**TRƯỜNG ĐẠI HỌC AN GIANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN**

**Chuyên đề Python (COS525)**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÁN SÁCH VỚI PYTHON, TKINTER VÀ MYSQL**

Giảng viên hướng dẫn: Ths. Nguyễn Ngọc Minh

Sinh viên thực hiện: <DTH235733>.<Lý Bá Phú>.<LớpDH24TH2\_NhómTH2\_TổTH2>

<DTH235694>.<Huỳnh Phú Lợi>.<LớpDH24TH2\_NhómTH2\_TổTH2>

**AN GIANG, 26-11-2025**

Mục lục

[I.Đặt vấn đề: 3](#_Toc214811942)

[**I.1** : Với đề tài quản lý sách, chúng ta thấy được tính cần thiết của đề tài: 3](#_Toc214811943)

[**I.2:** Mặc dù hệ thống đã giải quyết được các bài toán nghiệp vụ cơ bản: 4](#_Toc214811944)

[I.3: Sự cần thiết của Đề tài 6](#_Toc214811945)

[I.4: Ý nghĩa về mặt Lý luận (Theoretical Significance) 6](#_Toc214811946)

[I.5: Ý nghĩa về mặt Thực tiễn (Practical Significance) 7](#_Toc214811947)

[I.6: Cơ sở để Cụ thể hóa Mục tiêu 8](#_Toc214811948)

[I.7: Các Định hướng Nội dung chính (Đã thực hiện) 9](#_Toc214811949)

[I.8: Mục tiêu Nghiên cứu của Đề tài 10](#_Toc214811950)

[II. Tổng quan và cơ sở lý thuyết: 11](#_Toc214811951)

[II.1. Bối cảnh xuất hiện vấn đề 11](#_Toc214811952)

[II.2. Nội dung và Bài toán đặt ra 12](#_Toc214811953)

[II.3 Lịch sử giải quyết vấn đề 13](#_Toc214811954)

[II.4: Phạm vi Giải quyết của Đề tài: 15](#_Toc214811955)

[II.5. Phạm vi Không Giải quyết: 16](#_Toc214811956)

[II.6: Phương pháp Nghiên cứu Lý thuyết 16](#_Toc214811957)

[II.7: Phương pháp Điều tra & Khảo sát (Investigative Research) 17](#_Toc214811958)

[II.8: Phương pháp Nghiên cứu Thực nghiệm (Experimental Research) 18](#_Toc214811959)

[II.9: tóm tắt cơ sở lý thuyết được sử dụng để giải quyết đề tài, tập trung vào ba trụ cột chính: 18](#_Toc214811960)

[II.10: Phân tích Kỹ năng Lập trình Python trong Giải quyết Vấn đề Thực tế 20](#_Toc214811961)

## I.Đặt vấn đề:

### **I.1** : Với đề tài quản lý sách, chúng ta thấy được tính cần thiết của đề tài:

Trong bối cảnh thị trường bán lẻ sách ngày càng cạnh tranh, việc chuyển đổi từ phương thức quản lý thủ công sang sử dụng công nghệ trở thành một yêu cầu cấp thiết đối với các nhà sách. Quá trình quản lý truyền thống bộc lộ nhiều yếu điểm, ảnh hưởng trực tiếp đến hiệu quả kinh doanh và khả năng cạnh tranh.

Do đó, việc xây dựng một **"Hệ thống Quản lý Bán sách"** là vô cùng cần thiết, xuất phát từ những nhu cầu thực tiễn sau:

**1. Giải quyết Bất cập của Quản lý Thủ công**

* **Tốn thời gian và công sức:** Nhân viên phải ghi chép mọi giao dịch, tồn kho bằng sổ sách hoặc các tệp Excel rời rạc. Quá trình này tốn thời gian, dễ mệt mỏi và làm giảm tốc độ phục vụ khách hàng.
* **Dễ sai sót và khó kiểm soát:** Việc nhập liệu bằng tay có rủi ro cao về sai số (nhầm giá, sai số lượng). Khi kiểm kho cuối ngày hoặc cuối tháng, việc đối chiếu giữa sổ sách và thực tế là một "thảm họa", dẫn đến thất thoát hàng hóa và tài chính không rõ nguyên nhân.
* **Không có dữ liệu tức thì (Real-time):** Người quản lý không thể biết ngay lập tức lượng tồn kho của một cuốn sách là bao nhiêu, hoặc doanh thu tại thời điểm hiện tại là bao nhiêu. Mọi thông tin đều có độ trễ, khiến việc ra quyết định kinh doanh (như nhập hàng mới, điều chỉnh giá) trở nên chậm chạp và thiếu chính xác.

**2. Nâng cao Hiệu quả Vận hành**

* **Tự động hóa quy trình:** Một hệ thống phần mềm sẽ tự động hóa các nghiệp vụ cốt lõi:
  + **Thanh toán nhanh:** Tự động tính tổng tiền, áp dụng giảm giá, tích điểm.
  + **Trừ kho tự động:** Số lượng sách trong CSDL được cập nhật ngay khi hóa đơn được xuất, đảm bảo số liệu tồn kho luôn chính xác.
* **Quản lý tồn kho hiệu quả:** Hệ thống cho phép tra cứu nhanh số lượng, vị trí, giá cả của bất kỳ cuốn sách nào. Nó cũng đưa ra cảnh báo khi sách sắp hết hàng, giúp chủ nhà sách chủ động trong việc tái nhập hàng, tránh tình trạng "đứt hàng" (out-of-stock) các đầu sách bán chạy.

**3. Cải thiện Trải nghiệm Khách hàng**

* **Thanh toán nhanh và chuyên nghiệp:** Thay vì chờ nhân viên tính toán thủ công, khách hàng được thanh toán nhanh chóng, hóa đơn rõ ràng (qua file .txt hoặc in).
* **Chăm sóc khách hàng (CRM):** Tính năng tích điểm qua SĐT (như trong đồ án) là một công cụ marketing hiệu quả. Nó khuyến khích khách hàng quay lại mua sắm để hưởng ưu đãi, tăng cường lòng trung thành của khách hàng – điều mà quản lý thủ công không thể làm được một cách có hệ thống.

**4. Phân quyền và Bảo mật Dữ liệu**

* **Kiểm soát nội bộ:** Quản lý thủ công không thể ngăn chặn việc nhân viên gian lận hoặc xem các thông tin nhạy cảm.
* **Hệ thống phân quyền:** Đề tài giải quyết vấn đề này bằng cách phân rõ vai trò "chủ" và "nhanvien". Nhân viên chỉ có thể thực hiện nghiệp vụ bán hàng, trong khi chủ cửa hàng có thể truy cập các chức năng quản trị (như quản lý nhân viên) và xem các báo cáo tổng thể (quản lý hóa đơn).

### **I.2:** Mặc dù hệ thống đã giải quyết được các bài toán nghiệp vụ cơ bản:

(Bán hàng, Quản lý kho, Tích điểm, Quản lý nhân viên) và đáp ứng mục tiêu ban đầu, dự án vẫn tồn tại một số hạn chế mang tính kỹ thuật và nghiệp vụ. Nếu được triển khai trong môi trường kinh doanh thực tế, các hạn chế này cần được khắc phục để đảm bảo tính ổn định, bảo mật và khả năng mở rộng.

**1. Hạn chế về Kỹ thuật và Kiến trúc**

Đây là những hạn chế cốt lõi liên quan đến công nghệ và thiết kế của hệ thống:

* Lỗ hổng Bảo mật Mật khẩu: Hạn chế nghiêm trọng nhất là mật khẩu đang được lưu dưới dạng văn bản thuần (plaintext) trong CSDL (cả trong code SQL và logic INSERT của Python). Bất kỳ ai có quyền truy cập vào CSDL (kể cả quản trị viên) đều có thể thấy mật khẩu của tất cả nhân viên.

Thực tế: Các hệ thống chuyên nghiệp buộc phải băm (hash) mật khẩu bằng các thuật toán như SHA-256 trước khi lưu.

* Rủi ro về Truy cập Đồng thời (Concurrency): Hệ thống không sử dụng Database Transactions (Giao dịch CSDL) khi thanh toán. Điều này tạo ra một kịch bản lỗi nghiêm trọng:

Ví dụ: Kho chỉ còn 1 cuốn sách "S001". Hai nhân viên (trên 2 máy khác nhau) cùng lúc thêm cuốn sách này vào giỏ hàng và cùng nhấn "Thanh toán". Cả hai đều vượt qua bước kiểm tra sl > ton. Hệ thống sẽ thực thi hai lệnh UPDATE liên tiếp, dẫn đến tồn kho bị âm (-1), gây sai lệch dữ liệu.

* Kiến trúc "Fat Client" (Ứng dụng Desktop):
  + Khó bảo trì/cập nhật: Toàn bộ logic nghiệp vụ (cách tính điểm, giảm giá,...) nằm trên máy của nhân viên. Khi cần thay đổi một quy tắc (ví dụ: đổi điểm tích lũy từ 2 lên 3 điểm), lập trình viên phải cài đặt lại phiên bản mới trên tất cả các máy tính trong nhà sách.
  + Phụ thuộc vào Hệ điều hành: Code Tkinter và các thư viện (PIL, Pymysql) chỉ được tối ưu và cài đặt cho một môi trường nhất định (ví dụ: Windows).
  + Không thể truy cập từ xa: Người chủ không thể xem báo cáo doanh thu từ điện thoại hoặc laptop ở nhà. Họ bắt buộc phải có mặt tại cửa hàng và sử dụng đúng máy tính đó.

**2. Hạn chế về Quy trình Nghiệp vụ**

Đây là những tính năng còn thiếu để hoàn thiện một quy trình bán lẻ chuẩn:

* Thiếu nghiệp vụ Nhập kho (Inbound): Hệ thống chỉ xử lý "bán ra" (xuất kho). Hoàn toàn không có chức năng "nhập vào" (nhập kho). Khi sách mới về từ nhà cung cấp, nhân viên không có giao diện để tạo "Phiếu nhập kho", ghi nhận giá vốn, tên nhà cung cấp, và tự động cộng số lượng vào tồn kho.

Hệ quả: Cách duy nhất để thêm hàng là vào tab "Quản lý Sách" và sửa thủ công số lượng, một quy trình rất dễ sai sót và không lưu lại lịch sử nhập hàng.

* Thiếu hệ thống Báo cáo & Phân tích (OLAP/BI): Tab "Quản lý Hóa đơn" hiện tại chỉ là một "bản ghi chép" (Log) để xem lại *chi tiết* từng giao dịch (OLTP). Nó không phải là một "báo cáo" (Report).

Hệ quả: Người chủ không thể trả lời các câu hỏi quản trị quan trọng như: "Doanh thu hôm nay/tuần này/tháng này là bao nhiêu?", "Top 10 cuốn sách bán chạy nhất là gì?", "Những cuốn nào sắp hết hàng cần nhập thêm?".

* Không xử lý các "Quy trình Ngoại lệ":
  + Trả hàng: Không có chức năng cho phép khách hàng trả lại sách đã mua.
  + Hủy đơn: Khi nhân viên đã lỡ nhấn "Thanh toán" nhưng khách đổi ý, không có cách nào để hoàn tác (rollback) giao dịch, dẫn đến việc phải điều chỉnh kho và sổ sách thủ công.
  + Điều chỉnh Kho: Không có nghiệp vụ xử lý hàng hư hỏng, mất cắp, hoặc kiểm kho thực tế bị lệch so với phần mềm.

**3. Hạn chế về Giao diện và Trải nghiệm (UI/UX)**

* Không tích hợp Phần cứng: Hạn chế lớn nhất về mặt vận hành là hệ thống không hỗ trợ máy quét mã vạch (Barcode Scanner).

Hệ quả: Nhân viên phải gõ tay từng mã sách (S001, S002...) khi bán hàng. Quy trình này rất chậm, dễ gõ nhầm mã, và thiếu chuyên nghiệp, gây ùn tắc tại quầy thu ngân.

* Giao diện Tĩnh: Giao diện Tkinter có tính thẩm mỹ không cao và là giao diện "tĩnh". Nó có thể bị vỡ, co giãn không đồng đều nếu chạy trên các màn hình có độ phân giải khác nhau.

### I.3: Sự cần thiết của Đề tài

Trong bối cảnh nền kinh tế thị trường và sự phát triển của công nghệ, các nhà sách (từ quy mô nhỏ lẻ đến các chuỗi cửa hàng) đang đối mặt với áp lực cạnh tranh gay gắt. Phương thức quản lý thủ công truyền thống (ghi chép sổ sách, sử dụng Excel rời rạc) ngày càng bộc lộ nhiều yếu điểm chí mạng:

* **Sai sót và Thất thoát:** Việc tính toán thủ công, ghi chép tồn kho bằng tay có rủi ro sai sót rất cao (nhầm giá, sai số lượng), dẫn đến thất thoát hàng hóa và tài chính không thể kiểm soát.
* **Quá tải và Chậm trễ:** Khi lượng sách lên đến hàng ngàn đầu mục và lượng giao dịch hàng trăm lượt mỗi ngày, việc tra cứu, thanh toán và kiểm kho thủ công trở nên quá tải, gây ùn tắc tại quầy thu ngân và làm giảm trải nghiệm khách hàng.
* **Thiếu Dữ liệu Tức thời:** Người quản lý không thể nắm bắt ngay lập tức tình hình kinh doanh (doanh thu, tồn kho) để ra quyết định kịp thời (ví dụ: nhập thêm sách "hot", xả hàng tồn).
* **Không thể Chăm sóc Khách hàng:** Quản lý thủ công không cho phép xây dựng các chương trình khách hàng thân thiết (như tích điểm) một cách hệ thống.

Từ những bất cập trên, việc xây dựng một hệ thống phần mềm chuyên biệt để **tự động hóa** và **chuẩn hóa** quy trình quản lý nhà sách là một nhu cầu **cần thiết và cấp bách**, giúp giải quyết các vấn-"đề-cốt-lõi-về-vận-hành-và-quản-trị.

### I.4: Ý nghĩa về mặt Lý luận (Theoretical Significance)

Đề tài không chỉ là việc viết code, mà còn là quá trình áp dụng và kiểm chứng các lý thuyết nền tảng của ngành Khoa học Máy tính vào một bài toán thực tế:

* **Kiểm chứng Mô hình Cơ sở dữ liệu Quan hệ (RDBMS):** Đề tài vận dụng lý thuyết về **thiết kế CSDL quan hệ** để mô hình hóa một nghiệp vụ kinh doanh.
  + Các bảng (qlsach, hoadon, banhang, khachhang, taikhoan) được tạo ra dựa trên nguyên tắc chuẩn hóa (normalization) để tránh trùng lặp và đảm bảo **tính toàn vẹn dữ liệu (Data Integrity)**.
  + Việc sử dụng các Khóa chính (Primary Key) và Khóa ngoại (Foreign Key - ví dụ: banhang.ma\_hd liên kết với hoadon.ma\_hd) chính là sự thể hiện rõ ràng của lý thuyết liên kết dữ liệu.
* **Áp dụng Lập trình Hướng đối tượng (OOP):** Thay vì viết code theo kiểu thủ tục (procedural), đề tài sử dụng các class trong Python (như SaleManager, BookManager, InvoiceManager...).
  + Điều này áp dụng các nguyên lý của OOP (như **tính đóng gói** - Encapsulation) để "gói" các chức năng nghiệp vụ phức tạp (ví dụ: toàn bộ logic thanh toán, tích điểm) vào một đối tượng duy nhất, giúp code trở nên rõ ràng, dễ bảo trì và dễ tái sử dụng.
* **Vận dụng Lý thuyết Tương tác Người-Máy (HCI - Human-Computer Interaction):**
  + Thiết kế giao diện (UI) bằng Tkinter là một bài thực hành về HCI. Việc phân chia ứng dụng thành các "Tab" (ttk.Notebook) là một giải pháp thiết kế có chủ đích, giúp **giảm tải nhận thức (Cognitive Load)** cho người dùng, cho phép họ tập trung vào một nghiệp vụ duy nhất tại một thời điểm (Bán hàng, Quản lý kho, hoặc Xem hóa đơn).

### I.5: Ý nghĩa về mặt Thực tiễn (Practical Significance)

Giá trị lớn nhất của đề tài nằm ở khả năng ứng dụng và giải quyết các vấn đề "đau đầu" trong vận hành hàng ngày của một nhà sách:

* **Tự động hóa toàn diện Quy trình Bán hàng (POS):**
  + Hệ thống tự động hóa hoàn toàn các bước: từ tìm sách, tính tổng tiền, tính điểm tích lũy, áp dụng giảm giá theo điểm, cho đến xuất hóa đơn và **tự động trừ tồn kho** theo thời gian thực.
  + **Ý nghĩa:** Giải phóng nhân viên khỏi việc tính toán thủ công, giảm sai sót xuống gần như bằng 0, tăng tốc độ thanh toán và nâng cao tính chuyên nghiệp.
* **Xây dựng Hệ thống Chăm sóc Khách hàng (CRM) tinh gọn:**
  + Chức năng tích điểm theo SĐT (khachhang) là một công cụ CRM thực thụ. Nó giúp nhà sách **nhận diện khách hàng trung thành** và tự động "thưởng" cho họ (bằng cách giảm giá).
  + **Ý nghĩa:** Tăng khả năng khách hàng quay lại (Customer Retention), xây dựng tệp khách hàng thân thiết, và tăng khả năng cạnh tranh so với các đối thủ không có hệ thống này.
* **Minh bạch hóa Dữ liệu và Hỗ trợ Ra quyết định (Decision Support):**
  + Tab "Quản lý Hóa đơn" cung cấp một **hệ thống tra cứu tin cậy (Audit Trail)**. Khi có bất kỳ khiếu nại hay nhầm lẫn nào, người quản lý có thể truy xuất ngay lập tức hóa đơn gốc và chi tiết của nó.
  + **Ý nghĩa:** Cung cấp dữ liệu *thực* cho người chủ. Họ có thể (dù hiện tại là thủ công) dựa vào lịch sử này để biết sách nào bán chạy (cần nhập thêm) và sách nào bán chậm (cần khuyến mãi), từ đó tối ưu hóa kế hoạch kinh doanh.
* **Kiểm soát và Phân quyền (Access Control):**
  + Việc phân chia tài khoản "chủ" và "nhanvien" đảm bảo chỉ những người có thẩm quyền mới được thực hiện các thao tác nhạy cảm (như thêm/xóa tài khoản).
  + **Ý nghĩa:** Tăng cường an ninh nội bộ, giảm thiểu rủi ro gian lận hoặc phá hoại dữ liệu từ bên trong.

### I.6: Cơ sở để Cụ thể hóa Mục tiêu

Cơ sở để cụ thể hóa các mục tiêu của đề tài xuất phát trực tiếp từ việc **phân tích các vấn đề và hạn chế của quy trình quản lý nhà sách thủ công (hoặc bán thủ công bằng Excel)**.

Mục tiêu của đề tài không được đặt ra một cách ngẫu nhiên, mà được "cụ thể hóa" để giải quyết chính xác những "nỗi đau" (pain points) sau đây:

* **Cơ sở 1: Vấn đề về Tốc độ và Sai sót (Nghiệp vụ Bán hàng):**
  + **Thực tế:** Nhân viên tính tiền thủ công, tra giá bằng sổ sách, tính nhẩm giảm giá. Quá trình này **chậm**, **dễ sai sót** (nhầm giá, tính sai tổng), và **gây ùn tắc** tại quầy thu ngân.
  + **Cụ thể hóa mục tiêu:** Cần một module **Bán hàng (POS)** tự động hóa 100% việc tính toán, áp dụng giảm giá, và xuất hóa đơn.
* **Cơ sở 2: Vấn đề về Quản lý Tồn kho (Nghiệp vụ Kho):**
  + **Thực tế:** Không thể biết chính xác số lượng tồn kho của một cuốn sách ngay lập tức. Việc kiểm kho cuối ngày/cuối tháng là một "thảm họa" vì dữ liệu bán ra và dữ liệu tồn kho không liên kết với nhau.
  + **Cụ thể hóa mục tiêu:** Cần một hệ thống **CSDL tập trung** (MySQL) và module **Quản lý Sách (CRUD)**. Quan trọng nhất, hệ thống phải **tự động trừ kho** ngay khi một giao dịch bán hàng hoàn tất.
* **Cơ sở 3: Vấn đề về Cạnh tranh (Nghiệp vụ Marketing):**
  + **Thực tế:** Quản lý thủ công không thể xây dựng chương trình khách hàng thân thiết một cách hệ thống. Việc "giảm giá cho khách quen" mang tính cảm tính, không đo lường được.
  + **Cụ thể hóa mục tiêu:** Cần một module **Quản lý Khách hàng (CRM)** đơn giản, cụ thể là **tích điểm qua SĐT**, để giữ chân khách hàng và tăng khả năng cạnh tranh.
* **Cơ sở 4: Vấn đề về An ninh và Quản trị (Nghiệp vụ Quản lý):**
  + **Thực tế:** Sổ sách dễ bị sửa đổi, gian lận, hoặc mất mát. Chủ cửa hàng không thể kiểm soát nhân viên (ví dụ: nhân viên tự ý giảm giá cho bạn bè).
  + **Cụ thể hóa mục tiêu:** Cần một hệ thống **Đăng nhập & Phân quyền** rõ ràng (Tài khoản 'chủ' và 'nhanvien') để kiểm soát các chức năng nhạy cảm.
* **Cơ sở 5: Vấn đề về Đối soát (Nghiệp vụ Kế toán):**
  + **Thực tế:** Khi sổ sách bị sai, việc tìm lại một hóa đơn cũ để đối chiếu là cực kỳ khó khăn.
  + **Cụ thể hóa mục tiêu:** Cần một module **Quản lý Hóa đơn** cho phép tra cứu lại lịch sử giao dịch và xem chi tiết của từng hóa đơn.

### I.7: Các Định hướng Nội dung chính (Đã thực hiện)

Từ 5 cơ sở phân tích ở trên, đề tài đã vạch ra 5 định hướng nội dung chính (tương ứng với 5 mục tiêu cụ thể) và đã thực hiện chúng trong code:

* **Định hướng 1: Phân tích và Thiết kế Cơ sở dữ liệu (Database Design)**
  + Đây là nội dung nền tảng.
  + **Thực hiện:** Đã thiết kế một CSDL quan hệ qlSach trên MySQL. Các bảng chính (qlsach, taikhoan, khachhang, hoadon, banhang) được tạo ra với các mối quan hệ (khóa chính, khóa ngoại) rõ ràng để đảm bảo tính toàn vẹn và nhất quán của dữ liệu.
* **Định hướng 2: Xây dựng Nghiệp vụ Bán hàng Tự động (POS)**
  + Đây là nội dung cốt lõi của đề tài.
  + **Thực hiện:** Xây dựng class SaleManager với giao diện "Bán Sách". Module này cho phép tìm kiếm sách, thêm vào giỏ hàng (self.cart), tự động tính tổng tiền, và xử lý thanh toán.
* **Định hướng 3: Xây dựng Nghiệp vụ Tích điểm (CRM)**
  + Đây là nội dung giải quyết bài toán cạnh tranh.
  + **Thực hiện:** Tích hợp trong quy trình thanh toán (checkout). Hệ thống cho phép "Kiểm tra SĐT", tự động tra cứu điểm từ bảng khachhang, tính toán giảm giá theo bậc (10% - 50%), và tự động cập nhật lại điểm (INSERT... ON DUPLICATE KEY UPDATE...) sau khi thanh toán.
* **Định hướng 4: Xây dựng Hệ thống Xác thực & Phân quyền (Security)**
  + Đây là nội dung giải quyết bài toán quản trị.
  + **Thực hiện:** Xây dựng class LoginScreen để xác thực người dùng. Trong class MainApp, hệ thống kiểm tra quyền (if quyen == 'chu') để quyết định có hiển thị tab "Quản lý Nhân viên" hay không, đảm bảo nhân viên không thể truy cập chức năng này.
* **Định hướng 5: Xây dựng Chức năng Tra cứu & Báo cáo (Reporting)**
  + Đây là nội dung giải quyết bài toán đối soát.
  + **Thực hiện:** Xây dựng class InvoiceManager (tab "Quản lý Hoá Đơn"). Module này cho phép tải danh sách tất cả hóa đơn từ bảng hoadon và khi nhấn vào một hóa đơn, hệ thống sẽ truy vấn bảng banhang (dựa trên ma\_hd) để hiển thị chi tiết các sản phẩm đã bán.

### I.8: Mục tiêu Nghiên cứu của Đề tài

Đề tài tập trung vào việc nghiên cứu, thiết kế và triển khai một hệ thống phần mềm nhằm giải quyết ba (03) mục tiêu chính sau đây:

**Mục tiêu 1: Thiết kế và Chuẩn hóa Quy trình Quản lý Kho sách**

* **Nội dung:** Nghiên cứu và thiết kế mô hình cơ sở dữ liệu (CSDL) quan hệ trên MySQL để lưu trữ thông tin sách một cách tập trung. Xây dựng một giao diện (module "Quản lý Sách") cho phép người dùng thực hiện đầy đủ các thao tác **CRUD (Thêm, Sửa, Xóa)** sách trong kho.
* **Tính khả thi & Đo lường (Định tính):** Mục tiêu này hoàn toàn khả thi. Mức độ hoàn thành được đo lường khi hệ thống có thể:
  + (a) Thêm một cuốn sách mới vào CSDL.
  + (b) Sửa thông tin (ví dụ: giá, số lượng) của một cuốn sách.
  + (c) Xóa một cuốn sách khỏi CSDL.

**Mục tiêu 2: Tự động hóa và Tối ưu hóa Nghiệp vụ Bán hàng (POS)**

* **Nội dung:** Xây dựng module "Bán Sách" có khả năng tự động hóa hoàn toàn quy trình thanh toán. Hệ thống phải có khả năng thêm sản phẩm vào giỏ hàng, tự động tính tổng tiền, và quan trọng nhất là **tự động cập nhật (trừ) số lượng tồn kho** trong CSDL ngay khi giao dịch hoàn tất.
* **Tính khả thi & Đo lường (Định tính):** Khả thi. Mục tiêu được xem là hoàn thành khi:
  + (a) Một giao dịch bán hàng được thực hiện.
  + (b) Ngay sau đó, kiểm tra CSDL (bảng qlsach), thấy cột so\_luong của sách vừa bán đã bị trừ đi chính xác.

**Mục tiêu 3: Xây dựng Hệ thống Tích điểm (CRM) và Phân quyền Bảo mật**

* **Nội dung:** Phát triển hai tính năng quản trị cốt lõi:
  1. **Bảo mật:** Xây dựng hệ thống đăng nhập (LoginScreen) và **phân quyền** (Auth) để phân tách vai trò "chủ" và "nhân viên", đảm bảo nhân viên không thể truy cập các chức năng quản trị.
  2. **Chăm sóc khách hàng:** Xây dựng logic **tích điểm qua SĐT** (khachhang), tự động tính toán giảm giá dựa trên bậc điểm, và lưu lại lịch sử giao dịch.
* **Tính khả thi & Đo lường (Định tính):** Khả thi. Mục tiêu hoàn thành khi:
  1. (a) Tài khoản "nhanvien" đăng nhập và **không thấy** tab "Quản lý Nhân viên".
  2. (b) Một khách hàng (qua SĐT) có thể sử dụng điểm để được giảm giá và số điểm đó được cập nhật chính xác trong CSDL sau khi thanh toán.

# II. Tổng quan và cơ sở lý thuyết:

### II.1. Bối cảnh xuất hiện vấn đề

Như đã đề cập ở trên, trong bối cảnh thị trường bán lẻ đang bước vào giai đoạn chuyển đổi số mạnh mẽ, các nhà sách truyền thống đang phải đối mặt với áp lực cạnh tranh gay gắt, không chỉ từ các đối thủ trực tiếp mà còn từ các sàn thương mại điện tử. Để tồn tại và phát triển, việc tối ưu hóa quy trình vận hành và nâng cao trải nghiệm khách hàng là yêu cầu bắt buộc.

Tuy nhiên, một thực trạng phổ biến tại nhiều nhà sách (đặc biệt là quy mô vừa và nhỏ) là vẫn duy trì phương thức quản lý thủ công, bán thủ công. Việc ghi chép sổ sách, quản lý tồn kho bằng các tệp Excel rời rạc, và tính tiền bằng máy tính tay bộc lộ nhiều bất cập nghiêm trọng:

* **Sai sót cao:** Việc nhập liệu thủ công (tra giá, tính tổng, ghi chép tồn kho) rất dễ dẫn đến nhầm lẫn, gây sai lệch doanh thu và thất thoát hàng hóa.
* **Hiệu suất thấp:** Quy trình thanh toán chậm chạp, tra cứu sách tốn thời gian, gây ùn tắc tại quầy thu ngân và làm giảm sự hài lòng của khách hàng.
* **Không thể kiểm soát tức thời:** Người quản lý không thể nắm bắt được số lượng tồn kho chính xác hay doanh thu theo thời gian thực, khiến việc ra quyết định (như nhập hàng mới, xả hàng tồn) trở nên chậm trễ và thiếu cơ sở.

Chính từ bối cảnh này, vấn đề về một giải pháp công nghệ "vừa đủ" – dễ triển khai, chi phí thấp, nhưng giải quyết được các nghiệp vụ cốt lõi – trở nên cấp thiết.

### II.2. Nội dung và Bài toán đặt ra

**Nội dung của Vấn đề**

Nội dung cốt lõi của vấn đề không chỉ nằm ở việc bán sách, mà là sự **thiếu liên kết và tự động hóa** giữa các quy trình nghiệp vụ cơ bản:

1. **Hàng hóa (Kho):** Số lượng sách trên kệ không "nói chuyện" được với sổ sách bán hàng. Sách bán ra nhưng tồn kho không tự động giảm.
2. **Bán hàng (Thu ngân):** Nhân viên phải thực hiện quá nhiều thao tác thủ công (tra giá, tính tiền, tính giảm giá), làm tăng rủi ro sai sót.
3. **Khách hàng (Marketing):** Không có công cụ nào để ghi nhận thông tin khách hàng thân thiết, không thể triển khai các chương trình tích điểm một cách có hệ thống.
4. **Quản trị (An ninh):** Không có sự phân quyền. Bất kỳ ai cũng có thể xem sổ sách, dễ dẫn đến gian lận hoặc sai sót dữ liệu không thể truy vết.

**Bài toán đặt ra cần giải quyết**

Từ các phân tích trên, đề tài đặt ra bài toán trọng tâm là: **"Nghiên cứu, thiết kế và xây dựng một hệ thống phần mềm quản lý bán sách (POS) tập trung, có khả năng tự động hóa các nghiệp vụ cốt lõi."**

Để giải quyết bài toán này, hệ thống phải đáp ứng được các yêu cầu cụ thể sau:

1. **Về Quản lý Kho:** Phải xây dựng một CSDL (MySQL) duy nhất và tập trung. Cung cấp giao diện cho phép người dùng **quản lý (Thêm, Sửa, Xóa)** thông tin sách một cách dễ dàng.
2. **Về Bán hàng:** Phải xây dựng một module thanh toán tự động 100%. Hệ thống phải **tự động trừ số lượng tồn kho** trong CSDL ngay khi một giao dịch được xác nhận.
3. **Về Khách hàng:** Phải tích hợp một module **CRM (Tích điểm qua SĐT)** đơn giản để ghi nhận khách hàng, tự động tính toán giảm giá khi đủ điều kiện.
4. **Về Quản trị:** Phải xây dựng cơ chế **Đăng nhập và Phân quyền** (ví dụ: 'chủ' và 'nhanvien') để đảm bảo an ninh dữ liệu, ngăn nhân viên truy cập các chức năng quản lý nhạy cảm.
5. **Về Đối soát:** Phải cung cấp khả năng **tra cứu lại lịch sử hóa đơn** đã giao dịch để phục vụ việc giải quyết khiếu nại hoặc kiểm toán nội bộ.

### II.3 Lịch sử giải quyết vấn đề

Bài toán "Quản lý bán lẻ" (Retail Management), trong đó có "Quản lý Nhà sách", không phải là một vấn đề mới. Vấn đề này đã và đang được giải quyết ở nhiều quy mô khác nhau trên toàn thế giới. Việc giải quyết bài toán này có thể chia thành ba nhóm chính:

* **1. Các giải pháp Thương mại quy mô lớn (Enterprise Solutions):**
  + **Ai/Ở đâu:** Các tập đoàn công nghệ lớn như **SAP, Oracle, Microsoft (với Dynamics 365)**.
  + **Kết quả:** Họ tạo ra các hệ thống **ERP (Enterprise Resource Planning)** khổng lồ, toàn diện. Các hệ thống này quản lý *mọi thứ*: từ chuỗi cung ứng, tài chính kế toán, nhân sự, đến bán hàng. Chúng cực kỳ mạnh mẽ, ổn định, bảo mật và thường chạy trên nền tảng điện toán đám mây (Cloud).
  + **Tồn tại:** Quá **phức tạp, cồng kềnh** và **chi phí triển khai/bảo trì cực kỳ cao**. Các hệ thống này hoàn toàn không phù hợp với các doanh nghiệp vừa, nhỏ, hoặc siêu nhỏ.
* **2. Các giải pháp POS/SaaS cho Doanh nghiệp vừa và nhỏ (SMB Solutions):**
  + **Ai/Ở đâu:** Các công ty phần mềm chuyên biệt, đặc biệt là các công ty cung cấp **SaaS (Phần mềm như một Dịch vụ)**. Ở Việt Nam, các tên tuổi nổi bật là **KiotViet, Sapo, MISA CukCuk**.
  + **Kết quả:** Đây là các giải pháp rất thành công. Chúng thường chạy trên nền tảng Web và App di động, cho phép chủ cửa hàng quản lý từ xa. Chúng giải quyết tốt các nghiệp vụ cốt lõi: bán hàng (POS), tích hợp máy quét mã vạch, quản lý tồn kho, và báo cáo doanh thu cơ bản.
  + **Tồn tại:**
    - **Chi phí định kỳ:** Các giải pháp này hoạt động theo mô hình **thuê bao (subscription)**. Người dùng phải trả phí hàng tháng/hàng năm. Đây là một rào cản tài chính đối với các hộ kinh doanh siêu nhỏ.
    - **Tính năng thừa (Feature Bloat):** Để phục vụ nhiều ngành nghề (thời trang, cafe, tạp hóa), các phần mềm này thường có nhiều tính năng mà một nhà sách nhỏ không bao giờ dùng đến, gây phức tạp không cần thiết.
    - **Khó tùy chỉnh:** Người dùng không thể tự ý thay đổi một quy tắc nghiệp vụ (ví dụ: thay đổi cách tính điểm tích lũy) mà phải phụ thuộc hoàn toàn vào nhà cung cấp.
* **3. Các giải pháp Mã nguồn mở (Open-Source):**
  + **Ai/Ở đâu:** Các cộng đồng lập trình viên trên toàn thế giới (ví dụ: các dự án trên GitHub). Nổi bật có **Odoo** (một nhánh của nó là POS) hoặc các dự án nhỏ lẻ.
  + **Kết quả:** Cung cấp các bộ code miễn phí, cho phép doanh nghiệp tự do tải về, cài đặt và tùy chỉnh.
  + **Tồn tại:** Đòi hỏi doanh nghiệp phải có **đội ngũ kỹ thuật riêng** (hoặc thuê ngoài) để cài đặt, cấu hình, bảo trì và tùy chỉnh. Đối với một chủ nhà sách nhỏ không rành về công nghệ, đây là điều không thể.

🟏Từ phân tích trên, chúng ta thấy một "khoảng trống" (gap) rõ ràng trên thị trường:

**🡺Đó là sự thiếu hụt một giải pháp "vừa đủ" – một hệ thống siêu nhẹ, miễn phí (không mất phí thuê bao), và được tùy chỉnh chính xác theo nghiệp vụ đặc thù của một nhà sách quy mô nhỏ.**

Các giải pháp thương mại thì **quá đắt và phức tạp**, còn giải pháp mã nguồn mở thì **quá khó để triển khai**.

Chính những tồn tại này là cơ sở để đề tài được thực hiện. Đề tài này **không** đặt mục tiêu cạnh tranh với KiotViet hay SAP, mà giải quyết một thị trường ngách:

1. **Giải quyết vấn đề Chi phí:** Xây dựng một ứng dụng Desktop (Tkinter) chạy độc lập. Chủ nhà sách chỉ cần cài đặt một lần và sử dụng **miễn phí trọn đời**, không mất phí thuê bao hàng tháng.
2. **Giải quyết vấn đề Phức tạp:** Hệ thống **loại bỏ mọi tính năng thừa** và chỉ tập trung vào 5 nghiệp vụ cốt lõi mà một nhà sách nhỏ *thực sự* cần (Sách, Bán hàng, Khách hàng, Hóa đơn, Nhân viên).
3. **Giải quyết vấn đề Tùy chỉnh:** Vì đây là sản phẩm "in-house" (tự phát triển), chủ nhà sách có toàn quyền yêu cầu lập trình viên (người thực hiện đề tài) **thay đổi bất kỳ quy tắc nghiệp vụ nào** (ví dụ: đổi "2 điểm/cuốn" thành "5 điểm/cuốn") một cách nhanh chóng mà không cần chờ đợi nhà cung cấp SaaS.

**🡺Tóm lại:** Đề tài này giải quyết bài toán tồn tại bằng cách cung cấp một giải pháp "may đo" (tailor-made), chi phí bằng không, và đủ đơn giản cho các hộ kinh doanh siêu nhỏ - đối tượng mà các phần mềm thương mại lớn đang bỏ ngỏ.

### II.4: Phạm vi Giải quyết của Đề tài:

Đề tài này tập trung vào việc xây dựng một **Hệ thống Quản lý Bán hàng (POS) tại chỗ (Local Desktop Application)**, được thiết kế đặc thù cho các nhà sách quy mô vừa và nhỏ.

Phạm vi của đề tài được giới hạn trong 5 nhóm nghiệp vụ chính sau:

* **Nghiệp vụ Kho (Quản lý Sách):**
  + Cho phép người dùng thực hiện các thao tác quản lý kho cơ bản: **Thêm** sách mới, **Sửa** thông tin (tên, tác giả, giá bán, số lượng), và **Xóa** sách.
  + Đề tài chỉ giải quyết việc **xuất kho (outbound)**, cụ thể là **tự động trừ số lượng tồn kho** trong CSDL (bảng qlsach) ngay khi một giao dịch bán hàng được xác nhận.
* **Nghiệp vụ Bán hàng (POS):**
  + Cung cấp giao diện cho nhân viên thu ngân thực hiện quy trình bán hàng: tìm kiếm sách, thêm vào giỏ hàng, tự động tính tổng tiền.
  + Xử lý việc **xuất hóa đơn** (lưu lại thông tin vào CSDL và cho phép lưu file .txt).
* **Nghiệp vụ Khách hàng (CRM cơ bản):**
  + Quản lý khách hàng thân thiết dựa trên **SĐT** (bảng khachhang).
  + Tự động **tích điểm** cho khách sau mỗi lần mua (+2 điểm/cuốn).
  + Tự động **áp dụng giảm giá** dựa trên bậc điểm tích lũy của khách (ví dụ: 10 điểm = 10%... tối đa 50%) khi thanh toán.
* **Nghiệp vụ Quản trị (Admin & Security):**
  + Xây dựng hệ thống **đăng nhập** để xác thực người dùng.
  + Thực hiện **phân quyền** ở mức cơ bản: phân tách rõ ràng vai trò 'chủ' và 'nhanvien', trong đó 'nhanvien' không thể truy cập các chức năng quản lý nhạy cảm (như tab "Quản lý Nhân viên").
* **Nghiệp vụ Đối soát (Reporting cơ bản):**
  + Cung cấp chức năng **tra cứu lịch sử giao dịch** (tab "Quản lý Hoá Đơn").
  + Cho phép người dùng xem lại danh sách tất cả các hóa đơn đã bán và xem chi tiết các sản phẩm trong từng hóa đơn đó.

### II.5. Phạm vi Không Giải quyết:

Để đảm bảo tính khả thi và tập trung của đề tài, các nghiệp vụ phức tạp hoặc nằm ngoài mục tiêu cốt lõi sau đây sẽ **không** được giải quyết:

* **Nghiệp vụ Nhập kho (Purchasing):** Đề tài **không** xây dựng module quản lý nhập hàng, không quản lý nhà cung cấp, không tạo phiếu nhập kho hay xử lý giá vốn.
* **Hệ thống Báo cáo Phân tích (BI/OLAP):** Đề tài **không** xây dựng các báo cáo quản trị phức tạp (ví dụ: báo cáo tổng doanh thu theo ngày/tháng/năm, báo cáo top 10 sách bán chạy, phân tích lợi nhuận).
* **Tích hợp Phần cứng:** Đề tài **không** tích hợp với các thiết bị phần cứng chuyên dụng như máy quét mã vạch (barcode scanner), máy in hóa đơn nhiệt, hay ngăn kéo đựng tiền.
* **Nền tảng Web/Cloud:** Đây là một ứng dụng **Desktop** chạy cục bộ (local). Đề tài **không** giải quyết bài toán triển khai trên nền tảng web, không thể truy cập từ xa qua internet.
* **Nghiệp vụ Kế toán Chuyên sâu:** Đề tài **không** xử lý các nghiệp vụ tài chính - kế toán như quản lý công nợ nhà cung cấp, chấm công, tính lương nhân viên, hay tính toán lợi nhuận (do không quản lý giá vốn).
* **Xử lý Ngoại lệ Phức tạp:** Các quy trình như **nhận trả hàng (returns)**, **hủy đơn đã thanh toán (voids)**, hay **kiểm kê/điều chỉnh kho (stock adjustments)** nằm ngoài phạm vi giải quyết của đề tài này.

### II.6: Phương pháp Nghiên cứu Lý thuyết

Đây là phương pháp nền tảng, tập trung vào việc nghiên cứu, tổng hợp và phân tích các kiến thức lý thuyết đã có để xây dựng cơ sở khoa học cho đề tài.

* **Nội dung:**
  1. **Nghiên cứu Công nghệ:** Phân tích, so sánh ưu nhược điểm của các công nghệ.
     + **Giao diện (UI):** Tại sao chọn **Tkinter**? (Vì nó tích hợp sẵn trong Python, nhẹ, đủ nhanh cho các ứng dụng quản lý, dễ phát triển).
     + **Logic (Backend):** Tại sao chọn **Python**? (Ngôn ngữ lập trình hướng đối tượng (OOP) mạnh mẽ, cú pháp rõ ràng, hệ sinh thái thư viện phong phú).
     + **CSDL (Database):** Tại sao chọn **MySQL**? (Hệ CSDL quan hệ (RDBMS) miễn phí, phổ biến, mạnh mẽ, và có tính ổn định cao cho việc xử lý giao dịch).
  2. **Nghiên cứu Mô hình Dữ liệu:** Nghiên cứu lý thuyết về **Cơ sở dữ liệu Quan hệ**, các dạng chuẩn hóa (Normalization) để thiết kế CSDL (qlSach, hoadon, banhang, khachhang, taikhoan) đảm bảo tính toàn vẹn, nhất quán và giảm thiểu trùng lặp dữ liệu.
  3. **Nghiên cứu Cấu trúc (Kiến trúc):** Nghiên cứu và áp dụng mô hình **Lập trình Hướng đối tượng (OOP)**. Thay vì viết code theo kiểu thủ tục, đề tài đã tổ chức code thành các class (như SaleManager, BookManager, InvoiceManager...). Mỗi class đóng gói toàn bộ dữ liệu và phương thức xử lý cho một nghiệp vụ cụ thể, giúp code dễ đọc, dễ bảo trì và dễ mở rộng.
* **Giải quyết bài toán:** Phương pháp này giúp trả lời câu hỏi "Làm thế nào?" và "Dùng cái gì?". Nó cung cấp **bản thiết kế (blueprint)** và **công cụ (tools)** để xây dựng hệ thống một cách khoa học và có cấu trúc rõ ràng.

### II.7: Phương pháp Điều tra & Khảo sát (Investigative Research)

Đây là phương pháp nghiên cứu thực tế, tập trung vào việc tìm hiểu "hiện trạng" và thu thập các "yêu cầu" (requirements) từ thực tế để xác định bài toán cần giải quyết.

* **Nội dung:**
  1. **Điều tra Hiện trạng:** Phân tích các quy trình quản lý nhà sách thủ công (hoặc bán thủ công bằng Excel). Điều tra này giúp phát hiện ra các "nỗi đau" (pain points) như:
     + Thanh toán chậm, dễ sai sót.
     + Không thể kiểm soát tồn kho tức thời.
     + Không có chương trình giữ chân khách hàng.
     + Không thể tra cứu lại hóa đơn cũ.
  2. **Khảo sát và Thu thập Yêu cầu:** "Khảo sát" các bên liên quan (đóng vai chủ cửa hàng, nhân viên) để xác định họ *cần* gì ở một phần mềm.
     + *Chủ cửa hàng cần:* "Tôi muốn xem lại lịch sử hóa đơn", "Tôi muốn quản lý tài khoản nhân viên" -> **Bài toán:** Xây dựng class InvoiceManager và EmployeeManager.
     + *Nhân viên bán hàng cần:* "Tôi muốn thanh toán nhanh", "Tôi muốn có cách tích điểm cho khách" -> **Bài toán:** Xây dựng class SaleManager và logic khachhang.
* **Giải quyết bài toán:** Phương pháp này giúp trả lời câu hỏi "Làm cái gì?" và "Tại sao?". Nó đảm bảo rằng sản phẩm được phát triển **đúng theo nhu cầu thực tế** và giải quyết được các vấn đề đang tồn tại, tránh việc xây dựng các tính năng thừa hoặc vô dụng.

### II.8: Phương pháp Nghiên cứu Thực nghiệm (Experimental Research)

Đây là phương pháp cốt lõi của việc phát triển phần mềm, biến các "lý thuyết" và "yêu cầu" thành một sản phẩm chạy được và kiểm chứng nó.

* **Nội dung:**
  1. **Xây dựng Nguyên mẫu (Prototyping):** Dựa trên lý thuyết (OOP) và điều tra (yêu cầu), chúng ta tiến hành "thí nghiệm" bằng cách viết code và xây dựng ứng dụng. Toàn bộ class SaleManager, BookManager... chính là các **nguyên mẫu thực nghiệm**.
  2. **Kiểm thử và Lặp lại (Testing & Iteration):** Đây là phần quan trọng nhất của thực nghiệm. Chúng ta kiểm tra giả thuyết: "Liệu hàm \_confirm\_checkout có *thực sự* trừ kho và cộng điểm đúng không?".
     + **Kiểm thử tích hợp:** Chúng ta đã kiểm tra xem việc nhấn nút "Thanh toán" (UI - Tkinter) có kích hoạt đúng hàm logic (Python) và có cập nhật đúng CSDL (MySQL) hay không.
     + **Báo cáo lỗi (Bug Reporting):** Chính bạn là người thực hiện bước này khi báo cáo: "Lỗi Unknown column 'password'" hoặc "Cửa sổ thanh toán bị che khuất".
  3. **Sửa lỗi và Tối ưu (Debugging & Refactoring):** Dựa trên kết quả kiểm thử, chúng ta đã tiến hành sửa lỗi (thay password bằng matkhau rồi lại đổi về password sau khi CSDL được chuẩn hóa) và tối ưu (tăng chiều cao cửa sổ pop-up từ 350 lên 400).
* **Giải quyết bài toán:** Phương pháp này giúp trả lời câu hỏi "Nó có chạy đúng không?". Nó biến các ý tưởng và thiết kế trên giấy (lý thuyết, điều tra) thành một **sản phẩm hữu hình, đã được kiểm chứng** là hoạt động chính xác theo đúng các yêu cầu nghiệp vụ đã đặt ra.

### II.9: tóm tắt cơ sở lý thuyết được sử dụng để giải quyết đề tài, tập trung vào ba trụ cột chính:

**1. Lý thuyết Cơ sở dữ liệu Quan hệ**

Đây là lý thuyết nền tảng để giải quyết bài toán "lưu trữ dữ liệu".

* **Nội dung:** Đề tài sử dụng lý thuyết về **Cơ sở dữ liệu Quan hệ** để thiết kế CSDL (MySQL). Lý thuyết này tập trung vào:
  + **Mô hình hóa Dữ liệu:** Chia nhỏ các đối tượng thực tế (Sách, Hóa đơn, Khách hàng) thành các bảng (tables).
  + **Chuẩn hóa (Normalization):** Áp dụng các quy tắc chuẩn hóa để đảm bảo dữ liệu không bị trùng lặp, giảm thiểu sai sót.
  + **Khóa (Keys):** Sử dụng **Khóa chính** (Primary Key, ví dụ: hoadon.ma\_hd) để định danh duy nhất cho mỗi dòng và **Khóa ngoại** (Foreign Key, ví dụ: banhang.ma\_hd) để tạo ra các mối liên kết logic giữa các bảng.
* **Giải quyết vấn đề:** Lý thuyết này giúp xây dựng một CSDL có cấu trúc, logic, và đảm bảo **tính toàn vẹn dữ liệu**. Nó đảm bảo rằng khi một hóa đơn được tạo, các chi tiết của nó (trong bảng banhang) được liên kết chính xác, không thể có dữ liệu "mồ côi".

**2. Lý thuyết Lập trình Hướng đối tượng**

Đây là lý thuyết nền tảng để giải quyết bài toán "xử lý logic" và "tổ chức code".

* **Nội dung:** Đề tài sử dụng ngôn ngữ **Python** và tuân thủ các nguyên tắc của OOP. Lý thuyết này tập trung vào:
  + **Lớp (Class) và Đối tượng (Object):** Thay vì viết code rời rạc, chúng ta "gói" các nghiệp vụ thành các Lớp. Ví dụ: class SaleManager là một bản thiết kế cho nghiệp vụ bán hàng, class BookManager là bản thiết kế cho nghiệp vụ kho.
  + **Tính Đóng gói (Encapsulation):** Mỗi class tự quản lý dữ liệu (biến) và hành vi (hàm) của riêng nó. Ví dụ, toàn bộ logic phức tạp của việc thanh toán (kiểm tra điểm, trừ kho, lưu CSDL) được "đóng gói" gọn gàng bên trong hàm \_confirm\_checkout của class SaleManager.
* **Giải quyết vấn đề:** Lý thuyết OOP giúp code trở nên **dễ đọc, dễ bảo trì, và dễ mở rộng**. Khi cần sửa logic bán hàng, chúng ta biết chính xác phải tìm ở đâu (class SaleManager) mà không sợ ảnh hưởng đến logic của "Quản lý Sách".

**3. Lý thuyết Tương tác Người-Máy (Giao diện người dùng)**

* **Nội dung:** Đề tài sử dụng thư viện **Tkinter** và áp dụng các nguyên tắc cơ bản của HCI.
  + **Thiết kế Giao diện (UI Design):** Sử dụng các thành phần trực quan (widgets) như Button, Entry, Treeview để xây dựng một giao diện mà người dùng có thể tương tác.
  + **Tổ chức Thông tin:** Áp dụng một cấu trúc điều hướng rõ ràng. Thay vì dồn tất cả chức năng lên một màn hình, đề tài sử dụng ttk.Notebook (Tab) để phân chia các khu vực nghiệp vụ (Bán hàng, Quản lý Sách, Quản lý Hóa đơn...).
* **Giải quyết vấn đề:** Lý thuyết này giúp **giảm tải nhận thức** cho người dùng. Bằng cách chia tab, hệ thống cho phép nhân viên tập trung hoàn toàn vào một nhiệm vụ tại một thời điểm (ví dụ: chỉ tập trung vào bán hàng), làm tăng hiệu suất và giảm sai sót khi thao tác.

### II.10: Phân tích Kỹ năng Lập trình Python trong Giải quyết Vấn đề Thực tế

Trong đề tài này, ngôn ngữ Python không chỉ là công cụ để viết code, mà các kỹ năng và cấu trúc (paradigms) của nó được vận dụng trực tiếp để mô hình hóa và giải quyết các bài toán nghiệp vụ phức tạp của một nhà sách.

**1 Kỹ năng Lập trình Hướng đối tượng (OOP)**

* **Vấn đề thực tế:** Nghiệp vụ của một nhà sách rất phức tạp và được chia thành nhiều mảng riêng biệt (Bán hàng, Quản lý kho, Quản lý nhân sự, Báo cáo). Nếu viết code theo kiểu thủ tục (procedural), các logic này sẽ bị trộn lẫn, tạo ra một mớ code "spaghetti" (mỳ Ý) rối rắm, cực kỳ khó bảo trì và nâng cấp.
* **Vận dụng kỹ năng Python:**
  + Đề tài đã vận dụng **Lập trình Hướng đối tượng (OOP)** bằng cách sử dụng class để "đóng gói" từng nghiệp vụ.
  + **class LoginScreen:** Đóng gói toàn bộ logic và giao diện liên quan đến việc xác thực.
  + **class SaleManager:** Đóng gói toàn bộ logic, biến số (self.cart), và giao diện phức tạp của riêng nghiệp vụ bán hàng.
  + **class BookManager**, **InvoiceManager**, **EmployeeManager**: Tương tự, mỗi class là một "bộ não" độc lập cho nghiệp vụ của nó.
* **Giải quyết:** Kỹ năng OOP giúp **mô hình hóa** thế giới thực vào code. Nó đảm bảo nguyên tắc **Đóng gói (Encapsulation)**, nghĩa là logic của việc bán hàng (SaleManager) không bị rò rỉ hay ảnh hưởng đến logic của việc quản lý sách (BookManager). Điều này giúp code sạch sẽ, dễ dàng sửa lỗi và cực kỳ dễ nâng cấp (ví dụ: chỉ cần sửa trong SaleManager khi muốn thay đổi luật tích điểm mà không sợ hỏng các tab khác).

**2. Kỹ năng Xử lý Cấu trúc Dữ liệu (Data Structures)**

* **Vấn đề thực tế:** Làm thế nào để mô phỏng một "giỏ hàng" (shopping cart)? Làm sao để xử lý một loạt dữ liệu (như danh sách sách, danh sách hóa đơn) một cách hiệu quả?
* **Vận dụng kỹ năng Python:**
  + **Kiểu list:** Biến self.cart = [] chính là vận dụng trực tiếp cấu trúc dữ liệu list của Python để làm giỏ hàng.
    - Kỹ năng **self.cart.append(...)** được dùng để "thêm hàng vào giỏ".
    - Kỹ năng **List Comprehension** (ví dụ self.cart = [c for c in self.cart if c[0] != ma]) được dùng để "xóa hàng khỏi giỏ" một cách cực kỳ hiệu quả và ngắn gọn.
    - Kỹ năng **for item in self.cart:** được dùng để duyệt giỏ hàng, kiểm tra trùng lặp, hoặc tính tổng tiền.
  + **Kiểu tuple:** Khi lấy dữ liệu từ Treeview hoặc chuẩn bị dữ liệu cho CSDL, việc sử dụng tuple (ví dụ values=(ma, ten, sl, gia, tong)) đảm bảo dữ liệu là **bất biến (immutable)**, giúp truyền dữ liệu đi một cách an toàn, tránh bị thay đổi ngoài ý muốn.
* **Giải quyết:** Kỹ năng sử dụng linh hoạt list và tuple giúp giải quyết bài toán "lưu trữ dữ liệu tạm thời" (như giỏ hàng) một cách trực quan, hiệu suất cao và dễ hiểu.

**3.Kỹ năng Xử lý Ngoại lệ và Logic Điều kiện**

* **Vấn đề thực tế:** Người dùng (nhân viên) sẽ không bao giờ nhập liệu đúng 100%. Họ có thể gõ "abc" vào ô "Số lượng", gõ số lượng lớn hơn tồn kho, hoặc quên nhập tên khách hàng. Nếu không xử lý, ứng dụng sẽ "crash" (sập) ngay lập-tức, gây-mất-dữ-liệu-và-gián-đoạn-bán-hàng.
* **Vận dụng kỹ năng Python:**
  + **try... except ValueError:**: Đây là kỹ năng tối quan trọng. Trong hàm add\_to\_cart, code đã bọc logic sl = int(self.so\_luong.get()) trong một khối try-except. Nếu người dùng nhập chữ, Python sẽ bắt được lỗi ValueError và hiển thị cảnh báo (messagebox) thay vì làm sập chương trình.
  + **Logic if... elif... else:**
    - if not user or not pw:: Kỹ năng kiểm tra "rỗng" (validation) cơ bản.
    - if sl > ton:: Kỹ năng xử lý logic nghiệp vụ, đảm bảo không bán "khống" hàng.
    - if quyen == 'chu':: Kỹ năng này được dùng để giải quyết **bài toán bảo mật (phân quyền)**. Nó quyết định việc có hiển thị tab "Quản lý Nhân viên" hay không.
* **Giải quyết:** Kỹ năng **Xử lý Ngoại lệ (Exception Handling)** đảm bảo tính **"Bền vững" (Robustness)** của ứng dụng. Nó lường trước các lỗi do con người và ngăn chặn sập hệ thống, giúp ứng dụng chạy ổn định trong môi trường thực tế.

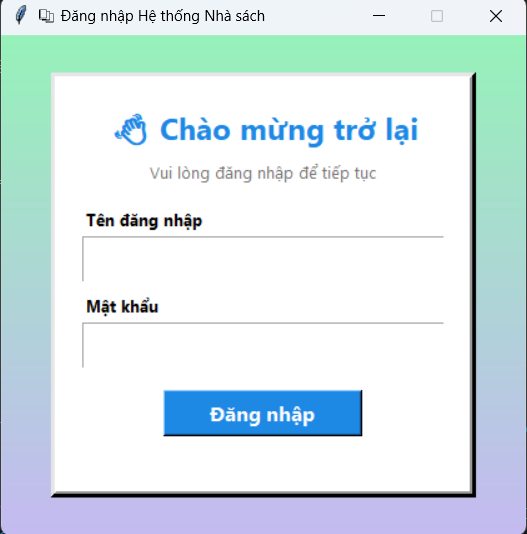
**4. Kỹ năng Lập trình Sự kiện (Event-Driven)**

* **Vấn đề thực tế:** Ứng dụng desktop không chạy tuần tự từ trên xuống dưới. Nó phải "chờ" người dùng hành động (click chuột, gõ phím). Làm thế nào để chương trình "phản ứng" lại với người dùng?
* **Vận dụng kỹ năng Python:**
  + **Hàm Callback (Hàm gọi lại):** Tkinter vận dụng kỹ năng này triệt để.
  + **command=self.login**: Đây là kỹ năng "gán" một hành động (command) với một hàm (self.login). Khi nút được nhấn, Python sẽ "gọi lại" hàm này.
  + **self.tree.bind("<ButtonRelease-1>", self.on\_invoice\_select)**: Đây là kỹF-năng-lắng-nghe-sự-kiện-(Event-Listener)-ở-mức-độ-phức-tạp-hơn.-Python-không-chỉ-chờ-click-nút,-mà-còn-"lắng-nghe"-sự-kiện-thả-chuột-trái-trên-một-widget-cụ-thể-(Treeview)-để-kích-hoạt-hàm-hiển-thị-chi-tiết.
* **Giải quyết:** Kỹ năng lập trình sự kiện biến một kịch bản (script) tĩnh thành một **ứng dụng tương tác (interactive)**, giải quyết bài toán cốt lõi về tương tác Người-Máy (HCI).

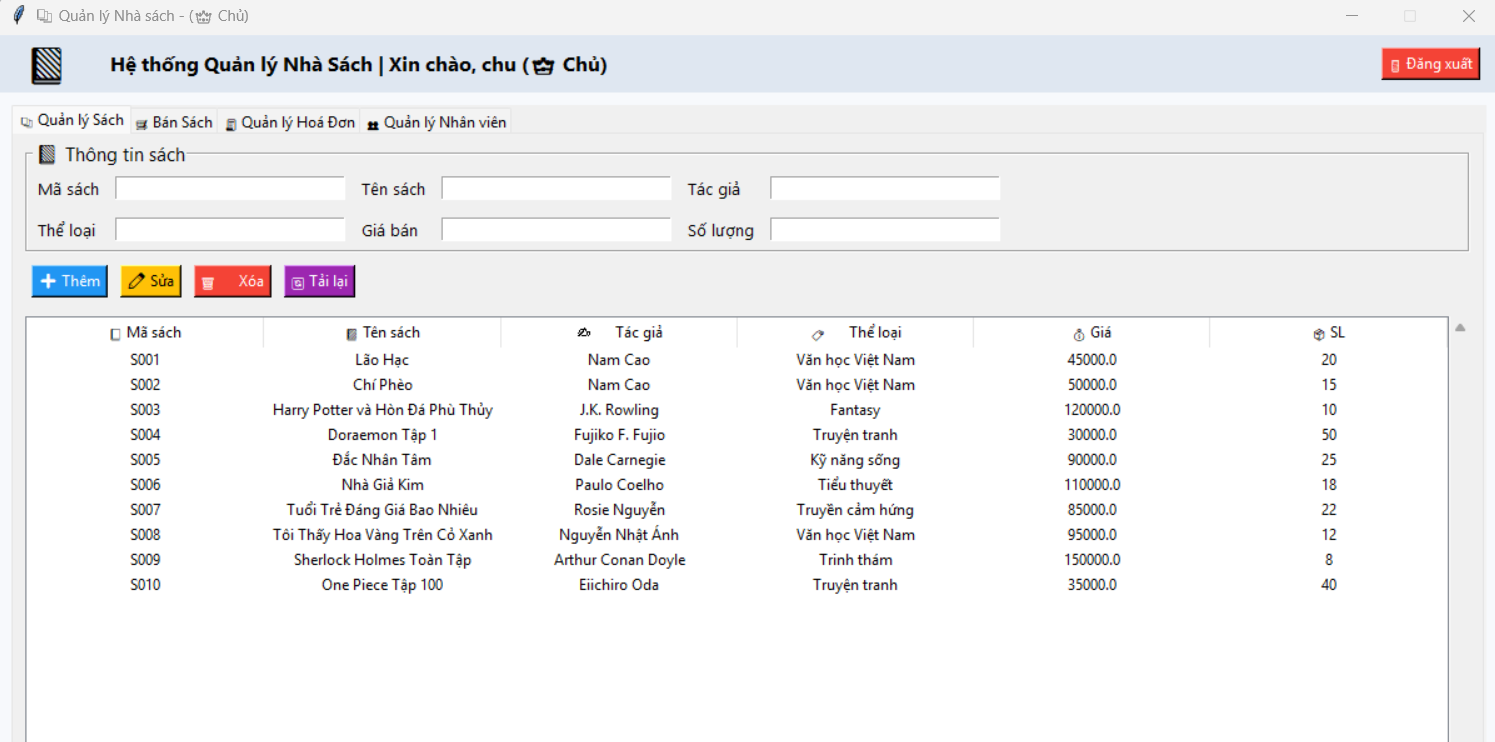
**5. Kỹ năng Giao tiếp CSDL và Bảo mật**

* **Vấn đề thực tế:** Dữ liệu phải được lưu trữ an toàn và nhất quán. Khi thanh toán, 4-5 hành động CSDL (Thêm hóa đơn, Thêm chi tiết bán, Trừ kho, Cập nhật điểm) phải diễn ra đồng thời.
* **Vận dụng kỹ năng Python:**
  + **Thư viện mysql-connector:** Đây là kỹ năng tích hợp thư viện bên thứ ba để giao tiếp với CSDL.
  + **Tham số hóa Truy vấn (Query Parameterization):**
    - **Code không an toàn (Dễ bị SQL Injection):** cur.execute(f"SELECT \* FROM taikhoan WHERE username='{user}'")
    - **Code an toàn (Đã vận dụng):** cur.execute("SELECT ... WHERE username=%s AND password=%s", (user, pw))
  + **Giải quyết:** Bằng cách truyền một tuple vào câu lệnh execute, thư viện mysql-connector sẽ tự động "làm sạch" (sanitize) dữ liệu. Đây là kỹ năng lập trình **phòng thủ (defensive programming)**, giúp giải quyết **lỗ hổng bảo mật SQL Injection** – một trong những rủi ro lớn nhất trong thực tế. Hơn nữa, việc nhóm các lệnh SQL vào một hàm (\_confirm\_checkout) là bước đầu tiên để tiến tới **Transaction (Giao dịch)**, đảm bảo tính nhất quán của dữ liệu.

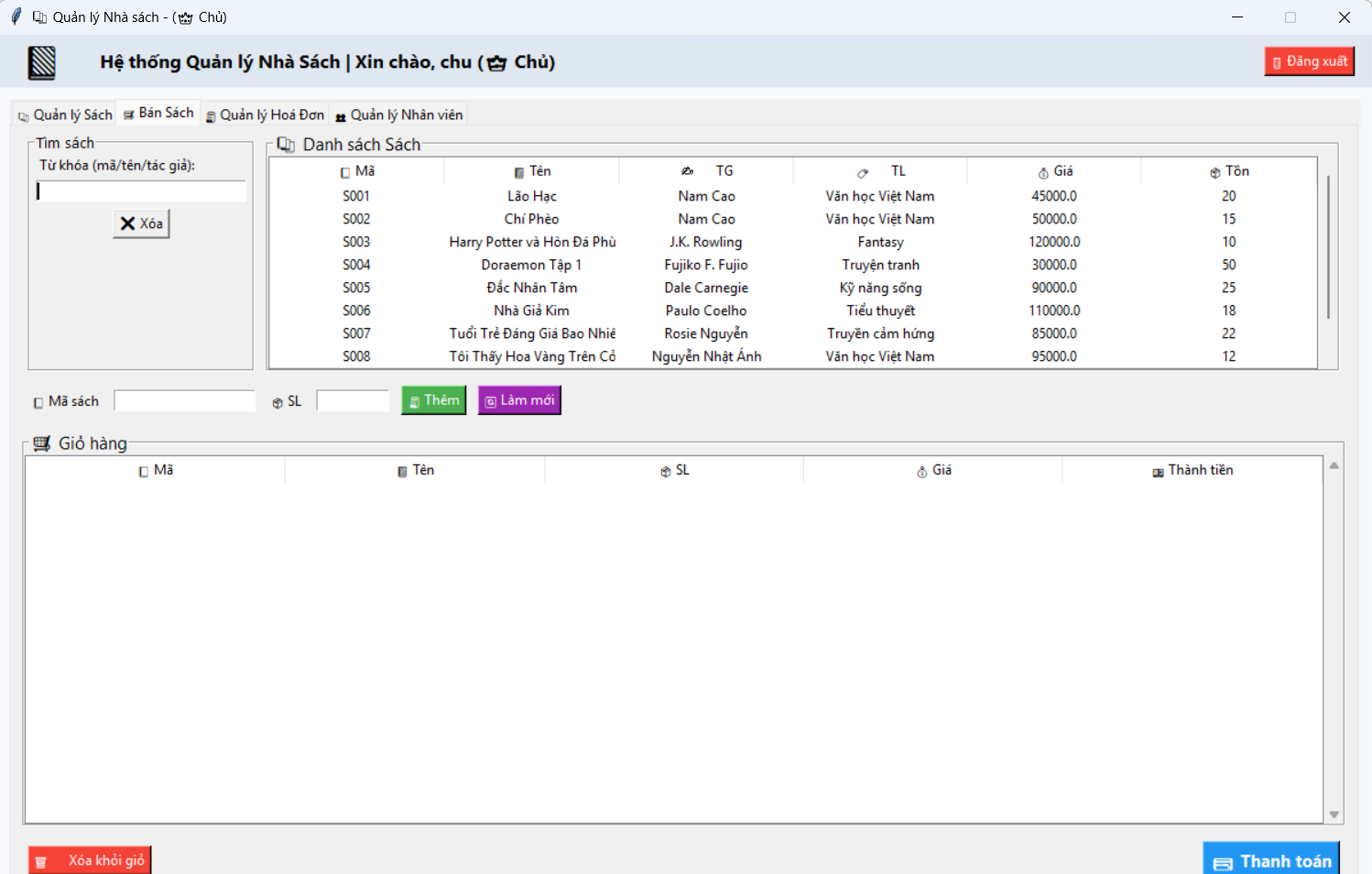
1. **THIẾT KẾ GIAO DIỆN QUẢN LÝ BÁN SÁCH:**

****

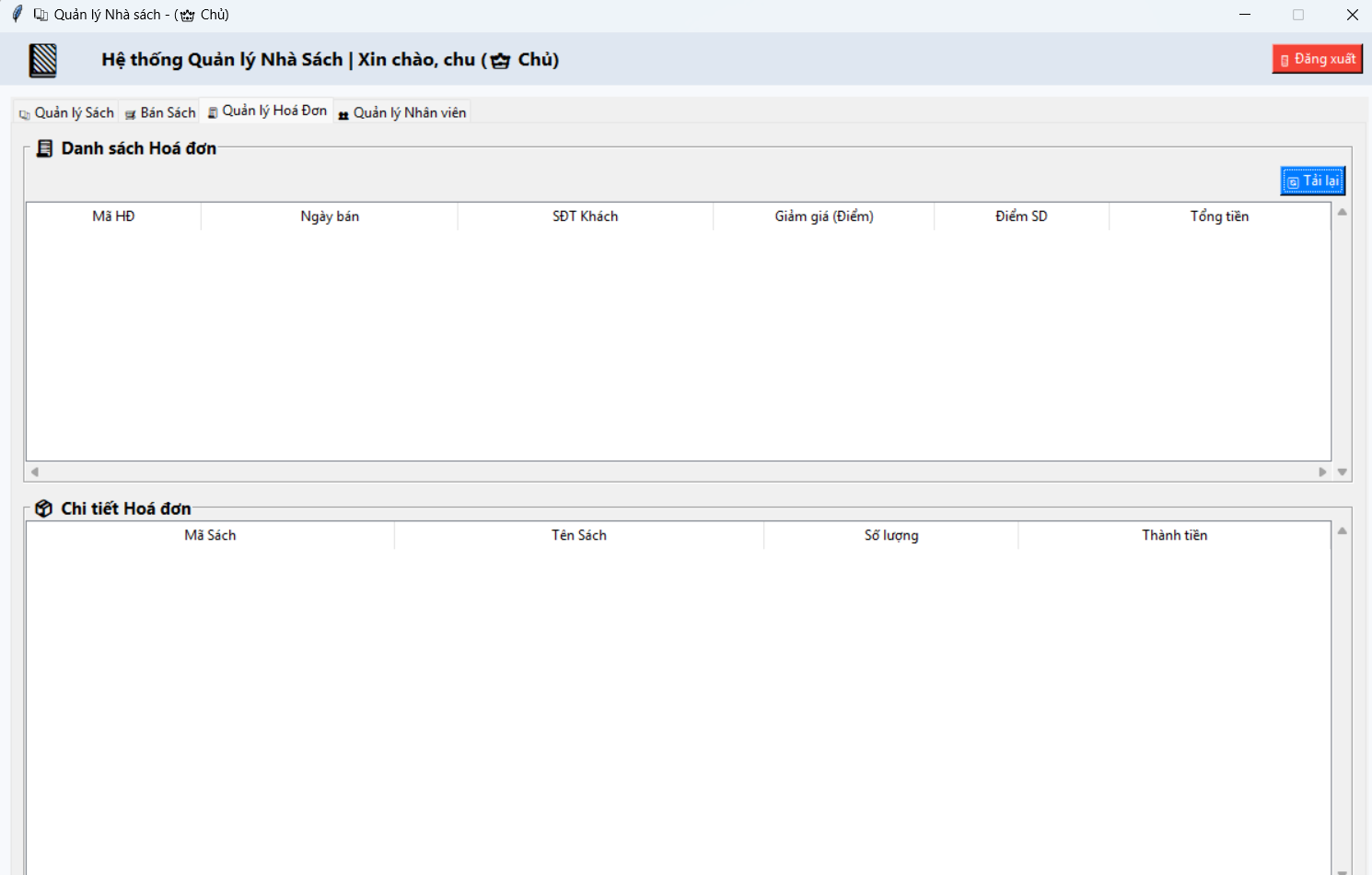
Đầu tiên sau chạy code(F5), kết quả cho ra đầu tiên sẽ là một cửa sổ đăng nhập. Sau khi đăng nhập tài khoản chủ ta sẽ được như hình bên dưới (có thể thay thế tài khoản bằng tài khoản của nhân viên).

****

**Tab quản lý bán sách**

****

**Tab bán sách (Thanh Toán)**

****

**Tab quản lý các thông tin của hoá đơn**

****

**Tab quản lý nhân viên**

1. CÁC BƯỚC XÂY DỰNG ỨNG DỤNG:

* **Để xây dựng ứng dụng ta cần các bước sau:**

**Bước 1: Thiết kế Cơ sở dữ liệu (CSDL) & Cài đặt Môi trường**

1. **Thiết kế CSDL**: Khởi động Mysql WorkBench. Tạo database tên qlsach. Dựa vaofys tưởng ban đầu cần ít nhất 5 bảng:
   * taikhoan (để đăng nhập, chứa username, password, quyen)
   * qlsach (danh sách sách, chứa ma\_sach, ten\_sach, tac\_gia, so\_luong, gia\_ban...)
   * khachhang (để tích điểm, chứa sdt, ten\_kh, diem\_tich\_luy)
   * hoadon (lưu thông tin thanh toán, chứa ma\_hd, ngay\_ban, tong\_tien, sdt\_kh...)
   * banhang (lưu chi tiết hóa đơn, chứa ma\_hd, ma\_sach, so\_luong...)
2. **Cài đặt Thư viện**: Mở terminal và chạy:



**Bước 2: Tạo Tệp Chính và các Hàm Hỗ trợ**

1. **Tạo tệp Python**: Tạo một tệp, ví dụ app.py.
2. **Import**: Thêm tất cả các thư viện cần thiết ở đầu tệp (Tkinter, mysql.connector, datetime, PIL...).
3. **Viết hàm tiện ích**: Sao chép 2 hàm quan trọng vào tệp:
   * connect\_db(): Để kết nối CSDL MySQL.
   * center\_window(): Để căn giữa các cửa sổ.

**Bước 3: Xây dựng Màn hình Đăng nhập (LoginScreen)**

1. **Tạo Lớp LoginScreen**: Đây là cửa ngõ vào ứng dụng.
2. **Giao diện (\_\_init\_\_)**: Thiết kế giao diện gồm logo, nền gradient (dùng draw\_gradient), ô nhập username/password và nút "Đăng nhập".
3. **Viết hàm login()**:
   * Lấy dữ liệu từ ô nhập.
   * Kết nối CSDL, truy vấn bảng taikhoan để kiểm tra.
   * Nếu đúng: Hủy cửa sổ đăng nhập (self.root.destroy()) và gọi main\_app() với quyền (quyen) và tên người dùng (user).
   * Nếu sai: Hiển thị messagebox.showerror.

**Bước 4: Xây dựng Khung Màn hình Chính (MainApp)**

1. **Tạo Lớp MainApp**: Lớp này sẽ chứa tất cả các tab chức năng.
2. **Giao diện (\_\_init\_\_)**:
   * Tạo cửa sổ chính (root), đặt tiêu đề, căn giữa.
   * Tạo header (thanh trên cùng) chứa logo, lời chào và nút "Đăng xuất".
   * Tạo một ttk.Notebook (đây là đối tượng sẽ chứa các tab).
3. **Viết hàm logout()**:
   * Hiển thị hộp thoại xác nhận.
   * Nếu đồng ý, hủy cửa sổ chính (self.root.destroy()) và gọi main\_app("login") để quay lại màn hình đăng nhập.

**Bước 5: Xây dựng các Tab Chức năng (Các Lớp Manager)**

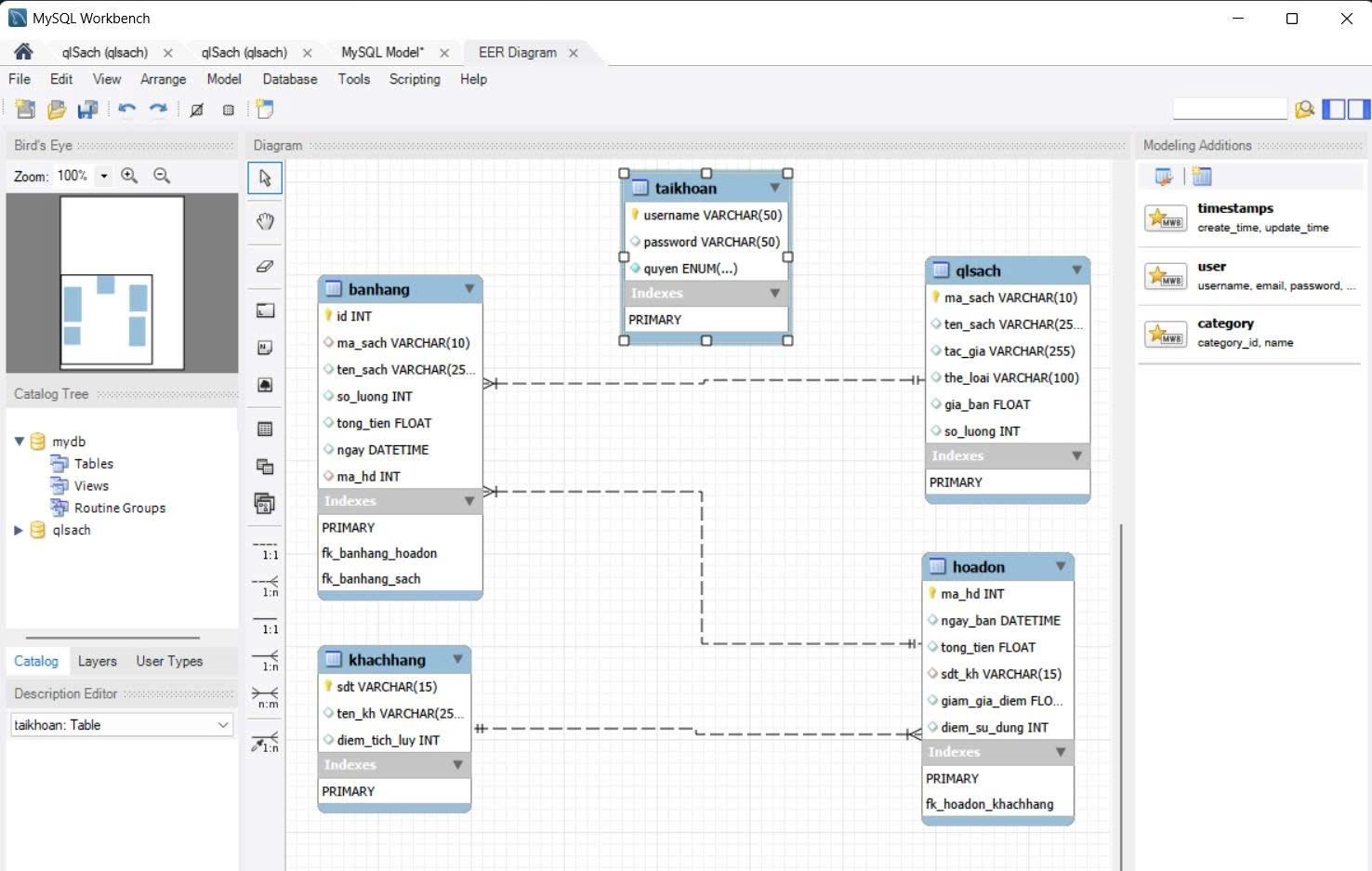
Đây là phần lõi, bạn sẽ tạo một lớp (Class) riêng cho mỗi tab:

1. **Tab Quản lý Sách (BookManager)**:
   * Tạo giao diện CRUD (ô nhập liệu, các nút Thêm/Sửa/Xóa, và một Treeview để hiển thị).
   * Viết các hàm load\_data (SELECT \*), add\_book (INSERT), update\_book (UPDATE), delete\_book (DELETE), và select\_row (để điền dữ liệu lên ô nhập khi nhấn vào).
2. **Tab Quản lý Bán Hàng (SaleManager)**:
   * **Giao diện**: Chia làm 2 phần chính: Bảng tìm kiếm/chọn sách và bảng "Giỏ hàng" (self.cart).
   * **Logic chính**:
     + load\_books(): Tải và tìm kiếm sách.
     + add\_to\_cart(): Thêm sách vào giỏ (self.cart), quan trọng là phải kiểm tra tồn kho (so\_luong) trước khi thêm.
     + checkout(): Mở cửa sổ pop-up thanh toán (Toplevel).
     + check\_customer\_points(): Xử lý logic phức tạp nhất: lấy SĐT, truy vấn bảng khachhang, tính toán giảm giá dựa trên \_calculate\_discount, và cập nhật các nhãn (Label) tiền.
     + \_\_confirm\_checkout(): "Chốt" đơn. Đây là một giao dịch (transaction) quan trọng:
       1. INSERT/UPDATE bảng khachhang (cập nhật điểm).
       2. INSERT vào bảng hoadon (tạo hóa đơn mới).
       3. UPDATE bảng qlsach (trừ tồn kho).
       4. INSERT vào bảng banhang (chi tiết hóa đơn).
       5. Xuất hóa đơn ra tệp .txt.
       6. Xóa giỏ hàng (self.cart.clear()).
3. **Tab Quản lý Hóa Đơn (InvoiceManager)**:
   * **Giao diện**: Chia 2 bảng (Treeview): Bảng trên (tree\_invoices) cho danh sách hóa đơn, bảng dưới (tree\_details) cho chi tiết.
   * **Logic liên kết**: Dùng sự kiện on\_invoice\_select. Khi nhấn vào một hóa đơn ở bảng trên, nó sẽ lấy ma\_hd và gọi load\_invoice\_details(ma\_hd) để hiển thị các món hàng của hóa đơn đó ở bảng dưới.
4. **Tab Quản lý Nhân viên (EmployeeManager)**:
   * Tương tự BookManager, đây là một giao diện CRUD đơn giản cho bảng taikhoan.
   * Bao gồm các hàm load\_users, add\_user, delete\_user.

**Bước 6: Tích hợp và Chạy Ứng dụng**

1. **Hoàn thiện MainApp**: Trong hàm \_\_init\_\_ của MainApp, sau khi tạo ttk.Notebook, bạn bắt đầu tạo các tab và gọi các lớp Manager:
   * BookManager(self.book\_tab)
   * SaleManager(self.sale\_tab, ...)
   * InvoiceManager(self.invoice\_tab)
   * **Quan trọng**: Dùng if quyen == "chu": để kiểm tra. Chỉ khi người dùng là "chủ" thì mới tạo tab EmployeeManager.
2. **Tạo Điểm vào (main\_app)**:
   * Tạo hàm main\_app(mode, ...) như bạn đã thấy (để quyết định mở LoginScreen hay MainApp).
3. Chạy app.

* **Thiết kế cơ sở dữ liệu Mysql Workbench:**

****

**Sơ đồ mối liên hệ Diargam**

* **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 các bảng hiển thị (sử dụng ttk.Notebook cho các tab) để quản lý Sách và Bán Hàng. Thiết kế giao diện cho phép người dùng thực hiện các chức năng sau:
  + Tab Quản lý Sách:
    - Thêm thông tin Sách mới.
    - Sửa thông tin Sách.
    - Xóa thông tin Sách.
    - Xem và tải lại danh sách tất cả Sách.
    - Tìm kiếm Sách.
  + Tab Bán Hàng:
    - Tìm kiếm Sách trong kho.
    - Thêm Sách vào giỏ hàng (với số lượng).
    - Xóa mặt hàng khỏi giỏ hàng.
    - Thực hiện Thanh toán:
      * Kiểm tra SĐT khách hàng để tích/dùng điểm.
      * Tính toán giảm giá và tổng tiền.
      * Xác nhận thanh toán và xuất hóa đơn.
  + Các chức năng khác: (VD: Tab xem lại Hóa đơn, Tab Quản lý Nhân viên - nếu đăng nhập với quyền "chủ").

4. Viết mã Python kết nối cơ sở dữ liệu:

* Sử dụng mysql.connector để thiết lập kết nối với cơ sở dữ liệu MySQL (qlsach).
* Viết các hàm:
  + Thêm Sách: Lấy dữ liệu từ các trường nhập liệu và INSERT vào bảng qlsach.
  + Sửa Sách: UPDATE thông tin Sách trong CSDL dựa trên ma\_sach.
  + Xóa Sách: DELETE Sách khỏi CSDL dựa trên ma\_sach.
  + Xem danh sách Sách: SELECT \* FROM qlsach và hiển thị lên Treeview.
  + Tìm kiếm Sách: SELECT từ qlsach với điều kiện LIKE tên sách hoặc mã sách.
  + Xử lý Thanh toán (Transaction):
    - UPDATE qlsach để trừ số lượng tồn kho đã bán.
    - INSERT INTO hoadon để tạo một hóa đơn mới.
    - INSERT INTO banhang để lưu chi tiết các mặt hàng thuộc hóa đơn đó.
    - INSERT ... ON DUPLICATE KEY UPDATE vào bảng khachhang để cập nhật điểm tích lũy.

1. **Hướng dẫn cài đặt và code ứng dụng:**

Ứng dụng sẽ có:

-Giao diện nhập thông tin các loại sách( Mã sách, Tên sách, Tác giả, Thể loại, Giá bán, Số lượng)

-Bảng danh sách các loại sách

-Chức năng CRUD ( thêm, sửa, xoá, lưu)

-Dữ liệu lưu trực tiếp vào Mysql WorkBench Database ( không mất khi tắt ứng dụng)

-Giao diện Tkinter chuẩn như hình minh hoạ trên

**1.Cài đặt môi trường cần thiết:**

Trước tiên cài đặt môi trường cần thiết:



Sau khi chạy lệnh trên, cần tiếp tục lệnh:

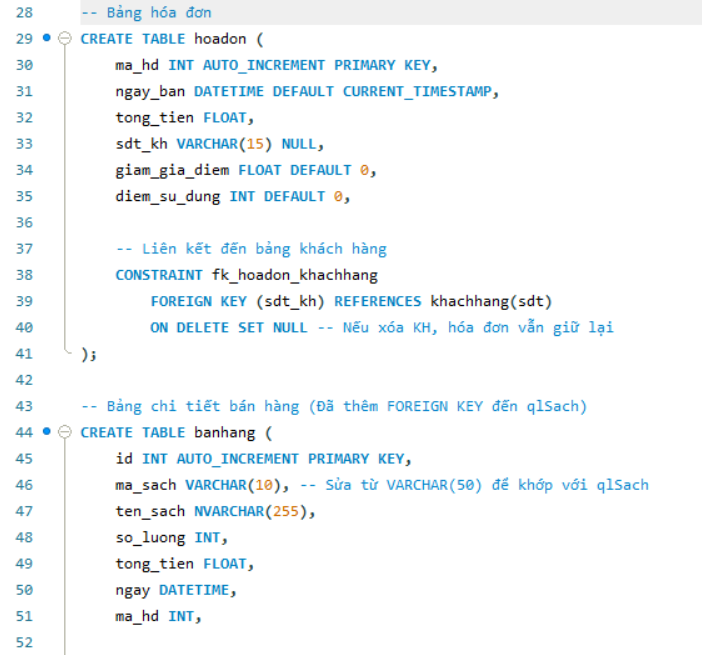
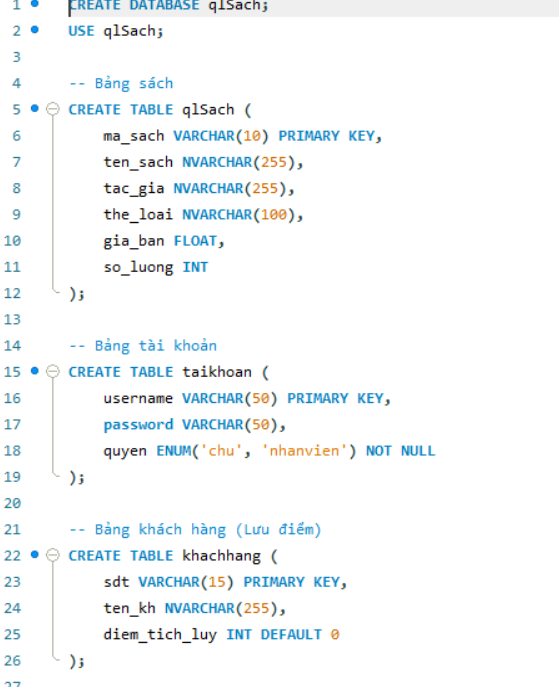


🡪Giải thích:

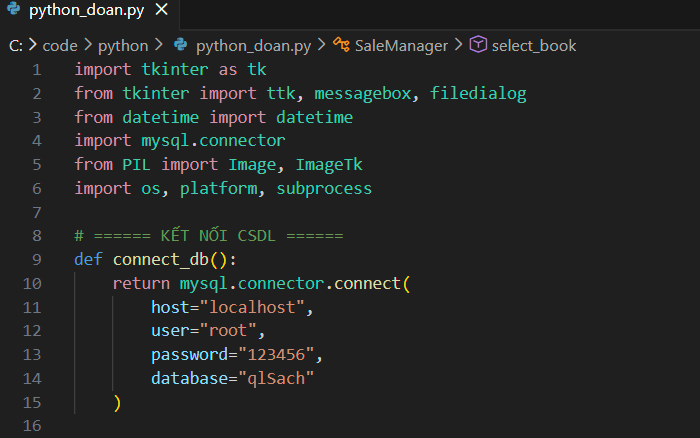
* **mysql-connector-python**: Đây là thư viện "cầu nối" chính thức. Nó cho phép code Python của bạn **nói chuyện (kết nối, truy vấn, cập nhật) với cơ sở dữ liệu MySQL**.
* **tkcalendar**: Đây là thư viện mở rộng cho Tkinter. Nó cung cấp các tiện ích (widget) để **chọn ngày tháng (như lịch pop-up DateEntry)** vì Tkinter cơ bản không có sẵn công cụ này.

**2.Viết code ứng dụng với Tkinter và Mysql:**

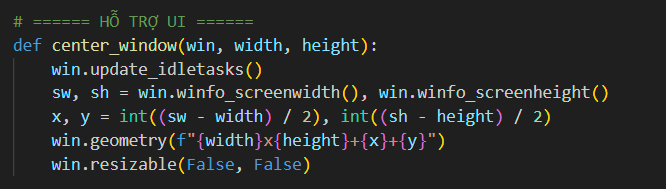
Mở Mysql WorkBench để tạo database và các bảng:



Mở Visual code và thực hiện viết code:

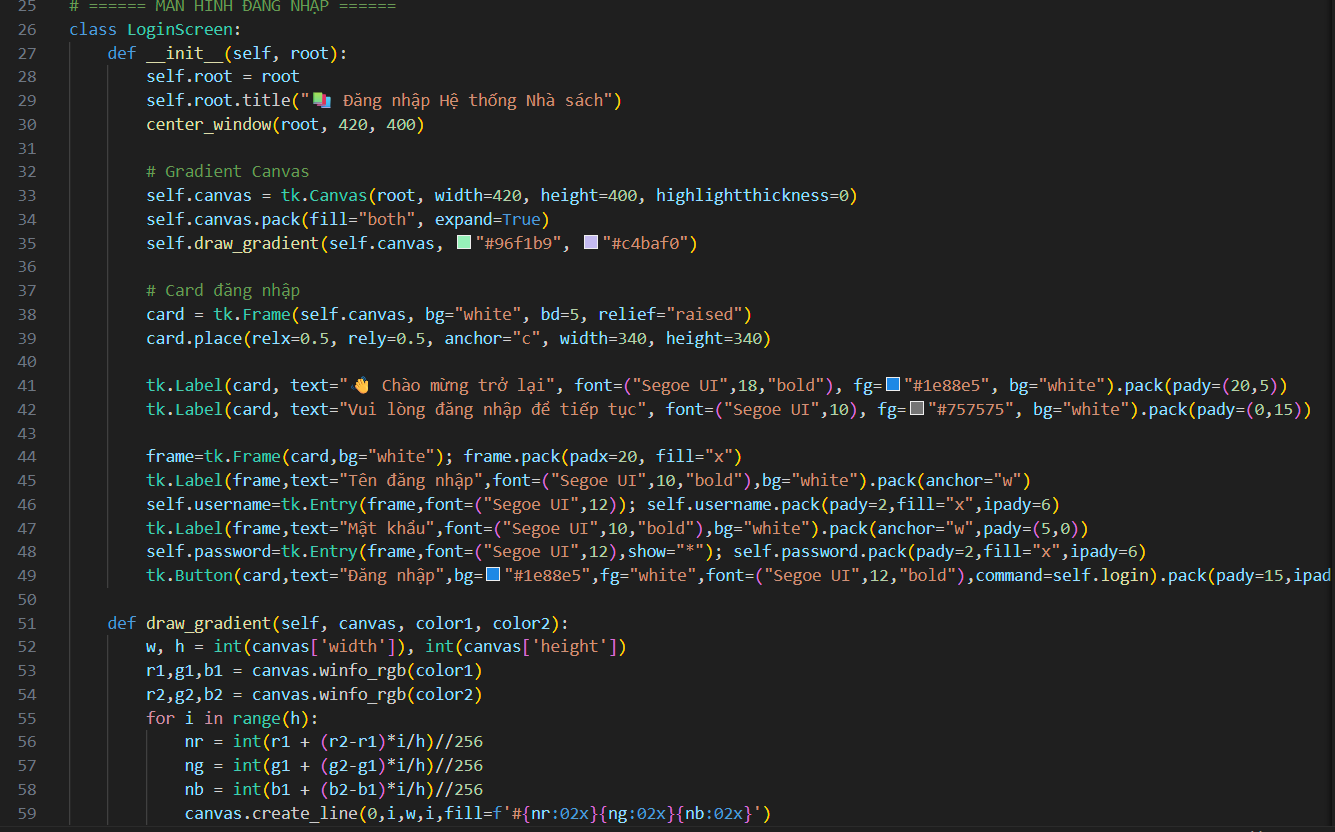
****

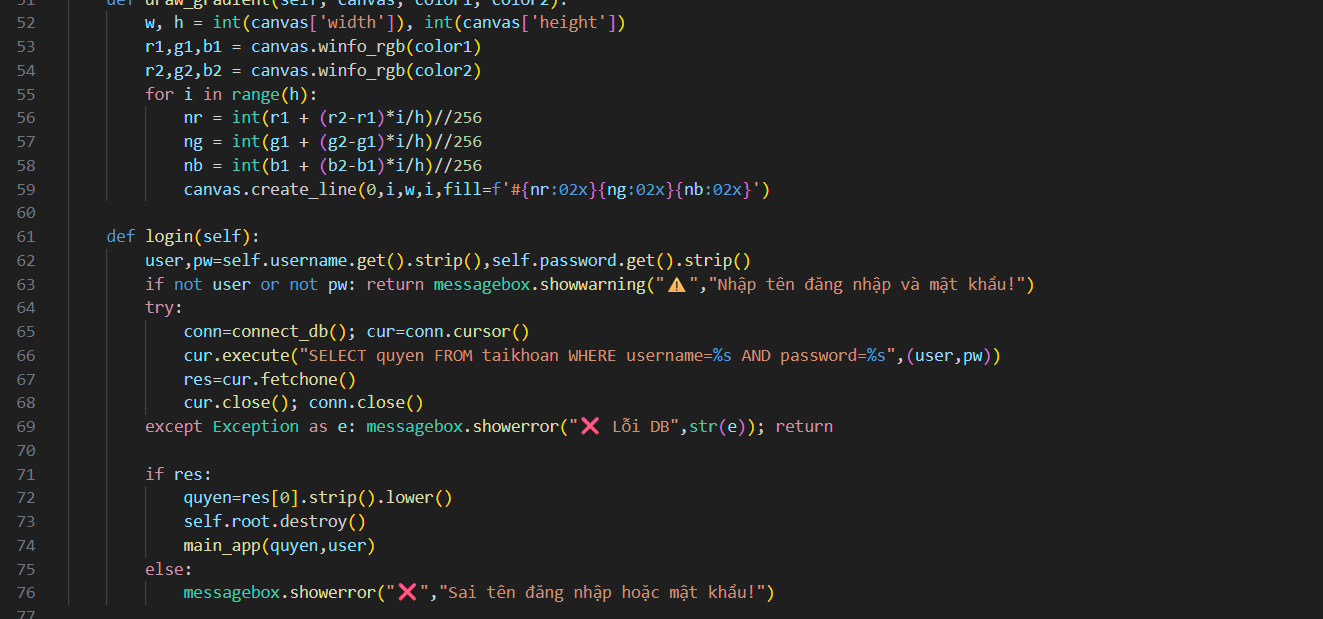
Đoạn mã này chuẩn bị các "công cụ" (thư viện) cần thiết cho một ứng dụng, bao gồm tkinter để tạo giao diện, mysql.connector để làm việc với cơ sở dữ liệu, và PIL để xử lý ảnh. Sau đó, nó định nghĩa một hàm tên là connect\_db() dùng để kết nối vào cơ sở dữ liệu MySQL có tên là "qlsach" đang chạy trên máy.



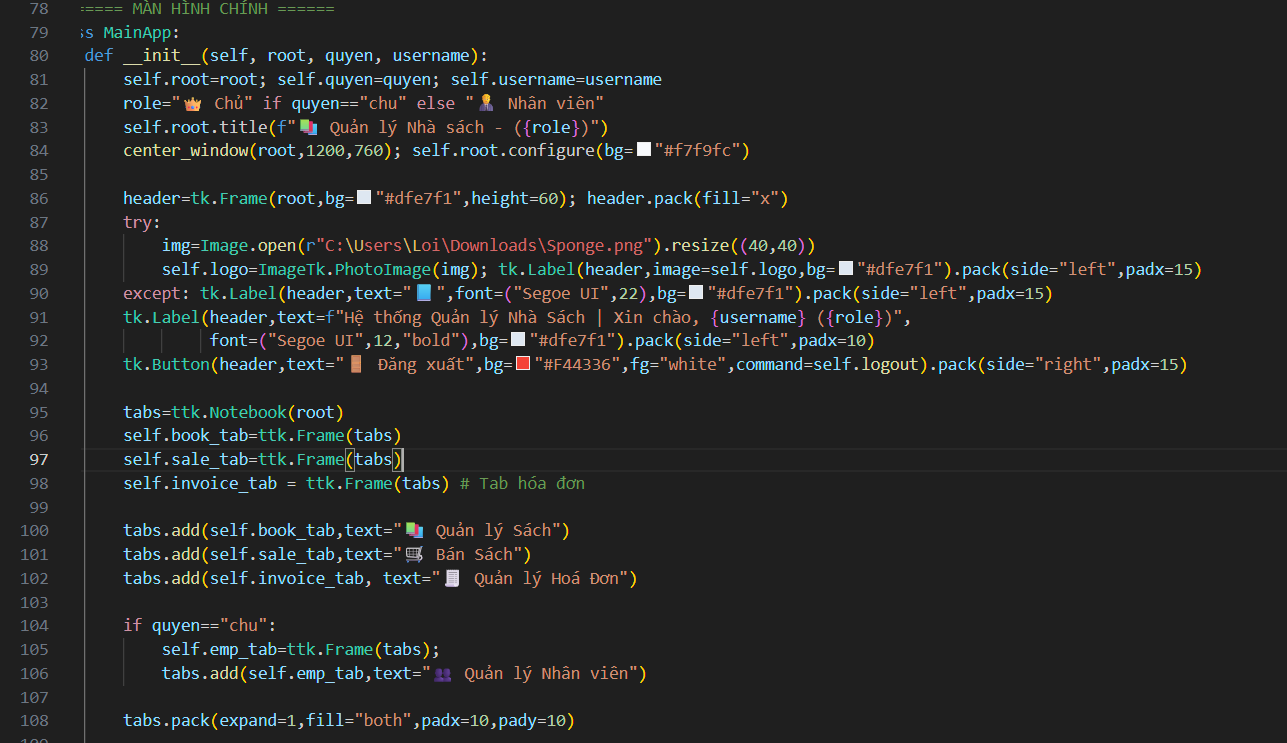
Hàm center\_window này dùng để **đưa một cửa sổ (win) ra chính giữa màn hình**

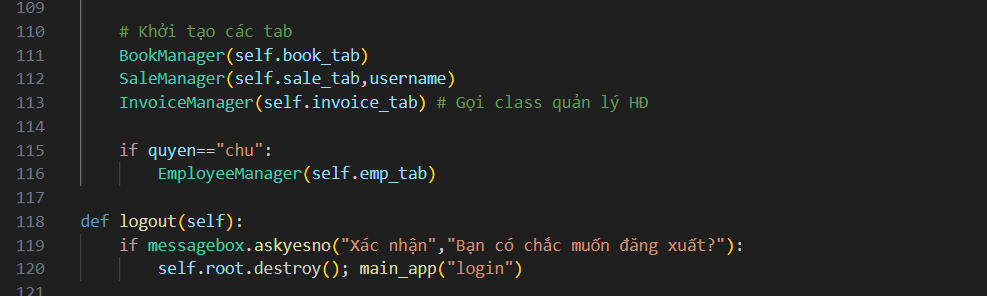
Nó hoạt động bằng cách lấy kích thước tổng thể của màn hình, sau đó tính toán tọa độ x và y (góc trên bên trái) cần thiết để cửa sổ hiển thị cân đối ở giữa. Cuối cùng, nó dùng win.geometry() để áp dụng vị trí và kích thước này, đồng thời win.resizable(False, False) để khóa, không cho người dùng thay đổi kích thước cửa sổ.

****

****

Đoạn mã này tạo ra một màn hình đăng nhập (class LoginScreen) bằng Tkinter. Khi khởi tạo (\_\_init\_\_), nó sẽ tạo một cửa sổ, căn giữa nó, và vẽ một nền dốc màu (gradient) đẹp mắt bằng hàm draw\_gradient. Trên nền đó, nó hiển thị các ô nhập "Tên đăng nhập", "Mật khẩu" và một nút "Đăng nhập". Khi bạn nhấn nút, hàm login sẽ được gọi. Hàm này lấy thông tin bạn nhập, kết nối với cơ sở dữ liệu MySQL (taikhoan table) để kiểm tra xem tên đăng nhập và mật khẩu có đúng không. Nếu đúng, nó sẽ đóng cửa sổ đăng nhập và mở ứng dụng chính (main\_app); nếu sai, nó sẽ hiển thị hộp thoại báo lỗi.





Hai đoạn mã này mô tả **Màn Hình Chính** của ứng dụng (MainApp), xuất hiện sau khi bạn đăng nhập thành công:

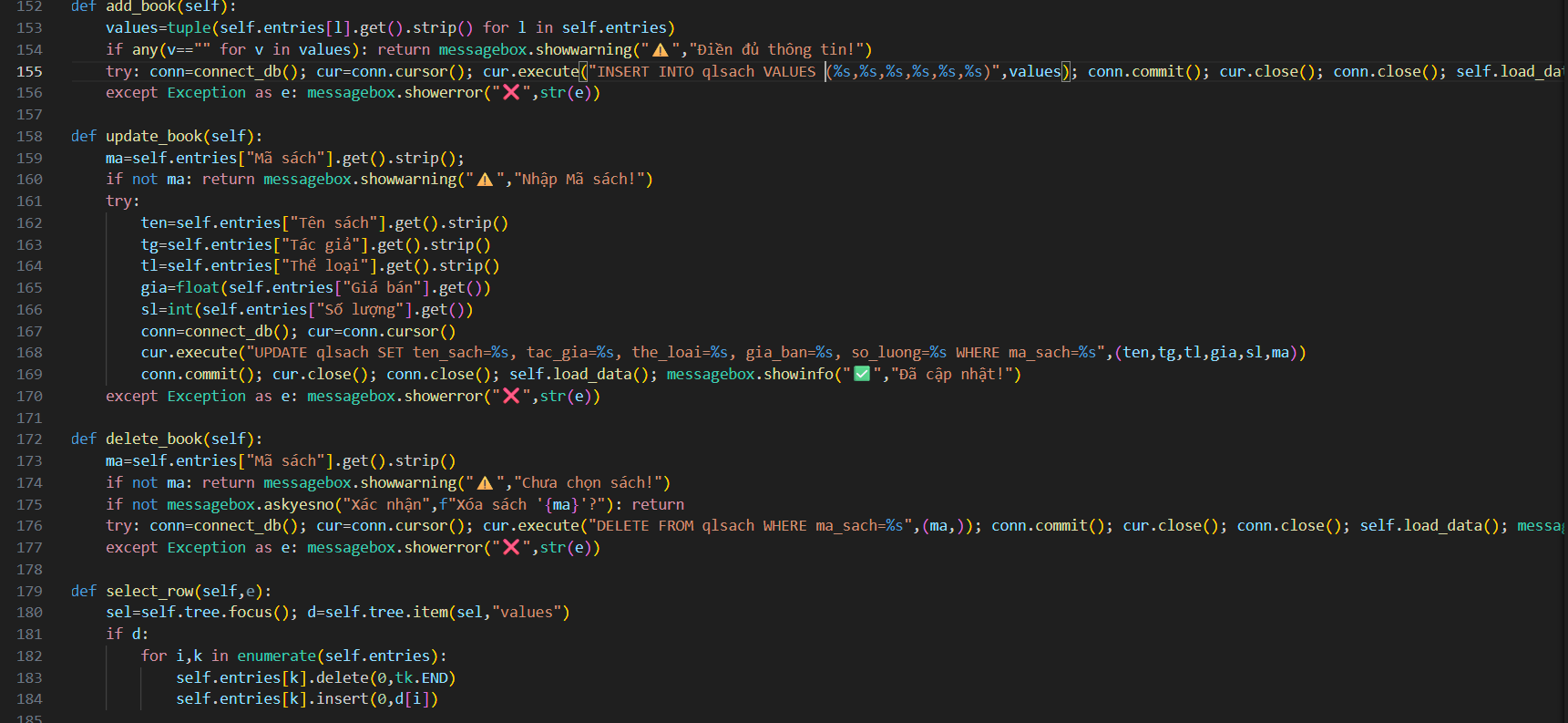
**Phần \_\_init\_\_ (Thiết lập ban đầu):**

* + Nó tạo cửa sổ chính, đặt tiêu đề và căn giữa.
  + Nó tạo một thanh header (đầu trang) chứa logo, lời chào (Xin chào, [username]) và nút "Đăng xuất".
  + Nó tạo một khu vực hiển thị dạng **tab** (ttk.Notebook).
  + Nó tạo các tab cơ bản: "Quản lý Sách", "Bán Sách", "Quản lý Hoá Đơn" và gọi các class tương ứng (BookManager, SaleManager,...) để đưa nội dung vào các tab đó.
  + **Phần quan trọng:** Nó kiểm tra quyền (quyen). Nếu quyen == "chu" (bạn là chủ), nó sẽ tạo thêm tab "Quản lý Nhân viên" (EmployeeManager). Nếu không, tab này sẽ bị ẩn đi.

**Phần logout (Đăng xuất):**

* + Đây là hàm được gọi khi nhấn nút "Đăng xuất".
  + Nó hiển thị một hộp thoại xác nhận ("Bạn có chắc muốn đăng xuất?").
  + Nếu bạn chọn "Yes", nó sẽ đóng cửa sổ chính hiện tại (self.root.destroy()) và gọi main\_app("login") để quay trở lại màn hình đăng nhập.





Đoạn mã trên định nghĩa lớp BookManager (Quản lý Sách), tạo nên giao diện và tất cả các chức năng cho **tab "Quản lý Sách"** của bạn. Đây là một giao diện CRUD (Thêm, Đọc, Cập nhật, Xóa) hoàn chỉnh.

**1. Hàm \_\_init\_\_ (Thiết lập Giao diện)**

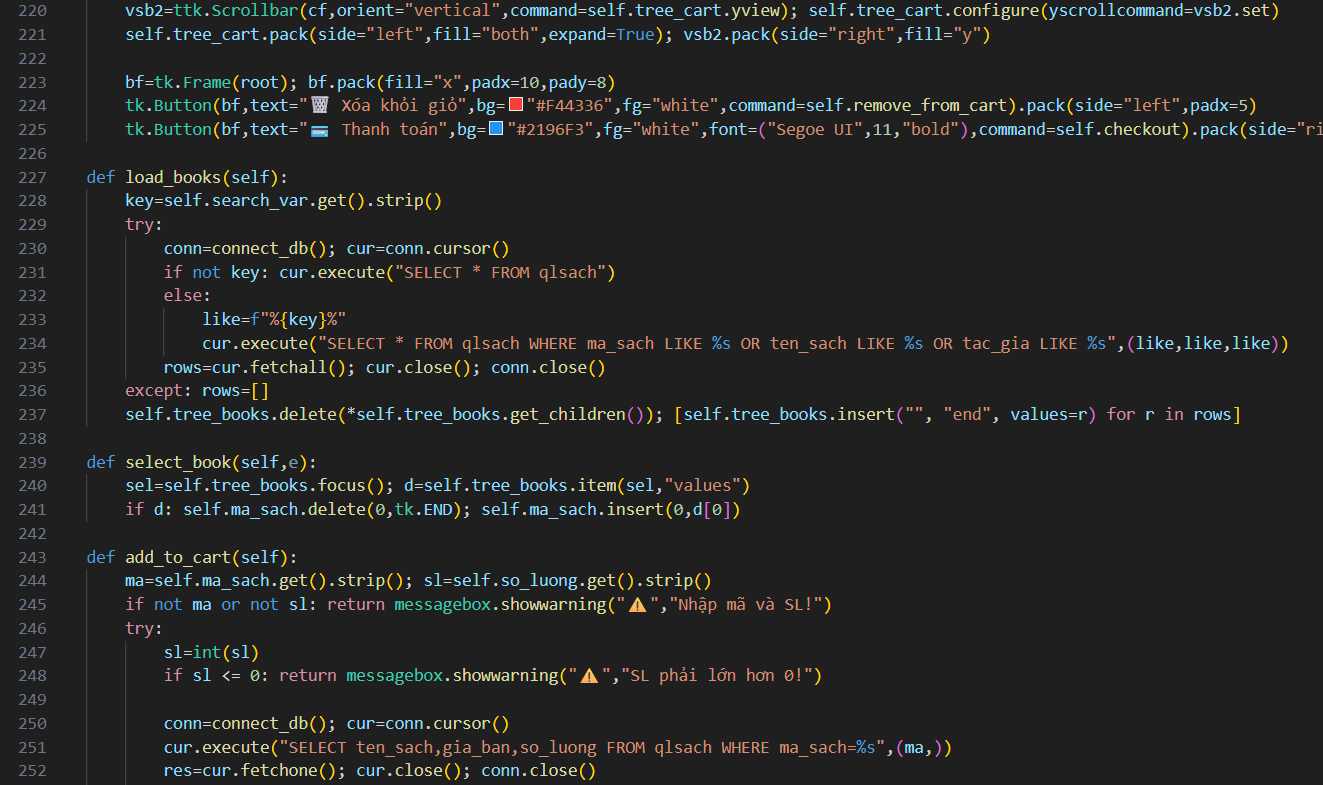
Hàm này sẽ chạy đầu tiên để "vẽ" lên tab quản lý sách:

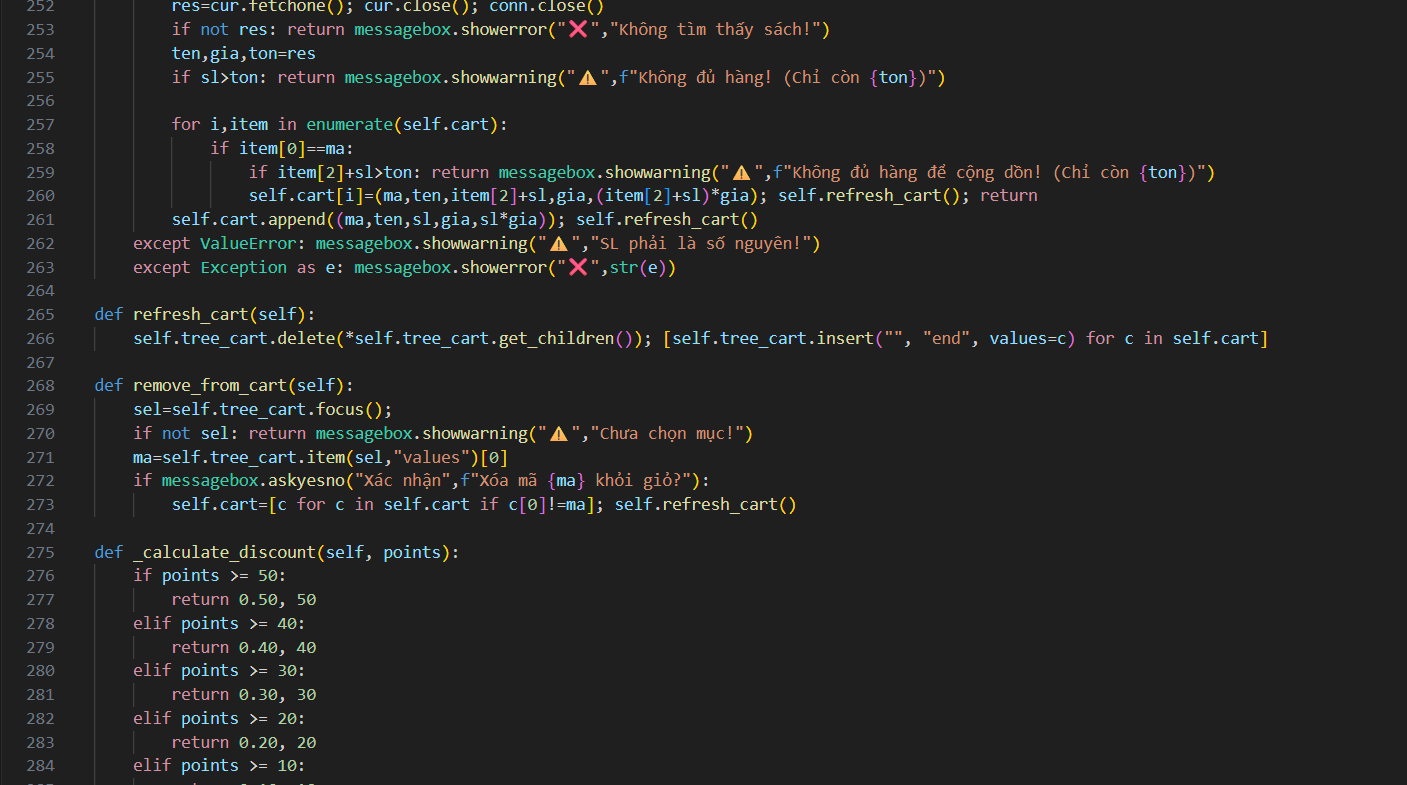
* **Ô nhập liệu (Entry):** Nó tự động tạo ra một loạt các nhãn (như "Mã sách", "Tên sách",...) và các ô nhập liệu tương ứng để bạn điền thông tin.
* **Các nút bấm (Button):** Nó tạo ra 4 nút: "Thêm", "Sửa", "Xóa", và "Tải lại". Mỗi nút được gán cho một lệnh (ví dụ: nút "Thêm" sẽ gọi hàm self.add\_book).
* **Bảng hiển thị (Treeview):** Nó tạo một bảng lớn ở dưới để hiển thị danh sách tất cả sách từ cơ sở dữ liệu (CSDL).
* **Sự kiện (Event):** Nó tạo một sự kiện (self.tree.bind...): Khi bạn **nhấn chuột vào một hàng trong bảng**, nó sẽ tự động gọi hàm self.select\_row.
* Cuối cùng, nó gọi self.load\_data() để tải dữ liệu lên bảng ngay khi mở tab.

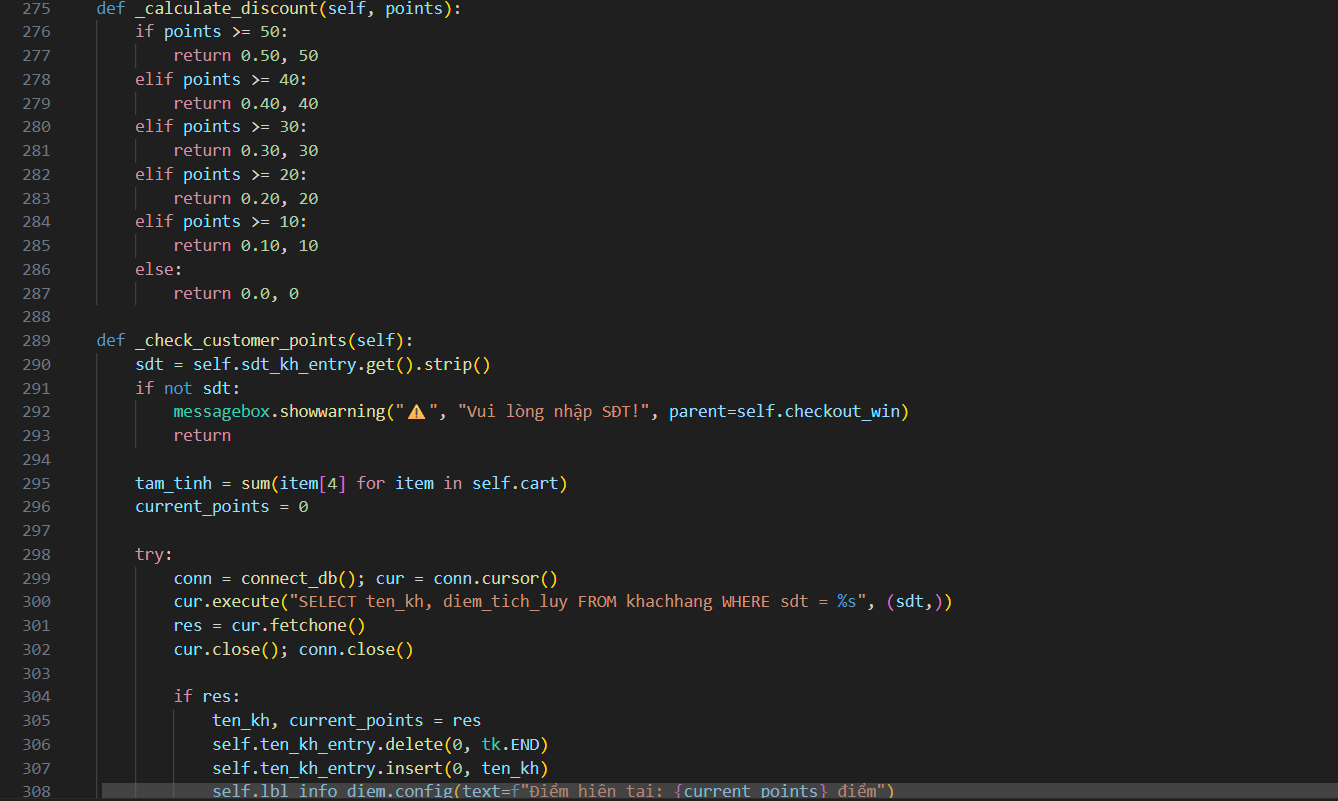
1. **Các hàm chức năng**

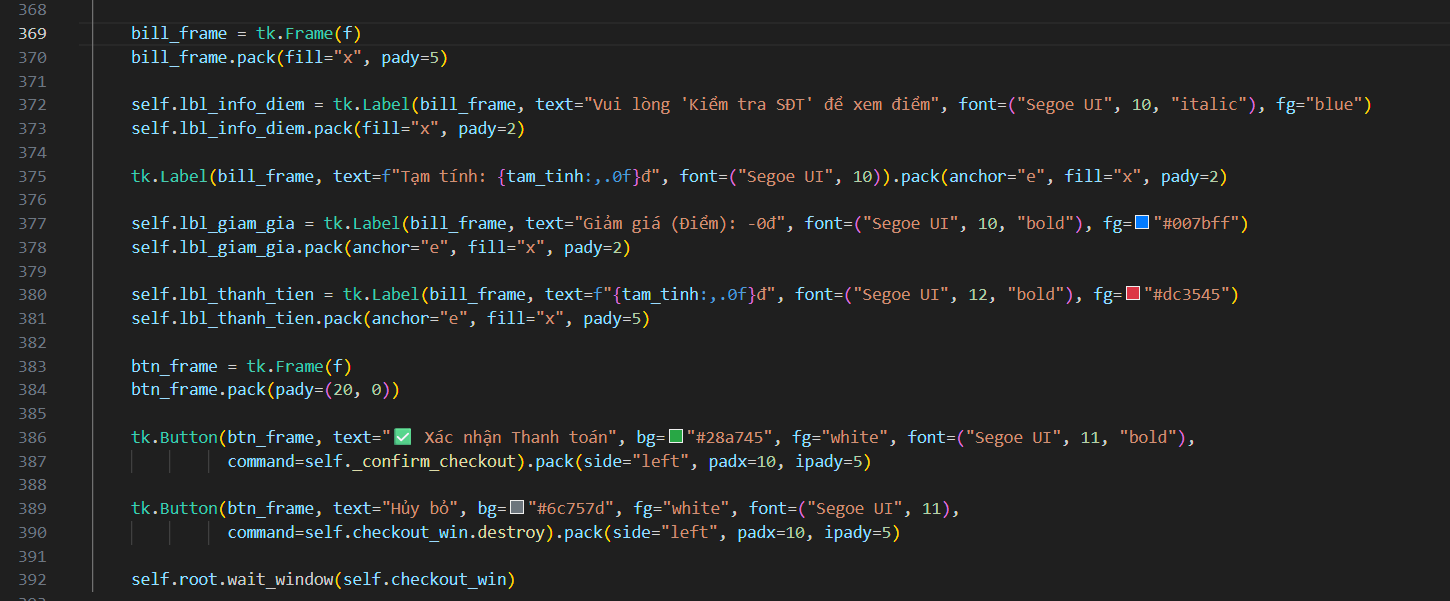
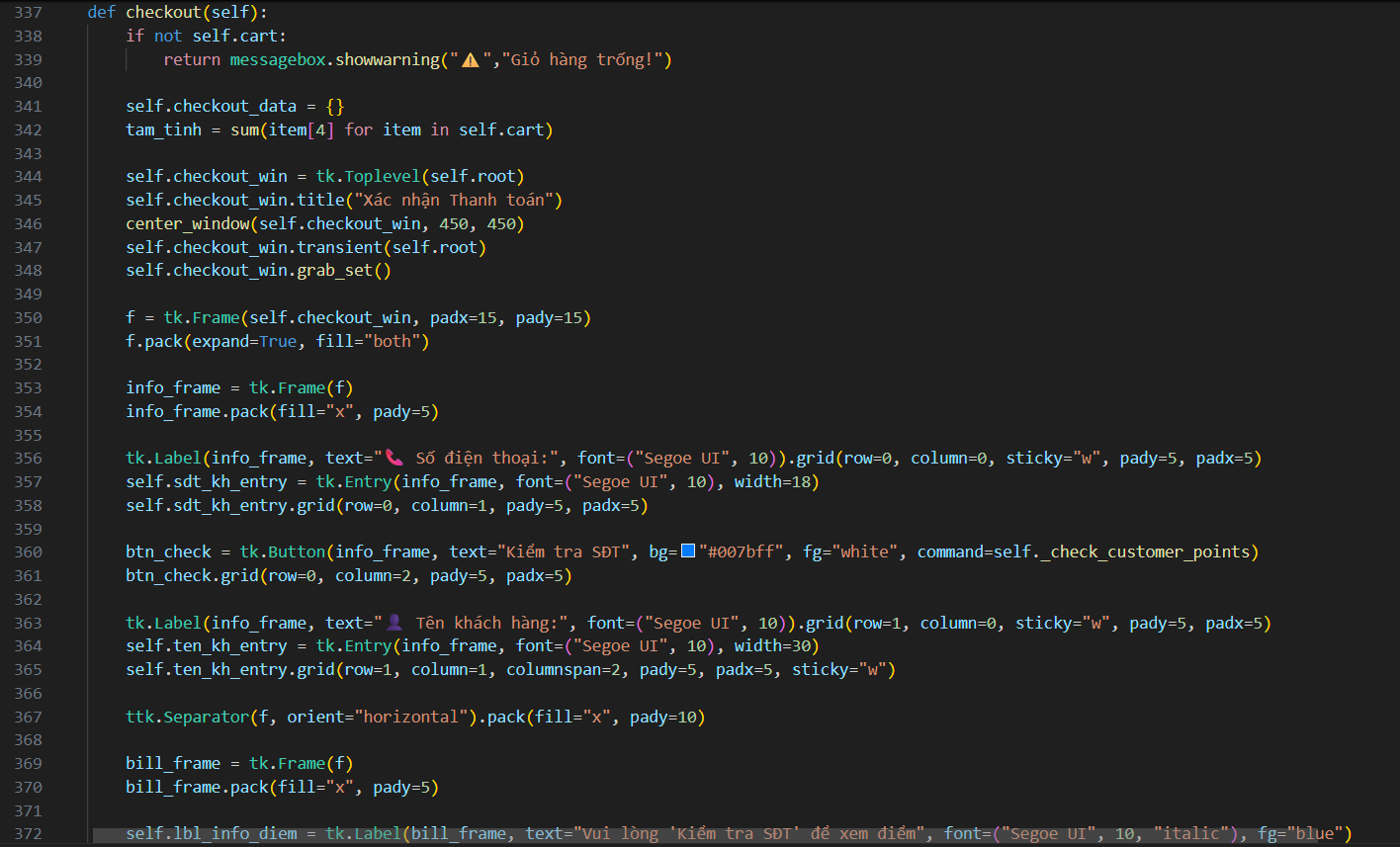
* **load\_data(self):**
  + Kết nối CSDL, chạy lệnh SELECT \* FROM qlsach để lấy *tất cả* sách.
  + Xóa sạch dữ liệu cũ trên bảng Treeview.
  + Đổ dữ liệu mới lấy được vào bảng.
* **add\_book(self):**
  + Được gọi khi nhấn nút "Thêm".
  + Lấy toàn bộ thông tin bạn đã nhập trong các ô.
  + Kiểm tra xem bạn có bỏ trống ô nào không (nếu có thì báo lỗi).
  + Chạy lệnh INSERT INTO qlsach... để thêm cuốn sách mới này vào CSDL.
  + Tự động gọi load\_data() để làm mới lại bảng.
* **update\_book(self):**
  + Được gọi khi nhấn nút "Sửa".
  + Lấy "Mã sách" từ ô nhập liệu (bắt buộc phải có).
  + Lấy *tất cả thông tin mới* từ các ô còn lại.
  + Chạy lệnh UPDATE qlsach SET ... WHERE ma\_sach=... để cập nhật thông tin cho cuốn sách đó.
  + Thông báo "Đã cập nhật!" khi thành công.
* **delete\_book(self):**
  + Được gọi khi nhấn nút "Xóa".
  + Lấy "Mã sách" từ ô nhập liệu.
  + Hiển thị hộp thoại "Xác nhận" hỏi bạn có chắc muốn xóa không.
  + Nếu bạn đồng ý, nó sẽ chạy lệnh DELETE FROM qlsach WHERE ma\_sach=... để xóa sách.
  + Tự động gọi load\_data() để làm mới lại bảng.
* **select\_row(self):**
  + Đây là hàm tiện ích, được gọi khi bạn *nhấn vào một hàng trong bảng*.
  + Nó lấy tất cả dữ liệu từ hàng bạn vừa nhấn.
  + Nó **điền ngược** dữ liệu đó lên các ô nhập liệu ở trên.
  + **Mục đích:** Giúp bạn chọn một cuốn sách để Sửa hoặc Xóa rất nhanh, không cần phải gõ lại "Mã sách".











Lớp này quản lý toàn bộ **tab "Bán Hàng"**, cho phép người dùng tìm sách, thêm sách vào giỏ hàng, và thực hiện quy trình thanh toán (checkout) có tích hợp hệ thống điểm khách hàng.

Dưới đây là giải thích từng phần:

**1. \_\_init\_\_ (Khởi tạo Giao diện)**

* **Mục đích**: Thiết lập toàn bộ giao diện cho tab bán hàng.
* **Hoạt động**:
  + Khởi tạo một giỏ hàng rỗng (self.cart = []).
  + Tạo khu vực **Tìm kiếm sách**: Gồm ô nhập liệu và nút "Xóa" (để xóa nội dung tìm kiếm).
  + Tạo bảng **Danh sách Sách** (Treeview): Hiển thị sách từ CSDL. Bảng này được gắn sự kiện select\_book (khi nhấn vào một hàng).
  + Tạo khu vực **Thêm vào giỏ**: Gồm ô nhập "Mã sách", "Số lượng" (SL) và các nút "Thêm", "Làm mới".
  + Tạo bảng **Giỏ hàng** (Treeview): Hiển thị các mặt hàng đang có trong giỏ.
  + Tạo các nút **Quản lý giỏ hàng**: Gồm "Xoá khỏi giỏ" (gọi remove\_from\_cart) và "Thanh toán" (gọi checkout).
  + Cuối cùng, gọi self.load\_books() để tải danh sách sách lên bảng.

**2. Tải sách và Thêm vào giỏ**

* **load\_books(self)**:
  + Kết nối CSDL.
  + Lấy nội dung từ ô tìm kiếm. Nếu ô tìm kiếm trống, nó chạy SELECT \* FROM qlsach (lấy tất cả sách).
  + Nếu có nội dung, nó chạy SELECT \*... WHERE ma\_sach LIKE ... OR ten\_sach LIKE ... để thực hiện tìm kiếm.
  + Sau đó, nó xóa dữ liệu cũ trên bảng và hiển thị kết quả tìm kiếm mới.
* **select\_book(self)**:
  + Hàm này được gọi khi bạn nhấn vào một cuốn sách trong bảng "Danh sách Sách".
  + Nó lấy mã sách của hàng được chọn và tự động điền vào ô "Mã sách" ở khu vực "Thêm vào giỏ", giúp bạn không cần gõ lại.
* **add\_to\_cart(self)**:
  + Đây là một hàm quan trọng, được gọi khi nhấn nút "Thêm".
  + **Kiểm tra đầu vào**: Đảm bảo bạn đã nhập "Mã sách" và "Số lượng" (phải là số và > 0).
  + **Kiểm tra kho**: Kết nối CSDL để lấy thông tin sách, đặc biệt là so\_luong (tồn kho).
  + Nếu số lượng bạn muốn mua (sl) lớn hơn số lượng tồn kho (ton), nó sẽ báo lỗi "Không đủ hàng".
  + **Thêm vào giỏ**: Nếu hàng đã có trong self.cart, nó sẽ cập nhật (cộng dồn) số lượng. Nếu chưa có, nó sẽ thêm mới vào danh sách self.cart.
  + Cuối cùng, gọi self.refresh\_cart() để cập nhật hiển thị trên bảng "Giỏ hàng".

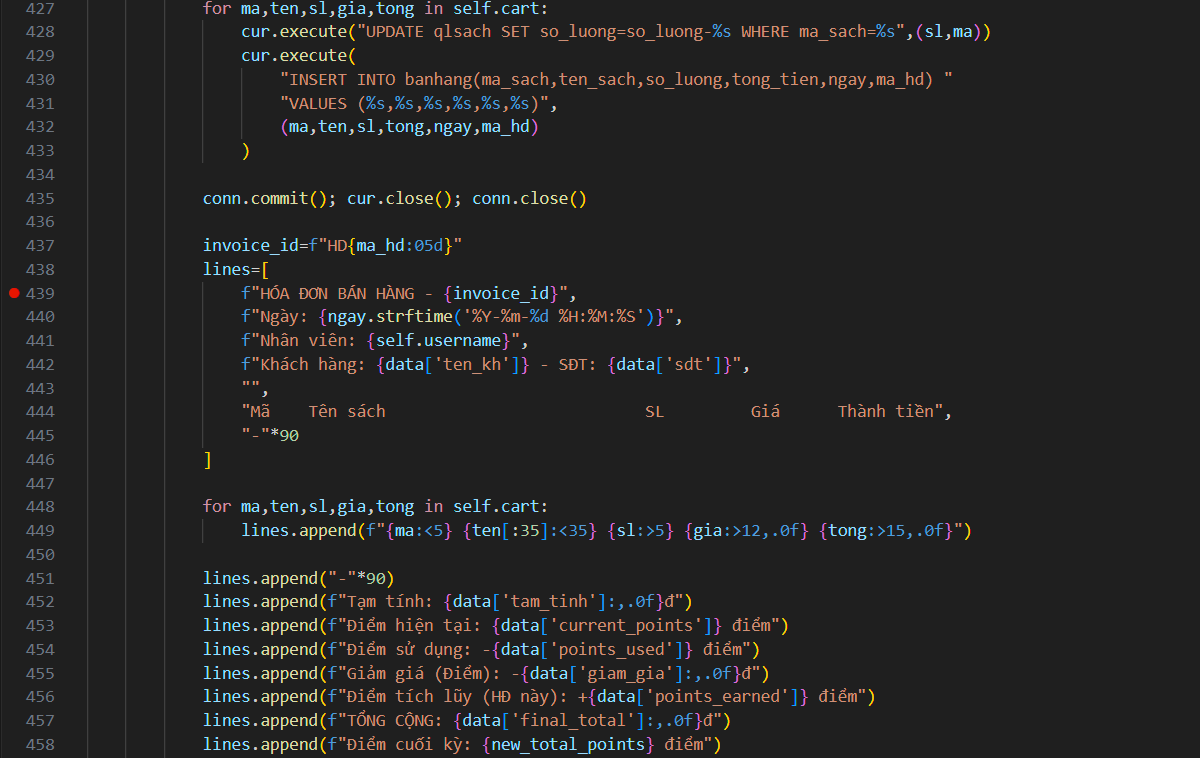
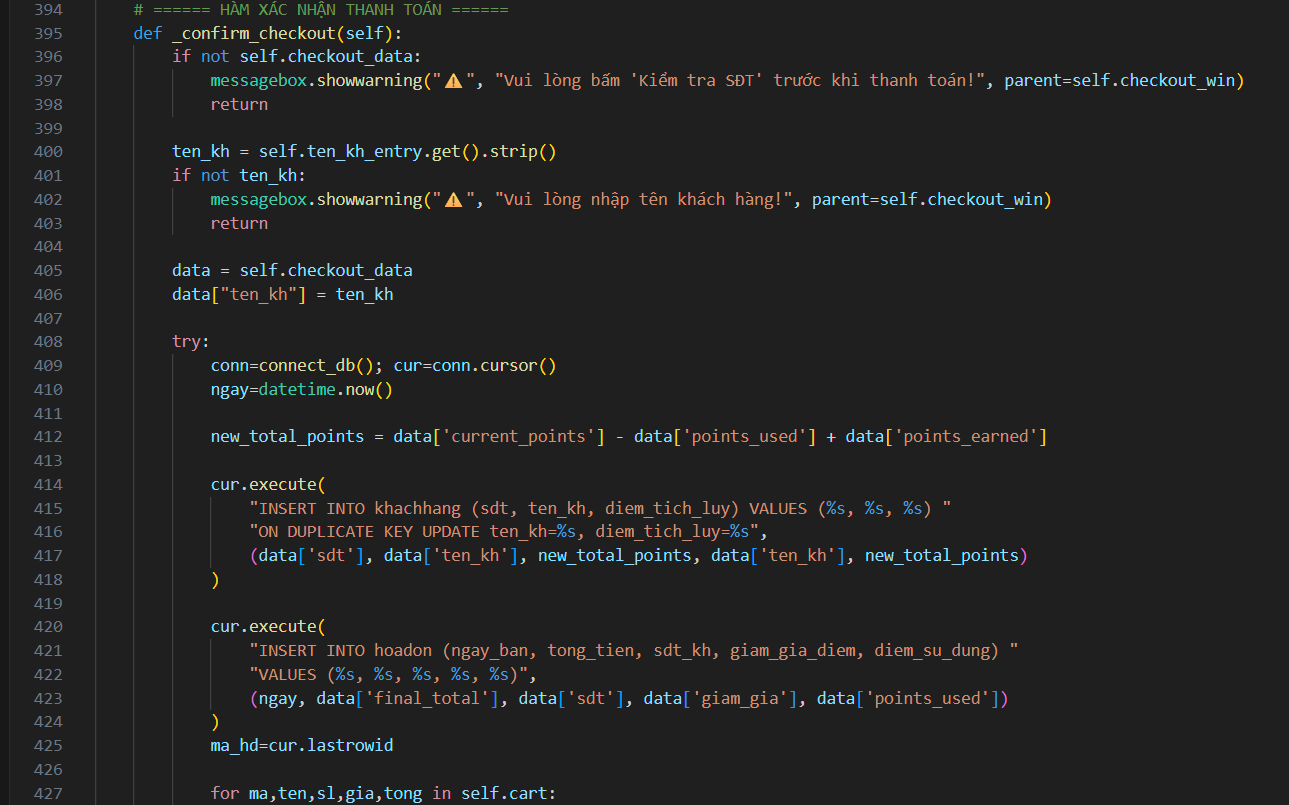
**3. Quản lý Giỏ hàng**

* **refresh\_cart(self)**:
  + Một hàm tiện ích đơn giản: Xóa tất cả các mục đang hiển thị trên bảng "Giỏ hàng" và vẽ lại từ đầu dựa trên danh sách self.cart mới nhất.
* **remove\_from\_cart(self)**:
  + Được gọi khi nhấn nút "Xoá khỏi giỏ".
  + Lấy mục đang được chọn trong bảng "Giỏ hàng".
  + Hiển thị hộp thoại xác nhận ("Bạn có chắc muốn xóa...?").
  + Nếu bạn đồng ý, nó sẽ xóa mặt hàng đó khỏi danh sách self.cart và gọi self.refresh\_cart() để cập nhật lại bảng.

**4. Quy trình Thanh toán (Checkout)**

Đây là phần phức tạp nhất, bao gồm nhiều hàm làm việc cùng nhau.

* **checkout(self) (Hàm chính - Dòng 337)**:
  + Được gọi khi nhấn nút "Thanh toán".
  + Kiểm tra xem giỏ hàng có trống không. Nếu trống, báo lỗi.
  + Nếu không trống, nó tạo một cửa sổ pop-up mới (Toplevel) tên là self.checkout\_win để xác nhận thanh toán.
  + Cửa sổ này chứa các ô nhập "Số điện thoại", "Tên khách hàng", và một nút "Kiểm tra SĐT" (gọi check\_customer\_points).
  + Nó cũng hiển thị các nhãn (Label) cho "Tạm tính", "Giảm giá (Điểm)", và "Thành tiền" (tổng cuối cùng).
  + Cuối cùng là 2 nút: "Xác nhận Thanh toán" (gọi self.confirm\_checkout - không thấy trong ảnh) và "Hủy bỏ".
* **check\_customer\_points(self) (Dòng 289)**:
  + Đây là "bộ não" của cửa sổ thanh toán, được gọi khi nhấn "Kiểm tra SĐT".
  1. **Lấy SĐT** và tính tam\_tinh (tạm tính) từ tổng giá trị giỏ hàng.
  2. **Truy vấn CSDL**: Dùng SĐT để SELECT ten\_kh, diem\_tich\_luy FROM khachhang.
  3. **Xử lý kết quả**:
     + Nếu **tìm thấy khách hàng**: Tự động điền "Tên khách hàng" và lấy current\_points (điểm tích lũy) của họ.
     + Nếu **không tìm thấy**: Đặt current\_points = 0 và hiển thị "khách hàng mới".
  4. **Tính toán**:
     + Gọi self.\_calculate\_discount(current\_points) để xem khách được giảm bao nhiêu % và dùng bao nhiêu điểm.
     + Tính giam\_gia (số tiền được giảm) và final\_total (số tiền cuối cùng phải trả).
     + Tính points\_earned (số điểm được cộng thêm từ đơn hàng này).
  5. **Cập nhật Giao diện**: Cập nhật các nhãn "Giảm giá" và "Thành tiền" với con số mới.
  6. **Lưu dữ liệu**: Lưu tất cả thông tin này (SĐT, tên, tạm tính, giảm giá, tổng tiền, điểm dùng, điểm nhận được) vào self.checkout\_data để chuẩn bị cho việc xác nhận và lưu hóa đơn.
* **\_calculate\_discount(self, points) (Dòng 275)**:
  + Đây là hàm logic kinh doanh, định nghĩa cách đổi điểm.
  + Nó hoạt động theo thang bậc:
    - > 50 điểm: Giảm 50%, dùng 50 điểm.
    - > 40 điểm: Giảm 40%, dùng 40 điểm.
    - ...
    - > 10 điểm: Giảm 10%, dùng 10 điểm.
    - else: Không giảm (0%).
  + Hàm này trả về 2 giá trị: discount\_percent (tỷ lệ giảm) và points\_used (số điểm đã dùng).



A black background with many colorful text

AI-generated content may be incorrect.

Đây là hàm được gọi khi nhấn nút **"Xác nhận Thanh toán"** trong cửa sổ pop-up. Đây là bước cuối cùng và quan trọng nhất, thực hiện 4 nhiệm vụ chính: Kiểm tra, cập nhật cơ sở dữ liệu (CSDL), xuất hóa đơn .txt, và dọn dẹp giỏ hàng.

**1. Kiểm tra và Chuẩn bị**

* **Kiểm tra checkout\_data**: Đảm bảo rằng bạn đã nhấn nút "Kiểm tra SĐT" trước đó. Nếu self.checkout\_data bị trống, nó sẽ báo lỗi và dừng lại.
* **Kiểm tra Tên khách hàng**: Đảm bảo ô "Tên khách hàng" không bị bỏ trống.
* **Lấy dữ liệu**: Nó lấy tất cả dữ liệu (tạm tính, giảm giá, điểm, SĐT...) đã được chuẩn bị bởi hàm check\_customer\_points và cập nhật tên khách hàng vào đó.

**2. Cập nhật Cơ sở dữ liệu**

Đây là phần quan trọng nhất, nó thực hiện một "giao dịch" (transaction) với CSDL:

* **Tính điểm mới**: Tính toán new\_total\_points (tổng điểm cuối cùng) của khách hàng.
* **Cập nhật khách hàng**: Chạy một lệnh SQL INSERT ... ON DUPLICATE KEY UPDATE. Lệnh này có nghĩa là:
  + Thử INSERT (thêm) khách hàng mới với SĐT này.
  + Nếu SĐT đã tồn tại (báo lỗi DUPLICATE KEY), thì thay vì báo lỗi, nó sẽ UPDATE (cập nhật) tên và diem\_tich\_luy (điểm tích lũy) cho khách hàng cũ đó.
* **Tạo hóa đơn**: INSERT một hàng mới vào bảng hoadon, lưu lại tổng tiền, SĐT khách, điểm đã dùng... và lấy ra ma\_hd (mã hóa đơn) vừa được tạo.
* **Cập nhật chi tiết**: Lặp qua từng món hàng trong self.cart:
  + **Trừ kho**: UPDATE qlsach để trừ số lượng sách đã bán (sl) ra khỏi tồn kho (so\_luong).
  + **Lưu chi tiết bán hàng**: INSERT INTO banhang để lưu chi tiết từng món hàng (mã sách, số lượng) vào đúng hóa đơn (ma\_hd) đó.
* **Chốt giao dịch**: conn.commit() được gọi. Lệnh này "chốt" tất cả các thay đổi ở trên vào CSDL. Nếu bất kỳ bước nào ở trên thất bại, giao dịch sẽ được hủy và CSDL không bị thay đổi gì.

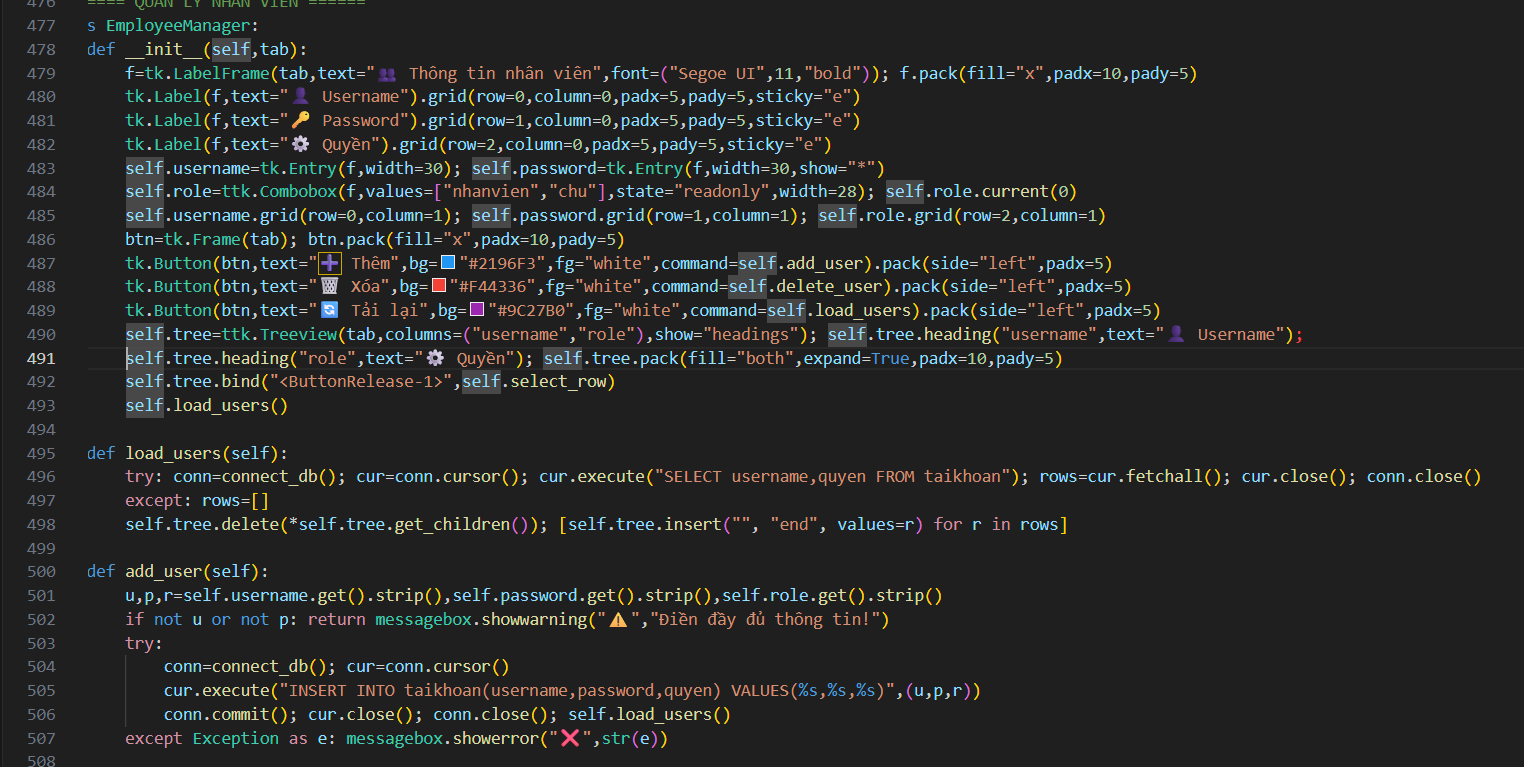
**3. Xuất Hóa Đơn .txt**

* **Tạo nội dung**: Tạo một mã hóa đơn đẹp hơn (ví dụ: HD00045). Sau đó, nó xây dựng nội dung của một tệp hóa đơn dưới dạng văn bản (lưu trong danh sách lines), bao gồm:
  + Thông tin (Ngày, Nhân viên, Khách hàng).
  + Danh sách các mặt hàng, số lượng, đơn giá, thành tiền.
  + Chi tiết thanh toán (Tạm tính, Điểm sử dụng, Giảm giá, Tổng cộng, Điểm tích lũy mới).
* **Lưu tệp**: Mở một hộp thoại "Save As..." (asksaveasfilename) để hỏi bạn muốn lưu tệp .txt này ở đâu.
* **Viết và Mở tệp**: Nếu bạn chọn lưu, nó sẽ ghi nội dung hóa đơn vào tệp. Sau đó, nó tự động dùng os.startfile (trên Windows) để mở tệp .txt đó cho bạn xem ngay lập tức.

**4. Dọn dẹp**

Sau khi mọi thứ thành công:

* Hiển thị thông báo "Thanh toán thành công!".
* **self.cart.clear()**: Xóa sạch giỏ hàng (trong bộ nhớ).
* **self.refresh\_cart()**: Cập nhật giao diện bảng giỏ hàng (cho nó trống rỗng).
* **self.load\_books()**: Tải lại danh sách sách (để bạn thấy số lượng tồn kho đã bị trừ).
* **self.checkout\_win.destroy()**: Đóng cửa sổ pop-up thanh toán.



Lớp này tạo ra giao diện và xử lý các chức năng cho **tab "Quản lý Nhân viên"**, cho phép người dùng (là "chủ") xem, thêm và xóa các tài khoản người dùng khác.

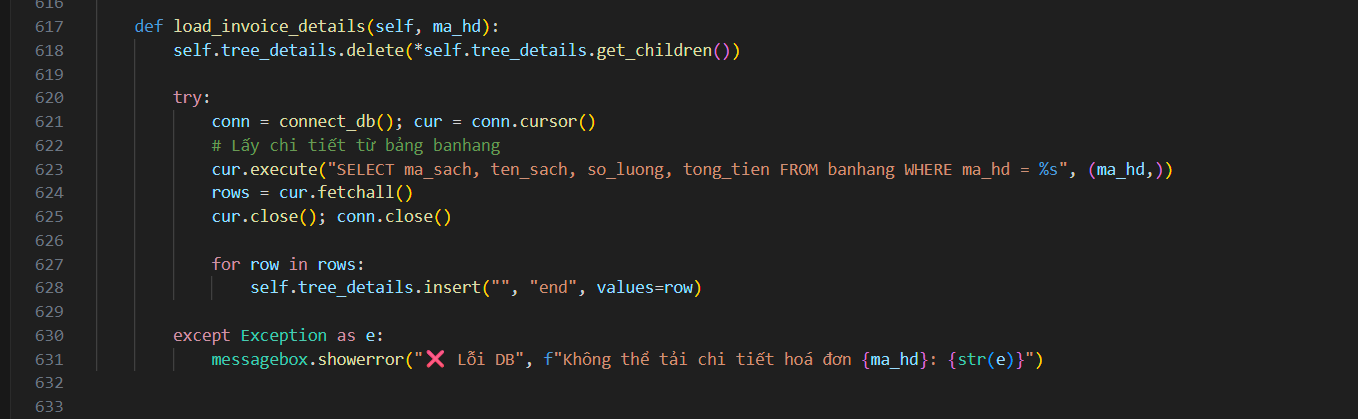
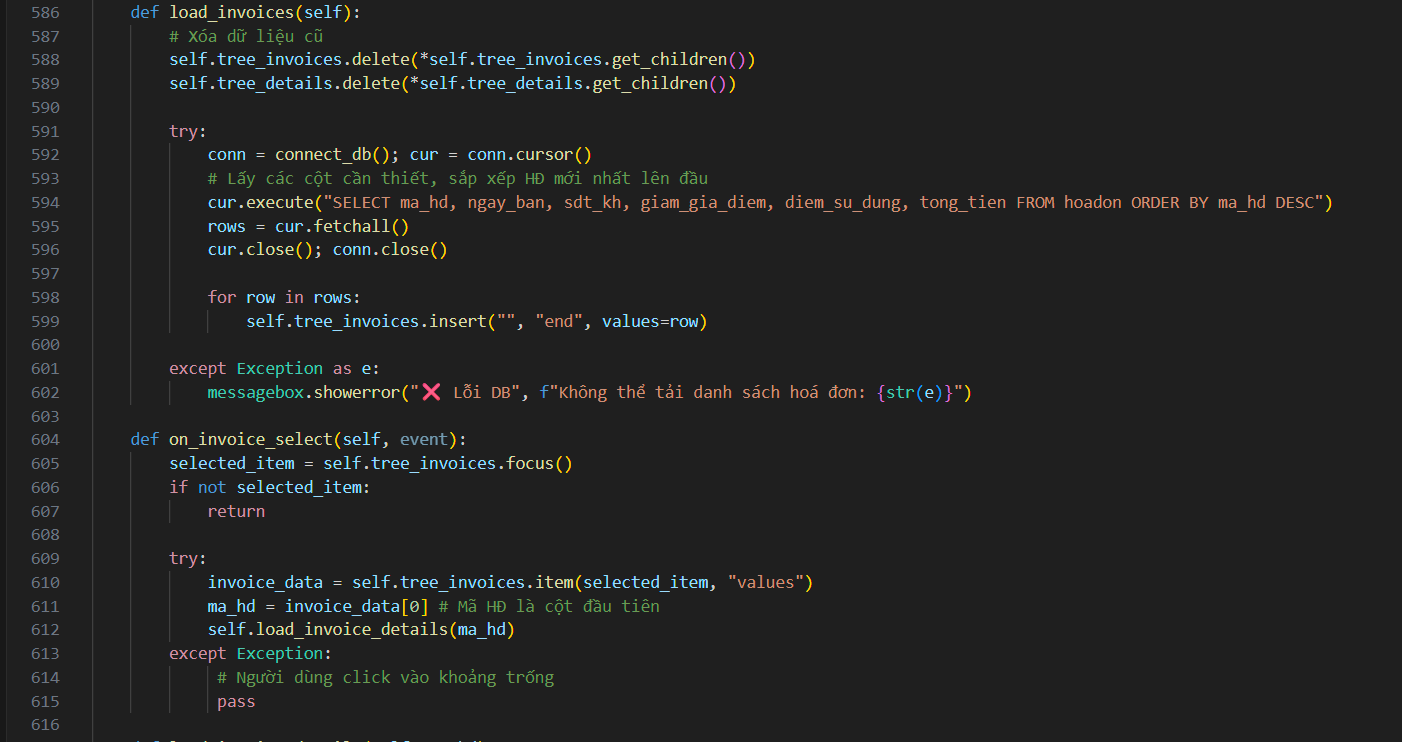
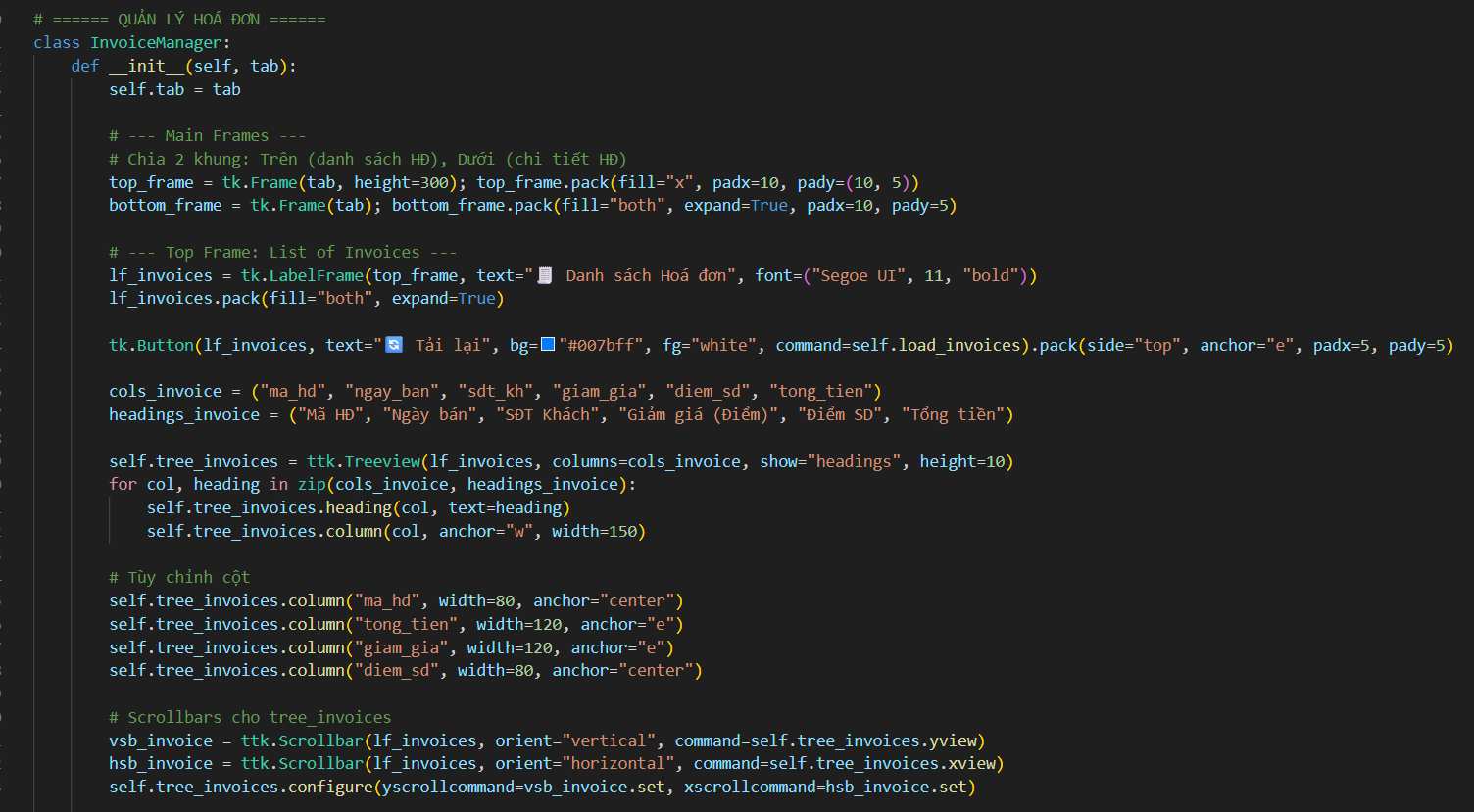
**1. Hàm \_\_init\_\_ (Thiết lập Giao diện)**

Hàm này "vẽ" ra toàn bộ giao diện cho tab:

* **Ô nhập liệu**: Tạo các nhãn và ô nhập cho:
  + Username (Tên đăng nhập)
  + Password (Mật khẩu)
  + Quyền (Quyền): Đây là một Combobox (hộp chọn) cho phép bạn chọn "nhanvien" (nhân viên) hoặc "chu" (chủ).
* **Các nút bấm**: Tạo 3 nút:
  + **"Thêm"**: Gọi hàm self.add\_user.
  + **"Xóa"**: Gọi hàm self.delete\_user.
  + **"Tải lại"**: Gọi hàm self.load\_users.
* **Bảng hiển thị (Treeview)**: Tạo một bảng để hiển thị danh sách Username và Quyền của tất cả người dùng.
* **Sự kiện**: Gắn một sự kiện (self.tree.bind...) để khi bạn **nhấn chuột vào một hàng trong bảng**, nó sẽ tự động gọi hàm self.select\_row.
* Cuối cùng, nó gọi self.load\_users() để tải danh sách nhân viên lên bảng ngay lập tức.

**2. Các hàm chức năng**

* **load\_users(self)**:
  + Kết nối CSDL, chạy lệnh SELECT username, quyen FROM taikhoan để lấy tên và quyền của *tất cả* người dùng.
  + Xóa sạch dữ liệu cũ trên bảng và hiển thị dữ liệu mới vừa lấy được.
* **add\_user(self)**:
  + Được gọi khi nhấn nút "Thêm".
  + Lấy thông tin username, password, và role (quyền) từ các ô nhập liệu.
  + Kiểm tra xem có bỏ trống ô nào không.
  + Chạy lệnh INSERT INTO taikhoan... để thêm người dùng mới này vào CSDL.
  + Tự động gọi self.load\_users() để làm mới lại bảng.
* **delete\_user(self)**:
  + Được gọi khi nhấn nút "Xóa".
  + Lấy username từ ô nhập liệu (ô này thường được điền tự động bởi hàm select\_row).
  + Hiển thị hộp thoại "Xác nhận" để hỏi bạn có chắc muốn xóa không.
  + Nếu bạn đồng ý, nó sẽ chạy lệnh DELETE FROM taikhoan WHERE username=... để xóa người dùng đó.
  + Tự động gọi self.load\_users() để làm mới lại bảng.
* **select\_row(self)**:
  + Đây là hàm tiện ích, được gọi khi bạn *nhấn vào một hàng trong bảng*.
  + Nó lấy username (d[0]) và role (d[1]) từ hàng bạn vừa nhấn.
  + Nó **điền ngược** thông tin này lên các ô Username và Quyền ở trên.
  + **Mục đích**: Giúp chọn một người dùng để Xóa rất nhanh mà không cần gõ lại tên.



Lớp này tạo ra giao diện cho **tab "Quản lý Hoá đơn"**. Điểm đặc biệt của tab này là nó được chia làm **hai phần** tương tác với nhau:

1. **Nửa trên (top\_frame)**: Hiển thị *danh sách của tất cả các hóa đơn* đã bán (self.tree\_invoices).
2. **Nửa dưới (bottom\_frame)**: Hiển thị *chi tiết các món hàng* (self.tree\_details) chỉ thuộc về *một* hóa đơn mà bạn chọn ở bảng trên.

Dưới đây là cách chúng hoạt động cùng nhau:

**1. Hàm \_\_init\_\_ (Thiết lập Giao diện)**

* Hàm này "vẽ" ra 2 khu vực (trên và dưới).
* **Khu vực trên**: Chứa bảng self.tree\_invoices (danh sách hóa đơn) và một nút "Tải lại" (gọi hàm self.load\_invoices).
* **Khu vực dưới**: Chứa bảng self.tree\_details (chi tiết món hàng).
* **Sự kiện quan trọng**: Nó gắn một sự kiện bind("<ButtonRelease-1>", ...) vào bảng self.tree\_invoices (bảng trên). Sự kiện này có nghĩa là: **"Khi người dùng nhấp chuột vào một hàng, hãy gọi hàm self.on\_invoice\_select"**.
* Cuối cùng, nó gọi self.load\_invoices() để tải dữ liệu lên bảng trên ngay khi mở tab.

**2. Hàm load\_invoices(self)**

* Hàm này được gọi khi mở tab hoặc khi nhấn nút "Tải lại".
* Nó xóa sạch dữ liệu cũ ở *cả hai bảng* (danh sách và chi tiết).
* Nó kết nối CSDL và chạy lệnh SELECT ... FROM hoadon ... để lấy **tất cả** các hóa đơn, xếp theo ID mới nhất trước.
* Nó đổ dữ liệu này vào self.tree\_invoices (bảng ở trên).

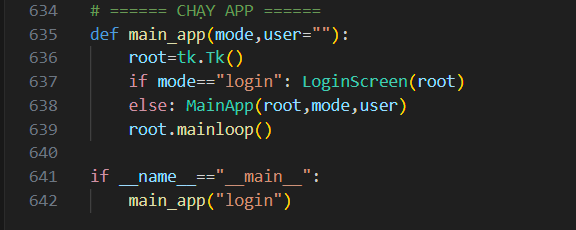
**3. Hàm on\_invoice\_select(self, event)**

* Đây là hàm được kích hoạt ngay khi bạn **nhấn vào một hóa đơn** ở bảng trên (do sự kiện bind trong \_\_init\_\_).
* Nó lấy thông tin của hàng bạn vừa chọn.
* Nó trích xuất ma\_hd (Mã hóa đơn) từ dữ liệu đó (là cột đầu tiên invoice\_data[0]).
* Sau đó, nó gọi hàm self.load\_invoice\_details(ma\_hd), truyền mã hóa đơn vừa lấy được vào.

**4. Hàm load\_invoice\_details(self, ma\_hd)**

* Hàm này nhận một ma\_hd (Mã hóa đơn) cụ thể từ hàm on\_invoice\_select.
* Nó xóa sạch bảng chi tiết (self.tree\_details - bảng ở dưới).
* Nó kết nối CSDL và chạy lệnh SELECT ... FROM banhang WHERE ma\_hd = %s để tìm **tất cả các món hàng** chỉ thuộc về hóa đơn đó.
* Cuối cùng, nó hiển thị các món hàng này lên self.tree\_details (bảng ở dưới).

**Tóm tắt quy trình:** Bạn mở tab -> load\_invoices tải danh sách (bảng trên) -> Bạn nhấn vào một hóa đơn -> on\_invoice\_select bắt được sự kiện -> load\_invoice\_details được gọi và tải chi tiết (bảng dưới).



Đây là "bộ điều khiển" chính để chạy toàn bộ ứng dụng của bạn.

1. **Hàm main\_app(mode, ...)**:
   * Hàm này quyết định hiển thị màn hình nào.
   * Nếu mode == "login", nó hiển thị LoginScreen (Màn hình Đăng nhập).
   * Nếu mode là khác (ví dụ: "chu" hoặc "nhanvien"), nó hiển thị MainApp (Màn hình Chính).
   * root.mainloop() là lệnh giữ cho cửa sổ ứng dụng luôn chạy.
2. **Khối if \_\_name\_\_ == "\_\_main\_\_":**:
   * Đây là điểm bắt đầu khi bạn chạy tệp Python này.
   * Nó gọi main\_app("login") để luôn hiển thị Màn hình Đăng nhập đầu tiên.