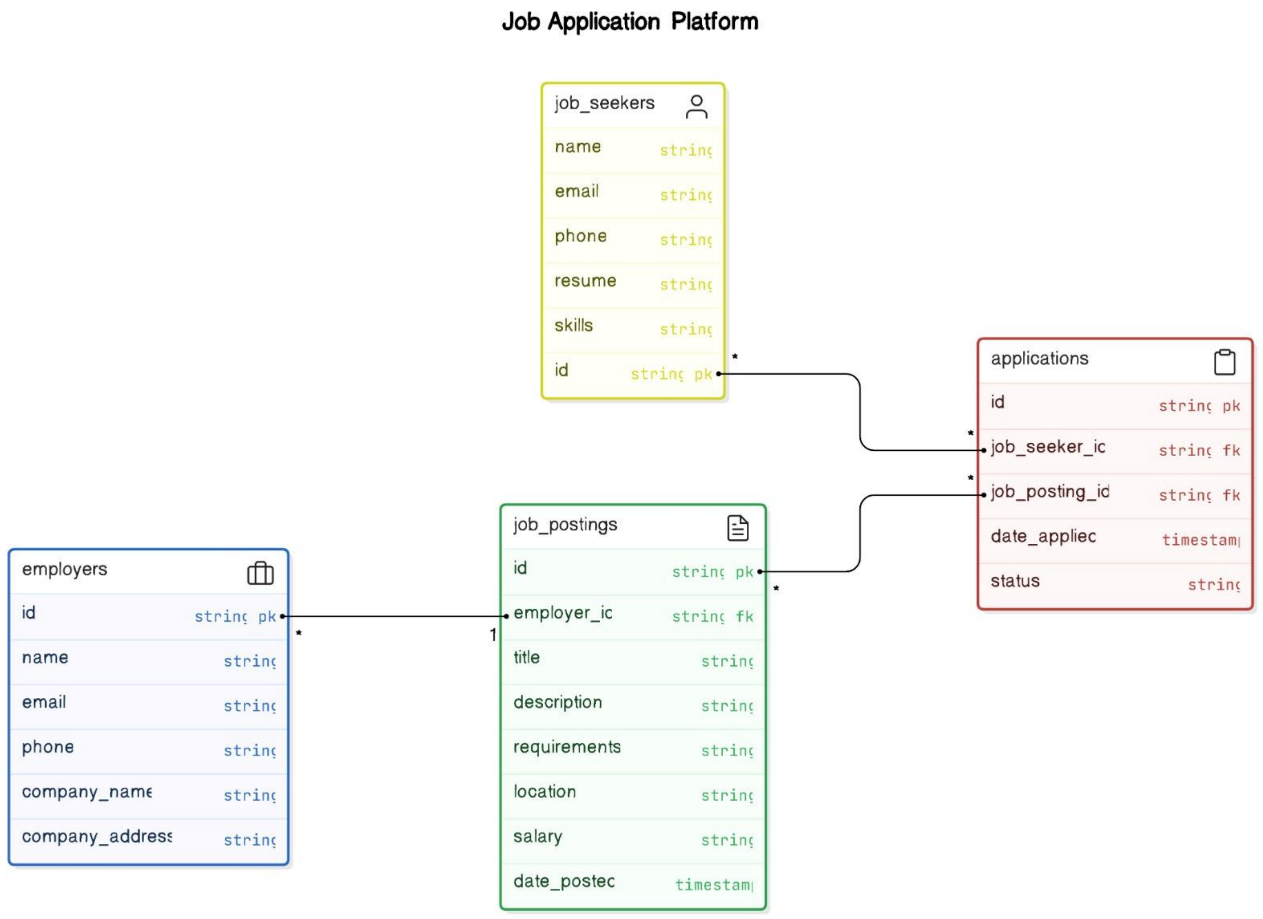
- Operating System: Windows, macOS, or Linux

- Python 3.x

- SQLit0065

3.3 Architecture Diagram

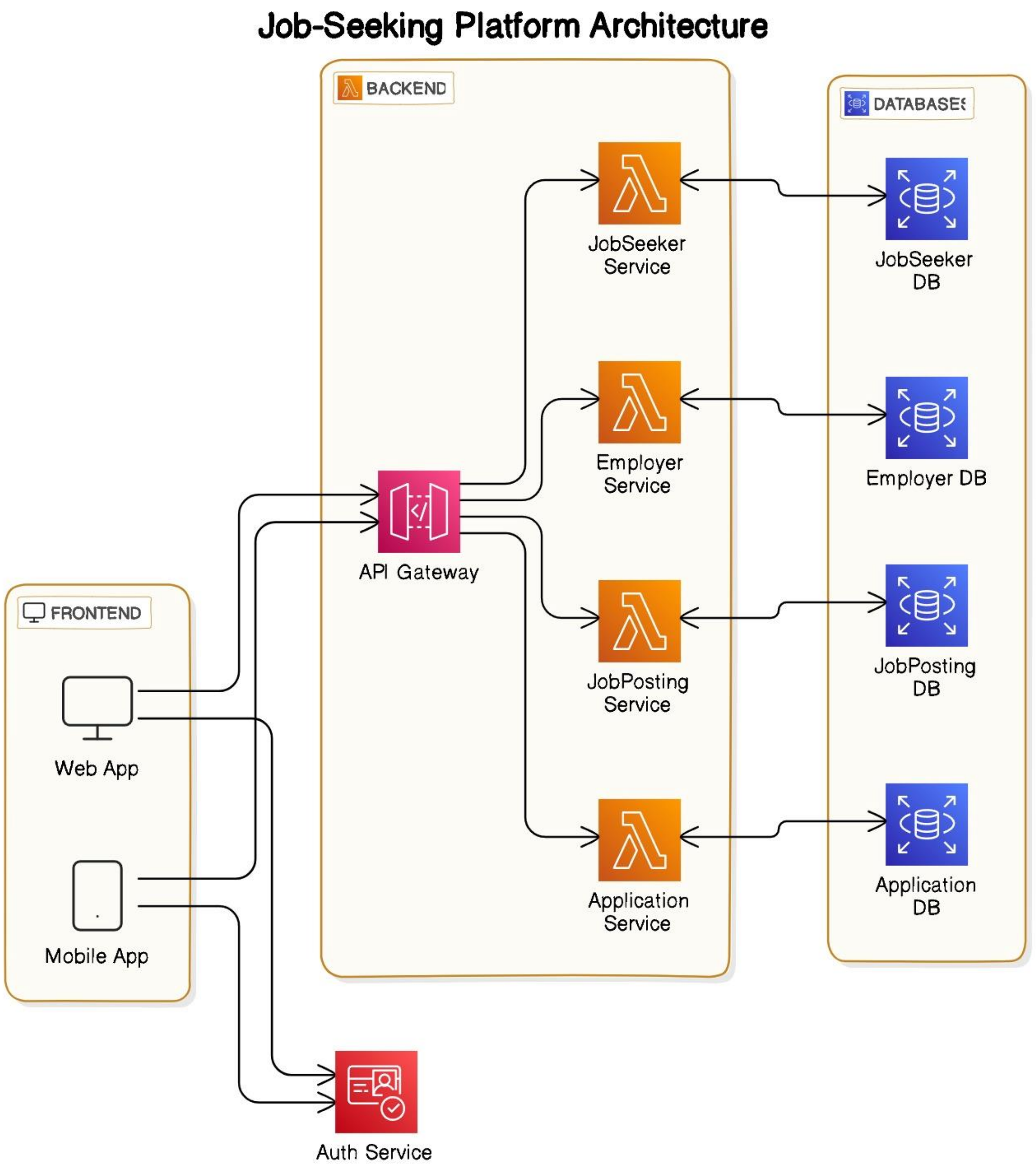
The architecture of the Student Portal follows a client-server model. The client-side application is developed using Java Swing, while the server-side consists of a database managed through JDBC.



3.4 ER Diagram

The Entity-Relationship (ER) diagram represents the database schema for the Student Portal. It includes entities such as Students, Teachers, Subjects, and Marks, along with their relationships.

\



4. PROGRAM CODE

MAIN.PY

```
from tkinter import *
from tkinter import messagebox
from modules.login import *


root = Tk()
root.geometry("1050x700")
root.title("Hire ME")
root.resizable(0, 0)
root.iconbitmap(r'elements\\favicon.ico')
log(root)
root.mainloop()
```

CLIENT.PY

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox, Label
from tkinter_uix.Entry import Entry
import mysql.connector as sql
import modules.login as l
from modules.creds import user_pwd


def get_details(email):
    global name, location, gen, clicid
    q = f'select CName,CLocation,CGender,CID from mydb.client where CEmail="{email}"'
    mycon = sql.connect(host='localhost', user='root',
                        passwd=user_pwd, database='mydb')
    cur = mycon.cursor()
    cur.execute(q)
    d = cur.fetchall()
    mycon.close()

    name = d[0][0]
    location = d[0][1]
    gen = d[0][2]
    clicid = d[0][3]


def logi(root):
    try:
        bg.destroy()
    except:
        pass
    l.log(root)


# ---------------------------------------------Apply a Job-----------------------------------------
def apply(table):
    # fetch cid,jid from treeview that is in available jobs function
    # code
    selectedindex = table.focus()     # that will return number index

    # that will return list of values with columns=['JID','JobRole', 'JobType', 'CompanyName',
'CompanyLocation', 'Qualification','MinExp', 'Salary']

    selectedvalues = table.item(selectedindex, 'values')

    ajid = selectedvalues[0]

    chkquery = f'SELECT * from mydb.application where cid={clicid} and jid={ajid}'

    mycon = sql.connect(host='localhost', user='root',

                        passwd=user_pwd, database='mydb')

    cur = mycon.cursor()

    cur.execute(chkquery)

    tempbuff = cur.fetchall()

    mycon.close()

    if(tempbuff):

        messagebox.showinfo(

            'Oops', 'It seems like you have already applied to this job')

    else:

        queryapplyjob = f'Insert into application values(NULL,(select rid from mydb.job
where job.jid={ajid}),{ajid},{clicid})'

        mycon = sql.connect(host='localhost', user='root',

                            passwd=user_pwd, database='mydb')

        cur = mycon.cursor()

        cur.execute(queryapplyjob)

        mycon.commit()

        mycon.close()

        messagebox.showinfo('Thanks', 'Your application has been submitted')
```

```
            # -------------------------------------------Delete A Job -----------------------------
        -----

def delet(table):
    selectedindex = table.focus()
    selectedvalues = table.item(selectedindex, 'values')
    aaid = selectedvalues[0]
    mycon = sql.connect(host='localhost', user='root',
                        passwd=user_pwd, database='mydb')
    cur = mycon.cursor()
    cur.execute(
        f'delete from mydb.application where aid={aaid}')
    mycon.commit()
    mycon.close()
    messagebox.showinfo('Thanks', 'Your application has been Deleted')
    myapp()


# ------------------------------------------ Sort Queries ---------------------------------------
def sort_alljobs(table):
    criteria = search_d.get()
    if(criteria == "Select"):
        pass
    else:
        table.delete(*table.get_children())
        mycon = sql.connect(host='localhost', user='root',
                            passwd=user_pwd, database='mydb')
        cur = mycon.cursor()
        cur.execute(
            f'select job.JID,job.JobRole,job.JobType, recruiter.CompanyName,
recruiter.CompanyLocation, job.Qualification, job.MinExp, job.Salary from mydb.job JOIN
mydb.recruiter ON job.rid=recruiter.rid order by {criteria}')
        jobs = cur.fetchall()
        mycon.close()
        i = 0
        for r in jobs:
            table.insert('', i, text="", values=(
                r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7]))
            i += 1


def sort_myapplications(table):
    criteria = search_d.get()
    if(criteria == "Select"):
        pass
    else:
        table.delete(*table.get_children())
        mycon = sql.connect(host='localhost', user='root',
                            passwd=user_pwd, database='mydb')
        cur = mycon.cursor()
        cur.execute(
            f'SELECT application.aid,job.JobRole, job.JobType, recruiter.CompanyName,
recruiter.CompanyLocation, job.qualification, job.minexp, job.salary FROM application
JOIN recruiter ON application.rid=recruiter.rid JOIN job ON application.jid=job.jid where
application.CID={clicid} order by {criteria}')

        jobs = cur.fetchall()

        mycon.close()

        i = 0

        for r in jobs:

            table.insert('', i, text="", values=(

                r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7]))

            i += 1


# ----------------------------------------My Applictions-----------------------------------------

def myapp():

    mycon = sql.connect(host='localhost', user='root',

                        passwd=user_pwd, database='mydb')

    cur = mycon.cursor()

    for widget in rt.winfo_children():

        widget.destroy()

    for widget in tab.winfo_children():

        widget.destroy()

    bgr.destroy()


    search_l = Label(rt, text="Order By : ", font=('normal', 18), bg="#ffffff")

    search_l.grid(row=0, column=0, padx=10, pady=10)

    global search_d

    search_d = ttk.Combobox(rt, width=12, font=(
        'normal', 18), state='readonly')
    search_d['values'] = ('Select', 'JobRole', 'JobType', 'CompanyLocation')
    search_d.current(0)
    search_d.grid(row=0, column=2, padx=0, pady=10)
    search = Button(rt, text="Sort", font=('normal', 12, 'bold'), bg="#00b9ed",
                    fg="#ffffff", command=lambda: sort_myapplications(table))
    search.grid(row=0, column=3, padx=10, pady=10, ipadx=15)

    dlt = Button(rt, text="Delete", font=('normal', 12, 'bold'),
                 bg="#00b9ed", fg="#ffffff", command=lambda: delet(table))
    dlt.grid(row=0, column=4, padx=10, pady=10, ipadx=5)

    scx = Scrollbar(tab, orient="horizontal")
    scy = Scrollbar(tab, orient="vertical")

    table = ttk.Treeview(tab, columns=('AID', 'JobRole', 'JobType', 'CompanyName',
'CompanyLocation', 'Qualification', 'MinExp', 'Salary'),
                         xscrollcommand=scx.set, yscrollcommand=scy.set)
    scx.pack(side="bottom", fill="x")
    scy.pack(side="right", fill="y")
    table.heading("AID", text="AID")
    table.heading("JobRole", text="JobRole")
    table.heading("JobType", text="JobType")
    table.heading("CompanyName", text='CompanyName')
    table.heading("CompanyLocation", text="CompanyLocation")
    table.heading("Qualification", text='Qualification')
    table.heading("MinExp", text='MinExp')

    table.heading("Salary", text="Salary")

    table['show'] = 'headings'


    scx.config(command=table.xview)

    scy.config(command=table.yview)


    table.column("AID", width=50)

    table.column("JobRole", width=150)

    table.column("JobType", width=150)

    table.column("CompanyName", width=150)

    table.column("CompanyLocation", width=150)

    table.column("Qualification", width=100)

    table.column("MinExp", width=100)

    table.column("Salary", width=150)

    show_myapplications(table)

    table.pack(fill="both", expand=1)

    mycon.close()
```

LOGIN.PY

```
from tkinter import *

from tkinter import messagebox

from tkinter_uix.Entry import Entry

import mysql.connector as sql

from modules.register import *
from modules.recruiter import *
from modules.client import *
from modules.creds import user_pwd

def success(root, email1):
    global f
    f1.destroy()
    try:
        r1.destroy()
    except:
        pass

    s = f'select type from users where email="{email1}"'
    mycon = sql.connect(host='localhost', user='root',
                        passwd=user_pwd, database='mydb')
    cur = mycon.cursor()
    cur.execute(s)
    q = cur.fetchall()
    mycon.close()
    print(q)

    if q[0][0] == "recruiter":
        rec(root, email1)
    else:
        cli(root, email1)
```

```
def submit(root):
    mycon = sql.connect(host='localhost', user='root',
                        passwd=user_pwd, database='mydb')
    cur = mycon.cursor()
    cur.execute('select email,password from users')
    total = cur.fetchall()
    mycon.close()
    email1 = email.get()
    password = pwd.get()
    if email1 and password:
        for i in total:
            if email1 == i[0] and password == i[1]:
                return success(root, email1)
            elif email1 == i[0] and password != i[1]:
                messagebox.showinfo('Alert!', 'Invalid Credentials')
                break
        else:
            messagebox.showinfo(
                'Alert!', 'Email is not registered, Please register')
    else:
        messagebox.showinfo(
            'Alert!', 'Please Enter both Email and Password')


def reg(root):
    try:
        f1.destroy()
    except:
        pass
    mai(root)


def log(root):
    global f1, email, pwd
    try:
        f2.destroy()
    except:
        pass
    f1 = Frame(root, width=1050, height=700, bg='#FFFFFF')
    f1.place(x=0, y=0)

    # Background
    f1.render = PhotoImage(file='elements\\bg.png')
    img = Label(f1, image=f1.render)
    img.place(x=0, y=0)

    # Email
    email_l = Label(f1, text="Email : ", bg='#FFFFFF',
                    font=('normal', 20, 'bold'), fg="#00B9ED")
    email_l.place(x=620, y=300)
    email = Entry(f1, width=24, placeholder="Enter your Email..")
    email.place(x=720, y=300)

    # Password
    pwd_l = Label(f1, text="Password : ", bg='#FFFFFF',
                  font=('normal', 20, 'bold'), fg="#00B9ED")
    pwd_l.place(x=565, y=350)
    pwd = Entry(f1, show="*", width=24, placeholder="Enter your Password..")
    pwd.place(x=720, y=350)

    # Buttons
    f1.bn = PhotoImage(file="elements\\login2.png")
    btn = Button(f1, image=f1.bn, bg='#FFFFFF', bd=0,
                 activebackground="#ffffff", command=lambda: submit(root))
    btn.place(x=820, y=420)

    f1.bn1 = PhotoImage(file="elements\\reg.png")
    btn1 = Button(f1, image=f1.bn1, bg='#FFFFFF', bd=0,
                  activebackground="#ffffff", command=lambda: reg(root))
    btn1.place(x=620, y=420)

# ---------------------------------------------Applicants-----------------------------------------

def show_applicants(table):
    mycon = sql.connect(host='localhost', user='root',
                        passwd=user_pwd, database='mydb')
    cur = mycon.cursor()
    cur.execute(
        f'SELECT job.JobRole, client.CName, client.CEmail, client.CAge, client.CLocation, client.CGender, client.CExp, client.CSkills, client.CQualification FROM application JOIN client ON application.cid=client.CID JOIN job ON job.jid=application.jid where job.rid={recid}')
    applicats = cur.fetchall()
    mycon.close()
    print(applicats)
    i = 0
    for x in applicats:
        table.insert('', i, text="", values=(
            x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8]))
        i += 1



# --------------------------------------------Post a Job--------------------------------------------
def create():
    global role, jtype, qual, exp, sal
    for widget in rt.winfo_children():
        widget.destroy()
    for widget in tab.winfo_children():
        widget.destroy()
    bgr.destroy()

    # Create Form
    f1 = Frame(rt, width=520)
    f1.load = PhotoImage(file="elements\\create.png")
```

```
img = Label(rt, image=f1.load, bg="#FFFFFF")
img.grid(row=0, column=1, padx=150, pady=10)

# Form
# Labels
role_l = Label(tab, text="Role :", font=(
   'normal', 18, 'bold'), bg="#FFFFFF")
role_l.grid(row=0, column=0, pady=10, padx=10)
type_l = Label(tab, text="Type :", font=(
   'normal', 18, 'bold'), bg="#FFFFFF")
type_l.grid(row=1, column=0, pady=10, padx=10)
qual_l = Label(tab, text="Qualification :", font=(
   'normal', 18, 'bold'), bg="#FFFFFF")
qual_l.grid(row=2, column=0, pady=10, padx=10)
exp_l = Label(tab, text="Experience :", font=(
   'normal', 18, 'bold'), bg="#FFFFFF")
exp_l.grid(row=3, column=0, pady=10, padx=10)
sal_l = Label(tab, text="Salary :", font=(
   'normal', 18, 'bold'), bg="#FFFFFF")
sal_l.grid(row=4, column=0, pady=10, padx=10)

# Entries
style = ttk.Style(tab)
style.configure("TCombobox", background="white",
            foreground="#696969")
```

```
role = Entry(tab, placeholder="Enter Job Role")
role.grid(row=0, column=1, pady=10, padx=10)
jtype = ttk.Combobox(tab, font=("normal", 18),
                  width=23, state='readonly')
jtype['values'] = ('Select', 'FullTime', 'PartTime', 'Intern')
jtype.current(0)
jtype.grid(row=1, column=1, pady=10, padx=10)
qual = Entry(tab, placeholder="Enter Job Qualifications")
qual.grid(row=2, column=1, pady=10, padx=10)
exp = Entry(tab, placeholder="Enter Minimum Experience")
exp.grid(row=3, column=1, pady=10, padx=10)
sal = Entry(tab, placeholder="Enter Expected salary")
sal.grid(row=4, column=1, pady=10, padx=10)

btn = Button(tab, text="Submit", font=(20), bg="#45CE30",
           fg="#FFFFFF", command=submit_job)
btn.grid(row=5, column=1, pady=15)
scy.config(command=table.yview)
```

REGISTER.PY

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox, Label
from tkinter_uix.Entry import Entry
import mysql.connector as sql
import modules.login as l
from modules.creds import user_pwd


def logi(root):
    try:
        r2.destroy()
        r3.destroy()
    except:
        pass
    l.log(root)



def mai(root):
    try:
        r2.destroy()
    except:
        pass
    global r1
    r1 = Frame(root, height=700, width=1050)

    r1.place(x=0, y=0)

    r1.render = PhotoImage(file="elements/Registration_bg.png")

    img = Label(r1, image=r1.render)

    img.place(x=0, y=0)

    r1.Img1 = PhotoImage(file="elements/recruiter_element.png")

    recruit = Button(r1, image=r1.Img1, border=0, bg="#03DDEE",

                     relief="raised", activebackground="#03EAFD", command=lambda:
recruiter_regis(root))

    recruit.place(x=140, y=340)

    r1.Img2 = PhotoImage(file="elements/client_element.png")

    recruit2 = Button(r1, image=r1.Img2, border=0, bg="#05edFC",

                      relief="raised", activebackground="#05F6FD", command=lambda:
client_regis(root))

    recruit2.place(x=360, y=340)

    r1.bn = PhotoImage(file="elements\\backlogin.png")

    btn = Button(r1, image=r1.bn, bg='#05e4f6',

                 bd=0, activebackground="#05e4f6", command=lambda: logi(root))

    btn.place(x=220, y=550)




def recruiter_regis(root):

    global name, email, pwd, cpwd

    print("hello recruiter")

    r1.destroy()

    r2 = Frame(root, height=700, width=1050)

    r2.place(x=0, y=0)
```

```
r2.render = PhotoImage(file="elements/reg_bg.png")
img = Label(r2, image=r2.render)
img.place(x=0, y=0)
name_l = Label(r2, text="Name : ", bg='#FFFFFF', fg="#00B9ED",
               font=('normal', 20, 'bold'))
name_l.place(x=100, y=250)
name = Entry(r2, placeholder='Enter Your Full Name...', width=20)
name.place(x=290, y=250)

email_l = Label(r2, text="Email : ", bg='#FFFFFF', fg="#00B9ED",
                font=('normal', 20, 'bold'))
email_l.place(x=100, y=300)
email = Entry(r2, placeholder='Email', width=20)
email.place(x=290, y=300)

pwd_l = Label(r2, text="Password : ", bg='#FFFFFF', fg="#00B9ED",
              font=('normal', 20, 'bold'))
pwd_l.place(x=100, y=350)
pwd = Entry(r2, placeholder='Password', show="*", width=20)
pwd.place(x=290, y=350)

con_pwd_l = Label(r2, text="Confirm : ", bg='#FFFFFF', fg="#00B9ED",
                  font=('normal', 20, 'bold'))
con_pwd_l.place(x=100, y=400)
cpwd = Entry(r2, placeholder='Confirm Password', show="*", width=20)
cpwd.place(x=290, y=400)
```

```
    r2.bn = PhotoImage(file="elements\\next1.png")
    btn = Button(r2, image=r2.bn, bg='#FFFFFF', bd=0,
                 activebackground="#ffffff", command=lambda: recruiter_check(root))
    btn.place(x=320, y=500)

    r2.back = PhotoImage(file="elements\\back.png")
    btn2 = Button(r2, image=r2.back, bg='#FFFFFF', bd=0,
                  activebackground="#ffffff", command=lambda: mai(root))
    btn2.place(x=120, y=500)


def recruiter_check(root):
    global name1, email1, pwd1, cpwd1
    name1 = name.get()
    email1 = email.get()
    pwd1 = pwd.get()
    cpwd1 = cpwd.get()
    print(name1, email1, pwd1, cpwd1)
    if name1 and email1 and pwd1 and cpwd1:
        mycon = sql.connect(host='localhost', user='root',
                            passwd=user_pwd, database='mydb')
        cur = mycon.cursor()
        cur.execute('select email from users')
        total = cur.fetchall()
        mycon.close()
        exist_email = []
        for i in total:
            exist_email.append(i[0])
        print("existing users:", exist_email)

        if email1 in exist_email:
            messagebox.showinfo('ALERT!', 'EMAIL ALREADY REGISTERED')
            email.delete(0, END)

        else:
            if pwd1 == cpwd1:
                recruit_complete(root)
            else:
                messagebox.showinfo('ALERT!', 'PASSWORDS DO NOT MATCH')

    else:
        messagebox.showinfo('ALERT!', 'ALL FIELDS ARE MUST BE FILLED')


def recruit_complete(root):
    print("hello ", name1, ", Let's complete your profile")
    r3 = Frame(root, height=700, width=1050)
    r3.place(x=0, y=0)
    r3.render = PhotoImage(file="elements/reg_bg.png")
    img = Label(r3, image=r3.render)
    img.place(x=0, y=0)
```

```
global gender, company, loc
gender = StringVar()

style = ttk.Style(r3)
style.configure("TRadiobutton", background="white",
                foreground="#696969", font=("arial", 16, "bold"))

gender_l = Label(r3, text="Gender : ", bg='#FFFFFF', fg="#00B9ED",
                 font=('normal', 20, 'bold'))
gender_l.place(x=100, y=250)
ttk.Radiobutton(r3, text="Male", value="M", variable=gender).place(
    x=300, y=250)
ttk.Radiobutton(r3, text="Female", value="F", variable=gender).place(
    x=400, y=250)

company_l = Label(r3, text="Company : ", bg='#FFFFFF', fg="#00B9ED",
                  font=('normal', 20, 'bold'))
company_l.place(x=100, y=300)
company = Entry(r3, placeholder='Company', width=20)
company.place(x=290, y=300)

loc_l = Label(r3, text="Location : ", bg='#FFFFFF', fg="#00B9ED",
              font=('normal', 20, 'bold'))
loc_l.place(x=100, y=350)
loc = Entry(r3, placeholder='Location', width=20)
```

```
    loc.place(x=290, y=350)


    r3.bn = PhotoImage(file="elements\\reg.png")

    btn = Button(r3, image=r3.bn, bg='#FFFFFF', bd=0,

                 activebackground="#ffffff", command=lambda: recruiter_submit(root))

    btn.place(x=320, y=500)



def recruiter_submit(root):

    global gender1, company1, loc1

    gender1 = gender.get()

    company1 = company.get()

    loc1 = loc.get()

    print(name1, email1, gender1, company1, loc1)

    if gender1 and company1 and loc1:

        exe = f'insert into users values("{name1}","{email1}","recruiter","{pwd1}")'

        exe1 = f'INSERT INTO mydb.Recruiter(RID, RName, REmail, CompanyName,
CompanyLocation ,RGender) VALUES
(NULL,"{name1}","{email1}","{company1}","{loc1}","{gender1}")'

        try:

            mycon = sql.connect(host='localhost', user='root',

                                passwd=user_pwd, database='mydb')

            cur = mycon.cursor()

            cur.execute(exe)

            cur.execute(exe1)

            name.delete(0, END)

            email.delete(0, END)
            pwd.delete(0, END)
            cpwd.delete(0, END)
            # gender.delete(0, END)
            loc.delete(0, END)
            company.delete(0, END)
            mycon.commit()
            mycon.close()
            messagebox.showinfo('SUCCESS!', 'Registration Successful')
            logi(root)
        except:
            pass

    else:
        messagebox.showinfo('ALERT!', 'ALL FIELDS ARE MUST BE FILLED')


def client_regis(root):
    global name, email, pwd, cpwd
    print("hello client")
    r1.destroy()
    r2 = Frame(root, height=700, width=1050)
    r2.place(x=0, y=0)
    r2.render = PhotoImage(file="elements/reg_bg.png")
    img = Label(r2, image=r2.render)
    img.place(x=0, y=0)
```

```
name_l = Label(r2, text="Name : ", bg='#FFFFFF', fg="#00B9ED",
               font=('normal', 20, 'bold'))
name_l.place(x=100, y=250)
name = Entry(r2, placeholder='Enter Your Full Name...', width=20)
name.place(x=290, y=250)

email_l = Label(r2, text="Email : ", bg='#FFFFFF', fg="#00B9ED",
                font=('normal', 20, 'bold'))
email_l.place(x=100, y=300)
email = Entry(r2, placeholder='Email', width=20)
email.place(x=290, y=300)

pwd_l = Label(r2, text="Password : ", bg='#FFFFFF', fg="#00B9ED",
              font=('normal', 20, 'bold'))
pwd_l.place(x=100, y=350)
pwd = Entry(r2, placeholder='Password', show="*", width=20)
pwd.place(x=290, y=350)

con_pwd_l = Label(r2, text="Confirm : ", bg='#FFFFFF', fg="#00B9ED",
                  font=('normal', 20, 'bold'))
con_pwd_l.place(x=100, y=400)
cpwd = Entry(r2, placeholder='Confirm Password', show="*", width=20)
cpwd.place(x=290, y=400)

r2.bn = PhotoImage(file="elements\\next1.png")
btn = Button(r2, image=r2.bn, bg='#FFFFFF', bd=0,
            activebackground="#ffffff", command=lambda: client_check(root))
    btn.place(x=320, y=500)

    r2.back = PhotoImage(file="elements\\back.png")
    btn2 = Button(r2, image=r2.back, bg='#FFFFFF', bd=0,
                  activebackground="#ffffff", command=lambda: mai(root))
    btn2.place(x=120, y=500)


def client_check(root):
    global name1, email1, pwd1, cpwd1
    name1 = name.get()
    email1 = email.get()
    pwd1 = pwd.get()
    cpwd1 = cpwd.get()
    print(name1, email1, pwd1, cpwd1)
    if name1 and email1 and pwd1 and cpwd1:
        mycon = sql.connect(host='localhost', user='root',
                            passwd=user_pwd, database='mydb')
        cur = mycon.cursor()
        cur.execute('select email from users')
        total = cur.fetchall()
        mycon.close()
        exist_email = []
        for i in total:
            exist_email.append(i[0])
        print("existing users:", exist_email)


        if email1 in exist_email:
            messagebox.showinfo('ALERT!', 'EMAIL ALREADY REGISTERED')
            email.delete(0, END)


        else:
            if pwd1 == cpwd1:
                client_complete(root)
            else:
                messagebox.showinfo('ALERT!', 'PASSWORDS DO NOT MATCH')


    else:
        messagebox.showinfo('ALERT!', 'ALL FIELDS ARE MUST BE FILLED')



def client_complete(root):
    print("hello ", name1, ", Let's complete your profile")
    r3 = Frame(root, height=700, width=1050)
    r3.place(x=0, y=0)
    r3.render = PhotoImage(file="elements/reg_bg.png")
    img = Label(r3, image=r3.render)
    img.place(x=0, y=0)


    global gender, age, loc, workxp, qualification, skills
    gender = StringVar()
```

```
style = ttk.Style(r3)
style.configure("TRadiobutton", background="white",
            foreground="#696969", font=("arial", 16, "bold"))

gender_l = Label(r3, text="Gender : ", bg='#FFFFFF', fg="#00B9ED",
             font=('normal', 20, 'bold'))
gender_l.place(x=100, y=200)
ttk.Radiobutton(r3, text="Male", value="M", variable=gender).place(
   x=300, y=200)
ttk.Radiobutton(r3, text="Female", value="F", variable=gender).place(
   x=400, y=200)

age_l = Label(r3, text="Age : ", bg='#FFFFFF', fg="#00B9ED",
           font=('normal', 20, 'bold'))
age_l.place(x=100, y=250)
age = Entry(r3, placeholder='Age', width=20)
age.place(x=290, y=250)

loc_l = Label(r3, text="Location : ", bg='#FFFFFF', fg="#00B9ED",
           font=('normal', 20, 'bold'))
loc_l.place(x=100, y=300)
loc = Entry(r3, placeholder='Location', width=20)
loc.place(x=290, y=300)

workxp_l = Label(r3, text="Experience : ", bg='#FFFFFF', fg="#00B9ED",
                font=('normal', 20, 'bold'))
    workxp_l.place(x=100, y=350)
    workxp = Entry(r3, placeholder='Work Experience(yrs)', width=20)
    workxp.place(x=290, y=350)

    qualification_l = Label(r3, text="Qualification : ",
                        bg='#FFFFFF', fg="#00B9ED", font=('normal', 20, 'bold'))
    qualification_l.place(x=100, y=400)
    qualification = Entry(r3, placeholder='Btech/BE...', width=20)
    qualification.place(x=290, y=400)

    skills_l = Label(r3, text="Skills : ", bg='#FFFFFF',
                fg="#00B9ED", font=('normal', 20, 'bold'))
    skills_l.place(x=100, y=450)
    skills = Entry(r3, placeholder='separated by comma', width=20)
    skills.place(x=290, y=450)

    r3.bn = PhotoImage(file="elements\\reg.png")
    btn = Button(r3, image=r3.bn, bg='#FFFFFF', bd=0,
             activebackground="#ffffff", command=lambda: client_submit(root))
    btn.place(x=320, y=550)


def client_submit(root):
    global gender1, age1, loc1, workxp1, qualification1, skills1
    gender1 = gender.get()
    age1 = age.get()

    loc1 = loc.get()

    workxp1 = workxp.get()

    qualification1 = qualification.get()

    skills1 = skills.get()

    print(name1, email1, gender1, age1, loc1, workxp1, qualification1, skills1)

    if gender1 and age1 and loc1 and workxp1:

        exe = f'insert into users values("{name1}","{email1}","client","{pwd1}")'

        exe1 = f'INSERT INTO mydb.Client(CID, CName , CEmail, CAge, CLocation,
CGender, CExp, CSkills, CQualification ) VALUES (NULL, "{name1}", "{email1}",
{age1}, "{loc1}", "{gender1}", {workxp1}, "{skills1}", "{qualification1}");'

        try:

            mycon = sql.connect(host='localhost', user='root',

                                passwd=user_pwd, database='mydb')

            cur = mycon.cursor()

            cur.execute(exe)

            cur.execute(exe1)

            name.delete(0, END)

            email.delete(0, END)

            pwd.delete(0, END)

            cpwd.delete(0, END)

            # gender.delete(0, END)

            loc.delete(0, END)

            age.delete(0, END)

            workxp.delete(0, END)

            qualification.delete(0, END)

            skills.delete(0, END)
            mycon.commit()
            mycon.close()
            messagebox.showinfo('SUCCESS!', 'Registration Successful')
            logi(root)
        except:
            pass

    else:
        messagebox.showinfo('ALERT!', 'ALL FIELDS ARE MUST BE FILLED')
```

5. RESULTS AND DISCUSSION

The project successfully implements a GUI-based DBMS for job postings and applications. Users can add, view, update, and delete job postings and applications through an intuitive interface. The search functionality allows users to filter jobs and applications based on various criteria. The use of Tkinter for the GUI and SQLite for the database ensures a lightweight and efficient application.

6. CONCLUSION

This project demonstrates the development of a comprehensive GUI-based DBMS for managing job postings and applications. The system covers essential CRUD operations and provides robust search capabilities, offering a user-friendly experience for both job seekers and employers.

7. REFERENCES

- Tkinter Documentation: [https://docs.python.org/3/library/tkinter.html]

- SQLite Documentation: [https://www.sqlite.org/docs.html]

- Python Documentation: [https://docs.python.org/3/]

- SQL Tutorial: [https://www.w3schools.com/sql/]

.