TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**

A red logo with white text

Description automatically generated

**BÁO CÁO ĐỒ ÁN MÔN HỌC HK241**

**LẬP TRÌNH PYTHON NÂNG CAO**

**XÂY DỰNG WEBSITE QUẢN LÍ SINH VIÊN BẰNG FLASK**

**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 website 3](#_Toc182259135)

[CHƯƠNG 2: Chức năng cơ bản 6](#_Toc182259136)

[1. Đăng nhập: 6](#_Toc182259137)

[2. Quản lí sinh viên: 7](#_Toc182259138)

[2.1. Thêm sinh viên: 7](#_Toc182259139)

[2.2. Sửa sinh viên. 8](#_Toc182259140)

[2.3. Xóa sinh viên 9](#_Toc182259141)

[2.4. Tìm kiếm sinh viên: 10](#_Toc182259142)

[CHƯƠNG 3: Mã nguồn 10](#_Toc182259143)

[1. Phần Model: 11](#_Toc182259144)

[2. Phần Controller 13](#_Toc182259145)

[3. Phần App: 15](#_Toc182259146)

[CHƯƠNG 4. GitHub 18](#_Toc182259147)

# CHƯƠNG 1. Giao diện website

Website sử dụng thư viện Flask để tạo giao diện người dùng và được xây dựng trên môi trường ảo (venv) đóng gói Docker. Giao diện chính bao gồm:

Trang Đăng nhập tài khoản:

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

A screenshot of a computer

Description automatically generated

Hình : Trang đăng nhập.

Sau khi đăng nhập thành công, chương trình sẽ chuyển hướng đến dang dashboard, ở đây

Ta có thể thực hiện các chức năng chính như thêm, tìm kiếm, xóa và sửa sinh viên.

A screenshot of a computer

Description automatically generated

Hình : Trang dashboard.

Trang thêm sinh viên: Nhập vào các trường thông tin cần thiết vào nhấn nút “Thêm” để thêm mới một sinh viên vào cơ sở dữ liệu.

A screenshot of a computer

Description automatically generated

Hình :Trang thêm sinh viên.

Trang tìm kiếm sinh viên: Nhập vào Mã số sinh viên để thực hiện tìm kiếm.

A screenshot of a computer

Description automatically generated

Hình :Trang tìm kiếm sinh viên.

Trang xóa sinh viên: Tương tự như tìm kiếm sinh viên, ta cần nhập Mã số sinh viên để xóa đi sinh viên đó ra khỏi cơ sở dữ liệu.

A screenshot of a computer

Description automatically generated

Hình : Trang xóa sinh viên.

Trang Sửa sinh viên: Nhập Mã số sinh viên để kiểm tra lại xem sinh viên có tồn tại không, nếu có thì sẽ tự động ghi sẵn thông tin sinh viên đó dựa trên Mã số sinh viên để người dùng có thể dễ dàng chỉnh sửa.

A screenshot of a login form

Description automatically generated

Hình : Trang Sửa sinh viên

# CHƯƠNG 2: Chức năng cơ bản

## Đăng nhập:

A screenshot of a computer

Description automatically generated

Hình : Trang đăng nhập.

Thực hiện kiểm tra Tên người dung và Mật khẩu đã được nhập vào có trùng khớp với Cơ sở dữ liệu trong PostgreSQL hay không, nếu không đúng sẽ truy cập vào dashboard.

A screenshot of a computer

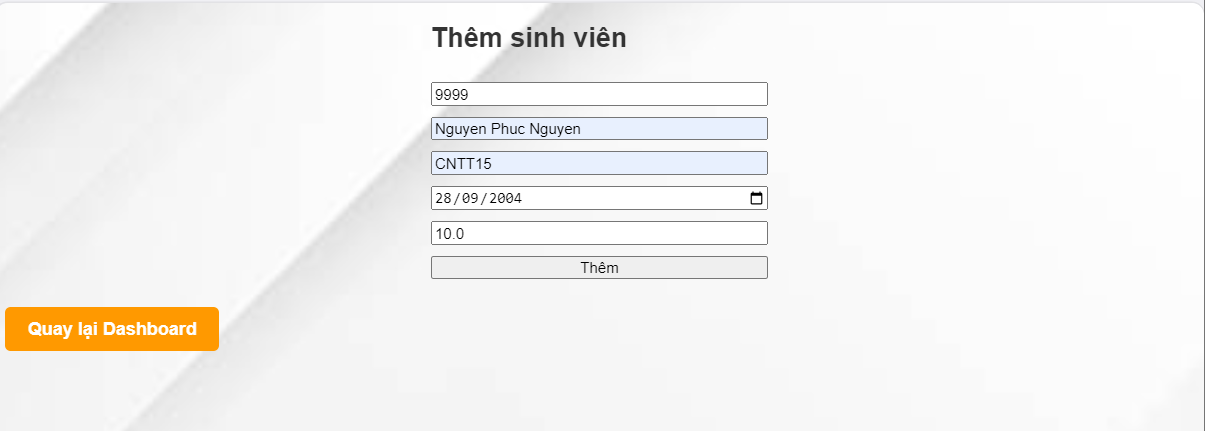
Description automatically generated

Hình : Trang dashboard.

## Quản lí sinh viên:

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

### Thêm sinh viên:



Hình : Thêm sinh viên mới vào.

A screenshot of a computer

Description automatically generated

Hình : Sau khi 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 them sinh viên vào CSDL và hiện thị lên trên trang Dashboard.

### 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

A screenshot of a computer

Description automatically generated

Hình : Nhập MSSV để kiểm tra.

A screenshot of a computer

Description automatically generated

Hình : Sửa thông tin sinh viên thành công.

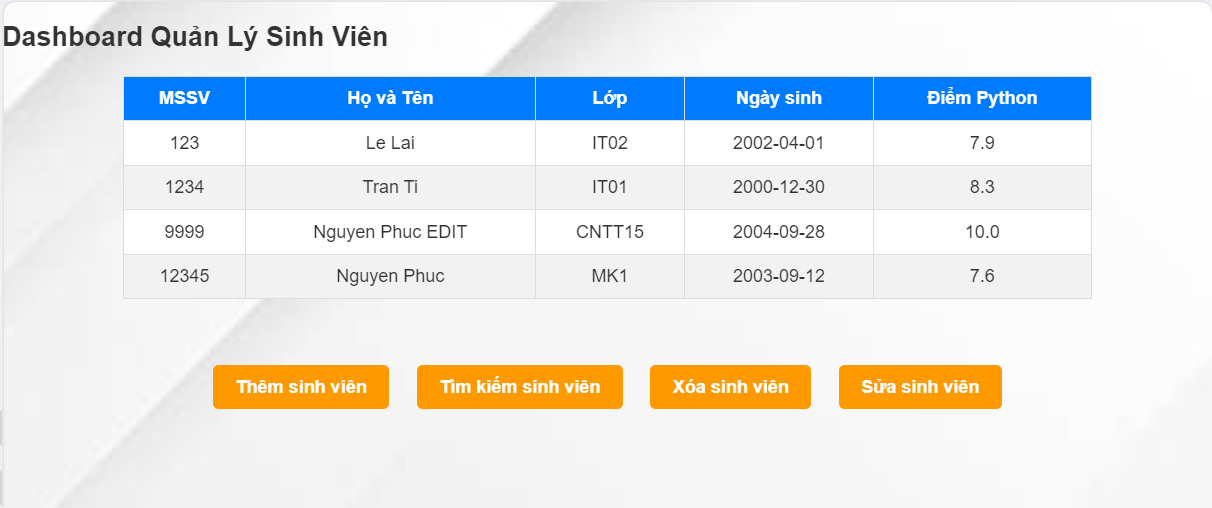
### Xóa sinh viên

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

A screenshot of a computer

Description automatically generated

Hình :Nhập MSSV để xóa sinh viên.



Hình : Kết quả sau khi xóa thành công.

### Tìm kiếm sinh viên:

Nhập MSSV của sinh viên cần tìm kiếm, chương trình sẽ hiện thị thông tin của sinh viên đó lên bảng thông tin.

A screenshot of a computer

Description automatically generated

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

# CHƯƠNG 3: Mã nguồn

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

Bao gồm các file chính:

* model.py: phần xử lý dữ liệu
* controller.py: phần điều phối giữa Model và App
* app.py: xây dựng các route và xử lí back-end cho trang web bằng flask
* login.html, dashboard.html, add\_student.html, delete\_student.html, edit\_student.html, search\_student: xây dựng giao diện web bằng html, kết nối với app.py giúp lấy và truyền dữ liệu cho các chức năng đặc trưng của từng trang web.
* style.css: thiết kế lại giao diện web trực quan và dễ nhìn.

### Phần Model:

import psycopg2

from psycopg2 import sql

class DbConn:

    def \_\_init\_\_(*self*, *connection*=None, *database*="student\_management", *user*="postgres", *password*="123456789", *host*="localhost", *port*="5432"):

        # Nếu đã có kết nối, sử dụng kết nối đó, nếu không tạo kết nối mới

        if *connection*:

*self*.conn = *connection*

        else:

*self*.conn = psycopg2.connect(

*database*=*database*,

*user*=*user*,

*password*=*password*,

*host*=*host*,

*port*=*port*

            )

*self*.cursor = *self*.conn.cursor()

    def \_\_enter\_\_(*self*):

        return *self*

    def \_\_exit\_\_(*self*, *exc\_type*, *exc\_value*, *traceback*):

*self*.cursor.close()

*self*.conn.close()

    def check\_login(*self*, *username*, *password*):

        query = "SELECT \* FROM users WHERE username = %s AND password = %s"

*self*.cursor.execute(query, (*username*, *password*))

        return *self*.cursor.fetchone() is not None

    def insert(*self*, \*\**kwargs*):

        try:

            query = sql.SQL("INSERT INTO students (mssv, fullname, class\_name, birthday, python\_score) VALUES (%s, %s, %s, %s, %s)")

*self*.cursor.execute(query, tuple(*kwargs*.values()))

*self*.conn.commit()

            return True

        except Exception as ex:

*self*.conn.rollback()

            print(f"Error inserting student: {ex}")

            return False

    def delete(*self*, \*\**conditions*):

        try:

            query = "DELETE FROM students WHERE mssv = %s"

*self*.cursor.execute(query, (*conditions*['mssv'],))

*self*.conn.commit()

            return True

        except Exception as ex:

*self*.conn.rollback()

            print(f"Error deleting student: {ex}")

            return False

    def select(*self*, \*\**conditions*):

        try:

            query = "SELECT mssv, fullname, class\_name, birthday, python\_score FROM students WHERE mssv = %s"

*self*.cursor.execute(query, (*conditions*['mssv'],))

            rows = *self*.cursor.fetchall()

            # Chuyển kết quả thành danh sách các dictionary

            results = []

            for row in rows:

                student = {

                    'mssv': row[0],

                    'fullname': row[1],

                    'class\_name': row[2],

                    'birthday': row[3],

                    'python\_score': row[4]

                }

                results.append(student)

            return results

        except Exception as e:

            print("Error fetching student:", e)

            return []

    def update\_student(*self*, *mssv*, *fullname*, *class\_name*, *birthday*, *python\_score*):

        try:

            sql = """

            UPDATE students

            SET fullname = %s, class\_name = %s, birthday = %s, python\_score = %s

            WHERE mssv = %s

            """

*self*.cursor.execute(sql, (*fullname*, *class\_name*, *birthday*, *python\_score*, *mssv*))

*self*.conn.commit()

            return True

        except Exception as e:

            print(f"Error updating student: {e}")

*self*.conn.rollback()

            return False

    def get\_student\_by\_mssv(*self*, *mssv*):

        try:

            sql = "SELECT \* FROM students WHERE mssv = %s"

*self*.cursor.execute(sql, (*mssv*,))

            return *self*.cursor.fetchone()

        except Exception as e:

            print(f"Error fetching student: {e}")

            return None

### Phần Controller

from model import DbConn

import psycopg2

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

# Thêm hàm này vào controller.py

def get\_top\_students():

    conn = psycopg2.connect(

*host*="localhost",

*database*="student\_management",

*user*="postgres",

*password*="123456789"

    )

    cursor = conn.cursor()

    query = """

        SELECT mssv, fullname, class\_name, birthday, python\_score

        FROM students

        ORDER BY mssv

        LIMIT 10;

    """

    cursor.execute(query)

    students = cursor.fetchall()

    cursor.close()

    conn.close()

    return students

### Phần App:

from flask import Flask, render\_template, request, redirect, url\_for, flash, session

from model import DbConn

app = Flask(\_\_name\_\_)

app.secret\_key = "pn"

# Trang đăng nhập

@app.route('/', *methods*=['GET', 'POST'])

def login():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        with DbConn() as db:

            if db.check\_login(username, password):

                session['username'] = username

                return redirect(url\_for('dashboard'))

            else:

                flash("Sai tên đăng nhập hoặc mật khẩu")

    return render\_template('login.html')

# Dashboard

@app.route('/dashboard')

def dashboard():

    if 'username' not in session:

        return redirect(url\_for('login'))

    from controller import get\_top\_students

    students = get\_top\_students()

    return render\_template('dashboard.html', *students*=students)

# Thêm sinh viên

@app.route('/add', *methods*=['GET', 'POST'])

def add\_student():

    if request.method == 'POST':

        data = {

            'mssv': request.form['mssv'],

            'fullname': request.form['fullname'],

            'class\_name': request.form['class\_name'],

            'birthday': request.form['birthday'],

            'python\_score': request.form['python\_score']

        }

        with DbConn() as db:

            if db.insert(\*\*data):

                flash("Thêm sinh viên thành công")

            else:

                flash("Sinh viên đã tồn tại")

        return redirect(url\_for('dashboard'))

    return render\_template('add\_student.html')

# Xóa sinh viên

@app.route('/delete\_student', *methods*=['GET', 'POST'])

def delete\_student():

    if request.method == 'POST':

        student\_id = request.form['student\_id']

        with DbConn() as db:

            if db.delete(*mssv*=student\_id):

                flash("Xóa sinh viên thành công")

            else:

                flash("Sinh viên không tồn tại")

        return redirect(url\_for('dashboard'))

    return render\_template('delete\_student.html')

# Tìm kiếm sinh viên

@app.route('/search', *methods*=['GET', 'POST'])

def search\_student():

    results = []

    if request.method == 'POST':

        keyword = request.form['keyword']

        with DbConn() as db:

            results = db.select(*mssv*=keyword)

    return render\_template('search\_student.html', *results*=results)

# Sửa sinh viên

@app.route('/edit\_student', *methods*=['GET', 'POST'])

def edit\_student():

    student = None

    if request.method == 'POST' and 'search' in request.form:

        # Lấy MSSV người dùng nhập vào

        mssv = request.form['mssv']

        # Tìm sinh viên từ cơ sở dữ liệu

        with DbConn() as db:

            student = db.select(*mssv*=mssv)  # Giả sử bạn có phương thức select để tìm sinh viên

            if student:

                student = student[0]  # Chỉ lấy sinh viên đầu tiên

            else:

                flash("Sinh viên không tồn tại")

                return redirect(url\_for('edit\_student'))  # Quay lại trang sửa nếu không tìm thấy sinh viên

    elif request.method == 'POST' and 'update' in request.form:

        # Khi người dùng gửi thông tin cập nhật

        mssv = request.form['mssv']

        fullname = request.form['fullname']

        class\_name = request.form['class\_name']

        birthday = request.form['birthday']

        python\_score = request.form['python\_score']

        # Cập nhật thông tin sinh viên trong cơ sở dữ liệu

        with DbConn() as db:

            # Sử dụng phương thức update\_student của DbConn để cập nhật thông tin

            if db.update\_student(mssv, fullname, class\_name, birthday, python\_score):

                flash("Cập nhật sinh viên thành công")

                return redirect(url\_for('dashboard'))  # Quay lại dashboard sau khi thành công

            else:

                flash("Lỗi cập nhật sinh viên")

                return redirect(url\_for('edit\_student'))  # Quay lại trang sửa nếu có lỗi

    return render\_template('edit\_student.html', *student*=student)

@app.route('/update\_student', *methods*=['GET', 'POST'])

def update\_student():

    student = None

    if request.method == 'POST':

        mssv = request.form.get('mssv')

        # Tìm kiếm sinh viên theo MSSV

        if mssv:

            with DbConn() as db:

                student\_data = db.get\_student\_by\_mssv(mssv)

                if student\_data:

                    student = {

                        'mssv': student\_data[0],

                        'fullname': student\_data[1],

                        'class\_name': student\_data[2],

                        'birthday': student\_data[3],

                        'python\_score': student\_data[4]

                    }

    # Render lại trang với thông tin sinh viên nếu tìm thấy

    return render\_template('update\_student.html', *student*=student)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(*debug*=True)

# CHƯƠNG 4. GitHub

Link github của dự án: <https://github.com/PhucNguyenne/Python_NC_Bai3>