TRƯỜNG ĐẠI HỌC HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

# Báo Cáo Python Tkinter

## Student Management System

Võ Minh Tân - 2274802010799

2024

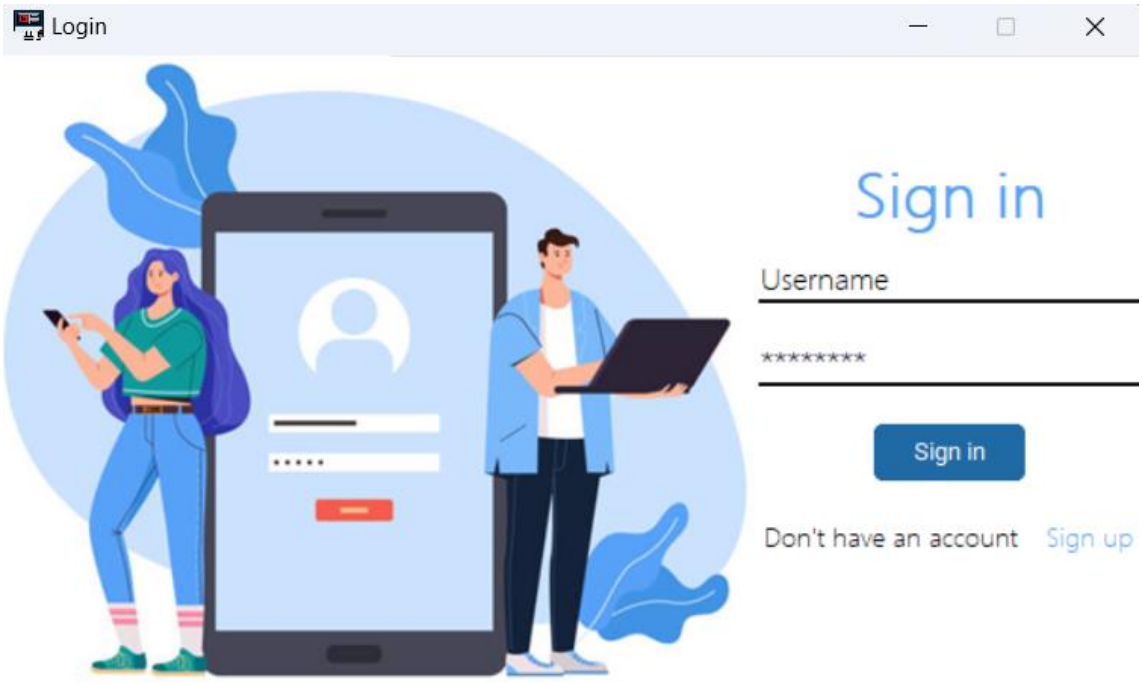
## Mục Lục

**I. Nội Dung** ..... 3
**1. Giao diện** ..... 3
**2. Chức năng** ..... 3
**2.1. Chức năng Đăng Nhập** ..... 3
**2.2. Chức năng Quản Lý Sinh Viên** ..... 4
**II. Source code** ..... 4
**III. Link github** ..... 8

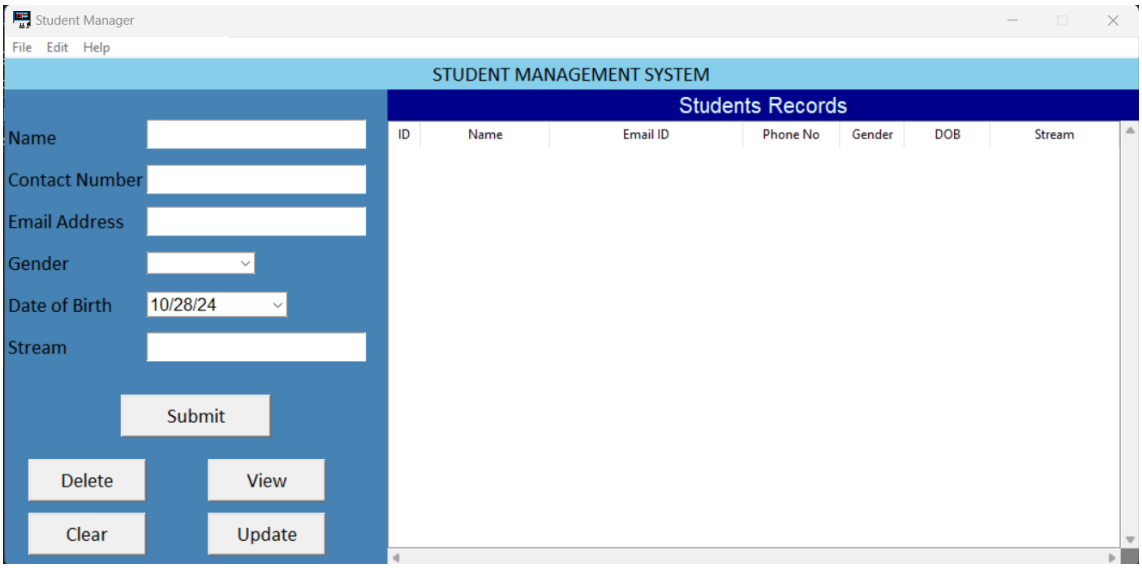
# I. Nội Dung

## 1. Giao diện

- giao diện login




- giao diện form chính




## 2. Chức năng

### 2.1. Chức năng Đăng Nhập

**- Giao diện Đăng Nhập:** Cung cấp giao diện cho người dùng để nhập tên đăng nhập và mật khẩu.

**- Xác thực**: Kiểm tra thông tin đăng nhập với cơ sở dữ liệu. Nếu đăng nhập thành công, chuyển đến giao diện quản lý sinh viên; nếu không, thông báo lỗi.

### 2.2. Chức năng Quản Lý Sinh Viên

#### 2.2.1. Nhập Thông Tin Sinh Viên

**- Trường Nhập:** Cho phép người dùng nhập tên, email, số điện thoại, giới tính, ngày sinh và dòng học của sinh viên.

**- Kiểm Tra Dữ Liệu:** Kiểm tra tính hợp lệ của email và đảm bảo rằng tất cả các trường đều được điền.

#### 2.2.2. Các Chức Năng Quản Lý

**- Thêm Sinh Viên:** Ghi thông tin sinh viên vào cơ sở dữ liệu.

**- Xóa Sinh Viên:** Xóa thông tin sinh viên đã chọn khỏi cơ sở dữ liệu và cập nhật giao diện.

**- Cập Nhật Thông Tin Sinh Viên:** Cho phép người dùng chỉnh sửa thông tin của sinh viên đã chọn và lưu lại thay đổi.

**- Xem Danh Sách Sinh Viên:** Hiển thị danh sách tất cả sinh viên từ cơ sở dữ liệu trong một bảng (Treeview).

#### 2.2.3. Chức Năng Xuất Nhập Dữ Liệu

- **Mở Tệp**: Cho phép người dùng mở tệp văn bản và hiển thị thông tin trong bảng.

- **Lưu Dữ Liệu**: Lưu danh sách sinh viên hiện tại vào tệp văn bản.

## II. Source code

### 1. Main.py

```python
import tkinter as tk
from data import MySQL
from tkinter import messagebox as ms
import customtkinter as ctk

from view_student import StudentManager

class Login:
    def __init__(self, root):
        self.root = root
        self.root.title("Login")
        self.root.configure(bg='#fff')
        self.root.resizable(0, 0)

        self.root.iconbitmap('img/icon_student.ico')

        self.etr_user = tk.StringVar()
        self.entr_pass = tk.StringVar()

        self.db = MySQL()

        self.img = tk.PhotoImage(file='img/login.png')

        self.create_widgets()

    def create_widgets(self):
        tk.Label(self.root, image=self.img, bg='white').grid(row=0, column=0, sticky='snew')

        self.frame = tk.Frame(self.root, width=350, height=350, bg='white')
        self.frame.grid(row=0, column=1)

        self.heading = tk.Label(self.frame, text='Sign in', fg='#57a1f8', bg='white', font=('Microsoft YaHei UI Light', 23, 'bold'))
        self.heading.grid(row=0, column=0, columnspan=2, sticky='ew')

        # Tạo trường nhập cho Username
        self.user_entry_frame = tk.Frame(self.frame, bg='white')
        self.user_entry_frame.grid(row=1, column=0, sticky='ew')

        self.user_entry = tk.Entry(self.user_entry_frame, textvariable=self.etr_user, width=25, fg='black', border=0, bg='white',
                                   font=('Microsoft Yahei UI Light', 11))
        self.user_entry.grid(row=0, column=0, pady=(10, 0))
        self.user_entry.insert(0, "Username")
        self.user_entry.bind('<FocusIn>', lambda e: self.clear_placeholder(self.user_entry, "Username"))
        self.user_entry.bind('<FocusOut>', lambda e: self.set_placeholder(self.user_entry, "Username"))

        self.user_underline = tk.Frame(self.user_entry_frame, bg='black', height=2)
        self.user_underline.grid(row=1, column=0, sticky='ew', pady=(0, 10))

        # Tạo trường nhập cho Password
        self.password_entry_frame = tk.Frame(self.frame, bg='white')
        self.password_entry_frame.grid(row=2, column=0, sticky='ew')

        self.password_entry = tk.Entry(self.password_entry_frame, textvariable=self.entr_pass, width=25, fg='black', border=0, bg='white',
                                       font=('Microsoft Yahei UI Light', 11), show='*')
        self.password_entry.grid(row=0, column=0, pady=(10, 0))
        self.password_entry.insert(0, "Password")
        self.password_entry.bind('<FocusIn>', lambda e: self.clear_placeholder(self.password_entry, "Password"))
        self.password_entry.bind('<FocusOut>', lambda e: self.set_placeholder(self.password_entry, "Password"))

        self.password_underline = tk.Frame(self.password_entry_frame, bg='black', height=2)
        self.password_underline.grid(row=1, column=0, sticky='ew', pady=(0, 10))

        # Button Login
        ctk.CTkButton(self.frame, width=80, height=30, text='Sign in', text_color='white', corner_radius=5,command=self.login_user).grid(row=3, columnspan=2, pady=(10, 0))
        lbl_forgot = tk.Label(self.frame, text="Don't have an account", fg='black', bg='white', cursor='hand2', font=('Microsoft Yahei UI Light', 10))
        lbl_forgot.grid(row=4, column=0, sticky='w')

        self.sign_up = tk.Button(self.frame, text="Sign up", fg='white', border=0, bg='white', cursor='hand2', foreground='#57a1f8', font=('Microsoft Yahei UI Light', 10))
        self.sign_up.grid(row=4, column=0, pady=15, sticky='e')

    def clear_placeholder(self, entry, placeholder):
        if entry.get() == placeholder:
            entry.delete(0, tk.END)

    def set_placeholder(self, entry, placeholder):
        if entry.get() == '':
            entry.insert(0, placeholder)

    def login_user(self):
        username = self.etr_user.get()
        password = self.entr_pass.get()

        if username == '' or password == '':
            ms.showerror("Error", "Please fill in all fields!")
            return

        if self.db.login(username, password):
            ms.showinfo("Success", "Login successful!")
            self.root.destroy()

            main_window = tk.Tk()
            StudentManager(main_window)
            main_window.mainloop()
        else:
            ms.showerror("Error", "Invalid username or password!")

if __name__ == '__main__':
    root = tk.Tk()
    login_gui = Login(root)
    root.mainloop()

```

## 2. data.py

```python
import mysql.connector as mysql
import configDB as guiConf

from tkinter import messagebox as ms

class MySQL:

    GUIDB = "SystemStudent"

    def connect(self):
        conn = mysql.connect(**guiConf.dbConfig)

        cursor = conn.cursor()

        return conn, cursor

    def close(self, cursor, conn):
        cursor.close()
        conn.close()

    def createGuiDB(self):
        conn, cursor = self.connect()

        try:
            cursor.execute(f"CREATE DATABASE IF NOT EXISTS {self.GUIDB}")
            conn.commit()
            print(f"Database {self.GUIDB} created successfully")
        except mysql.Error as error:
            print(f"Failed to create database: {error}")

        self.close(cursor, conn)

    def dropGuiDB(self):

        conn, cursor = self.connect()
        try:
            cursor.execute(f"DROP DATABASE IF EXISTS {self.GUIDB}")
            conn.commit()
            print(f"Database {self.GUIDB} dropped successfully")
        except mysql.Error as error:
            print(f"Failed to drop database: {error}")

        self.close(cursor, conn)

    def useGuiDB(self, cursor):
        cursor.execute(f"USE {self.GUIDB}")

    def createTables(self):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)

        cursor.execute('''
            CREATE TABLE IF NOT EXISTS students (
                id INT AUTO_INCREMENT PRIMARY KEY,
                name VARCHAR(100),
                email VARCHAR(100),
                contact_number VARCHAR(15),
                gender VARCHAR(10),
                dob DATE,
                stream VARCHAR(50)
            )
        ''')
        print("Bảng 'students' đã được tạo hoặc đã tồn tại.")

        cursor.execute('''
            CREATE TABLE IF NOT EXISTS users (
                id INT AUTO_INCREMENT PRIMARY KEY,
                username VARCHAR(50) UNIQUE,
                password VARCHAR(255)
            )
        ''')
        print("Bảng 'users' đã được tạo hoặc đã tồn tại.")

        conn.commit()
        print("Tables created successfully.")


        self.close(cursor, conn)

    def dropTables(self):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)
        try:
            cursor.execute("DROP TABLE IF EXISTS students")
            cursor.execute("DROP TABLE IF EXISTS users")
            conn.commit()
            print("Tables dropped successfully.")
        except mysql.connector.Error as err:
            print(f"Error dropping tables: {err}")
        finally:
            self.close(cursor, conn)

    def insert_student(self, name, email, contact_number, gender, dob, stream):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)
        try:
            cursor.execute('''
                INSERT INTO students (name, email, contact_number, gender, dob, stream)
                VALUES (%s, %s, %s, %s, %s, %s)
            ''', (name, email, contact_number, gender, dob, stream))
            conn.commit()
            print("Student inserted successfully.")
        except mysql.Error as err:
            print(f"Error inserting student: {err}")
        finally:
            self.close(cursor, conn)

    def showStudent(self):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)
        try:
            cursor.execute("SELECT * FROM students")
            rows = cursor.fetchall()
            # for row in rows:
            #    print(row)
        except mysql.Error as err:
            print(f"Error fetching students: {err}")
        finally:
            self.close(cursor, conn)

            return rows

    def update_record(self, student_id, name, email, contact_number, gender, dob, stream):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)

        try:
            cursor.execute('''
                UPDATE students
                SET name = %s, email = %s, contact_number = %s, gender = %s, dob = %s, stream = %s
                WHERE id = %s
            ''', (name, email, contact_number, gender, dob, stream, student_id))

            conn.commit()
            print("Record updated successfully.")
        except mysql.Error as err:
            print(f"Error updating record: {err}")
        finally:
            self.close(cursor, conn)

    def delete_record(self, student_id):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)

        try:
            cursor.execute("DELETE FROM students WHERE id = %s", (student_id,))
            conn.commit()
            print("Record deleted successfully.")
        except mysql.Error as e:
            print(f"Error deleting record: {e}")
        finally:
            self.close(cursor, conn)

    def login(self, username, password):
        conn, cursor = self.connect()
        self.useGuiDB(cursor)
        try:
            cursor.execute("SELECT password FROM users WHERE username = %s", (username,))
            result = cursor.fetchone()
            if result and result[0] == password:
                print("Login successful!")
                return True
            else:
                print("Invalid username or password.")
                return False
        except mysql.Error as err:
            print(f"Error during login: {err}")
            return False
        finally:
            self.close(cursor, conn)



if __name__ == '__main__':
    db = MySQL()
    # db.createGuiDB()
    db.createTables()
```

## 3. view_student.py

**III. Link github**

https://github.com/VMT200104/Python_Advance_Tkinter