

TRƯỜNG ĐẠI HỌC HỌC VĂN LANG  
KHOA CÔNG NGHỆ THÔNG TIN

**VAN LANG**  
UNIVERSITY



**BÁO CÁO ĐỒ ÁN MÔN HỌC HK241  
LẬP TRÌNH PYTHON NÂNG CAO**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ SINH VIÊN  
BẰNG PYTHON**

Sinh viên thực hiện: Nguyễn Phúc Nguyên

MSSV: 2274802010586

GVHD: Huỳnh Thái Học

TP. Hồ Chí Minh – năm 2024

## Mục lục

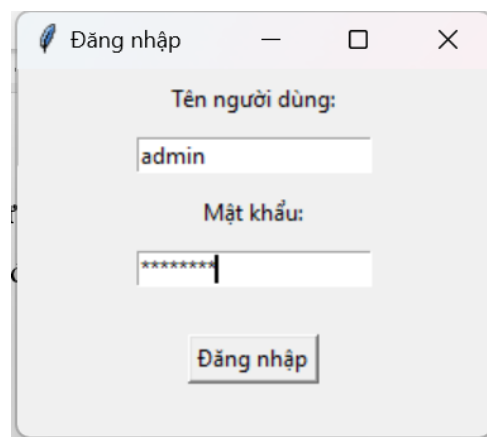
CHƯƠNG 1. Giao diện người dùng (GUI).....	3
1.  Cửa sổ Đăng nhập tài khoản: .....	3
2.  Cửa sổ Quản lí sinh viên: .....	3
CHƯƠNG 2: Chức năng cơ bản .....	4
1.  Đăng nhập:.....	4
2.  Quản lí sinh viên:.....	5
2.1.  Thêm sinh viên:.....	5
2.2.  Sửa sinh viên. ....	5
2.3.  Xóa sinh viên .....	6
2.4.  Tìm kiếm sinh viên: .....	7
CHƯƠNG 3: Mã nguồn .....	7
1.  Phần Model: .....	8
2.  Phần view .....	11
3.  Phần Controller .....	19
4.  Phần Main .....	20
CHƯƠNG 4. GitHub.....	21

## CHƯƠNG 1. Giao diện người dùng (GUI)

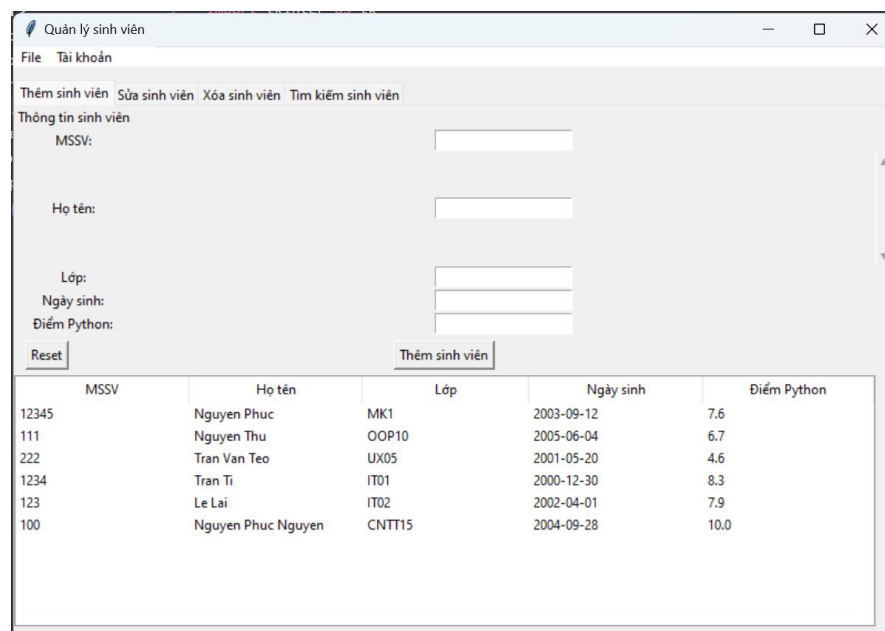
Ứng dụng sử dụng thư viện Tkinter để tạo giao diện đồ họa. Giao diện chính bao gồm 2 cửa sổ:

### 1. Cửa sổ Đăng nhập tài khoản:

Sử dụng để đăng nhập tài khoản, người dùng cần nhập tên người dùng và mật khẩu để đăng nhập



### 2. Cửa sổ Quản lý sinh viên:

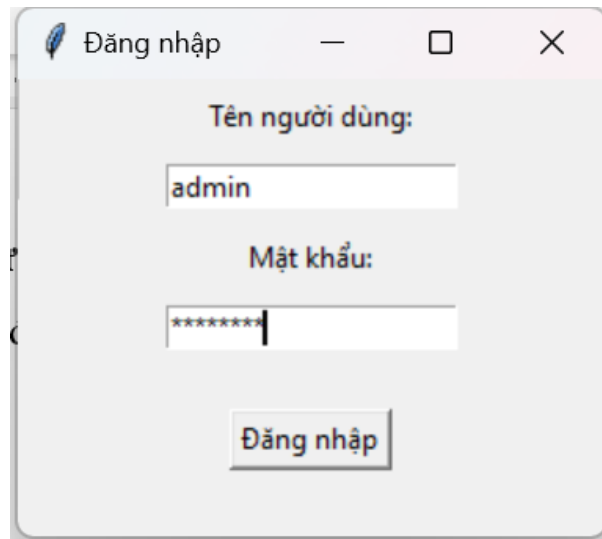


MSSV	Họ tên	Lớp	Ngày sinh	Điểm Python
12345	Nguyen Phuc	MK1	2003-09-12	7.6
111	Nguyen Thu	OOP10	2005-06-04	6.7
222	Tran Van Teo	UX05	2001-05-20	4.6
1234	Tran Ti	IT01	2000-12-30	8.3
123	Le Lai	IT02	2002-04-01	7.9
100	Nguyen Phuc Nguyen	CNTT15	2004-09-28	10.0

Đây là cửa sổ thực hiện các thao tác cơ bản như thêm, xóa, sửa và tìm kiếm sinh viên. Người dùng có thể nhập thông tin và thao tác trên các nút bấm.

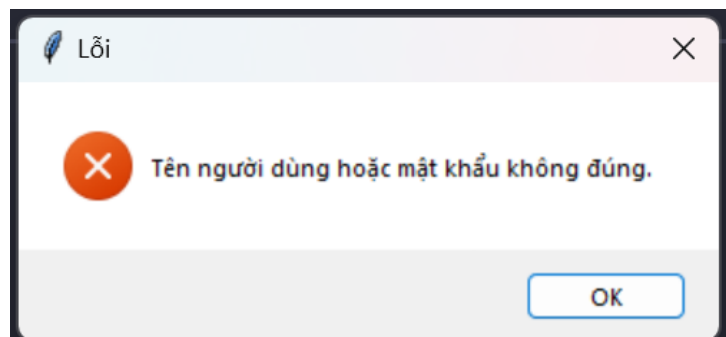
## CHƯƠNG 2: Chức năng cơ bản

### 1. Đăng nhập:



Hình 1: Cửa sổ đăng nhập.

Thực hiện kiểm tra Tên người dùng và Mật khẩu đã được nhập vào có trùng khớp với Cơ sở dữ liệu trong PostgreSQL hay không, nếu không đúng thì thông báo và buộc người dùng phải nhập lại chính xác.



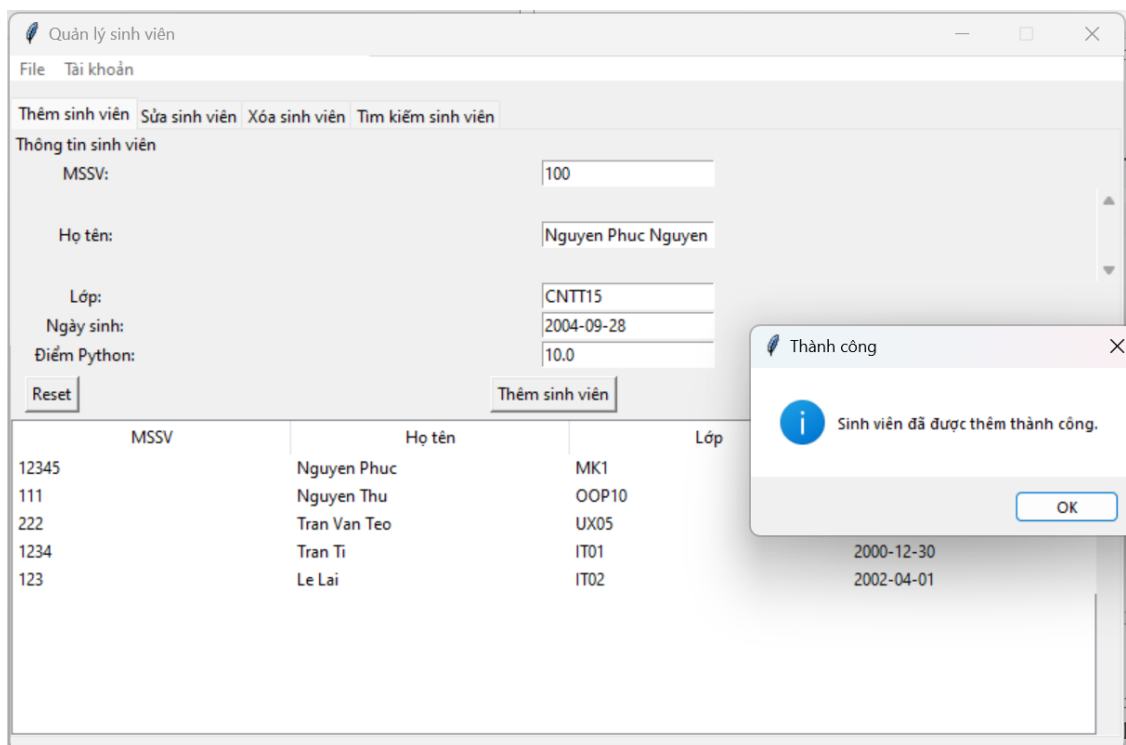
Hình 2: Thông báo nếu nhập sai.

Sau khi đã kiểm tra thông tin được nhập vào là chính xác, chương trình sẽ tắt cửa sổ đăng nhập và mở cửa sổ Quản lý sinh viên để người dùng thao tác.

## 2. Quản lí sinh viên:

Gồm các chức năng cơ bản như Thêm sinh viên, Sửa sinh viên, Xóa sinh viên và Tìm kiếm sinh viên:

### 2.1. Thêm sinh viên:



Hình 3: Thêm sinh viên thành công.

Cho phép người dung thêm mới sinh viên, cần nhập vào các thông tin cơ bản như MSSV, Họ tên, Lớp, Ngày sinh và Điểm Python. Trong đó MSSV sẽ là khóa chính để thực hiện các thao tác sau này. Sau khi kiểm tra các thông tin nhập vào đã đúng định dạng và đầy đủ hay, thì chương trình sẽ thực hiện thêm sinh viên vào CSDL và hiện thị lên trên bảng

### 2.2. Sửa sinh viên.

Ta cần nhập vào MSSV trước và “Kiểm tra” xem sinh viên có tồn tại hay không, nếu tồn tại sẽ tự động điền tất cả thông tin vào để người dung có thể sửa lại dễ dàng

Nguyễn Phúc Nguyễn  
2274802010586

MSSV	Họ tên	Lớp	Ngày sinh	Điểm Python
12345	Nguyễn Phúc	MK1	2003-09-12	7.6
222	Tran Van Teo	UX05	2001-05-20	4.6
1234	Tran Ti	IT01	2000-12-30	8.3
123	Le Lai	IT02	2002-04-01	7.9
100	Nguyễn Phúc Nguyễn	CNTT15	2004-09-28	10.0
111	Nguyễn Ha	OOP10	2005-06-04	6.7
9	Ly Van	BC12	2005-08-12	3.5

Hình 4: Nhập MSSV để kiểm tra.

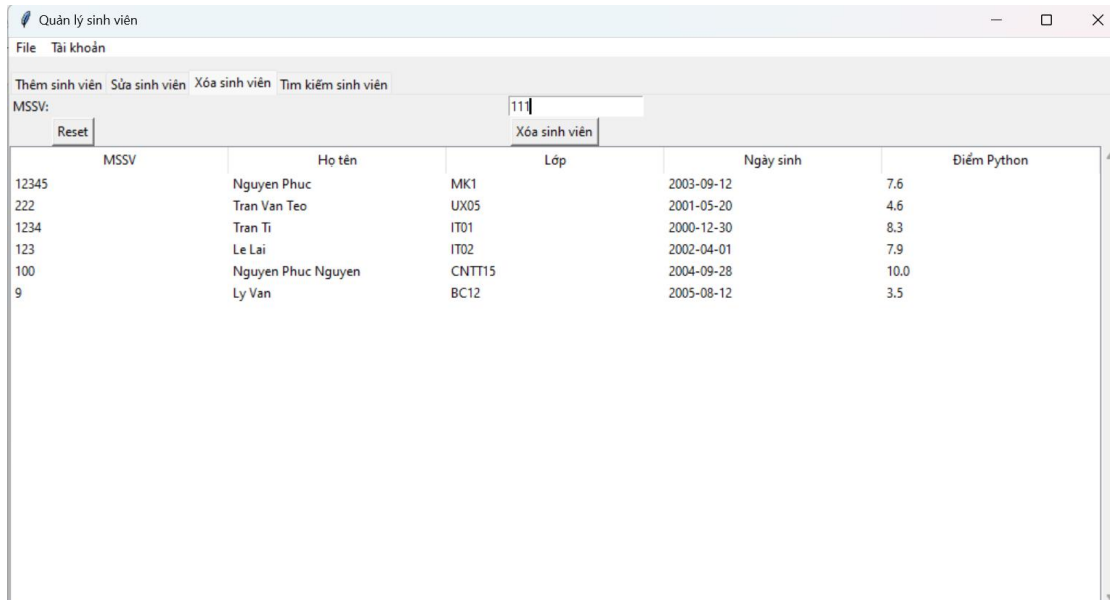
### 2.3. Xóa sinh viên

Nhập vào MSSV cần xóa và nhấn nút xóa sinh viên, chương trình sẽ xóa đi sinh viên đó trong database

MSSV	Họ tên	Lớp	Ngày sinh	Điểm Python
12345	Nguyễn Phúc	MK1	2003-09-12	7.6
222	Tran Van Teo	UX05	2001-05-20	4.6
1234	Tran Ti	IT01	2000-12-30	8.3
123	Le Lai	IT02	2002-04-01	7.9
100	Nguyễn Phúc Nguyễn	CNTT15	2004-09-28	10.0
9	Ly Van	BC12	2005-08-12	3.5
111	Nguyễn Z	OOP10	2005-06-04	6.7

Hình 5: Nhập MSSV để xóa sinh viên.

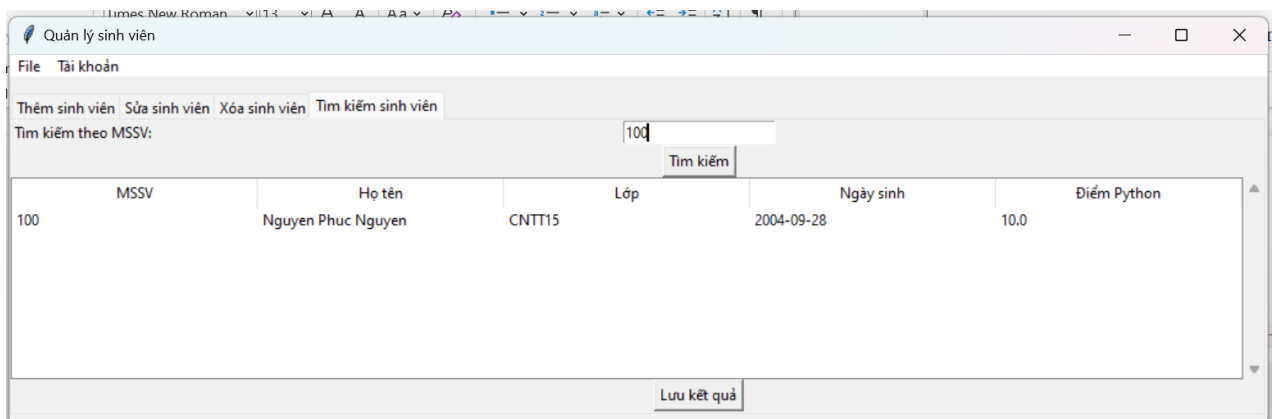
Nguyễn Phúc Nguyễn  
2274802010586



Hình 6: Kết quả sau khi xóa thành công.

#### 2.4. Tìm kiếm sinh viên:

Nhập MSSV của sinh viên cần tìm kiếm, chương trình sẽ hiển thị thông tin của sinh viên đó lên treeview



Hình 7: Nhập MSSV để tìm kiếm và hiển thị kết quả.

## CHƯƠNG 3: Mã nguồn

Tổng quan thư mục chương trình: Được xây dựng trên mô hình MVC

Bao gồm các file:

- model.py: phần xử lý dữ liệu

- controller.py: phần điều phối giữa Model và View
- view.py: phần giao diện người dùng
- main.py: phần chạy chương trình

1. Phần Model:

```
import psycopg2
from psycopg2 import sql

class DbConn:
    def __init__(self, database="student_management", table="students",
user="postgres", password="123456789", host="localhost", port="5432"):
        self.database = database
        self.table = table
        self.user = user
        self.password = password
        self.host = host
        self.port = port
        self.conn = None
        self.cur = None

    def __enter__(self):
        try:
            self.conn = psycopg2.connect(
                database=self.database,
                user=self.user,
                password=self.password,
                host=self.host,
                port=self.port
            )
            self.cur = self.conn.cursor()
        except Exception as ex:
            print(f"Error connecting to database: {ex}")
            raise ex
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        if self.cur:
            self.cur.close()
        if self.conn:
            self.conn.close()

    def check_student_exists(self, mssv):
```



```
        query = sql.SQL("SELECT * FROM {table} WHERE mssv = %s").format(
            table=sql.Identifier(self.table)
        )
        self.cur.execute(query, (mssv,))
        return self.cur.fetchone() is not None

    def select(self, columns=None, **conditions):
        if columns is None:
            columns = ['*']
        columns_str = ', '.join(columns)
        query = f'SELECT {columns_str} FROM "{self.table}"'
        if conditions:
            condition_clauses = [f'"{key}" = %s' for key in conditions.keys()]
            query += ' WHERE ' + ' AND '.join(condition_clauses)
        with self.conn.cursor() as cursor:
            cursor.execute(query, tuple(conditions.values()))
            return cursor.fetchall()

    def insert(self, **kwargs):
        mssv = kwargs.get('mssv')
        if mssv and self.check_student_exists(mssv):
            print(f"Sinh viên với MSSV {mssv} đã tồn tại.")
            return False

        try:
            columns = kwargs.keys()
            values = kwargs.values()
            query = sql.SQL("INSERT INTO {table} ({fields}) VALUES
({placeholders})").format(
                table=sql.Identifier(self.table),
                fields=sql.SQL(', ').join(map(sql.Identifier, columns)),
                placeholders=sql.SQL(', ').join(sql.Placeholder() *
len(columns))
            )
            self.cur.execute(query, tuple(values))
            self.conn.commit()
            print("Insert thành công")
            return True
        except Exception as ex:
            self.conn.rollback()
            print(f"Error during insert: {ex}")
            return False
```

```
def update(self, update_data, **conditions):
    if not conditions:
        raise ValueError("No conditions provided for update.")

    mssv = conditions.get('mssv')
    if mssv and not self.check_student_exists(mssv):
        print(f"Sinh viên với MSSV {mssv} không tồn tại.")
        return False

    updates = [sql.SQL("{} = {}".format(sql.Identifier(k),
sql.Placeholder())) for k in update_data.keys()]
    conds = [sql.SQL("{} = {}".format(sql.Identifier(k),
sql.Placeholder())) for k in conditions.keys()]

    query = sql.SQL("UPDATE {table} SET {updates} WHERE {conds}").format(
        table=sql.Identifier(self.table),
        updates=sql.SQL(", ").join(updates),
        conds=sql.SQL(" AND ").join(conds)
    )

    try:
        self.cur.execute(query, tuple(update_data.values()) +
tuple(conditions.values()))
        self.conn.commit()
        print("Update thành công")
        return True
    except Exception as ex:
        self.conn.rollback()
        print(f"Error during update: {ex}")
        return False

def delete(self, **conditions):
    if not conditions:
        raise ValueError("No conditions provided for deletion.")

    mssv = conditions.get('mssv')
    if mssv and not self.check_student_exists(mssv):
        print(f"Sinh viên với MSSV {mssv} không tồn tại.")
        return False

    conds = [sql.SQL("{} = {}".format(sql.Identifier(k),
sql.Placeholder())) for k in conditions.keys()]
    query = sql.SQL("DELETE FROM {table} WHERE {conds}").format(
```

```
        table=sql.Identifier(self.table),
        conds=sql.SQL(" AND ").join(conds)
    )
    self.cur.execute(query, tuple(conditions.values()))
    self.conn.commit()
    print("Delete thành công")
    return True

def check_login(self, username, password):
    query = sql.SQL("SELECT * FROM users WHERE username = %s AND password = %s")
    try:
        self.cur.execute(query, (username, password))
        user = self.cur.fetchone()
        if user:
            print("Login thành công")
            return True
        else:
            print("Sai tên đăng nhập hoặc mật khẩu")
            return False
    except Exception as ex:
        print(f"Error during login: {ex}")
        return False
```

## 2. Phần view

```
import tkinter as tk
from tkinter import ttk, messagebox
import controller
from tkinter import filedialog

class LoginView:
    def __init__(self, master, on_login_success):
        self.master = master
        self.on_login_success = on_login_success
        self.setup_login_view()
        self.master.title("Đăng nhập")
    def setup_login_view(self):
        self.master.geometry("250x200")
        tk.Label(self.master, text="Tên người dùng:").pack(pady=5)
        self.username_entry = tk.Entry(self.master)
        self.username_entry.pack(pady=5)
```

```
tk.Label(self.master, text="Mật khẩu:").pack(pady=5)
self.password_entry = tk.Entry(self.master, show='*')
self.password_entry.pack(pady=5)

self.login_button = tk.Button(self.master, text="Đăng nhập",
command=self.login)
self.login_button.pack(pady=20)

def login(self):
    username = self.username_entry.get()
    password = self.password_entry.get()

    with controller.DbConn() as db:
        if db.check_login(username, password):
            self.on_login_success() # Gọi hàm khi đăng nhập thành công
        else:
            messagebox.showerror("Lỗi", "Tên người dùng hoặc mật khẩu không
đúng.")

class StudentManagementApp:
    def __init__(self, root, logout_callback):
        self.root = root
        self.logout_callback = logout_callback
        self.root.title("Quản lý sinh viên")
        self.root.geometry("")
        # Tạo thanh menu
        self.menu_bar = tk.Menu(self.root)
        self.root.config(menu=self.menu_bar)

        # Thêm mục "File"
        file_menu = tk.Menu(self.menu_bar, tearoff=0)
        file_menu.add_command(label="Thoát", command=self.root.quit)
        self.menu_bar.add_cascade(label="File", menu=file_menu)

        # Thêm mục "Tài khoản"
        account_menu = tk.Menu(self.menu_bar, tearoff=0)
        account_menu.add_command(label="Đăng xuất",
command=self.logout_callback)
        self.menu_bar.add_cascade(label="Tài khoản", menu=account_menu)

        # Tạo Notebook để chứa các tab
```

```
self.notebook = ttk.Notebook(root)
self.notebook.pack(pady=10, expand=True)

# Thêm các tab
self.add_tab = ttk.Frame(self.notebook)
self.edit_tab = ttk.Frame(self.notebook)
self.delete_tab = ttk.Frame(self.notebook)
self.search_tab = ttk.Frame(self.notebook)

self.notebook.add(self.add_tab, text="Thêm sinh viên")
self.notebook.add(self.edit_tab, text="Sửa sinh viên")
self.notebook.add(self.delete_tab, text="Xóa sinh viên")
self.notebook.add(self.search_tab, text="Tìm kiếm sinh viên")

self.setup_add_tab()
self.setup_edit_tab()
self.setup_delete_tab()
self.setup_search_tab()

def create_treeview(self, parent, columns):
    # Tạo Treeview để hiển thị kết quả
    treeview = ttk.Treeview(parent, columns=columns, show="headings",
height=10)

    # Đặt tên cho các cột
    for col in columns:
        treeview.heading(col, text=col)

    treeview.grid(row=2, column=0, columnspan=2, sticky="nsew")

    # Thêm Scrollbar
    scrollbar = ttk.Scrollbar(parent, orient="vertical",
command=treeview.yview)
    scrollbar.grid(row=2, column=2, sticky="ns")
    treeview.configure(yscrollcommand=scrollbar.set)

    # Đặt grid cho các hàng và cột
    parent.grid_rowconfigure(2, weight=1)
    parent.grid_columnconfigure(1, weight=1)

    return treeview
```

```
def setup_add_tab(self):

    tk.Label(self.add_tab, text="Thông tin sinh viên").grid(row=0,
column=0)
    # Sử dụng hàm tiện ích để tạo Treeview và Scrollbar
    columns = ("MSSV", "Họ tên", "Lớp", "Ngày sinh", "Điểm Python")
    self.add_tree = self.create_treeview(self.add_tab, columns)
    self.add_tree.grid(column=0, row =7)
    self.load_all_students()

    tk.Label(self.add_tab, text="MSSV:").grid(row=1, column=0)
    tk.Label(self.add_tab, text="Họ tên:").grid(row=2, column=0)
    tk.Label(self.add_tab, text="Lớp:").grid(row=3, column=0)
    tk.Label(self.add_tab, text="Ngày sinh:").grid(row=4, column=0)
    tk.Label(self.add_tab, text="Điểm Python:").grid(row=5, column=0)

    self.mssv_entry = tk.Entry(self.add_tab)
    self.name_entry = tk.Entry(self.add_tab)
    self.class_entry = tk.Entry(self.add_tab)
    self.birthday_entry = tk.Entry(self.add_tab)
    self.python_score_entry = tk.Entry(self.add_tab)

    self.mssv_entry.grid(row=1, column=1)
    self.name_entry.grid(row=2, column=1)
    self.class_entry.grid(row=3, column=1)
    self.birthday_entry.grid(row=4, column=1)
    self.python_score_entry.grid(row=5, column=1)

    self.add_button = tk.Button(self.add_tab, text="Thêm sinh viên",
command=self.add_student)
    self.add_button.grid(row=6, columnspan=2, padx=10, pady=5)

    self.reset_button = tk.Button(self.add_tab, text="Reset",
command=self.load_all_students)
    self.reset_button.grid(row=6, columnspan=2, padx=10, pady=5,
sticky="w")

def add_student(self):
    mssv = self.mssv_entry.get()
    fullname = self.name_entry.get()
    class_name = self.class_entry.get()
```

```
birthday = self.birthday_entry.get()
python_score = self.python_score_entry.get()

with controller.DbConn() as db:
    if db.insert(mssv=mssv, fullname=fullname, class_name=class_name,
birthday=birthday, python_score=python_score):
        messagebox.showinfo("Thành công", "Sinh viên đã được thêm thành
công.")
        self.load_all_students()
    else:
        messagebox.showerror("Lỗi", "Sinh viên với MSSV đã tồn tại.")

def setup_edit_tab(self):
    tk.Label(self.edit_tab, text="MSSV:").grid(row=0, column=0)
    self.edit_mssv_entry = tk.Entry(self.edit_tab)
    self.edit_mssv_entry.grid(row=0, column=1)

    columns = ("MSSV", "Họ tên", "Lớp", "Ngày sinh", "Điểm Python")
    self.add_tree = self.create_treeview(self.edit_tab, columns)
    self.add_tree.grid(column=0, row =8)
    self.load_all_students()

    self.check_button = tk.Button(self.edit_tab, text="Kiểm tra",
command=self.check_student)
    self.check_button.grid(row=5, columnspan=2, padx= 5, pady=5)

    tk.Label(self.edit_tab, text="Họ tên:").grid(row=1, column=0)
    tk.Label(self.edit_tab, text="Lớp:").grid(row=2, column=0)
    tk.Label(self.edit_tab, text="Ngày sinh:").grid(row=3, column=0)
    tk.Label(self.edit_tab, text="Điểm Python:").grid(row=4, column=0)

    self.edit_name_entry = tk.Entry(self.edit_tab, state='readonly')
    self.edit_class_entry = tk.Entry(self.edit_tab, state='readonly')
    self.edit_birthday_entry = tk.Entry(self.edit_tab, state='readonly')
    self.edit_python_score_entry = tk.Entry(self.edit_tab,
state='readonly')

    self.edit_name_entry.grid(row=1, column=1)
    self.edit_class_entry.grid(row=2, column=1)
    self.edit_birthday_entry.grid(row=3, column=1)
    self.edit_python_score_entry.grid(row=4, column=1)
```

```
        self.update_button = tk.Button(self.edit_tab, text="Cập nhật",
command=self.update_student)
        self.update_button.grid(row=6, columnspan=2, padx= 5, pady=5)

        self.reset_button = tk.Button(self.edit_tab, text="Reset",
command=self.load_all_students)
        self.reset_button.grid(row=7, columnspan=2, padx=5, pady=5)

def check_student(self):
    mssv = self.edit_mssv_entry.get()
    with controller.DbConn() as db:
        student = db.select(mssv=mssv)
        if student:
            self.edit_name_entry.config(state='normal')
            self.edit_class_entry.config(state='normal')
            self.edit_birthday_entry.config(state='normal')
            self.edit_python_score_entry.config(state='normal')

            self.edit_name_entry.delete(0, tk.END)
            self.edit_name_entry.insert(0, student[0][1]) # Họ tên
            self.edit_class_entry.delete(0, tk.END)
            self.edit_class_entry.insert(0, student[0][2]) # Lớp
            self.edit_birthday_entry.delete(0, tk.END)
            self.edit_birthday_entry.insert(0, student[0][3]) # Ngày sinh
            self.edit_python_score_entry.delete(0, tk.END)
            self.edit_python_score_entry.insert(0, student[0][4]) # Điểm

        else:
            messagebox.showerror("Lỗi", "Sinh viên không tồn tại.")

def update_student(self):
    mssv = self.edit_mssv_entry.get()
    update_data = {
        'fullname': self.edit_name_entry.get(),
        'class_name': self.edit_class_entry.get(),
        'birthday': self.edit_birthday_entry.get(),
        'python_score': self.edit_python_score_entry.get(),
    }

    with controller.DbConn() as db:
        if db.update(update_data, mssv=mssv):
```

Python



```
        messagebox.showinfo("Thành công", "Thông tin sinh viên đã được  
cập nhật.")  
        self.load_all_students()  
    else:  
        messagebox.showerror("Lỗi", "Cập nhật không thành công.")  
  
    def setup_delete_tab(self):  
        tk.Label(self.delete_tab, text="MSSV:").grid(row=0, column=0)  
        self.delete_mssv_entry = tk.Entry(self.delete_tab)  
        self.delete_mssv_entry.grid(row=0, column=1)  
  
        columns = ("MSSV", "Họ tên", "Lớp", "Ngày sinh", "Điểm Python")  
        self.add_tree = self.create_treeview(self.delete_tab, columns)  
        self.add_tree.grid(column=0, row =2)  
        self.load_all_students()  
  
        self.delete_button = tk.Button(self.delete_tab, text="Xóa sinh viên",  
command=self.delete_student)  
        self.delete_button.grid(row=1, columnspan=2)  
  
        self.reset_button = tk.Button(self.delete_tab, text="Reset",  
command=self.load_all_students)  
        self.reset_button.grid(row=1, column=1, sticky="w")  
  
    def delete_student(self):  
        mssv = self.delete_mssv_entry.get()  
  
        with controller.DbConn() as db:  
            if db.delete(mssv=mssv):  
                messagebox.showinfo("Thành công", "Sinh viên đã được xóa thành  
công.")  
                self.load_all_students()  
            else:  
                messagebox.showerror("Lỗi", "Sinh viên không tồn tại.")  
  
    def setup_search_tab(self):  
        tk.Label(self.search_tab, text="Tìm kiếm theo MSSV:").grid(row=0,  
column=0)  
        self.search_mssv_entry = tk.Entry(self.search_tab)  
        self.search_mssv_entry.grid(row=0, column=1)
```

```
        self.search_button = tk.Button(self.search_tab, text="Tìm kiếm",
command=self.search_student)
        self.search_button.grid(row=1, column=1) # Đưa nút tìm kiếm về cột 0

        # Nút lưu kết quả tìm kiếm
        self.save_button = tk.Button(self.search_tab, text="Lưu kết quả",
command=self.save_results_to_file)
        self.save_button.grid(row=3, column=1) # Đặt nút lưu ở cột 1 cùng hàng
với nút tìm kiếm

        # Sử dụng hàm tiện ích để tạo Treeview và Scrollbar
        columns = ("MSSV", "Họ tên", "Lớp", "Ngày sinh", "Điểm Python")
        self.results_tree = self.create_treeview(self.search_tab, columns)
        self.results_tree.grid(column=0, row=2, columnspan=2) # Điều chỉnh vị
trí cho Treeview

    def search_student(self):
        mssv = self.search_mssv_entry.get()

        # Kết nối với database và tìm kiếm sinh viên theo MSSV
        with controller.DbConn() as db:
            results = db.select(mssv=mssv)

        # Xóa tất cả nội dung cũ trong Treeview trước khi hiển thị kết quả
mới
        self.results_tree.delete(*self.results_tree.get_children()) # Xóa
tất cả các hàng

        # Kiểm tra và hiển thị kết quả tìm kiếm
        if results:
            for row in results:
                self.results_tree.insert('', 'end', values=row) # Thêm
từng kết quả vào Treeview

        else:
            messagebox.showinfo("Thông báo", "Không tìm thấy sinh
viên.") # Thông báo nếu không tìm thấy

    def save_results_to_file(self):
        # Mở hộp thoại để chọn vị trí và tên file
        file_name = filedialog.asksaveasfilename(defaultextension=".txt",
```

```
filetypes=[("Text files",
            "*.txt"), ("All files", "*.*")],
            title="Lưu kết quả tìm kiếm")
    if file_name: # Kiểm tra nếu người dùng không hủy bỏ
        try:
            with open(file_name, 'w') as f:
                # Giả sử bạn có một list kết quả rows từ tìm kiếm trước đó
                for row in self.results_tree.get_children():
                    row_values = self.results_tree.item(row, 'values')
                    f.write(', '.join(row_values) + '\n')
                print(f"Kết quả tìm kiếm đã được lưu vào {file_name}")
            except Exception as ex:
                print(f"Lỗi khi lưu file: {ex}")

def load_all_students(self):
    with controller.DbConn() as db:
        results = db.select() # Lấy tất cả sinh viên
        # Xóa tất cả nội dung hiện tại trong Treeview
        self.add_tree.delete(*self.add_tree.get_children())
        # Chèn từng dòng kết quả vào Treeview
        for row in results:
            self.add_tree.insert('', 'end', values=row)

def main():
    root = tk.Tk()

    def on_login_success():
        login_view.master.destroy() # Đóng cửa sổ đăng nhập
        StudentManagementApp(root, logout_callback)

    logout_callback = lambda: print("Đăng xuất thành công!") # Thay đổi theo
    yêu cầu của bạn

    login_view = LoginView(root, on_login_success)
    root.mainloop()
```

### 3. Phần Controller

```
from model import DbConn
```

```
class StudentController:
    def __init__(self, treeview):
        self.treeview = treeview

    def load_all_students(self):
        # Sử dụng DbConn để lấy dữ liệu sinh viên từ database
        with DbConn() as db:
            results = db.select() # Lấy tất cả sinh viên từ bảng
            # Xóa tất cả nội dung hiện tại trong Treeview
            self.treeview.delete(*self.treeview.get_children())
            # Chèn từng dòng kết quả vào Treeview
            for row in results:
                self.treeview.insert('', 'end', values=row)

    def insert_student(self, **student_data):
        with DbConn() as db:
            success = db.insert(**student_data)
            return success

    def update_student(self, update_data, **conditions):
        with DbConn() as db:
            success = db.update(update_data, **conditions)
            return success

    def delete_student(self, **conditions):
        with DbConn() as db:
            success = db.delete(**conditions)
            return success
```

#### 4. Phần Main

```
import tkinter as tk
from view import LoginView, StudentManagementApp

class MainApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Quản lý sinh viên")

        # Khởi tạo trang đăng nhập
        self.login_view = LoginView(self.root, self.show_student_management)
```

```
def show_student_management(self):
    # Xóa giao diện hiện tại và hiển thị trang quản lý sinh viên
    self.clear_widgets() # Gọi hàm để xóa widget cũ
    # Khởi tạo trang quản lý sinh viên
    self.student_management_app = StudentManagementApp(self.root,
self.logout)

def show_login_view(self):
    # Xóa giao diện hiện tại và hiển thị trang đăng nhập
    self.clear_widgets() # Gọi hàm để xóa widget cũ
    self.login_view = LoginView(self.root, self.show_student_management)

def logout(self):
    # Thực hiện các thao tác khi người dùng đăng xuất
    print("Người dùng đã đăng xuất")
    self.show_login_view() # Quay về trang đăng nhập

def clear_widgets(self):
    # Hàm để xóa tất cả các widget trên giao diện hiện tại
    for widget in self.root.winfo_children():
        widget.destroy()

if __name__ == "__main__":
    root = tk.Tk()
    app = MainApp(root)
    root.mainloop()

5.
```

## CHƯƠNG 4. GitHub

Link github của dự án: [https://github.com/PhucNguyenne/Python\\_NC\\_Bai2](https://github.com/PhucNguyenne/Python_NC_Bai2)