**TRƯỜNG ĐẠI HỌC AN GIANG- ĐẠI HỌC QUỐC GIA TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

A yellow circle with blue and green text

Description automatically generated

**BÁO CÁO ĐỒ ÁN**

**Chuyên đề Python (COS525)**

**Nhóm-15**

# XÂY DỰNG ỨNG DỤNG QUẢN LÝ KÍ TÚC XÁ

**VỚI PYTHON, TKINTER VÀ MS SQL SEVER**

**Giảng viên hướng dẫn:**

**Ths. Nguyễn Ngọc Minh**

**Sinh viên thực hiện:**

**DTH235622 - Phạm Cẩm Đào**

**DTH235620 - Đinh Thanh Danh**

**An Giang, Ngày 10, Tháng 11, Năm 2025.**

**LỜI NÓI ĐẦU**

**1. Tính Cần Thiết của Đề Tài**

Trong bối cảnh số hóa và hiện đại hóa các hoạt động quản lý tại các cơ sở giáo dục, việc ứng dụng công nghệ thông tin vào công tác quản lý ký túc xá đang trở thành nhu cầu cấp thiết. Hiện nay, nhiều ký túc xá sinh viên vẫn đang sử dụng các phương pháp quản lý thủ công hoặc bán thủ công, dẫn đến những hạn chế như: khó khăn trong việc tra cứu thông tin, mất nhiều thời gian xử lý hồ sơ, dễ xảy ra sai sót trong quản lý sinh viên và phòng ở, cũng như thiếu tính minh bạch trong các giao dịch tài chính.

Nhận thức được những vấn đề trên, đồ án này tập trung vào việc **"Xây dựng ứng dụng quản lý ký túc xá"** sử dụng ngôn ngữ lập trình Python và thư viện đồ họa tkinter. Ứng dụng được thiết kế nhằm số hóa và tự động hóa các quy trình quản lý, bao gồm: quản lý thông tin sinh viên, phân bổ phòng ở, theo dõi thanh toán phí ký túc xá, và quản lý các dịch vụ tiện ích. Thông qua việc ứng dụng các thư viện mạnh mẽ của Python, đồ án hướng tới mục tiêu tạo ra một giải pháp phần mềm thân thiện, dễ sử dụng và có khả năng mở rộng trong tương lai.

1. **Mục Tiêu Nghiên Cứu**

Mục tiêu tổng quát:

Xây dựng ứng dụng quản lý ký túc xá sử dụng ngôn ngữ lập trình **Python** với giao diện đồ họa **Tkinter**, kết nối cơ sở dữ liệu **MS SQL Server**, nhằm **số hóa** và **tự động hóa** các quy trình quản lý, **nâng cao hiệu quả** công tác quản lý ký túc xá sinh viên.

Mục tiêu cụ thể:

* **Xây dựng hệ thống quản lý thông tin sinh viên** đầy đủ, chính xác với các chức năng **thêm, sửa, xóa, tìm kiếm** và **phân loại** theo nhiều tiêu chí..
* Thiết kế **module quản lý phòng ở linh hoạt, theo dõi tình trạng phòng** trống/đầy/bảo trì theo thời gian thực..
* **Đảm bảo tính toàn vẹn, bảo mật dữ liệu** thông qua cơ chế **phân quyền, mã hóa mật khẩu** và **sao lưu dữ liệu định kỳ**.
* Tạo cơ sở cho việc **mở rộng và nâng cấp hệ thống** trong tương lai.

Thông qua việc đạt được các mục tiêu trên, đề tài hướng tới việc xây dựng một **giải pháp phần mềm hoàn chỉnh**, có khả năng ứng dụng thực tế tại các ký túc xá sinh viên, góp phần **hiện đại hóa** công tác quản lý và **nâng cao chất lượng dịch vụ**.

**3. Cấu Trúc Báo Cáo**

Báo cáo được trình bày gồm các chương chính sau:

* Chương 1: Đặt Vấn Đề (Tính cần thiết, Mục tiêu)
* Chương 2: Tổng Quan và Cơ Sở Lý Thuyết (Python, Tkinter, PostgreSQL)
* Chương 3: Phân Tích, Thiết Kế và Triển Khai Ứng Dụng (Thiết kế CSDL, GUI, chi tiết mã nguồn CRUD)
* Chương 4: Kết Quả Đạt Được và Đánh Giá (Đánh giá tính năng, hướng phát triển)

1. **Đặt vấn đề**

**1.1. Đặt vấn đề**

Để xây dựng một **ứng dụng quản lý ký túc xá hiệu quả**, việc nêu lên **bối cảnh xuất hiện vấn đề** và **nội dung bài toán đặt ra** là cần thiết. Như đã phân tích ở phần trên, vấn đề chính mà đề tài cần giải quyết là việc **số hóa và tự động hóa các quy trình quản lý ký túc xá**, từ **quản lý thông tin sinh viên, phòng ở, hợp đồng**, đến **thanh toán và báo cáo thống kê**.

Bài toán đặt ra là: Làm thế nào để xây dựng một **hệ thống phần mềm** có khả năng **quản lý toàn diện** các hoạt động của ký túc xá, đảm bảo tính **chính xác, nhanh chóng, dễ sử dụng** và có khả năng **mở rộng** trong tương lai? Hệ thống cần phải giải quyết được các vấn đề về **lưu trữ dữ liệu lớn, xử lý đồng thời nhiều người dùng, bảo mật thông tin** và **tạo ra các báo cáo thống kê hữu ích** cho công tác quản lý.

**1.2. Lịch sử giải quyết vấn đề**

Vấn đề quản lý ký túc xá đã được nhiều cá nhân và tổ chức nghiên cứu và giải quyết ở các mức độ khác nhau. Trong quá trình tìm hiểu, chúng tôi nhận thấy:

Các giải pháp đã có:

* Nhiều trường đại học vẫn sử dụng phương pháp quản lý **thủ công** bằng **sổ sách giấy tờ**, dễ thất thoát và khó tra cứu.
* Một số đơn vị đã ứng dụng file **Excel** để quản lý, tuy nhiên phương pháp này vẫn còn nhiều hạn chế về khả năng **xử lý dữ liệu lớn**, **không hỗ trợ đồng thời nhiều người dùng** và **dễ gây sai sót**.
* Một số trường đã đầu tư xây dựng **phần mềm quản lý chuyên nghiệp**, nhưng thường có **chi phí cao** và **không phù hợp với mọi quy mô** ký túc xá.
* Các đề tài sinh viên trước đây cũng đã nghiên cứu xây dựng ứng dụng tương tự nhưng thường sử dụng các công nghệ khác nhau (**Java, C#, Web-based**) với **phạm vi và chức năng còn hạn chế**.

**Đánh giá các giải pháp:**

Qua khảo sát, chúng tôi nhận thấy rằng các giải pháp hiện có hoặc là quá **đơn giản** (thủ công, Excel), hoặc là quá **phức tạp và tốn kém** (phần mềm thương mại). Đề tài này hướng đến việc xây dựng một **giải pháp cân bằng**: vừa đủ chức năng để đáp ứng nhu cầu thực tế, vừa dễ triển khai và bảo trì với **chi phí thấp**.

*Điểm mới của đề tài:*

· Sử dụng **Python** - ngôn ngữ lập trình hiện đại, dễ học, dễ bảo trì.

· Ứng dụng **Tkinter** để tạo giao diện **desktop thân thiện**, **không yêu cầu kết nối internet**.

· Tích hợp **MS SQL Server** đảm bảo **quản lý dữ liệu chuyên nghiệp**.

· Thiết kế theo **mô hình phân lớp**, dễ **mở rộng** và **nâng cấp**.

· Tập trung vào **trải nghiệm người dùng** và **tính thực tiễn**.

**2. Phạm vi đề tài**

Để đảm bảo tính khả thi và hiệu quả trong thời gian thực hiện đồ án, phạm vi của đề tài được xác định rõ ràng như sau:

**2.1. *Phạm vi chức năng.***

Đề tài tập trung xây dựng các **chức năng cốt lõi** của hệ thống quản lý ký túc xá phục vụ công tác nội trú, bao gồm:

Dành cho quản lý ký túc xá

* Quản lý thông tin sinh viên (thêm – sửa – xóa – tìm kiếm).
* Quản lý danh sách phòng ở theo từng tòa, theo loại phòng và trạng thái phòng.
* Theo dõi và cập nhật tình trạng đóng tiền phòng của sinh viên.
* Quản lý tài khoản đăng nhập (lấy từ bảng TaiKhoan trong CSDL).
* Xem và quản lý lịch sử đóng tiền của từng sinh viên.
* Xem thống kê số lượng sinh viên theo tòa nhà, theo phòng, theo giới tính.

Dành cho sinh viên

* Đăng nhập và xem thông tin cá nhân.
* Xem phòng đang ở, tòa nhà, và giá phòng.
* Xem trạng thái đóng tiền theo tháng/năm.
* Theo dõi lịch sử đóng tiền của bản thân.

Không bao gồm

* Không hỗ trợ đăng ký phòng online.
* Không hỗ trợ phân phòng tự động.
* Không có chức năng xuất báo cáo ra PDF/Excel.
* Không có tính năng gửi thông báo hoặc email

**2.2. Phạm vi kỹ thuật.**

* Ngôn ngữ lập trình: python.
* Giao diện: Tkinter đầy đủ các chức năng CRUD.
* Cơ sở dữ liệu: MS SQL Server.
* Kết nối CSDL: PyODBC.
* Ứng dụng chạy offline trên máy tính cục bộ (local machine).
* Hệ thống không triển khai trên Web hoặc Mobile.

**2.3. Phạm vi người dùng.**

* Quản lý ký túc xá tại trường học
* Sinh viên nội trú đang ở trong ký túc xá.
* Những đối tượng khác như phụ huynh, ban giám hiệu, hoặc bộ phận tài chính không nằm trong phạm vi sử dụng trực tiếp của hệ thống.

**2.4. Phạm vi triển khai.**

* Chạy trên máy tính Windows có cài Python, ODBC Driver 17 (hoặc các phiên bản cũ hơn) và SQL Server.
* Không triển khai mạng LAN hoặc hệ thống phân tán.

**2.5. Phạm vi dữ liệu.**

Cơ sở dữ liệu được thiết kế và sử dụng với các bảng chính:

* ToaNha: danh sách 4 tòa A, B, C, D.
* Phong: thông tin phòng ở (loại 4 hoặc 6 người).
* SinhVien: hơn 100 sinh viên mẫu (phân chia theo tòa).
* QuanLy: danh sách quản lý ký túc xá.
* TaiKhoan: tài khoản đăng nhập cho quản lý và sinh viên.
* LichSuDongTien: lưu thông tin đóng tiền theo tháng

Dữ liệu trên là dữ liệu mẫu được tạo ra chủ yếu nhằm mục đích thử nghiệm chức năng ứng dụng.

**2.6. Phương pháp nghiên cứu và hướng giải quyết vấn đề.**

Thực hiện khảo sát thực tế: Tiến hành đi khảo sát thực tế tại ký túc xá tại Trường Đại học An Giang, thu thập dữ liệu và thông tin cần thiết bằng cách phỏng vấn sinh viên nội trú và quản lý ký túc xá, tìm hiểu thêm các tài liệu tham khảo.

Phân tích yêu cầu:

* Xác định vấn đề cốt lõi: Phân tích các hạn chế của phương pháp quản lý thủ công và ứng dụng cơ bản đã được phát triển trước đó. Xác định vấn đề cốt lõi là sự thiếu minh bạch và tự động hóa trong quản lý phân phòng và thu phí ký túc xá.
* Xác định phạm vi: Xác định chính xác các đối tượng người dùng (quản lý và sinh viên) và các nghiệp vụ cần giải quyết (CRUD, phân phòng, ghi nhận đóng tiền, xem lịch sử)
* Phân tích ràng buộc: Xác định các quy tắc nghiệp vụ cần tuân thủ trong quá trình lập trình, ví dụ: bắt buộc chọn mã phòng khi thêm sinh viên mới.

Xây dựng cơ sở dữ liệu: Sử dụng hệ quản trị cơ sở dữ liệu SQL Sever để tạo và lưu trữ dữ liệu.

Thực hiện xây dựng ứng dụng : Dùng ngôn ngữ lập trình Python, Tkinter và SQL Sever để xây dựng giao diện, quản lý dữ liệu và cài đặt chương trình.

Kiểm thử và hoàn thiện ứng dụng: chạy thử, sửa lỗi kết nối, kiểm tra tính năng CRUD (Create, Read, Update, Delete).

**2.7. Cơ sở lý thuyết.**

Python: Là ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng, mạnh về khả năng xử lý dữ liệu và phát triển ứng dụng nhanh (Ts. Huỳnh Phước Hải, 2023). Với những ưu điểm:

* Cú pháp ngắn gọn, dễ đọc, dễ nhớ, dễ viết mã và dễ bảo trì.
* Có khả năng xử lý dữ liệu lớn, thư viện có nhiều hàm hỗ trợ các công nghệ mới trong trong các lĩnh vực trí tuệ nhân tạo, khai phá dữ liệu, cơ sở dữ liệu.
* Tương thích tốt với các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, SQLite.
* Đáp ứng được yêu cầu của bài toán lập trình cổ điển và cả bài toán lập trình mới.

Tkinter: Là thư viện thiết kế giao diện tươn g tác người dùng Grapic User Interface (GUI) tiêu chuẩn của Python, cho phép tạo giao diện ứng dụng nhanh chóng. Tkinter cung cấp các widget như:

* Label, Button, Entry, Frame, Treeview, Combobox, Scrollbar...  
  Giúp tạo các form đăng nhập, bảng quản lý dữ liệu, và hộp thoại thông báo. Trong đồ án này, Treeview được dùng để hiển thị danh sách sinh viên và phòng; LabelFrame để nhóm thông tin; Button để thao tác thêm/sửa/xóa.

SQL Server: Structured Query Language (SQL) là một loại phần mềm được phát triển bởi Microsoft, dùng để lưu trữ và truy xuất dữ liệu dựa theo tiêu chuẩn RDBMS (Relational Database Management System). Một RDBMS bao gồm: cơ sở dữ liệu (databases), công cụ cơ sở dữ liệu (database engine), các ứng dụng quản lý dữ liệu và các bộ phận khác.

* Dễ học và dễ sử dụng, khả năng truy vấn mạnh mẽ.
* Nhất quán và toàn vẹn dữ liệu.
* Có khả năng mở rộng và bảo mật tốt.
* Có thể hoạt động trên nhiều nền tảng khác nhau, là công cụ linh hoạt và đa năng

Các công cụ hỗ trợ phát triển

* Visual Studio Code / PyCharm: môi trường để soạn thảo, chạy, và debug mã Python.
* ODBC Driver 17 for SQL Server: trình điều khiển giúp Python kết nối với SQL Server.
* Gói thư viện bổ sung: tkinter, pyodbc, datetime, os, messagebox.

Cơ sở lý thuyết mới xây dựng (nghiệp vụ ký túc xá)

* Mô hình phân quyền đơn giản: Xây dựng chức năng kiểm tra vai trò người dùng (check\_credentials\_in\_sqlserver) ngay tại lớp Login để quyết định mở giao diện Admin hay Student, đảm bảo tính bảo mật và phân quyền cơ bản.
* Logic quản lý thanh toán (tiền tổng): Xây dựng logic để tổng hợp giá trị từ 2 cột CSDL (Phong.Gia và Phong.TienDienNuoc) để tính ra tổng số tiền phải đóng và sử dụng 1 cột duy nhất (LichSuDongTien) để kiểm tra trạng thái đóng tiền trong tháng hiện tại.

- Kỹ thuật Pop-up lịch sử/LogPayment: Xây dựng 2 lớp Toplevel mới (StudentPaymentHistoryWindow và LogPaymentWindow) để xử lý các nghiệp vụ phụ trợ phức tạp:LogPaymentWindow: Tự động điền thông tin và tổng tiền cần đóng.

- StudentPaymentHistoryWindow: Hiển thị dữ liệu từ bảng LichSuDongTien theo tiêu chí tháng/năm, định dạng ngày/tiền tệ chuyên nghiệp.

Quy Trình Thực Hiện Đồ Án

* Bước 1: Thu thập và phân tích yêu cầu

Khảo sát thực tế

Phân tích nghiệp vụ

* Bước 2: Thiết kế hệ thống

Thiết kế kiến trúc tổng thể \* Thiết kế cơ sở dữ liệu

Thiết kế giao diện

* Bước 3: Triển khai

Xây dựng cơ sở dữ liệu

Lập trình từng module theo thứ tự ưu tiên

Tích hợp các module

* Bước 4: Kiểm thử

Kiểm thử từng module (Unit Testing)

Kiểm thử tích hợp (Integration Testing)

Kiểm thử hệ thống (System Testing)

Sửa lỗi và tối ưu hóa

**3 . Kết quả đạt được và demo code**

**3.1. Thiết kế giao diện**

**Chức năng đăng nhập.**



Hình 1. Giao diện đăng nhập hệ thống

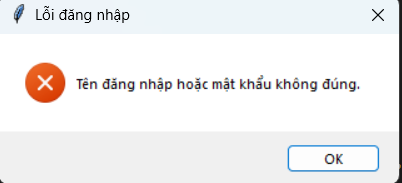
Đối với việc đăng nhập cần thì người dùng cần phải có tài khoản được cấp mới có thể đăng nhập vào hệ thống vì vậy có thể thấy ở giao diện đăng nhập không có chức năng đăng ký. Tài khoản đăng nhập cần được kiểm tra qua các bước:

* Kiểm tra có nhập đủ thông tin gồm tên đăng nhập và mật khẩu, nếu nhập thiếu cả hai hoặc một trong hai thông tin hệ thống sẽ hiển thị thông báo



Hình 2. Thông báo thiếu thông tin

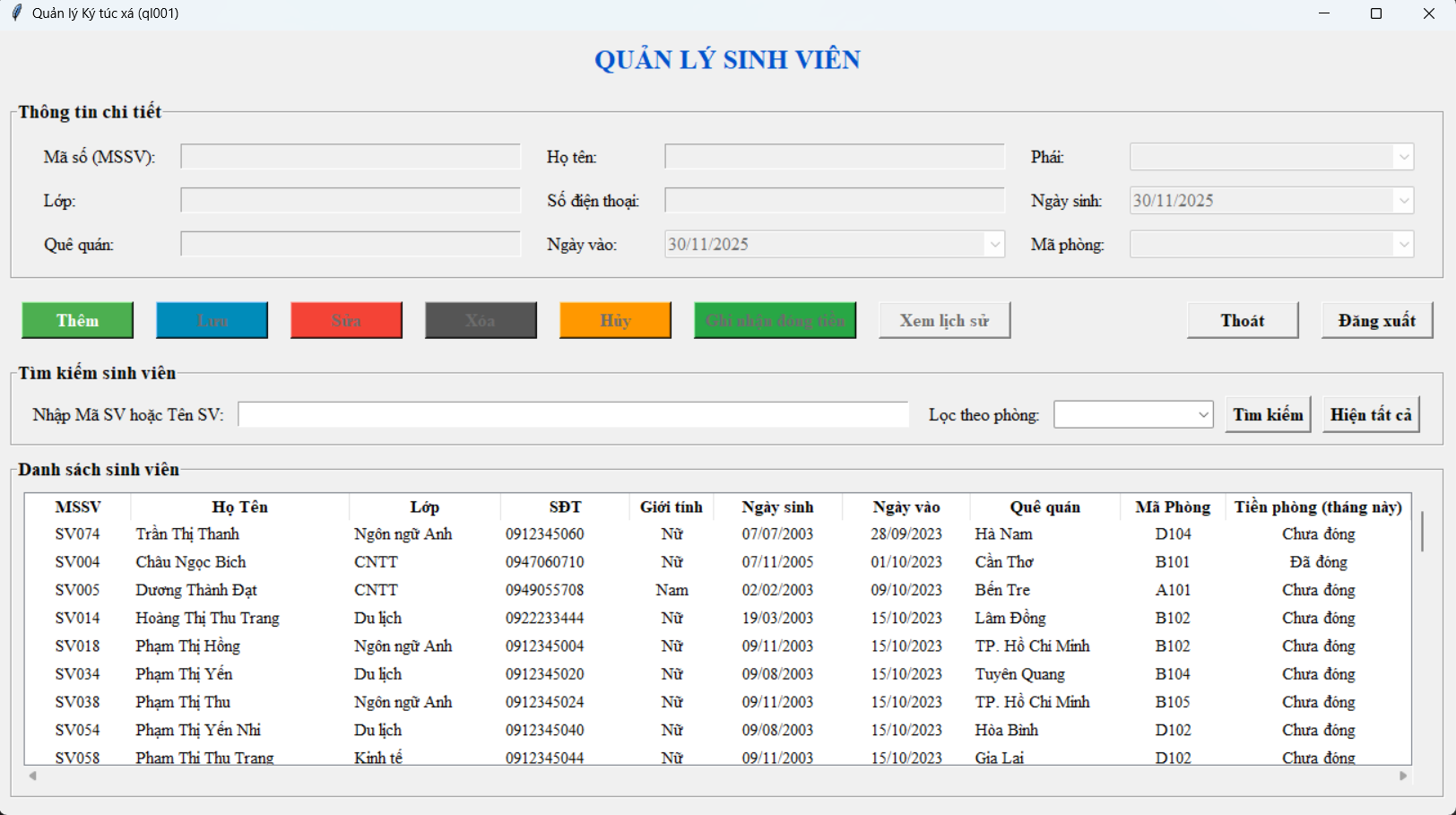
* Kiểm tra tên đăng nhập và mật khẩu có chính xác không,
  + Nếu cả hai hoặc một trong hai thông tin không chính xác hệ thống sẽ hiển thị thông báo



Hình 3. Thông báo lỗi đăng nhập

Nếu tên đăng nhập và mật khẩu đều đúng thì đăng nhập thành công và hệ thống sẽ kiểm tra tài khoản vừa đăng nhập là tài khoản quản lý hay tài khoản sinh viên và mở form tương ứng.

Tài khoản quản lý thì hệ thống sẽ hiển thị giao diện Quản lý



Hình 4. Giao diện quản lý

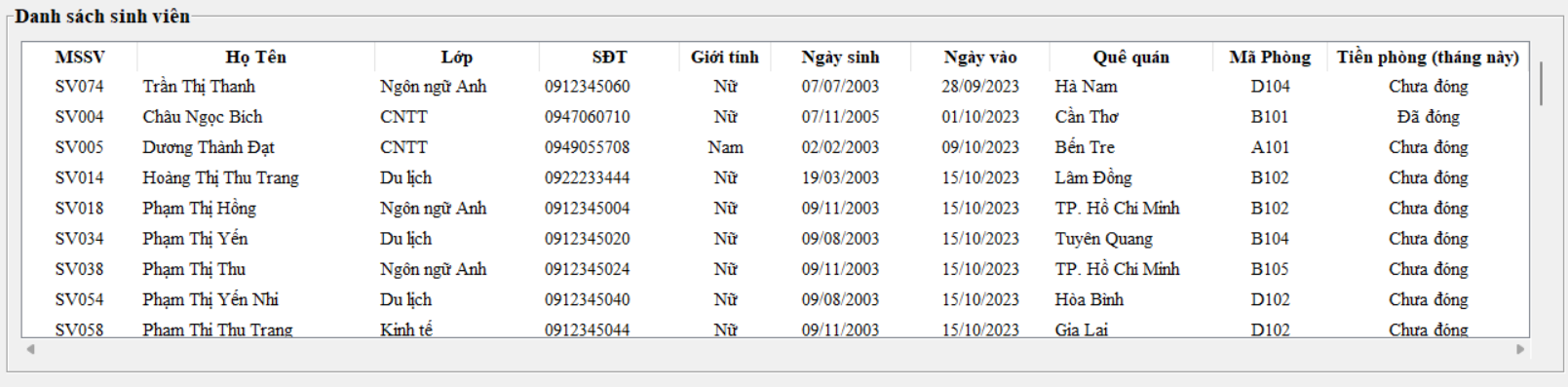
Tài khoản sinh viên thì hệ thống sẽ hiển thị giao diện Sinh viên



Hình 5. Giao diện sinh viên

**Chức năng của tài khoản quản lý.**

Xem danh sách sinh viên: Khi người dùng đăng nhập với tài khoản quản lý, mã Python sẽ truy vấn cơ sở dữ liệu, lấy tất cả thông tin sinh viên và hiển thị chúng dưới dạng một danh sách trên giao diện Tkinter “Quản lý”.

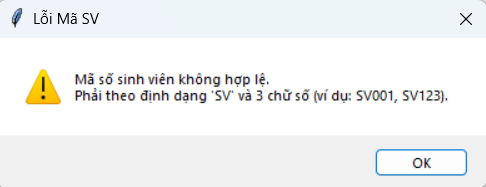


Hình 6. Danh sách sinh viên

Thêm: Khi quản lý nhấn nút “Thêm” các ô nhập liệu sẽ được mở khóa để người dùng nhập thông tin đồng thời vô hiệu hóa các nút “Sửa” và “Xóa”. Hệ thống sẽ tự động lọc và hiển thị danh sách các phòng trống của những tòa nhà theo giới tính sinh viên để người quản lý dễ dàng thêm sinh viên vào các phòng phù hợp.

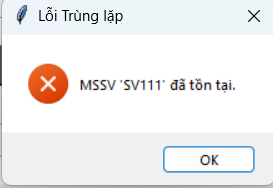
* Thông tin nhập vào chưa chính xác

Nhập sai định dạng mã số sinh viên



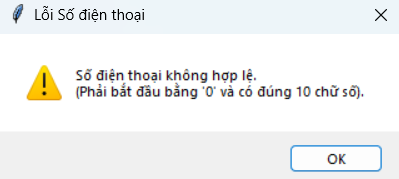
Hình 7. Thông báo lỗi mã sinh viên

Mã số sinh viên đã tồn tại



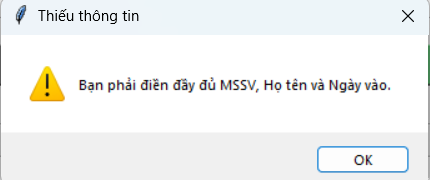
Hình 8. Thông báo lỗi trùng lặp

Nhập sai định dạng số điện thoại



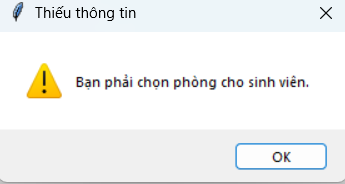
Hình 9. Thông báo lỗi số điện thoại

Nhập thiếu thông tin MSSV, Họ tên, Ngày vào



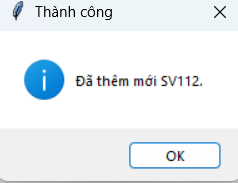
Hình 10. Thông báo thiếu thông tin

Chưa chọn Mã phòng



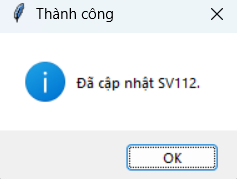
Hình 11. Thông báo thiếu thông tin

* Thông tin đầy đủ và chính xác: Khi người dùng nhập đầy đủ và chính xác thông tin và nhấn nút "Lưu", mã Python sẽ lấy các giá trị này và thêm một bản ghi mới vào bảng SinhVien trong SQL Sever, hệ thống hiển thị thông báo thêm thành công. Sau khi thêm thành công hệ thống sẽ xóa trống các ô nhập liệu.



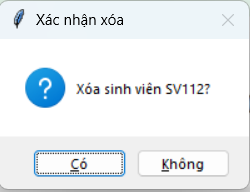
Hình 12. Thông báo thêm thành công

Sửa: Khi người dùng nhấn chọn một sinh viên bất kì từ “Danh sách sinh viên” rồi nhấn nút “Sửa” thì các ô nhập liệu sẽ được mở khóa để người dùng cập nhật thông tin sinh viên tin đồng thời vô hiệu hóa các nút “Thêm”, “Xóa”. Việc sửa thông tin chỉ được thực hiện trên các thông tin không phải là khóa chính của bảng “Sinhvien”. Sau khi sửa xong, hệ thống sẽ cập nhật lại trạng thái phòng.

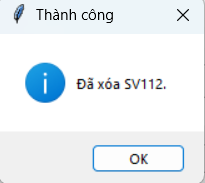


Hình 13. Thông báo cập nhật thành công

Xóa: Khi người dùng nhấn chọn một sinh viên bất kì từ “Danh sách sinh viên” rồi nhấn nút “Xóa” hệ thống sẽ hiện thông báo xác nhận xóa vĩnh viễn, nếu người dùng chọn “Yes” hệ thống sẽ xóa sinh viên đã chọn khỏi cơ sở dữ liệu. Hệ thống sẽ lấy “MaPhong” trước khi xóa sinh viên khỏi bảng SinhVien, MaPhong sẽ được dùng để cập nhật trạng thái phòng sau khi xóa. Xóa tài khoản tương ứng trong bảng TaiKhoan. Bảng LichSuDongTien sẽ tự động xóa các bản ghi liên quan nhờ ràng buộc ON DELETE CASCADE trên FK MSSV trong SQL script.



Hình 14. Thông báo xác nhận xóa



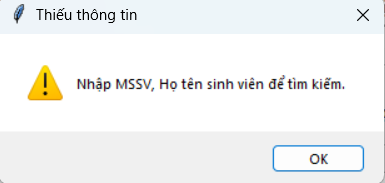
Hình 15. Thông báo xóa thành công

Lưu:

* Chịu trách nhiệm chính trong việc ghi dữ liệu mới hoặc cập nhật dữ liệu hiện có vào cơ sở dữ liệu, đồng thời xử lý các nghiệp vụ KTX quan trọng.
* Lấy tất cả giá trị từ các ô nhập liệu
* Xác định chế độ (thêm mới hay sửa)
* Kiểm tra tính hợp lệ của dữ liệu trước khi thực hiện các thao tác.
* Tương tác với cơ sở dữ liệu để hoàn tất thêm hoặc sửa

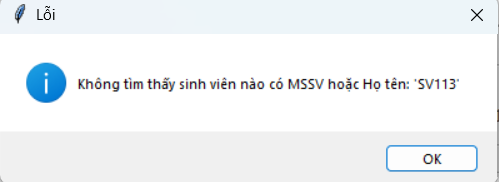
Chức năng tìm kiếm theo mã số sinh viên hoặc họ tên sinh viên: Hàm search\_student\_by\_name lấy từ khóa tìm kiếm (search\_term) từ trường nhập liệu.

* Thông tin chưa đầy đủ hệ thống sẽ hiển thị thông báo thiếu thông tin



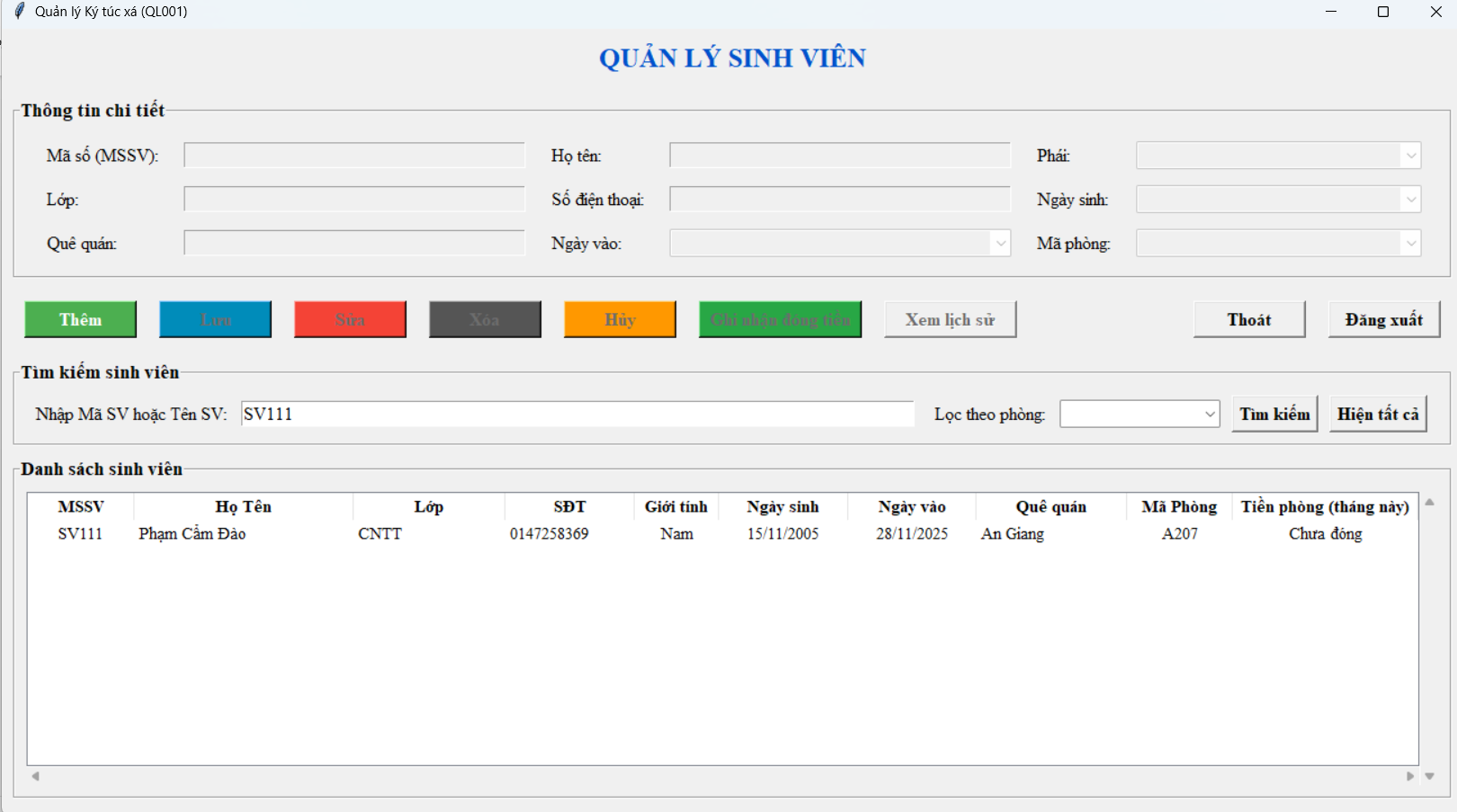
Hình 16. Thông báo thiếu thông tin

* Mã số sinh viên, họ tên, tên sinh viên không tồn tại hoặc không chính xác hệ thống sẽ hiện thông báo lỗi tìm kiếm.



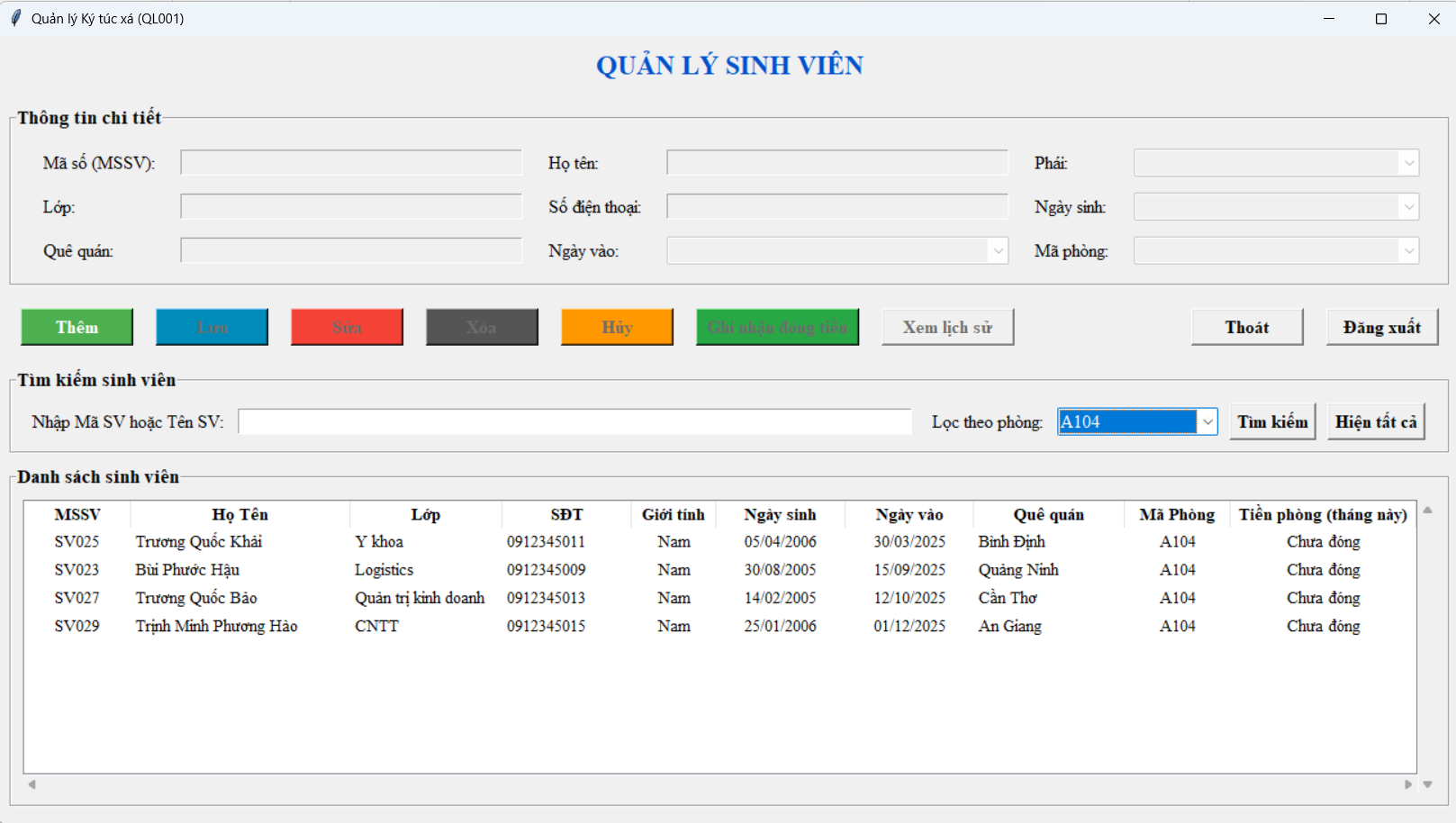
Hình 17. Thông báo lỗi tìm kiếm

* Thông tin tìm kiếm đầy đủ và sinh viên có tồn tại, hệ thống hiển thị danh sách sinh viên chứa thông tin sinh viên vừa tìm kiếm.



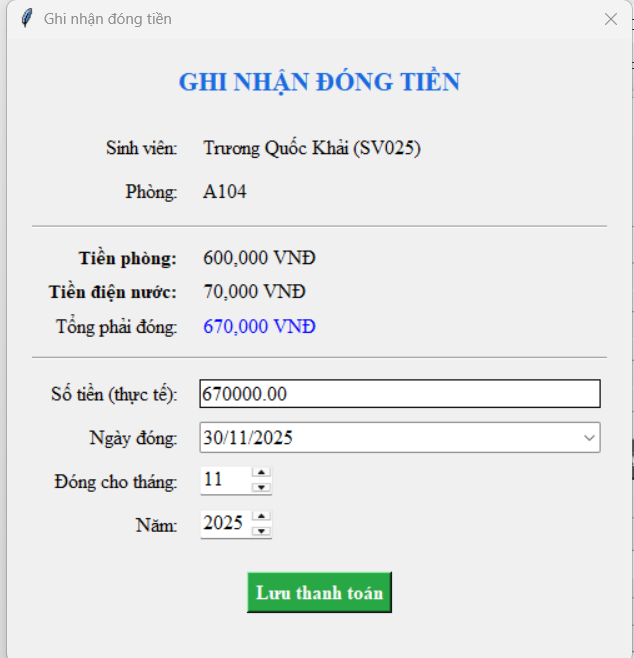
Hình 18. Thông tin sinh viên vừa tìm kiếm

Chức năng lọc sinh viên theo phòng: Khi quản lý chọn phòng muốn lọc trong combobox, Hệ thống đếm số lượng sinh viên thỏa mãn điều kiện và hiển thị những sinh viên thuộc phòng đó.

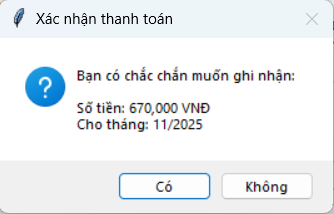


Hình 19. Thông tin sinh viên theo từng phòng

Ghi nhận đóng tiền: Chọn một sinh viên từ “Danh sách sinh viên” thì nút “Ghi thanh toán” được kích hoạt, khi quản lý nhấp chọn nút “Ghi thanh toán” thì form “Ghi nhận đóng tiền” được mở ra để quản lý lưu thanh toán cho sinh viên đã chọn bằng cách nhấp vào nút “Lưu thanh toán”. Hệ thống hiện thông báo xác nhận thanh toán và quản lý chọn “Yes” thì hệ thống thực hiện việc ghi nhận đóng tiền, lưu bản ghi giao dịch vào cơ sở dữ liệu và lập tức cập nhật trạng thái thanh toán của sinh viên đó trên bảng danh sách.

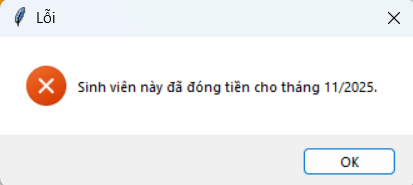


Hình 20. Bản ghi "Ghi nhận đóng tiền"



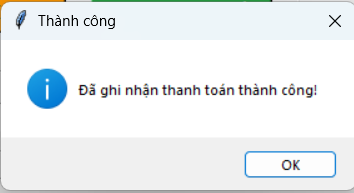
Hình 21. Thông báo xác nhận thanh toán

* Nếu có xuất hiện trùng lặp: Sinh viên đã có bản ghi đóng tiền cho tháng/năm được chọn thì hệ thống hiện thông báo lỗi



Hình 22. Thông báo lỗi trùng lặp

* Nếu không xuất hiện trùng lặp: Sinh viên chưa có bản ghi đóng tiền cho tháng/năm được chọn thì hệ thống hiện thông báo thanh toán thành công.



Hình 23. Thông báo ghi nhận thanh toán thành công

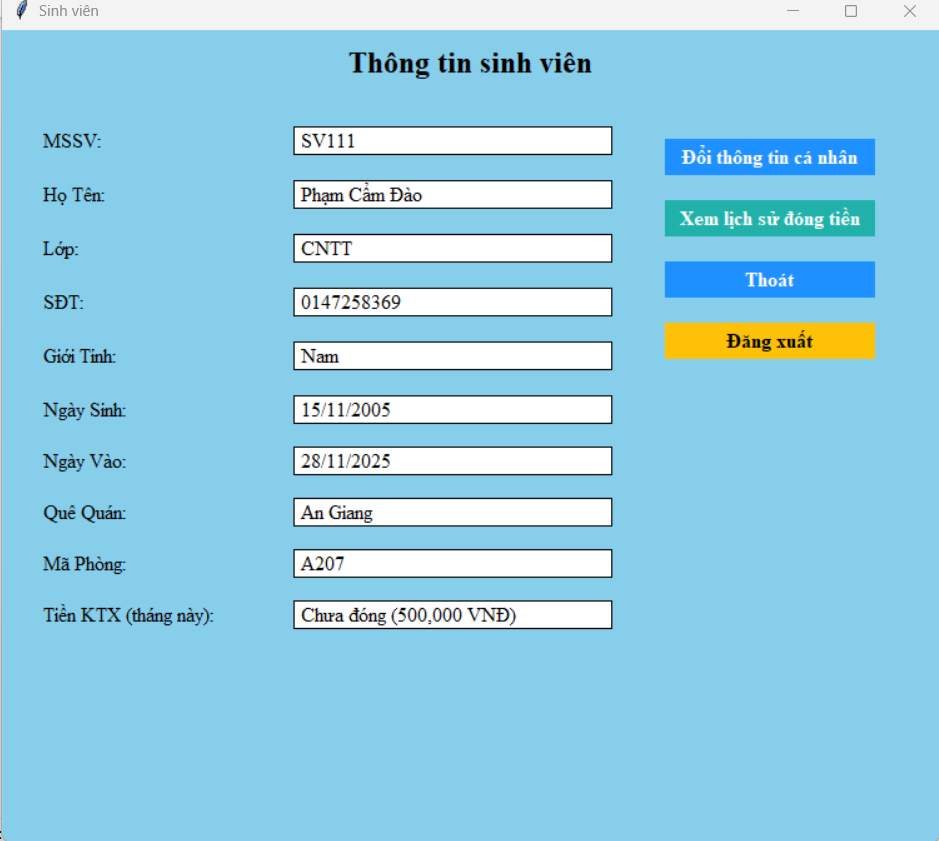
Chức năng xem lịch sử đóng tiền: Nút "Xem lịch sử" chỉ được kích hoạt sau khi quản lý nhấp chọn một sinh viên trên bảng danh sách, hệ thống lấy mã số sinh viên (MSSV) của sinh viên đang được chọn và thực hiện truy vấn vào bảng LichSuDongTien để lấy các bản ghi thanh toán có MSSV tương ứng. Các bản ghi được truy vấn sẽ được sắp xếp theo thời gian (năm và tháng giảm dần) và hiển thị trong một bảng (Treeview) để quản lý dễ dàng theo dõi chi tiết từng lần đóng tiền (tháng/năm, số tiền, ngày đóng), giúp kiểm tra đối soát và truy vết lỗi.



Hình 24. Thông tin lịch sử đóng tiền

**Chức năng của tài khoản sinh viên.**

Xem thông tin cá nhân: Khi người dùng đăng nhập với tài khoản sinh viên, mã Python sẽ truy vấn cơ sở dữ liệu, lấy tất cả thông tin sinh viên của mã sinh viên vừa đăng nhập và hiển thị chúng trên giao diện Tkinter “Sinh viên”. Đảm bảo dữ liệu chỉ ở chế độ xem, không cho phép chỉnh sửa trực tiếp, từ đó bảo vệ tính toàn vẹn của dữ liệu sinh viên trong CSDL

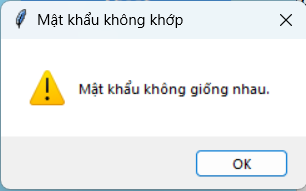


Hình 25. Thông tin sinh viên

Đổi thông tin cá nhân: Mở một cửa sổ pop-up độc lập (tk.Toplevel) để sinh viên nhập dữ liệu mới mà không ảnh hưởng đến giao diện chính. Sau khi cập nhật thành công, cửa sổ pop-up bị hủy (self.destroy()), và giao diện chính (StudentForm) được cập nhật ngay lập tức với SĐT mới.

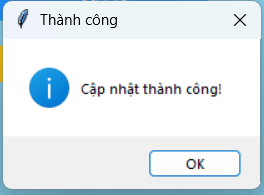
* Đổi mật khẩu.

Xác thực mật khẩu không trùng khớp.



Hình 26. Thông báo mật khẩu không trùng khớp

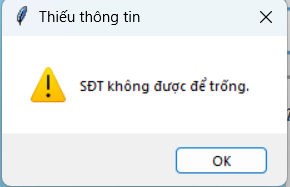
Xác thực mật khẩu trùng khớp.



Hình 27. Thông báo cập nhật thành công

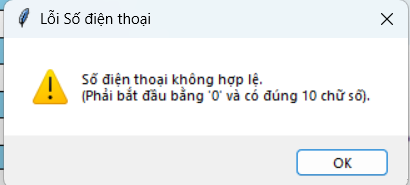
* Đổi số điện thoại.

Chưa nhập thông tin số điện thoại.



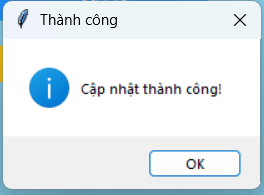
Hình 28. Thông báo thiếu thông tin số điện thoại

Thông tin số điện thoại không hợp lệ.



Hình 29. Thông báo số điện thoại không hợp lệ

Nhập đầy đủ thông tin số điện thoại và đúng định dạng.



Hình 30. Thông báo cập nhật thành công

Xem lịch sử đóng tiền: Khi sinh viên nhấn nút, hệ thống tự động lấy Mã số sinh viên (MSSV) của tài khoản đang đăng nhập. Cửa sổ StudentPaymentHistoryWindow sử dụng MSSV để thực hiện truy vấn select vào bảng LichSuDongTien để lấy tất cả các bản ghi thanh toán liên quan đến MSSV đó. Các bản ghi được truy vấn sẽ được sắp xếp theo thời gian (năm và tháng giảm dần) và hiển thị trong một bảng (Treeview) để sinh viên dễ dàng theo dõi chi tiết từng lần đóng tiền (tháng/năm, số tiền, ngày đóng).



Hình 31. Thông tin lịch sử đóng tiền

* 1. **Các bước xây dựng ứng dụng.**

**Cài đặt các thư viện và phần mềm cần thiết**

Python: Đảm bảo đã cài đặt Python trên máy tính.  
SQL Sever: Cài đặt máy chủ SQL Sever và tạo một cơ sở dữ liệu cho ứng dụng.

Tkinter: xây dựng giao diện đồ họa (GUI) cho ứng dụng.

PyODBC: giúp kết nối PyThon với SQL Sever

* Tìm xem ODBC Driver 17 for SQL Server hoặc phiên bản khác (như 13) có trong danh sách không, nếu không có hãy tải xuống và cài đặt.
* pip install pyodbc

tklendar: Để chọn ngày sinh, ngày vào bằng DateEntry

* pip install tkcalendar

Ttk (thuộc Tkinter): dùng để nâng cấp giao diện với các widget hiện đại hơn.

**Thiết kế cơ sở dữ liệu SQL Sever**

Sử dụng SQL Sever để tạo database và các table cần thiết.

* Cơ sở dữ liệu QL\_KyTucXa01
* Gồm 6 bảng

SinhVien

TaiKhoan

QuanLy

ToaNha

Phong

LichSuDongTien

**Xây dựng giao diện người dùng (GUI) bằng Tkinter.**

Sử dụng thư viện Tkinter để tạo các cửa sổ, trường nhập liệu, nút bấm và bảng hiển thị thông tin sinh viên. Thiết kế giao diện cho phép người dùng thực hiện các chức năng sau:

* Người dùng quản lý
  + Thêm thông tin sinh viên mới.
  + Lưu thông tin sinh viên
  + Sửa thông tin sinh viên.
  + Hủy thông tin sinh viên.
  + Xóa thông tin sinh viên.
  + Ghi nhận đóng tiền.
  + Xem lịch sử đóng tiền
  + Xem danh sách tất cả sinh viên.
  + Tìm kiếm sinh viên theo MSSV hoặc họ tên
  + Lọc sinh viên theo phòng
* Người dùng sinh viên
  + Xem thông tin cá nhân
  + Thay đổi thông tin cá nhân
    - Mật khẩu
    - Số điện thoại
  + Xem lịch sử đóng tiền

Liên kết các nút bấm trên giao diện Tkinter với các hàm Python tương ứng để xử lý dữ liệu và tương tác với cơ sở dữ liệu.

**Viết mã Python kết nối cơ sở dữ liệu.**

Sử dụng pyodbc để thiết lập kết nối với cơ sở dữ liệu MySQL

Gồm 5 module chính:

* Login\_form: Giao diện đăng nhập và điều phối, khởi chạy ứng dụng, xác thực tài khoản và phân quyền người dùng (Quản lý/Sinh viên) để mở giao diện tương ứng.
* Admin \_dashboard: Module quản lý và nghiệp vụ, giao diện chính cho admin thực hiện các thao tác CRUD sinh viên, quản lý phân phòng, ghi nhận đóng tiền, và xem lịch sử thanh toán.
* Student\_dashboard: Module sinh viên và tra cứu, giao diện cho sinh viên. Cho phép tra cứu thông tin cá nhân, cập nhật SĐT/Mật khẩu, và kiểm tra trạng thái đóng tiền tháng hiện tại.
* Db\_connector: Kết nối cơ sở dữ liệu, chứa chuỗi kết nối đến SQL Server và hàm check\_credentials\_in\_sqlserver để thực hiện xác thực tài khoản.
* App\_styles: Định nghĩa phong cách, chứa các hằng số định nghĩa font chữ (FONT\_TITLE, FONT\_LABEL) dựa trên Times New Roman để đảm bảo tính nhất quán cho toàn bộ giao diện Tkinter.

Gồm các hàm:

* Thêm sinh viên: Lấy dữ liệu từ các trường nhập liệu trên giao diện và chèn vào bảng SinhVien
* Lưu thông tin sinh viên: Tùy thuộc vào chế độ (add hoặc edit), thực hiện thêm hoặc cập nhật dữ liệu vào bảng SinhVien và TaiKhoan. Cập nhật trạng thái phòng cũ/mới nếu có thay đổi.
* Sửa thông tin sinh viên: Sửa đổi thông tin sinh viên trong cơ sở dữ liệu dựa trên mã số sinh viên.
* Hủy thông tin sinh viên: Đặt lại (reset) tất cả các trường nhập liệu trên giao diện về trạng thái rỗng và đưa các nút bấm về trạng thái ban đầu (init).
* Xóa thông tin sinh viên: Xóa sinh viên khỏi cơ sở dữ liệu dựa trên mã số sinh viên.
* Xem danh sách tất cả sinh viên: Truy vấn cơ sở dữ liệu để lấy tất cả sinh viên và hiển thị chúng trong bảng trên giao diện.
* Tìm kiếm sinh viên: Gửi câu lệnh đến cơ sở dữ liệu để truy vấn bản ghi khớp với điều kiện tìm kiếm và hiển thị kết quả trong danh sách.
* Ghi nhận đóng tiền: Kiểm tra trùng lặp thanh toán, thực hiện thêm vào bảng LichSuDongTien và cập nhật cột trạng thái trên thông tin cá nhân của sinh viên đó thành "Đã đóng".
* Xem lịch sử đóng tiền: Truy vấn vào bảng LichSuDongTien của cơ sở dữ liệu và sắp xếp theo thời gian (năm và tháng giảm dần) để hiển thị các giao dịch.
* Tìm kiếm sinh viên theo mã số sinh viên (MSSV) hoặc họ tên: Thực hiện truy vấn vào bảng SinhVien để tìm kiếm theo mã số sinh viên và họ tên phù hợp.
* Lọc sinh viên theo phòng: Thực hiện truy vấn vào cơ sở dữ liệu để chỉ hiển thị các sinh viên có mã phòng khớp chính xác với phòng đã chọn.
* Cập nhật trạng thái: Cập nhật trạng thái (Đầy / Trống) của một phòng trong bảng Phong. Hàm này được gọi sau mỗi thao tác Thêm, Xóa, hoặc Sửa phòng của sinh viên.
* Phân phòng: Lọc và cập nhật danh sách các phòng "Trống" và phù hợp với Giới tính của sinh viên vào Combobox Mã phòng.
* Quản lý giao diện: Quản lý trạng thái bật/tắt (normal/disabled) của các nút bấm (thêm, lưu, sửa, xóa) và các trường nhập liệu (Entry, Combobox), đảm bảo luồng CRUD chính xác.
* Thao tác bảng: Xử lý sự kiện khi người dùng nhấp chọn một hàng trên bảng Treeview. Hàm này chịu trách nhiệm lấy dữ liệu của sinh viên được chọn và đổ (load) vào các ô nhập liệu.
* Thao tác ứng dụng: Xóa thông tin đăng nhập, đóng cửa sổ dashboard hiện tại, và quay trở lại cửa sổ login\_form.
* Nghiệp vụ sinh viên: Chịu trách nhiệm cập nhật số điện thoại và mật khẩu mới của sinh viên vào các bảng SinhVien và TaiKhoan.

# **TÀI LIỆU THAM KHẢO**

Ts. Huỳnh Phước Hải, T. N. (2023). *Giáo trình lập trình python.* An Giang:

Đại học Quốc gia thành phố Hồ Chí Minh.

Trợ lý ảo gemini.

Bài toán quản lý bài hát đặt ra trong đề tài này xuất phát từ nhu cầu kiểm soát và cá nhân hóa thư viện âm nhạc số. Bối cảnh chính là sự dịch chuyển từ các ứng dụng quản lý tập trung, đa chức năng (như iTunes, Windows Media Player) sang các giải pháp **gọn nhẹ, mã nguồn mở** và **chuyên biệt hơn**.

Nội dung bài toán đặt ra cần giải quyết bao gồm:

1. **Xây dựng một giao diện người dùng (GUI)** hiện đại, trực quan, cho phép người dùng dễ dàng duyệt, tìm kiếm và tương tác với bài hát.
2. **Quản lý metadata** (siêu dữ liệu) của bài hát một cách có hệ thống (Tên bài hát, Nghệ sĩ, Album, Thể loại, Thời lượng).
3. **Đảm bảo tính ổn định và bảo mật** của dữ liệu thông qua việc sử dụng một hệ quản trị cơ sở dữ liệu (HQTCSDL) mạnh mẽ.

**2.2. Lịch Sử Giải Quyết Vấn Đề**

**2.2.1. Lịch sử quản lý thư viện số**

Vấn đề quản lý thư viện nhạc đã được giải quyết từ lâu với sự ra đời của các ứng dụng tiêu biểu như **Winamp, Windows Media Player** (Microsoft), và nổi bật nhất là **iTunes** (Apple). Các ứng dụng này cung cấp chức năng quản lý, phân loại và phát nhạc. Tuy nhiên, các giải pháp này thường có tồn tại:

* Phụ thuộc vào hệ điều hành hoặc nền tảng phần cứng nhất định.
* Cấu trúc phức tạp, khó mở rộng hoặc tùy chỉnh.
* Dữ liệu thường được lưu cục bộ hoặc phụ thuộc vào máy chủ của nhà cung cấp, thiếu sự linh hoạt trong việc trích xuất và quản lý dữ liệu gốc.

**2.2.2. Sự cần thiết của giải pháp mới**

Với sự phát triển của lập trình Python, xu hướng xây dựng các công cụ **tùy chỉnh và tối ưu** đã xuất hiện. Các dự án mã nguồn mở sử dụng Tkinter (hoặc PyQt, Kivy) kết hợp với các HQTCSDL quan hệ đã chứng minh được tính hiệu quả trong việc tạo ra các ứng dụng desktop nhẹ, đáp ứng các yêu cầu cụ thể của người dùng, tránh phụ thuộc vào các giải pháp thương mại.

**2.3. Phạm Vi của Đề Tài**

Đề tài tập trung giải quyết các vấn đề trong phạm vi sau:

* **Công nghệ sử dụng:** Giới hạn trong bộ ba **Python**, thư viện GUI **Tkinter** và HQTCSDL **PostgreSQL**.
* **Chức năng chính:** Tập trung vào các chức năng cơ bản của một ứng dụng quản lý bài hát: **CRUD** (Create, Read, Update, Delete) cho thông tin Bài hát, Album và Nghệ sĩ.
* **Quản lý người dùng:** Bao gồm chức năng **Đăng nhập/Đăng ký** để quản lý phiên làm việc và tùy chọn cá nhân hóa.
* **Giới hạn:** Không bao gồm các tính năng phức tạp như phát nhạc nâng cao (chỉ phát cơ bản), đồng bộ hóa đám mây (cloud synchronization) hoặc xử lý tín hiệu số chuyên sâu.

**2.4. Phương Pháp Nghiên Cứu và Hướng Giải Quyết**

Đề tài áp dụng kết hợp **Phương pháp Nghiên cứu Lý thuyết** và **Phương pháp Phát triển Thực nghiệm**.

**2.4.1. Phương pháp Nghiên cứu Lý thuyết**

* **Thu thập tài liệu:** Nghiên cứu về kiến trúc ứng dụng desktop, mô hình CSDL quan hệ (PostgreSQL), và các thư viện Python chuyên dụng.
* **Phân tích và Tổng hợp:** Phân tích các ưu điểm và nhược điểm của các công nghệ liên quan để chọn lựa và tích hợp tối ưu (ví dụ: lý do chọn PostgreSQL thay vì các CSDL khác).

**2.4.2. Phương pháp Phát triển Thực nghiệm (Mô hình lặp)**

* Áp dụng mô hình phát triển lặp: **Thiết kế CSDL -> Thiết kế GUI -> Viết mã kết nối CSDL và xử lý logic -> Kiểm thử và Lặp lại**.
* **Hướng giải quyết:** Sử dụng cách tiếp cận **Module hóa** trong Python, chia ứng dụng thành các thành phần quản lý riêng biệt như Database, Music, Pages/HomePage, Pages/MusicPage, v.v. để dễ dàng bảo trì và mở rộng.

**2.5. Trình Bày Tóm Tắt Cơ Sở Lý Thuyết**

**2.5.1. Ngôn ngữ Lập trình Python**

* **Khái niệm:** Python là ngôn ngữ lập trình đa năng, thông dịch, hướng đối tượng, nổi bật với cú pháp rõ ràng, dễ đọc, và thư viện tiêu chuẩn phong phú.
* **Vai trò:** Là ngôn ngữ cốt lõi để xây dựng logic ứng dụng, xử lý dữ liệu và kết nối giữa giao diện (Tkinter) và cơ sở dữ liệu (PostgreSQL).

**2.5.2. Thư viện Tkinter**

* **Khái niệm:** Tkinter là giao diện lập trình ứng dụng (API) chuẩn của Python để tạo ra ứng dụng giao diện người dùng đồ họa (GUI) desktop.
* **Vai trò:** Được sử dụng để thiết kế toàn bộ giao diện, bao gồm các thành phần hiện đại (custom components) và quản lý luồng chuyển trang trong ứng dụng.

**2.5.3. Hệ quản trị Cơ sở dữ liệu PostgreSQL**

* **Khái niệm:** PostgreSQL (còn gọi là Postgres) là một HQTCSDL quan hệ đối tượng (ORDBMS) mã nguồn mở tiên tiến, nổi tiếng về độ tin cậy, tính năng mạnh mẽ và khả năng mở rộng tuân thủ chuẩn SQL.
* **Vai trò:** Được sử dụng để lưu trữ và quản lý tất cả dữ liệu ứng dụng một cách bền vững, bao gồm thông tin bài hát, album, nghệ sĩ, và thông tin người dùng. Để kết nối Python với PostgreSQL, thư viện **psycopg2** là lựa chọn phổ biến nhất và sẽ được sử dụng trong đề tài này (thay thế cho mysql.connector trong ngữ cảnh ban đầu).

**2.5.4. Mô hình Cơ sở dữ liệu Quan hệ**

* **Khái niệm:** Là mô hình tổ chức dữ liệu dựa trên các bảng (relations), trong đó dữ liệu được liên kết thông qua các khóa (Primary Key và Foreign Key), đảm bảo tính toàn vẹn và nhất quán của dữ liệu.
* **Vai trò:** Áp dụng mô hình này để thiết kế các bảng như Tracks, Albums, Artists và Users, tối ưu hóa việc truy vấn thông tin phức tạp (ví dụ: tìm tất cả bài hát của một nghệ sĩ cụ thể).

**3.Kết quả đạt được và Demo Code**

**3.1. Tổng Quan về Kết Quả Đạt Được**

Sau quá trình nghiên cứu và phát triển, đề tài đã hoàn thành thành công các mục tiêu đã đề ra, xây dựng được một ứng dụng quản lý bài hát hoàn chỉnh trên nền tảng Python, Tkinter và PostgreSQL. Ứng dụng không chỉ là một công cụ quản lý dữ liệu cơ bản mà còn cung cấp trải nghiệm người dùng hiện đại, module hóa.

Các kết quả đạt được khẳng định tính khả thi của việc kết hợp các công nghệ Mã nguồn mở (Python, Tkinter) và hệ quản trị cơ sở dữ liệu mạnh mẽ (PostgreSQL) để giải quyết bài toán quản lý thư viện âm nhạc cá nhân.

**3.2. Các Kết Quả Chi Tiết**

**3.2.1. Thiết Kế và Triển Khai Giao Diện Người Dùng (GUI)**

* **GUI Chuyên nghiệp bằng Tkinter:** Ứng dụng đã được phát triển với một kiến trúc giao diện hiện đại, tối ưu hóa trải nghiệm người dùng desktop, sử dụng thư viện Tkinter và các thành phần tùy chỉnh (modern\_components.py).
* **Hệ thống Chuyển trang (Navigation):** Triển khai thành công hệ thống module hóa với các trang chính bao gồm **Home, Browse, Search, Music Page, Album Page** và **Artist Page**.
* **Thanh điều khiển nhạc (Player Bar):** Phát triển thanh điều khiển cố định ở phía dưới (bottom bar) cho phép người dùng tương tác trực tiếp với các chức năng phát nhạc cơ bản (Play, Pause, Next, Previous, Volume).

**3.2.2. Triển Khai Cơ Sở Dữ Liệu PostgreSQL**

* **Thiết lập CSDL:** Đã thiết lập thành công kết nối và cấu trúc dữ liệu trên hệ quản trị PostgreSQL, đảm bảo khả năng lưu trữ ổn định và truy xuất dữ liệu hiệu suất cao.
* **Cấu trúc Schema Rõ ràng:** Xây dựng thành công schema CSDL quan hệ, bao gồm các bảng chính để quản lý:
  + **Bài hát (Tracks):** Lưu trữ metadata chi tiết (tên, nghệ sĩ, album, thời lượng).
  + **Người dùng (Users):** Lưu trữ thông tin đăng nhập và quản lý phiên.
  + **Album/Artist:** Hỗ trợ phân loại và duyệt nhạc theo nghệ sĩ/album.

**3.2.3. Triển Khai Chức Năng Cốt Lõi và Nâng Cao**

* **Chức năng CRUD (Quản lý Bài hát):** Đã triển khai đầy đủ các thao tác cơ bản để **Thêm, Đọc, Cập nhật** và **Xóa** thông tin bài hát, cho phép người dùng duy trì và cập nhật thư viện nhạc cá nhân.
* **Quản lý Tài khoản Người dùng:** Xây dựng module **Đăng nhập và Đăng ký** (UserAuthentication) để cá nhân hóa trải nghiệm và quản lý phiên người dùng (user\_session.py), cho phép ứng dụng nhận diện và lưu trữ tùy chọn cá nhân.
* **Tính năng Tìm kiếm và Lọc:** Đã triển khai chức năng **Tìm kiếm** chính xác và phân loại bài hát theo các bộ lọc khác nhau (FilterFrame.py), giúp người dùng dễ dàng định vị các bài hát trong thư viện lớn.

**3.3. Đánh Giá và Hướng Phát Triển**

**3.3.1. Đánh giá**

* **Ưu điểm:** Ứng dụng hoạt động ổn định, GUI trực quan và dễ sử dụng. Việc sử dụng PostgreSQL đã mang lại hiệu suất truy vấn dữ liệu tốt và bảo mật cao. Kiến trúc module hóa giúp mã nguồn dễ đọc và bảo trì.
* **Hạn chế:** Do giới hạn phạm vi đề tài, ứng dụng chưa có các tính năng phức tạp như đồng bộ đám mây, chỉnh sửa metadata nâng cao cho tệp nhạc (chỉ quản lý metadata trong CSDL) hoặc hệ thống playlist tùy chỉnh hoàn toàn.

**3.3.2. Hướng phát triển**

Trong tương lai, ứng dụng có thể được nâng cấp với các tính năng:

* **Phân tích Tệp nhạc:** Tích hợp thư viện để tự động đọc và cập nhật metadata (ID3 tags) từ tệp nhạc khi thêm mới.
* **Playlist Động:** Xây dựng hệ thống playlist thông minh (Smart Playlists) dựa trên tiêu chí của người dùng.
* **Báo cáo Thống kê:** Phát triển chức năng báo cáo thống kê về tần suất nghe, thể loại yêu thích, v.v.

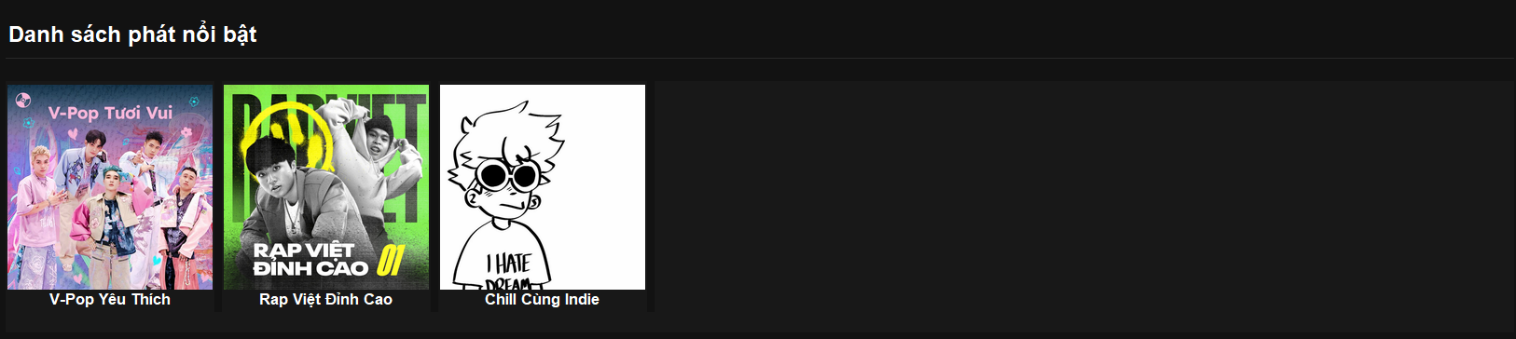
**A.THIẾT KẾ GIAO DIỆN QUẢN LÝ BÀI HÁT**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**B.CÁC BƯỚC XÂY DỰNG**

Để xây dựng ứng dụng Quản lý Bài hát bằng Python, Tkinter và PostgreSQL, đề tài đã thực hiện một quy trình phát triển module hóa, được chia thành các bước chính sau:

**1. Chuẩn bị Môi trường và Công cụ Phát triển**

Bước này nhằm đảm bảo môi trường làm việc đã sẵn sàng để phát triển ứng dụng:

* **Cài đặt Python và PostgreSQL:** Đảm bảo phiên bản Python (ví dụ: Python 3.x) và Máy chủ PostgreSQL đã được cài đặt và cấu hình sẵn trên máy tính.
* **Cài đặt Thư viện:** Sử dụng trình quản lý gói pip để cài đặt các thư viện Python cần thiết:
  + **psycopg2** (hoặc thư viện tương đương): Dùng để thiết lập kết nối giữa Python và PostgreSQL (thay thế cho mysql-connector-python trong đề tài gốc).
  + **tkinter (tiêu chuẩn):** Thư viện GUI chính.
  + **Các thư viện âm nhạc (ví dụ: pygame hoặc thư viện xử lý metadata):** Để xử lý các tệp âm thanh và tính năng phát nhạc cơ bản.

**2. Thiết kế Cơ sở Dữ liệu (CSDL) PostgreSQL**

Thiết kế CSDL là nền tảng cốt lõi để lưu trữ và quản lý thông tin bài hát:

* **Tạo Database:** Tạo một cơ sở dữ liệu mới trong PostgreSQL, ví dụ: qlbaihat (tương tự qlnhanvien trong yêu cầu cũ).
* **Thiết kế Schema:** Xây dựng cấu trúc các bảng quan hệ để quản lý dữ liệu:
  + **Bảng Users:** Lưu trữ thông tin người dùng (cho chức năng Đăng nhập/Đăng ký).
  + **Bảng Tracks:** Lưu trữ thông tin chi tiết của bài hát (tên, nghệ sĩ, album, đường dẫn tệp, thời lượng).
  + **Bảng Albums và Artists:** Liên kết với bảng Tracks bằng Foreign Key để phân loại dữ liệu.
* **Viết mã Khởi tạo Schema:** Viết các script Python (init\_schema.py) để tự động tạo các bảng và cấu trúc CSDL khi ứng dụng chạy lần đầu.

**3. Xây dựng Khung Giao diện Ứng dụng (GUI Framework)**

Tập trung vào cấu trúc cơ bản và tính thẩm mỹ của ứng dụng:

* **Khởi tạo Cửa sổ chính:** Sử dụng Tkinter để tạo cửa sổ chính (root window) và cấu hình tiêu đề, kích thước, và các thuộc tính cơ bản (main.py).
* **Thiết kế Theme và Components:** Xây dựng các thành phần giao diện tùy chỉnh (modern\_components.py) và định nghĩa theme màu sắc (theme.py) để đảm bảo giao diện thống nhất, đẹp mắt.
* **Cấu trúc Module Page:** Xây dựng hệ thống quản lý trang (Page Management) để dễ dàng chuyển đổi giữa các màn hình (Home, MusicPage, SearchPage, v.v.).

**4. Triển khai các Module Chức năng Chính**

**4.1. Module Kết nối CSDL và CRUD**

* **Thiết lập Kết nối:** Viết hàm kết nối cơ sở dữ liệu PostgreSQL (connect\_db) trong module Database.py.
* **Xây dựng Data Model:** Tạo các lớp (Class) Python tương ứng với các bảng trong CSDL (ví dụ: Track class trong Music/track.py).
* **Ánh xạ CRUD:** Viết các hàm (method) để thực hiện các thao tác **Thêm, Sửa, Xóa, Truy vấn** dữ liệu bài hát, album, và nghệ sĩ, sử dụng các câu lệnh SQL được thực thi qua psycopg2.

**4.2. Module Đăng nhập và Quản lý Phiên**

* **Triển khai Giao diện Xác thực:** Xây dựng các trang AuthBase, Frame1 (Đăng nhập), Frame2 (Đăng ký) bằng Tkinter.
* **Quản lý Phiên:** Xây dựng class UserSession để lưu trữ trạng thái đăng nhập của người dùng, đảm bảo ứng dụng chỉ hiển thị nội dung cá nhân hóa khi người dùng đã xác thực.

**4.3. Module Phát triển Giao diện Chức năng**

* **Trang Quản lý Bài hát (MusicPage):** Thiết kế bảng hiển thị bài hát, tích hợp các nút chức năng (Thêm, Sửa, Xóa) và các thành phần lọc/tìm kiếm (FilterFrame).
* **Trang Tìm kiếm (SearchPage):** Xây dựng thanh tìm kiếm và hiển thị kết quả theo thời gian thực.
* **Thanh điều khiển Phát nhạc (bottomMusicPage):** Thiết kế thanh cố định ở dưới đáy ứng dụng, bao gồm các nút Play/Pause, Next/Previous và thanh trượt âm lượng.

**5. Tích hợp và Kiểm thử**

* **Tích hợp:** Liên kết các sự kiện trên giao diện (như nhấn nút "Thêm", "Sửa", "Tìm kiếm") với các hàm CRUD đã được viết trong module Database và Music.
* **Kiểm thử đơn vị và tích hợp:** Thực hiện kiểm tra từng module nhỏ (hàm CRUD, hàm kết nối CSDL) và sau đó kiểm tra toàn bộ luồng ứng dụng (ví dụ: đăng nhập thành công có chuyển đến trang Home không, thêm bài hát có được lưu vào PostgreSQL không).
* **Hoàn thiện:** Chạy ứng dụng và sửa lỗi để đảm bảo tính ổn định và đáp ứng đầy đủ các mục tiêu đề ra ban đầu.

**C.HƯỚNG DẪN CÀI ĐẶT VÀ CODE ỨNG DỤNG**

Tuyệt vời! Tôi sẽ giúp bạn phác thảo chi tiết phần **HƯỚNG DẪN CÀI ĐẶT VÀ CODE ỨNG DỤNG** cho báo cáo của bạn, tập trung vào việc chuyển đổi từ MySQL sang **PostgreSQL** và cấu trúc của ứng dụng quản lý bài hát.

**D. HƯỚNG DẪN CÀI ĐẶT VÀ CODE ỨNG DỤNG**

Phần này mô tả các bước cần thiết để thiết lập môi trường và cấu trúc mã nguồn đã triển khai.

**1. HƯỚNG DẪN CÀI ĐẶT MÔI TRƯỜNG**

Để ứng dụng Quản lý Bài hát có thể chạy được, cần cài đặt môi trường phát triển (Python) và máy chủ Cơ sở dữ liệu (PostgreSQL).

**1.1. Cài đặt các Công cụ Nền tảng**

* **Python:** Đảm bảo đã cài đặt Python (phiên bản 3.8+ được khuyến nghị).
* **PostgreSQL Server:** Cài đặt và cấu hình máy chủ PostgreSQL trên máy tính của bạn. Tạo một người dùng (user) và mật khẩu (password) riêng cho ứng dụng.

**1.2. Cài đặt các Thư viện Python**

Sau khi có Python, mở Terminal (hoặc Command Prompt) và chạy lệnh sau để cài đặt tất cả các gói phụ thuộc cần thiết.

* **Kết nối PostgreSQL:** Thư viện **psycopg2** được sử dụng để kết nối Python với PostgreSQL, thay thế cho mysql-connector-python trong yêu cầu ban đầu.

Bash

# Cài đặt Psycopg2 cho PostgreSQL và các thư viện khác (ví dụ: pygame, Tkinter)

pip install psycopg2-binary # Sử dụng binary nếu gặp lỗi biên dịch

# Giả định các thư viện khác trong requirements.txt bao gồm:

# pip install pygame Pillow

**Lưu ý:** Tkinter thường đi kèm với bản cài đặt Python trên Windows và macOS, nhưng trên Linux (như Ubuntu), có thể cần cài đặt thêm gói python3-tk.

**1.3. Khởi tạo Cơ sở Dữ liệu (CSDL)**

Sau khi cài đặt thư viện và có thông tin kết nối, ứng dụng cần chạy script để tạo cấu trúc bảng:

* Mở công cụ quản lý CSDL của PostgreSQL (như pgAdmin hoặc terminal).
* Tạo Database mới, ví dụ: db\_music\_manager.
* Ứng dụng sẽ tự động chạy script (init\_schema.py) để tạo các bảng như Users, Tracks, Albums, và Artists khi khởi chạy lần đầu.

**2. CẤU TRÚC CODE ỨNG DỤNG VÀ CHỨC NĂNG**

Mã nguồn được tổ chức theo kiến trúc module hóa để dễ dàng quản lý và phát triển.

**2.1. Thiết lập Kết nối PostgreSQL (Tương tự connect\_db)**

Module Database.py chứa hàm thiết lập kết nối tới PostgreSQL.

