TRƯỜNG ĐẠI HỌC VĂN LANG

KHOA CÔNG NGHỆ THÔNG TIN

TRƯỜNG ĐẠI HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**

Ảnh có chứa văn bản, hình mẫu

Mô tả được tạo tự động

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

LẬP TRÌNH PYTHON NÂNG CAO

***Chủ đề:***

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

**SVTH: ĐẶNG NHẬT BẢO**

**MSSV: 2174802010301**

**GVHD: ThS Trần Kim Mỹ Vân**

TP. Hồ Chí Minh – năm 2024

TP. Hồ Chí Minh – năm 2023

TP. Hồ Chí Minh – năm 2023

TP. Hồ Chí Minh – năm 2023

TP. Hồ Chí Minh, ngày… tháng … năm 2023

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

Nguyễn Ngọc Thảo Vy

**MỤC LỤC**

[**CHƯƠNG 1: TỔNG QUAN 3**](#_Toc182296259)

[**1.1. Tổng quan 3**](#_Toc182296260)

[**1.2. Mục Tiêu Phát Triển Ứng Dụng 3**](#_Toc182296261)

[**1.3. Các Tính Năng Chi Tiết 3**](#_Toc182296262)

[**1.4. Công Nghệ Sử Dụng 4**](#_Toc182296263)

[**CHƯƠNG 2: KẾT LUẬN 5**](#_Toc182296264)

[**CHƯƠNG 3: DEMO 6**](#_Toc182296265)

[**3.1. Thư mục chứa sản phẩm 6**](#_Toc182296266)

[**3.2. Mô tả demo 14**](#_Toc182296267)

[**3.3. Giao diện chương trình 14**](#_Toc182296268)

Cập nhật file WORD/PDF đồ án cuối kì

# CHƯƠNG 1: TỔNG QUAN

## 1.1. Tổng quan

Ứng dụng quản lý sinh viên là sản phẩm mà em phát triển nhằm hỗ trợ công tác quản lý thông tin sinh viên, tạo điều kiện cho việc cập nhật, tìm kiếm và xóa thông tin một cách dễ dàng và nhanh chóng. Với vai trò là một ứng dụng quản lý dữ liệu trực tuyến, ứng dụng của em hướng đến việc đơn giản hóa quy trình quản lý dữ liệu và đảm bảo tính chính xác, an toàn cho thông tin sinh viên.

Ứng dụng sử dụng ngôn ngữ lập trình Python kết hợp với Flask để xây dựng giao diện backend và PostgreSQL làm hệ quản trị cơ sở dữ liệu. Ngoài ra, HTML, CSS và JavaScript được sử dụng để tạo giao diện người dùng, giúp người dùng dễ dàng thao tác trên các trình duyệt web thông dụng.

**Các tính năng chính của ứng dụng bao gồm:**

* Đăng nhập và bảo mật phiên làm việc của người dùng.
* Thêm mới thông tin sinh viên vào cơ sở dữ liệu.
* Xóa thông tin sinh viên khỏi cơ sở dữ liệu.
* Tìm kiếm thông tin sinh viên dựa trên các tiêu chí khác nhau.
* Hiển thị thông tin sinh viên sau khi thêm hoặc xóa nhằm đảm bảo dữ liệu hiển thị luôn được cập nhật.

## 1.2. Mục Tiêu Phát Triển Ứng Dụng

**Ứng dụng được phát triển với những mục tiêu cụ thể sau:**

* Giảm thiểu công sức và thời gian khi nhập và quản lý dữ liệu sinh viên.
* Cải thiện độ chính xác của thông tin, tránh các sai sót thủ công khi quản lý trên giấy tờ.
* Cung cấp một giải pháp quản lý sinh viên an toàn, bảo mật.
* Hỗ trợ tìm kiếm thông tin nhanh chóng, dễ dàng.
* Đảm bảo cập nhật thông tin sinh viên ngay lập tức sau khi thực hiện các thao tác thêm và xóa.

## 1.3. Các Tính Năng Chi Tiết

* Đăng nhập: Đảm bảo chỉ người dùng đã xác thực mới có quyền truy cập vào hệ thống.
* Thêm sinh viên: Người dùng có thể nhập thông tin sinh viên mới bao gồm họ tên, mã số sinh viên (MSSV), ngành học và năm sinh. Sau khi thêm, hệ thống sẽ hiển thị thông báo thành công, và bảng dữ liệu sẽ tự động cập nhật để hiển thị thông tin sinh viên mới thêm.
* Xóa sinh viên: Người dùng có thể xóa sinh viên bằng cách nhập mã số sinh viên. Hệ thống sẽ xóa sinh viên và hiển thị thông báo thành công, đồng thời bảng dữ liệu tự động cập nhật sau khi xóa.
* Tìm kiếm sinh viên: Ứng dụng hỗ trợ tìm kiếm sinh viên dựa trên các trường thông tin như họ tên, MSSV, ngành, và năm sinh. Kết quả tìm kiếm được hiển thị tức thì, giúp người dùng nhanh chóng tìm được thông tin cần thiết.
* Cập nhật bảng dữ liệu tức thì: Sau mỗi thao tác thêm hoặc xóa, bảng dữ liệu sẽ được tự động tải lại để đảm bảo tính nhất quán của thông tin.

## 1.4. Công Nghệ Sử Dụng

Ứng dụng sử dụng một số công nghệ và thư viện như:

* Python và Flask: Được sử dụng làm nền tảng chính để xây dựng backend, giúp xử lý các yêu cầu HTTP và giao tiếp với cơ sở dữ liệu.
* PostgreSQL: Đảm bảo hiệu năng cao cho việc lưu trữ và truy xuất dữ liệu.
* HTML, CSS và JavaScript: Được sử dụng để xây dựng giao diện người dùng đơn giản, thân thiện và dễ sử dụng.

# CHƯƠNG 2: KẾT LUẬN

Ứng dụng quản lý sinh viên đã hoàn thành các mục tiêu đặt ra và đạt được hiệu quả tốt trong việc quản lý thông tin sinh viên. Hệ thống có khả năng mở rộng thêm các tính năng nâng cao như chỉnh sửa thông tin sinh viên, hoặc tạo các báo cáo thống kê theo yêu cầu của người dùng. Nhờ vào việc áp dụng các công nghệ hiện đại và cách thức tổ chức dữ liệu rõ ràng, ứng dụng đảm bảo cung cấp một môi trường quản lý hiệu quả và dễ sử dụng.

**Những lợi ích và kết quả đạt được:**

Ứng dụng giúp cải thiện quy trình quản lý dữ liệu sinh viên nhờ vào khả năng lưu trữ dữ liệu tập trung và tìm kiếm nhanh chóng, giúp tiết kiệm thời gian và giảm sai sót so với cách quản lý thủ công.

**Những thách thức trong quá trình phát triển:**

* Đảm bảo tính ổn định khi kết nối cơ sở dữ liệu và xử lý các lỗi khi thực hiện các thao tác thêm hoặc xóa dữ liệu.
* Tối ưu giao diện sao cho người dùng dễ dàng thao tác, đồng thời đảm bảo tính năng bảo mật của hệ thống.

**Hướng phát triển trong tương lai:**

* Tích hợp các tính năng bổ sung như cập nhật thông tin sinh viên, sắp xếp dữ liệu theo các tiêu chí khác nhau.
* Tối ưu hóa giao diện và trải nghiệm người dùng (UI/UX).
* Xây dựng ứng dụng trên nền tảng web động để mở rộng cho nhiều người dùng hơn.

# CHƯƠNG 3: DEMO

## 3.1. Thư mục chứa sản phẩm

A screenshot of a computer

Description automatically generated

**File style.css:**

body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    color: #333;

    margin: 0;

    padding: 20px;

}

*.container* {

    max-width: 950px;

    margin: auto;

    background: white;

    padding: 20px;

    border-radius: 5px;

    box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

*.container2* {

    max-width: 600px;

    height: 310px;

    margin: auto;

    background: white;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0,0,0,0.1);

    margin-top: 220px;

    margin-right: 650px;

}

h1 {

    text-align: center;

}

label {

    display: block;

    margin: 10px 0 5px;

}

input[*type*="text"], input[*type*="password"], input[*type*="number"] {

    width: 100%;

    padding: 10px;

    margin: 5px 0 20px;

    border: 1px solid #ddd;

    border-radius: 5px;

}

button {

    background-color: #5d74c7;

    color: white;

    padding: 10px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

button*:hover* {

    background-color: #2f49aa;

}

table {

    width: 100%;

    border-collapse: collapse;

}

th, td {

    padding: 10px;

    text-align: left;

    border-bottom: 1px solid #ddd;

}

th {

    background-color: #f2f2f2;

}

*.logout-button* {

    display: inline-block;

    background-color: #ff0000;

    color: white;

    padding: 10px 15px;

    text-align: center;

    text-decoration: none;

    border-radius: 5px;

    margin-top: 20px;

}

*.logout-button:hover* {

    background-color: #ff1b1b;

}

*.flashes* {

    list-style-type: none;

    padding: 0;

    margin: 10px 0;

}

*.flashes* li {

    padding: 10px;

    background-color: #eaf8e5;

    border: 1px solid #d4edda;

    color: #155724;

    border-radius: 5px;

    margin-bottom: 5px;

}

*.delete-button* {

    background-color: red;

    color: white;

    padding: 10px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

*.delete-button:hover* {

    background-color: darkred;

}

*.form-row* {

    display: flex;

    align-items: center;

    gap: 10px;

}

*.form-row* label {

    margin: 0;

    white-space: nowrap;

}

*.form-row* input[*type*="text"], *.form-row* input[*type*="number"] {

    width: 170px;

    margin: 0;

}

*.form-row* button {

    margin-left: 10px;

    padding: 10px 15px;

}

*.form-row* button*.Add-student* {

    margin-top: 17px;

}

*.form-row* input[*type*="text"]*.Search\_student*, *.form-row* input[*type*="number"]*.Search\_student*{

    width: 600px;

}

*.form-login* input[*type*="text"], *.form-login* input[*type*="password"]{

    width: 575px;

}

**File login.html:**

<!DOCTYPE html>

<html *lang*="vi">

<head>

    <meta *charset*="UTF-8">

    <title>Login</title>

    <link *rel*="stylesheet" *href*="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <div *class*="container2">

        <h2>Đăng Nhập</h2>

        <div>

        <form *action*="/login" *method*="POST">

            <div *class* = "form-login">

                <label *for*="username">Tên tài khoản:</label>

                <input *type*="text" *id*="username" *name*="username" *required*>

                <label *for*="password">Mật khẩu:</label>

                <input *type*="password" *id*="password" *name*="password" *required*>

                <button *type*="submit">Đăng nhập</button>

            </div>

        </form>

        </div>

        {% with messages = get\_flashed\_messages() %}

            {% if messages %}

                <ul *class*="flashes">

                {% for message in messages %}

                    <li>{{ message }}</li>

                {% endfor %}

                </ul>

            {% endif %}

        {% endwith %}

    </div>

</body>

</html>

**File student\_management.html:**

<!DOCTYPE html>

<html *lang*="vi">

<head>

    <meta *charset*="UTF-8">

    <title>Quản lý Sinh Viên</title>

    <link *rel*="stylesheet" *href*="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <div *class*="container">

        <h1>QUẢN LÝ SINH VIÊN</h1>

        {% with messages = get\_flashed\_messages() %}

            {% if messages %}

                <ul *class*="flashes">

                {% for message in messages %}

                    <li>{{ message }}</li>

                {% endfor %}

                </ul>

            {% endif %}

        {% endwith %}

        <form *method*="POST">

            <h3>Thêm sinh viên</h3>

            <div *class*="form-row">

                <div *class*="input-group">

                    <label *for*="mssv">MSSV:</label>

                    <input *type*="text" *id*="mssv" *name*="mssv" *required*>

                </div>

                <div *class*="input-group">

                    <label *for*="hoten">Họ tên:</label>

                    <input *type*="text" *id*="hoten" *name*="hoten" *required*>

                </div>

                <div *class*="input-group">

                    <label *for*="nganh">Ngành:</label>

                    <input *type*="text" *id*="nganh" *name*="nganh" *required*>

                </div>

                <div *class*="input-group">

                    <label *for*="namsinh">Năm sinh:</label>

                    <input *type*="number" *id*="namsinh" *name*="namsinh" *required*>

                </div>

                <button *class* = "Add-student" *type*="submit" *name*="add\_student">Thêm sinh viên</button>

            </div>

        </form>

        <h3>Tìm kiếm sinh viên</h3>

        <form *method*="POST">

            <div *class*="form-row search-row">

                <label *for*="search\_value">Nhập thông tin cần tìm:</label>

                <input *class* = "Search\_student" *type*="text" *id*="search\_value" *name*="search\_value" *required*>

                <button *type*="submit" *name*="search\_student">Tìm kiếm</button>

            </div>

        </form>

        <h3>Danh sách sinh viên</h3>

        <table>

            <thead>

                <tr>

                    <th>Mssv</th>

                    <th>Họ tên</th>

                    <th>Ngành</th>

                    <th>Năm sinh</th>

                    <th>Hành động</th>

                </tr>

            </thead>

            <tbody>

                {% for student in students %}

                <tr>

                    <td>{{ student[0] }}</td>

                    <td>{{ student[1] }}</td>

                    <td>{{ student[2] }}</td>

                    <td>{{ student[3] }}</td>

                    <td>

                        <form *method*="POST" *style*="display:inline;">

                            <input *type*="hidden" *name*="mssv" *value*="{{ student[0] }}">

                            <button *type*="submit" *name*="delete\_student" *class*="delete-button" *onclick*="return confirm('Bạn có chắc chắn muốn xóa sinh viên này?');">Xóa</button>

                        </form>

                    </td>

                </tr>

                {% endfor %}

            </tbody>

        </table>

        <a *href*="/logout" *class*="logout-button">Đăng xuất</a>

    </div>

</body>

</html>

**File app.py:**

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

import psycopg2

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

app.secret\_key = 'your\_secret\_key'

# Database connection information

DB\_NAME = 'dbtest'

USER = 'postgres'

PASSWORD = '123456'

HOST = 'localhost'

PORT = '5432'

def connect\_db():

    return psycopg2.connect(*dbname*=DB\_NAME, *user*=USER, *password*=PASSWORD, *host*=HOST, *port*=PORT)

@app.route('/')

def home():

    return render\_template('login.html')

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

def login():

    username = request.form['username']

    password = request.form['password']

    if username == "admin" and password == "123":

        session['logged\_in'] = True

        return redirect(url\_for('student\_management'))

    else:

        flash('Invalid username or password!')

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

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

def student\_management():

    if not session.get('logged\_in'):

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

    conn = connect\_db()

    cur = conn.cursor()

    if request.method == 'POST':

        # Adding a new student

        if 'add\_student' in request.form:

            hoten = request.form['hoten']

            mssv = request.form['mssv']

            nganh = request.form['nganh']

            namsinh = request.form['namsinh']

            try:

                insert\_query = "INSERT INTO sinhvien (hoten, mssv, nganh, namsinh) VALUES (%s, %s, %s, %s)"

                cur.execute(insert\_query, (hoten, mssv, nganh, namsinh))

                conn.commit()

                flash('Thêm sinh viên thành công!')

            except Exception as e:

                conn.rollback()

                flash(f'Lỗi khi thêm sinh viên: {e}')

        # Deleting a student

        elif 'delete\_student' in request.form:

            mssv = request.form['mssv']

            try:

                delete\_query = "DELETE FROM sinhvien WHERE mssv = %s"

                cur.execute(delete\_query, (mssv,))

                conn.commit()

                flash('Sinh viên đã được xóa!')

            except Exception as e:

                conn.rollback()

                flash(f'Lỗi khi xóa sinh viên: {e}')

        # Searching for students

        elif 'search\_student' in request.form:

            search\_value = request.form['search\_value']

            search\_query = "SELECT \* FROM sinhvien WHERE hoten ILIKE %s OR CAST(mssv AS TEXT) ILIKE %s OR nganh ILIKE %s OR CAST(namsinh AS TEXT) ILIKE %s"

            search\_pattern = f"%{search\_value}%"

            cur.execute(search\_query, (search\_pattern, search\_pattern, search\_pattern, search\_pattern))

            students = cur.fetchall()  # Fetch searched students and display them

            cur.close()  # Close cursor to avoid conflicts

            conn.close()  # Close connection after search

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

    # If it's a GET request or no POST action

    cur.execute("SELECT \* FROM sinhvien")

    students = cur.fetchall()  # Always fetch all students at the end

    cur.close()

    conn.close()

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

@app.route('/logout')

def logout():

    session.pop('logged\_in', None)

    return redirect(url\_for('home'))

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

    app.run(*debug*=True)

## 3.2. Mô tả demo

* Bước 1: Người dùng đăng nhập vào hệ thống bằng thông tin tài khoản.
* Bước 2: Sau khi đăng nhập, người dùng có thể xem danh sách sinh viên hiện tại.
* Bước 3: Người dùng có thể thực hiện các thao tác thêm hoặc xóa sinh viên, với thông báo trạng thái sau mỗi hành động.
* Bước 4: Tính năng tìm kiếm cho phép người dùng dễ dàng tra cứu thông tin sinh viên.

## 3.3. Giao diện chương trình

Giao diện đăng nhập:

**A screenshot of a computer

Description automatically generated**

Giao diện quản lý sinh viên sau khi đăng nhập thành công:

**A screenshot of a computer

Description automatically generated**

Giao diện thêm sinh viên thành công sau khi thực hiện thao tác thêm sinh viên:

A screenshot of a computer

Description automatically generated

Giao diện xóa sinh viên thành công sau thực hiện thao tác xóa:

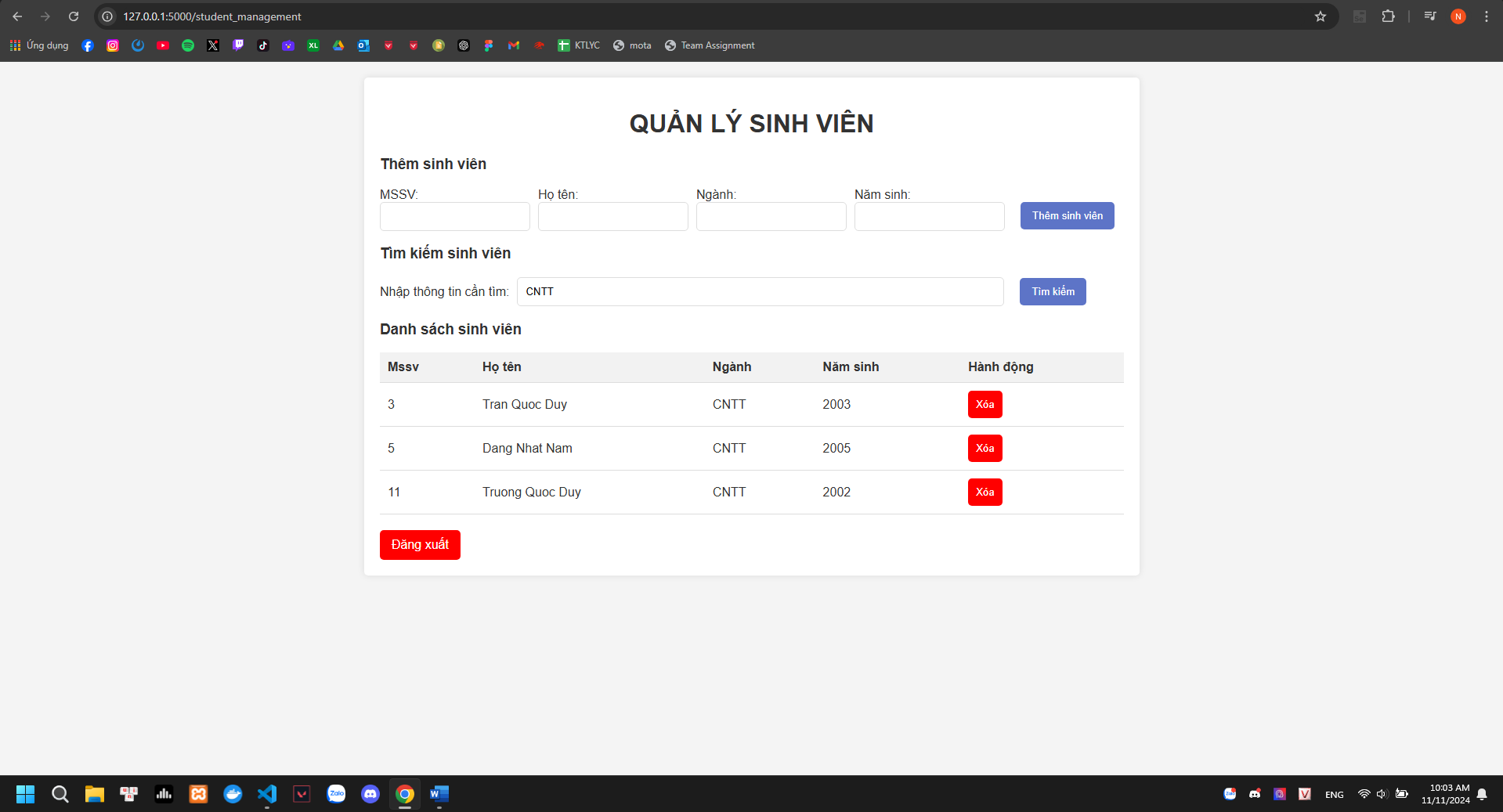
A screenshot of a computer

Description automatically generated

Giao diện tìm kiếm thông tin sinh viên theo cột thông tin:

A screenshot of a computer

Description automatically generated



**Hết**