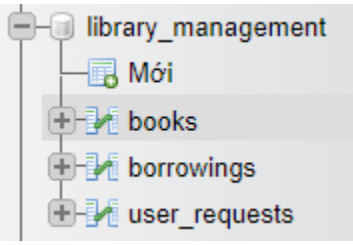
Họ tên: Ngô Long Vũ

Msv: 20210463

Yêu cầu: csdl



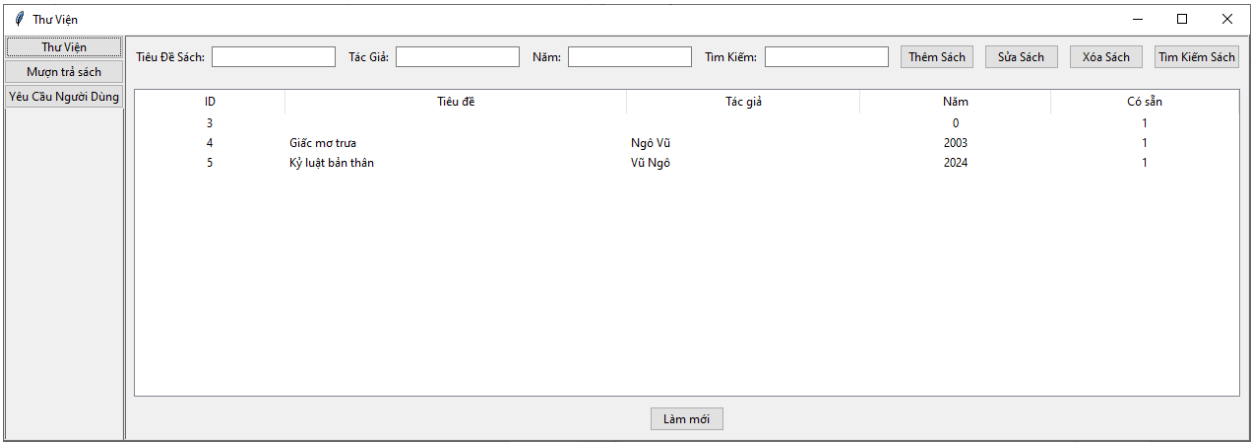
Đoạn mã tạo csdl

```
import mysql.connector

# Kết nối đến MySQL server
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password=""
)

# Tạo cursor
cursor = mydb.cursor()

# Tạo database
cursor.execute("CREATE DATABASE IF NOT EXISTS library_management")

# Sử dụng database vừa tạo
cursor.execute("USE library_management")

# Tạo bảng books
cursor.execute("""
CREATE TABLE IF NOT EXISTS books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    publication_year INT,
    available BOOLEAN DEFAULT TRUE
)
""")

# Tạo bảng borrowings
cursor.execute("""
CREATE TABLE IF NOT EXISTS borrowings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    book_id INT,
    user_name VARCHAR(255) NOT NULL,
    borrow_date DATE,
    return_date DATE,
    FOREIGN KEY (book_id) REFERENCES books(id)
)
""")
```

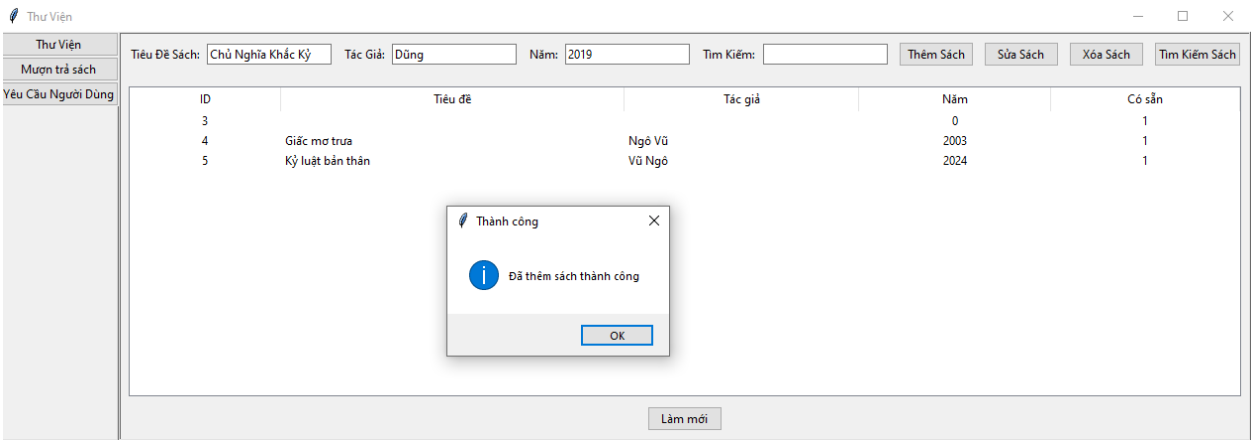
```
# Tạo bảng user_requests (cho yêu cầu số 6)
cursor.execute("""
CREATE TABLE IF NOT EXISTS user_requests (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_name VARCHAR(255) NOT NULL,
    request_content TEXT,
    request_date DATETIME DEFAULT CURRENT_TIMESTAMP
)
""")

# Commit các thay đổi
mydb.commit()

# Đóng kết nối
cursor.close()
mydb.close()

print("Database và các bảng đã được tạo thành công.")
```

---

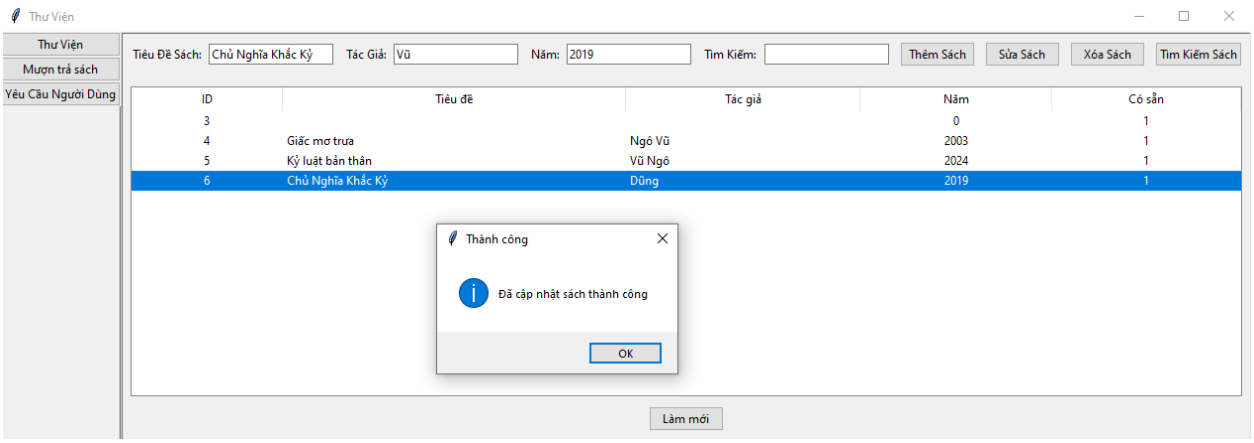
form tổng:



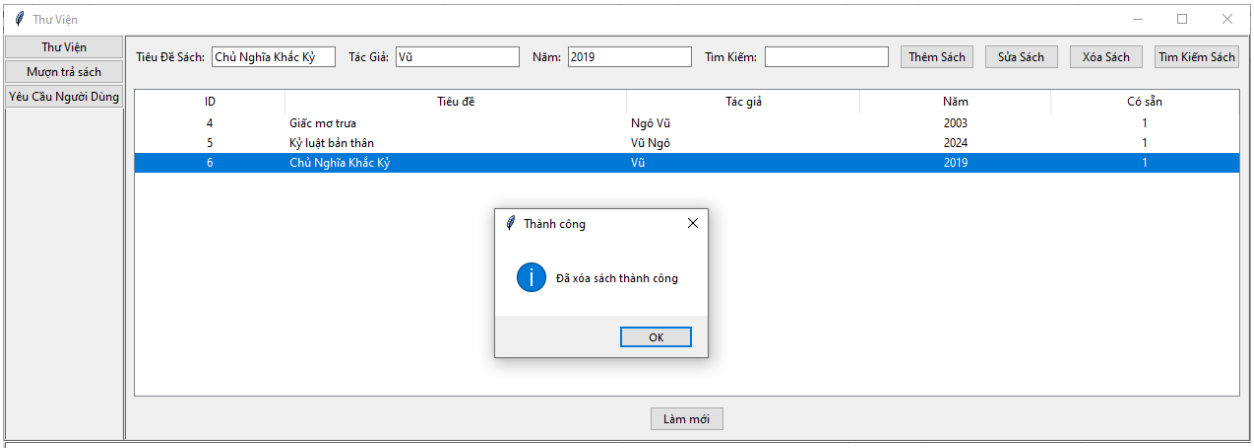
Thêm sách thành công:

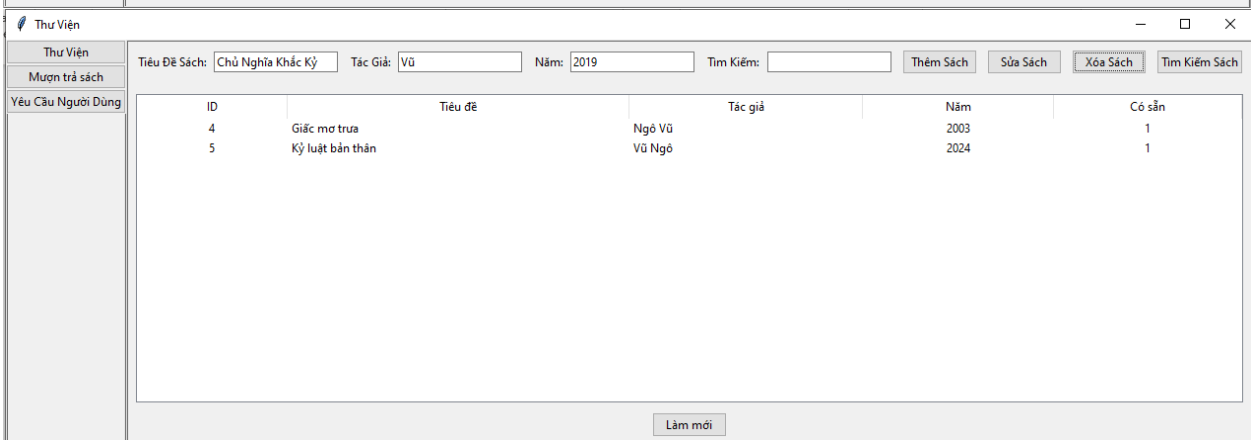


Sửa sách thành công:



Xóa sách thành công:





Đoạn mã:

```
import tkinter as tk
from tkinter import ttk, messagebox, simpledialog
import mysql.connector
from datetime import datetime

root = tk.Tk()
```

```
root.title("Thư Viện")
root.geometry("1080x720")


def connect_to_db():
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="library_management"
    )
    return connection


left_frame = ttk.Frame(root, width=480, height=720, relief=tk.RIDGE)
left_frame.pack(side="left", fill="y")

right_frame = ttk.Frame(root, width=600, height=720, relief=tk.RIDGE)
right_frame.pack(side="right", fill="both", expand=True)


def clear_right_frame():
    for widget in right_frame.winfo_children():
        widget.destroy()


def show_thu_vien():
    clear_right_frame()
    frame = ttk.Frame(right_frame)
    frame.pack(padx=10, pady=10, fill='x')

    ttk.Label(frame, text="Tiêu Đề Sách:").pack(side='left', padx=(0, 5))
    title_entry = ttk.Entry(frame)
    title_entry.pack(side='left', expand=True)

    ttk.Label(frame, text="Tác Giả:").pack(side='left', padx=(10, 5))
    author_entry = ttk.Entry(frame)
    author_entry.pack(side='left', expand=True)

    ttk.Label(frame, text="Năm:").pack(side='left', padx=(10, 5))
    year_entry = ttk.Entry(frame)
    year_entry.pack(side='left', expand=True)

    ttk.Label(frame, text="Tìm Kiếm:").pack(side='left', padx=(10, 5))
    search_entry = ttk.Entry(frame)
    search_entry.pack(side='left', expand=True)

    def add_book():
        title = title_entry.get()
        author = author_entry.get()
        year = year_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            cursor.execute("INSERT INTO books (title, author,
publication_year) VALUES (%s, %s, %s)",
                           (title, author, year))
```

```
            db.commit()
            messagebox.showinfo("Thành công", "Đã thêm sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_table()

    def update_book():
        selected = tree.selection()
        if not selected:
            messagebox.showwarning("Cảnh báo", "Vui lòng chọn một cuốn sách
để cập nhật")
            return
        book_id = tree.item(selected[0])['values'][0]
        title = title_entry.get()
        author = author_entry.get()
        year = year_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            cursor.execute("UPDATE books SET title=%s, author=%s,
publication_year=%s WHERE id=%s",
                           (title, author, year, book_id))
            db.commit()
            messagebox.showinfo("Thành công", "Đã cập nhật sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_table()

    def delete_book():
        selected = tree.selection()
        if not selected:
            messagebox.showwarning("Cảnh báo", "Vui lòng chọn một cuốn sách
để xóa")
            return
        book_id = tree.item(selected[0])['values'][0]
        if messagebox.askyesno("Xác nhận", "Bạn có chắc chắn muốn xóa cuốn
sách này?"):
            db = connect_to_db()
            cursor = db.cursor()
            try:
                cursor.execute("DELETE FROM books WHERE id=%s", (book_id,))
                db.commit()
                messagebox.showinfo("Thành công", "Đã xóa sách thành công")
            except mysql.connector.Error as err:
                messagebox.showerror("Lỗi", str(err))
            finally:
                cursor.close()
                db.close()
            refresh_table()

    def search_books():
```

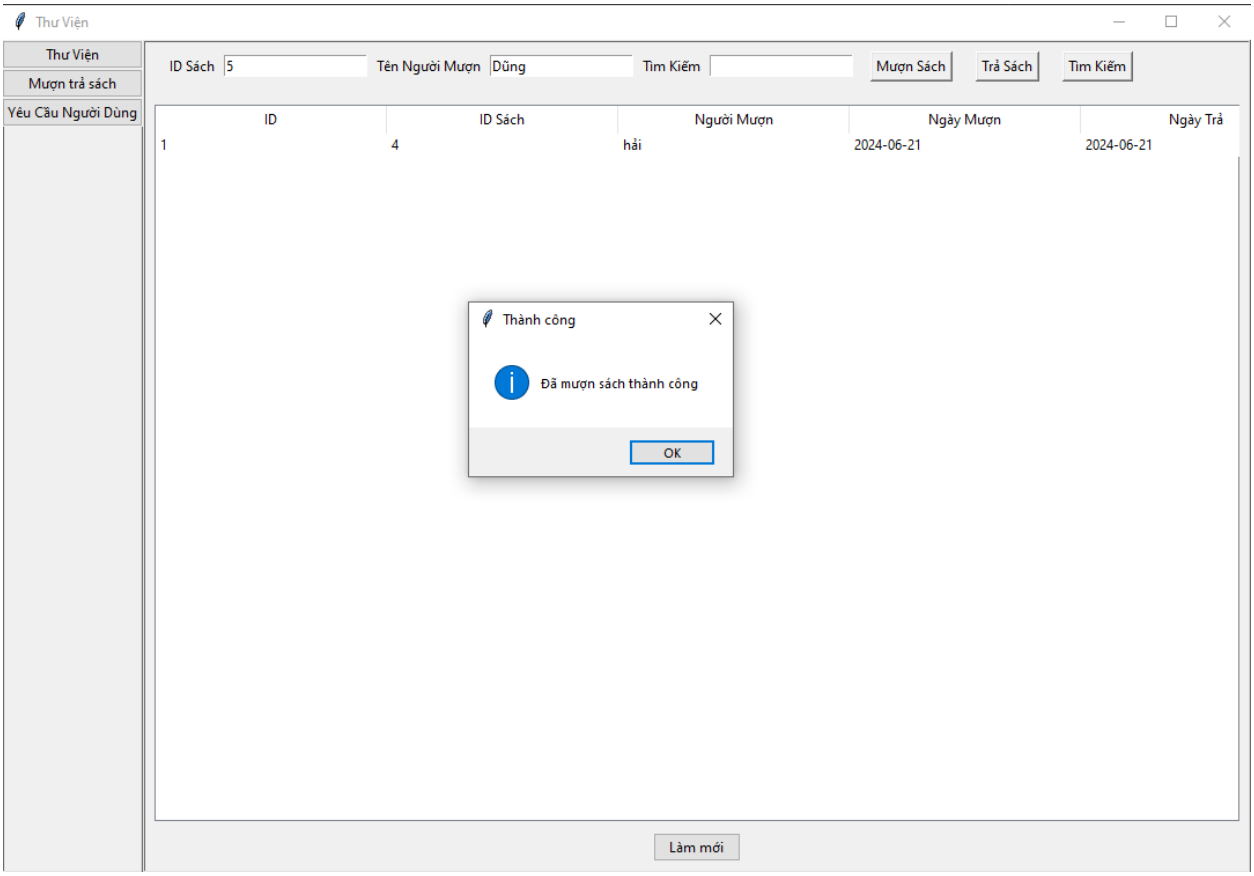
```
        search_query = search_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            cursor.execute("SELECT * FROM books WHERE title LIKE %s OR author
LIKE %s",
                           ('%' + search_query + '%', '%' + search_query +
'%'))
            results = cursor.fetchall()
            for i in tree.get_children():
                tree.delete(i)
            for book in results:
                tree.insert('', 'end', values=book)
        finally:
            cursor.close()
            db.close()

    add_button = ttk.Button(frame, text="Thêm Sách", command=add_book)
    add_button.pack(side='left', padx=(10, 0))

    sua_button = ttk.Button(frame, text="Sửa Sách", command=update_book)
    sua_button.pack(side='left', padx=(10, 0))

    xoa_button = ttk.Button(frame, text="Xóa Sách", command=delete_book)
    xoa_button.pack(side='left', padx=(10, 0))

    tim_kiem_button = ttk.Button(frame, text="Tìm Kiếm Sách",
command=search_books)
    tim_kiem_button.pack(side='left', padx=(10, 0))

    tree_frame = ttk.Frame(right_frame)
    tree_frame.pack(fill='both', expand=True, padx=10, pady=10)

    tree = ttk.Treeview(tree_frame, columns=("ID", "Title", "Author", "Year",
"Available"), show="headings")
    tree.heading("ID", text="ID")
    tree.heading("Title", text="Tiêu đề")
    tree.heading("Author", text="Tác giả")
    tree.heading("Year", text="Năm")
    tree.heading("Available", text="Có sẵn")
    tree.column("ID", width=50, anchor="center")
    tree.column("Title", width=250)
    tree.column("Author", width=150)
    tree.column("Year", width=100, anchor="center")
    tree.column("Available", width=100, anchor="center")
    tree.pack(fill='both', expand=True)

    def fetch_books():
        db = connect_to_db()
        cursor = db.cursor()
        cursor.execute("SELECT id, title, author, publication_year, available
FROM books")
        rows = cursor.fetchall()
        cursor.close()
        db.close()
        return rows
```

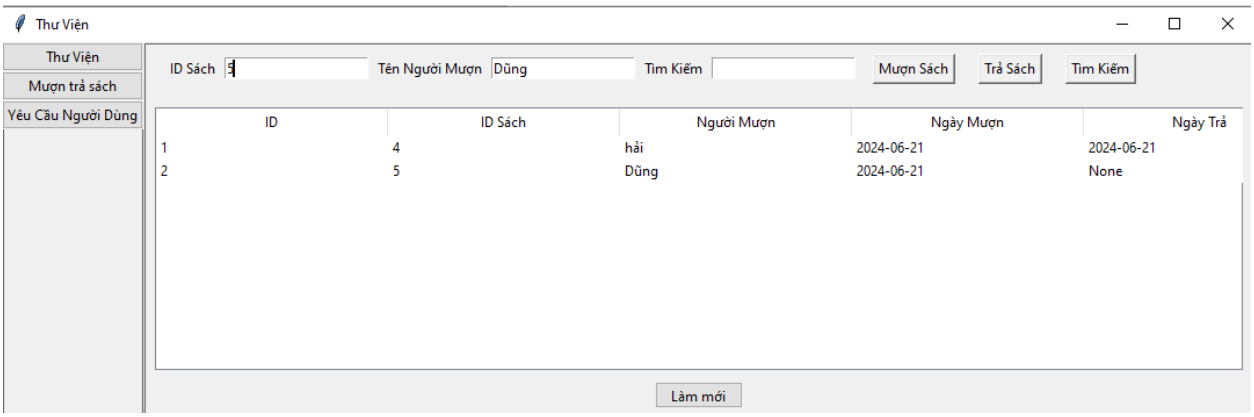
```
    def refresh_table():
        for i in tree.get_children():
            tree.delete(i)
        books = fetch_books()
        for book in books:
            tree.insert('', 'end', values=book)

    refresh_button = ttk.Button(right_frame, text="Làm mới",
command=refresh_table)
    refresh_button.pack(pady=(0, 10))

    refresh_table()

ttk.Button(left_frame, text="Thư Viện", command=show_thu_vien).pack(fill='x')
ttk.Button(left_frame, text="Mượn trả sách",
command=show_muon_tra).pack(fill='x')
ttk.Button(left_frame, text="Yêu Cầu Người Dùng",
command=yeu_cau_nguoi_dung).pack(fill='x')
```

---

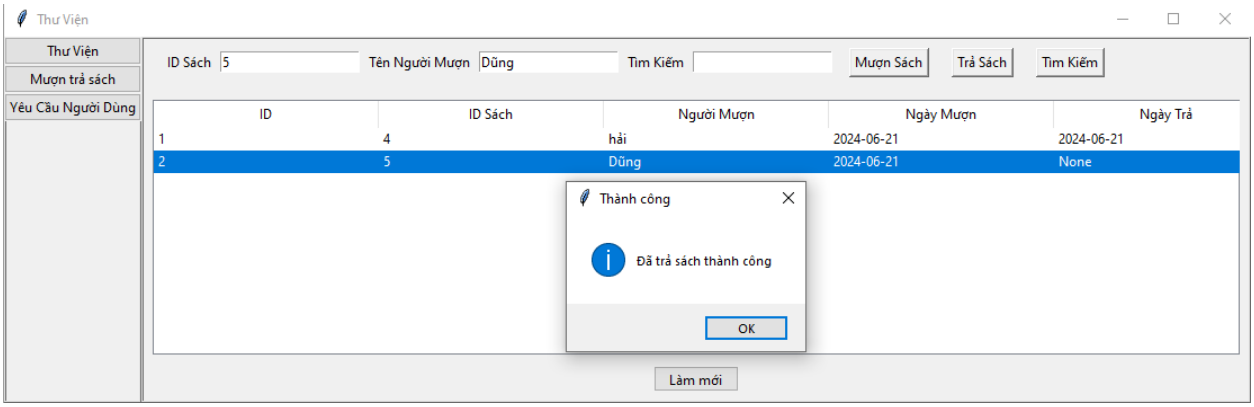
form mượn trả:

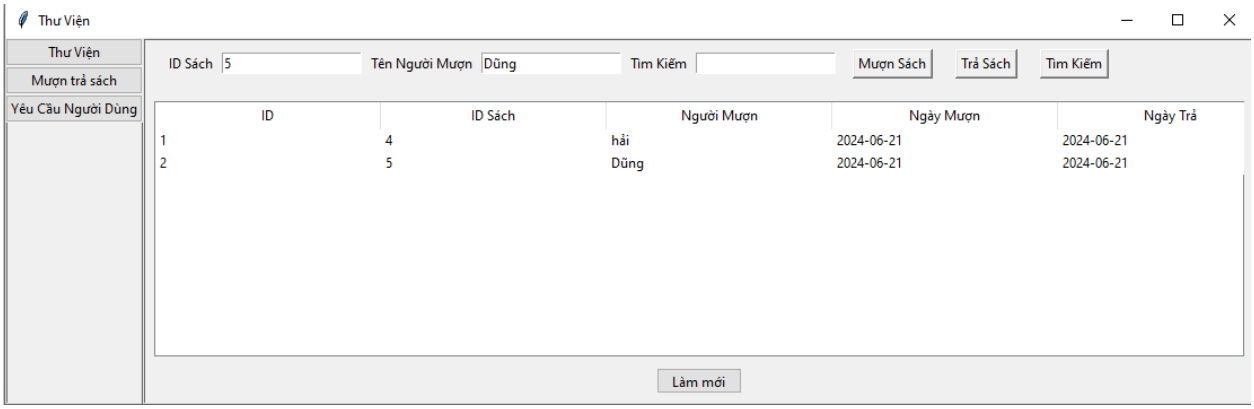
mượn thành công:



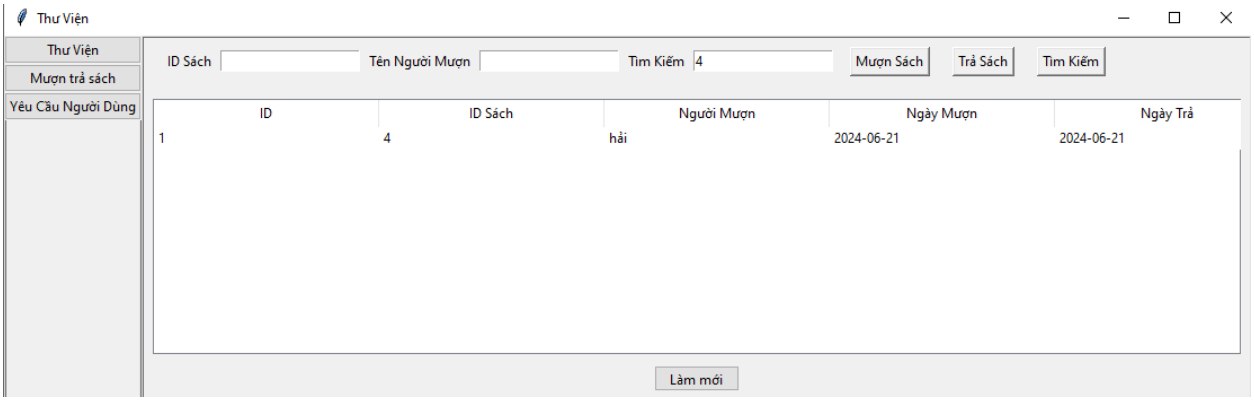


Trả thành công:





Tìm kiếm:



Đoạn mã:

```
def show_muon_tra():
    clear_right_frame()
    frame_borrow = tk.Frame(right_frame)
    frame_borrow.pack(fill=tk.X, padx=20, pady=10)

    tk.Label(frame_borrow, text="ID Sách").pack(side=tk.LEFT)
    book_id_entry = tk.Entry(frame_borrow)
    book_id_entry.pack(side=tk.LEFT, padx=5)

    tk.Label(frame_borrow, text="Tên Người Mượn").pack(side=tk.LEFT)
    user_name_entry = tk.Entry(frame_borrow)
    user_name_entry.pack(side=tk.LEFT, padx=5)

    tk.Label(frame_borrow, text="Tìm Kiếm").pack(side=tk.LEFT)
    search_muon_entry = tk.Entry(frame_borrow)
    search_muon_entry.pack(side=tk.LEFT, padx=5)

    def borrow_book():
        book_id = book_id_entry.get()
        user_name = user_name_entry.get()
        borrow_date = datetime.now().date()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            # Kiểm tra xem sách có sẵn không
            cursor.execute("SELECT available FROM books WHERE id = %s",
(book_id,))
            result = cursor.fetchone()
            if not result or not result[0]:
                messagebox.showerror("Lỗi", "Sách không có sẵn hoặc không tồn
tại")
                return

            # Cập nhật trạng thái sách
            cursor.execute("UPDATE books SET available = FALSE WHERE id =
%s", (book_id,))

            # Thêm vào bảng mượn
            cursor.execute(
```

```
                "INSERT INTO borrowings (book_id, user_name, borrow_date)
VALUES (%s, %s, %s)",
                (book_id, user_name, borrow_date))
            db.commit()
            messagebox.showinfo("Thành công", "Đã mượn sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_borrow_table()

    def return_book():
        selected = borrow_tree.selection()
        if not selected:
            messagebox.showwarning("Cảnh báo", "Vui lòng chọn một mục để trả
sách")
            return
        borrow_id = borrow_tree.item(selected[0])['values'][0]
        book_id = borrow_tree.item(selected[0])['values'][1]
        return_date = datetime.now().date()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            # Cập nhật ngày trả trong bảng mượn
            cursor.execute("UPDATE borrowings SET return_date = %s WHERE id =
%s", (return_date, borrow_id))

            # Cập nhật trạng thái sách
            cursor.execute("UPDATE books SET available = TRUE WHERE id = %s",
(book_id,))

            db.commit()
            messagebox.showinfo("Thành công", "Đã trả sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_borrow_table()

    def search_muon_tra():
        search_query = search_muon_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            # Tìm kiếm trong bảng borrowings dựa trên ID sách, tên người mượn
hoặc ngày mượn
            cursor.execute("""
                SELECT * FROM borrowings
                WHERE book_id LIKE %s
                OR user_name LIKE %s
                OR DATE(borrow_date) = %s
            """, ('%' + search_query + '%', '%' + search_query + '%',
search_query))
            results = cursor.fetchall()
```

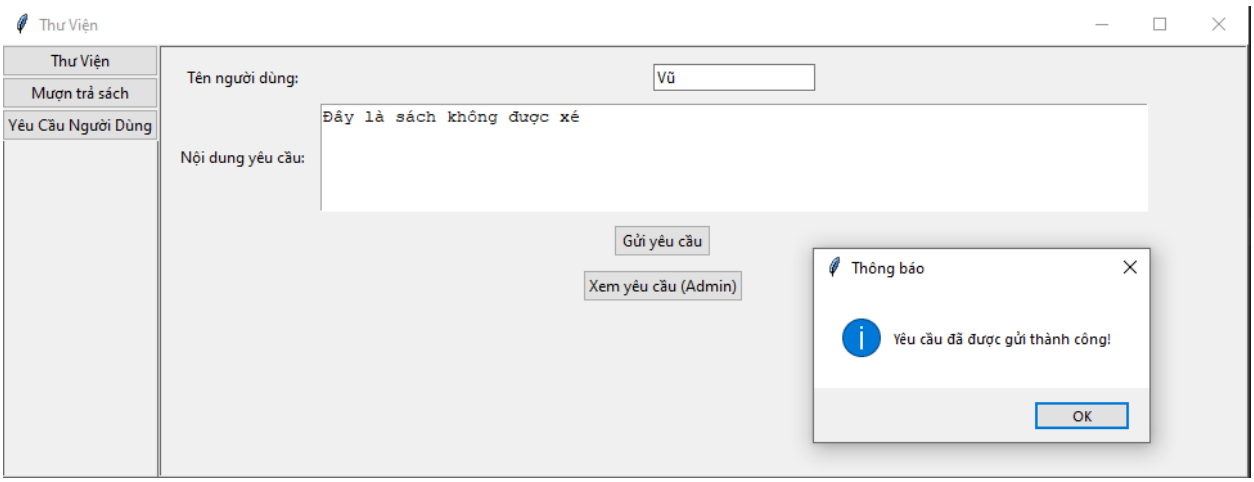
```
            # Xóa dữ liệu cũ trong bảng
            for i in borrow_tree.get_children():
                borrow_tree.delete(i)

            # Hiển thị kết quả tìm kiếm
            for borrowing in results:
                borrow_tree.insert('', 'end', values=borrowing)

            if not results:
                messagebox.showinfo("Kết quả", "Không tìm thấy kết quả phù
hợp.")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()


    borrow_button = tk.Button(frame_borrow, text="Mượn Sách",
command=borrow_book)
    borrow_button.pack(side=tk.LEFT, padx=10)

    return_button = tk.Button(frame_borrow, text="Trả Sách",
command=return_book)
    return_button.pack(side=tk.LEFT, padx=10)

    search_button = tk.Button(frame_borrow, text="Tìm Kiếm",
command=search_muon_tra)
    search_button.pack(side=tk.LEFT, padx=10)

    borrow_tree_frame = ttk.Frame(right_frame)
    borrow_tree_frame.pack(fill='both', expand=True, padx=10, pady=10)

    borrow_tree = ttk.Treeview(borrow_tree_frame, columns=("ID", "Book ID",
"User Name", "Borrow Date", "Return Date"),
                               show="headings")
    borrow_tree.heading("ID", text="ID")
    borrow_tree.heading("Book ID", text="ID Sách")
    borrow_tree.heading("User Name", text="Người Mượn")
    borrow_tree.heading("Borrow Date", text="Ngày Mượn")
    borrow_tree.heading("Return Date", text="Ngày Trả")
    borrow_tree.pack(fill='both', expand=True)

    def fetch_borrowings():
        db = connect_to_db()
        cursor = db.cursor()
        cursor.execute("SELECT * FROM borrowings")
        rows = cursor.fetchall()
        cursor.close()
        db.close()
        return rows

    def refresh_borrow_table():
        for i in borrow_tree.get_children():
            borrow_tree.delete(i)
        borrowings = fetch_borrowings()
        for borrowing in borrowings:
```

```
            borrow_tree.insert('', 'end', values=borrowing)

    refresh_borrow_button = ttk.Button(right_frame, text="Làm mới",
command=refresh_borrow_table)
    refresh_borrow_button.pack(pady=(0, 10))

    refresh_borrow_table()
```

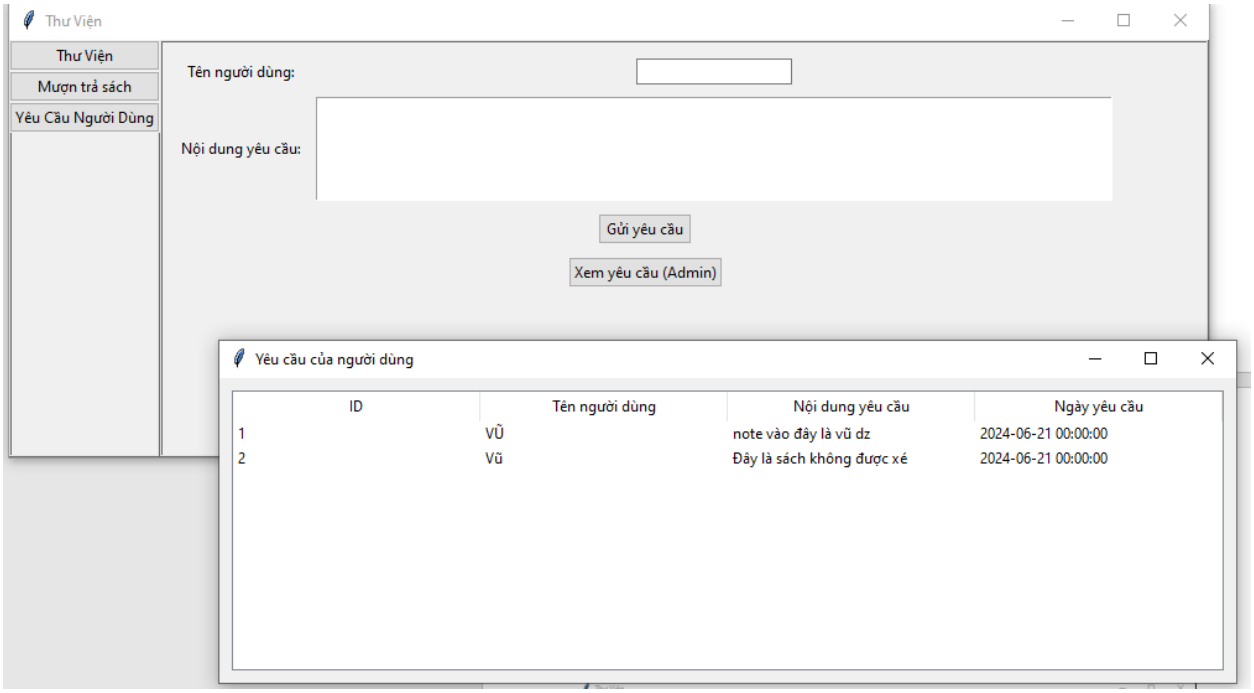
---

form yêu cầu người dùng:

yêu cầu:



Xem yêu cầu:



đoạn mã:

```
def yeu_cau_nguoi_dung():
    clear_right_frame()

    request_frame = ttk.Frame(right_frame)
    request_frame.pack(padx=10, pady=10, fill='both', expand=True)

    request_name_label = ttk.Label(request_frame, text="Tên người dùng:")
    request_name_label.grid(row=0, column=0, padx=5, pady=5)
    request_name_entry = ttk.Entry(request_frame)
    request_name_entry.grid(row=0, column=1, padx=5, pady=5)

    request_text_label = ttk.Label(request_frame, text="Nội dung yêu cầu:")
    request_text_label.grid(row=1, column=0, padx=5, pady=5)
    request_text_entry = tk.Text(request_frame, height=5)
    request_text_entry.grid(row=1, column=1, padx=5, pady=5)

    submit_request_button = ttk.Button(request_frame, text="Gửi yêu cầu",
                                       command=lambda:
submit_request(request_name_entry, request_text_entry))
    submit_request_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

    view_requests_button = ttk.Button(request_frame, text="Xem yêu cầu
(Admin)", command=view_requests)
    view_requests_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5)


def submit_request(name_entry, text_entry):
    user_name = name_entry.get()
    request_text = text_entry.get("1.0", tk.END).strip()
    if user_name and request_text:
        try:
            conn = connect_to_db()
            cursor = conn.cursor()
            cursor.execute(
                "INSERT INTO user_requests (user_name, request_content,
request_date) VALUES (%s, %s, %s)",
                (user_name, request_text, datetime.now().date()))
            conn.commit()
            messagebox.showinfo("Thông báo", "Yêu cầu đã được gửi thành
công!")
            name_entry.delete(0, tk.END)
            text_entry.delete("1.0", tk.END)
        except Exception as e:
            messagebox.showerror("Lỗi", f"Đã xảy ra lỗi: {e}")
        finally:
            conn.close()
    else:
        messagebox.showerror("Lỗi", "Vui lòng điền đầy đủ thông tin yêu
cầu!")


def view_requests():
    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM user_requests")
```

```
        requests = cursor.fetchall()
    except Exception as e:
        messagebox.showerror("Lỗi", f"Đã xảy ra lỗi: {e}")
    finally:
        conn.close()

    request_window = tk.Toplevel(root)
    request_window.title("Yêu cầu của người dùng")
    request_tree = ttk.Treeview(request_window,
                                columns=("ID", "Tên người dùng", "Nội dung
yêu cầu", "Ngày yêu cầu"),
                                show="headings")
    request_tree.heading("ID", text="ID")
    request_tree.heading("Tên người dùng", text="Tên người dùng")
    request_tree.heading("Nội dung yêu cầu", text="Nội dung yêu cầu")
    request_tree.heading("Ngày yêu cầu", text="Ngày yêu cầu")
    for request in requests:
        request_tree.insert("", "end", values=request)
    request_tree.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
```

Tất cả đoạn mã:

```
import tkinter as tk
from tkinter import ttk, messagebox, simpledialog
import mysql.connector
from datetime import datetime

root = tk.Tk()
root.title("Thư Viện")
root.geometry("1080x720")


def connect_to_db():
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="library_management"
    )
    return connection


left_frame = ttk.Frame(root, width=480, height=720, relief=tk.RIDGE)
left_frame.pack(side="left", fill="y")

right_frame = ttk.Frame(root, width=600, height=720, relief=tk.RIDGE)
right_frame.pack(side="right", fill="both", expand=True)


def clear_right_frame():
    for widget in right_frame.winfo_children():
        widget.destroy()


def show_thu_vien():
```

```
    clear_right_frame()
    frame = ttk.Frame(right_frame)
    frame.pack(padx=10, pady=10, fill='x')

    ttk.Label(frame, text="Tiêu Đề Sách:").pack(side='left', padx=(0, 5))
    title_entry = ttk.Entry(frame)
    title_entry.pack(side='left', expand=True)

    ttk.Label(frame, text="Tác Giả:").pack(side='left', padx=(10, 5))
    author_entry = ttk.Entry(frame)
    author_entry.pack(side='left', expand=True)

    ttk.Label(frame, text="Năm:").pack(side='left', padx=(10, 5))
    year_entry = ttk.Entry(frame)
    year_entry.pack(side='left', expand=True)

    ttk.Label(frame, text="Tìm Kiếm:").pack(side='left', padx=(10, 5))
    search_entry = ttk.Entry(frame)
    search_entry.pack(side='left', expand=True)

    def add_book():
        title = title_entry.get()
        author = author_entry.get()
        year = year_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            cursor.execute("INSERT INTO books (title, author,
publication_year) VALUES (%s, %s, %s)",
                           (title, author, year))
            db.commit()
            messagebox.showinfo("Thành công", "Đã thêm sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_table()

    def update_book():
        selected = tree.selection()
        if not selected:
            messagebox.showwarning("Cảnh báo", "Vui lòng chọn một cuốn sách
để cập nhật")
            return
        book_id = tree.item(selected[0])['values'][0]
        title = title_entry.get()
        author = author_entry.get()
        year = year_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            cursor.execute("UPDATE books SET title=%s, author=%s,
publication_year=%s WHERE id=%s",
                           (title, author, year, book_id))
            db.commit()
            messagebox.showinfo("Thành công", "Đã cập nhật sách thành công")
```

```
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_table()

    def delete_book():
        selected = tree.selection()
        if not selected:
            messagebox.showwarning("Cảnh báo", "Vui lòng chọn một cuốn sách
để xóa")
            return
        book_id = tree.item(selected[0])['values'][0]
        if messagebox.askyesno("Xác nhận", "Bạn có chắc chắn muốn xóa cuốn
sách này?"):
            db = connect_to_db()
            cursor = db.cursor()
            try:
                cursor.execute("DELETE FROM books WHERE id=%s", (book_id,))
                db.commit()
                messagebox.showinfo("Thành công", "Đã xóa sách thành công")
            except mysql.connector.Error as err:
                messagebox.showerror("Lỗi", str(err))
            finally:
                cursor.close()
                db.close()
            refresh_table()

    def search_books():
        search_query = search_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            cursor.execute("SELECT * FROM books WHERE title LIKE %s OR author
LIKE %s",
                           ('%' + search_query + '%', '%' + search_query +
'%'))
            results = cursor.fetchall()
            for i in tree.get_children():
                tree.delete(i)
            for book in results:
                tree.insert('', 'end', values=book)
        finally:
            cursor.close()
            db.close()

    add_button = ttk.Button(frame, text="Thêm Sách", command=add_book)
    add_button.pack(side='left', padx=(10, 0))

    sua_button = ttk.Button(frame, text="Sửa Sách", command=update_book)
    sua_button.pack(side='left', padx=(10, 0))

    xoa_button = ttk.Button(frame, text="Xóa Sách", command=delete_book)
    xoa_button.pack(side='left', padx=(10, 0))

    tim_kiem_button = ttk.Button(frame, text="Tìm Kiếm Sách",
```

```
command=search_books)
    tim_kiem_button.pack(side='left', padx=(10, 0))

    tree_frame = ttk.Frame(right_frame)
    tree_frame.pack(fill='both', expand=True, padx=10, pady=10)

    tree = ttk.Treeview(tree_frame, columns=("ID", "Title", "Author", "Year",
"Available"), show="headings")
    tree.heading("ID", text="ID")
    tree.heading("Title", text="Tiêu đề")
    tree.heading("Author", text="Tác giả")
    tree.heading("Year", text="Năm")
    tree.heading("Available", text="Có sẵn")
    tree.column("ID", width=50, anchor="center")
    tree.column("Title", width=250)
    tree.column("Author", width=150)
    tree.column("Year", width=100, anchor="center")
    tree.column("Available", width=100, anchor="center")
    tree.pack(fill='both', expand=True)

    def fetch_books():
        db = connect_to_db()
        cursor = db.cursor()
        cursor.execute("SELECT id, title, author, publication_year, available
FROM books")
        rows = cursor.fetchall()
        cursor.close()
        db.close()
        return rows

    def refresh_table():
        for i in tree.get_children():
            tree.delete(i)
        books = fetch_books()
        for book in books:
            tree.insert('', 'end', values=book)

    refresh_button = ttk.Button(right_frame, text="Làm mới",
command=refresh_table)
    refresh_button.pack(pady=(0, 10))

    refresh_table()


def show_muon_tra():
    clear_right_frame()
    frame_borrow = tk.Frame(right_frame)
    frame_borrow.pack(fill=tk.X, padx=20, pady=10)

    tk.Label(frame_borrow, text="ID Sách").pack(side=tk.LEFT)
    book_id_entry = tk.Entry(frame_borrow)
    book_id_entry.pack(side=tk.LEFT, padx=5)

    tk.Label(frame_borrow, text="Tên Người Mượn").pack(side=tk.LEFT)
    user_name_entry = tk.Entry(frame_borrow)
    user_name_entry.pack(side=tk.LEFT, padx=5)
```

```
    tk.Label(frame_borrow, text="Tìm Kiếm").pack(side=tk.LEFT)
    search_muon_entry = tk.Entry(frame_borrow)
    search_muon_entry.pack(side=tk.LEFT, padx=5)

    def borrow_book():
        book_id = book_id_entry.get()
        user_name = user_name_entry.get()
        borrow_date = datetime.now().date()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            # Kiểm tra xem sách có sẵn không
            cursor.execute("SELECT available FROM books WHERE id = %s",
(book_id,))
            result = cursor.fetchone()
            if not result or not result[0]:
                messagebox.showerror("Lỗi", "Sách không có sẵn hoặc không tồn
tại")
                return

            # Cập nhật trạng thái sách
            cursor.execute("UPDATE books SET available = FALSE WHERE id =
%s", (book_id,))

            # Thêm vào bảng mượn
            cursor.execute(
                "INSERT INTO borrowings (book_id, user_name, borrow_date)
VALUES (%s, %s, %s)",
                (book_id, user_name, borrow_date))
            db.commit()
            messagebox.showinfo("Thành công", "Đã mượn sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_borrow_table()

    def return_book():
        selected = borrow_tree.selection()
        if not selected:
            messagebox.showwarning("Cảnh báo", "Vui lòng chọn một mục để trả
sách")
            return
        borrow_id = borrow_tree.item(selected[0])['values'][0]
        book_id = borrow_tree.item(selected[0])['values'][1]
        return_date = datetime.now().date()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            # Cập nhật ngày trả trong bảng mượn
            cursor.execute("UPDATE borrowings SET return_date = %s WHERE id =
%s", (return_date, borrow_id))

            # Cập nhật trạng thái sách
            cursor.execute("UPDATE books SET available = TRUE WHERE id = %s",
(book_id,))
```

```
            db.commit()
            messagebox.showinfo("Thành công", "Đã trả sách thành công")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()
        refresh_borrow_table()

    def search_muon_tra():
        search_query = search_muon_entry.get()
        db = connect_to_db()
        cursor = db.cursor()
        try:
            # Tìm kiếm trong bảng borrowings dựa trên ID sách, tên người mượn
hoặc ngày mượn
            cursor.execute("""
                SELECT * FROM borrowings
                WHERE book_id LIKE %s
                OR user_name LIKE %s
                OR DATE(borrow_date) = %s
            """, ('%' + search_query + '%', '%' + search_query + '%',
search_query))
            results = cursor.fetchall()

            # Xóa dữ liệu cũ trong bảng
            for i in borrow_tree.get_children():
                borrow_tree.delete(i)

            # Hiển thị kết quả tìm kiếm
            for borrowing in results:
                borrow_tree.insert('', 'end', values=borrowing)

            if not results:
                messagebox.showinfo("Kết quả", "Không tìm thấy kết quả phù
hợp.")
        except mysql.connector.Error as err:
            messagebox.showerror("Lỗi", str(err))
        finally:
            cursor.close()
            db.close()


    borrow_button = tk.Button(frame_borrow, text="Mượn Sách",
command=borrow_book)
    borrow_button.pack(side=tk.LEFT, padx=10)

    return_button = tk.Button(frame_borrow, text="Trả Sách",
command=return_book)
    return_button.pack(side=tk.LEFT, padx=10)

    search_button = tk.Button(frame_borrow, text="Tìm Kiếm",
command=search_muon_tra)
    search_button.pack(side=tk.LEFT, padx=10)

    borrow_tree_frame = ttk.Frame(right_frame)
```

```
    borrow_tree_frame.pack(fill='both', expand=True, padx=10, pady=10)

    borrow_tree = ttk.Treeview(borrow_tree_frame, columns=("ID", "Book ID",
"User Name", "Borrow Date", "Return Date"),
                               show="headings")
    borrow_tree.heading("ID", text="ID")
    borrow_tree.heading("Book ID", text="ID Sách")
    borrow_tree.heading("User Name", text="Người Mượn")
    borrow_tree.heading("Borrow Date", text="Ngày Mượn")
    borrow_tree.heading("Return Date", text="Ngày Trả")
    borrow_tree.pack(fill='both', expand=True)

    def fetch_borrowings():
        db = connect_to_db()
        cursor = db.cursor()
        cursor.execute("SELECT * FROM borrowings")
        rows = cursor.fetchall()
        cursor.close()
        db.close()
        return rows

    def refresh_borrow_table():
        for i in borrow_tree.get_children():
            borrow_tree.delete(i)
        borrowings = fetch_borrowings()
        for borrowing in borrowings:
            borrow_tree.insert('', 'end', values=borrowing)

    refresh_borrow_button = ttk.Button(right_frame, text="Làm mới",
command=refresh_borrow_table)
    refresh_borrow_button.pack(pady=(0, 10))

    refresh_borrow_table()


def yeu_cau_nguoi_dung():
    clear_right_frame()

    request_frame = ttk.Frame(right_frame)
    request_frame.pack(padx=10, pady=10, fill='both', expand=True)

    request_name_label = ttk.Label(request_frame, text="Tên người dùng:")
    request_name_label.grid(row=0, column=0, padx=5, pady=5)
    request_name_entry = ttk.Entry(request_frame)
    request_name_entry.grid(row=0, column=1, padx=5, pady=5)

    request_text_label = ttk.Label(request_frame, text="Nội dung yêu cầu:")
    request_text_label.grid(row=1, column=0, padx=5, pady=5)
    request_text_entry = tk.Text(request_frame, height=5)
    request_text_entry.grid(row=1, column=1, padx=5, pady=5)

    submit_request_button = ttk.Button(request_frame, text="Gửi yêu cầu",
                                       command=lambda:
submit_request(request_name_entry, request_text_entry))
    submit_request_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

    view_requests_button = ttk.Button(request_frame, text="Xem yêu cầu
```

```
(Admin)", command=view_requests)
    view_requests_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5)


def submit_request(name_entry, text_entry):
    user_name = name_entry.get()
    request_text = text_entry.get("1.0", tk.END).strip()
    if user_name and request_text:
        try:
            conn = connect_to_db()
            cursor = conn.cursor()
            cursor.execute(
                "INSERT INTO user_requests (user_name, request_content,
request_date) VALUES (%s, %s, %s)",
                (user_name, request_text, datetime.now().date()))
            conn.commit()
            messagebox.showinfo("Thông báo", "Yêu cầu đã được gửi thành
công!")
            name_entry.delete(0, tk.END)
            text_entry.delete("1.0", tk.END)
        except Exception as e:
            messagebox.showerror("Lỗi", f"Đã xảy ra lỗi: {e}")
        finally:
            conn.close()
    else:
        messagebox.showerror("Lỗi", "Vui lòng điền đầy đủ thông tin yêu
cầu!")


def view_requests():
    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM user_requests")
        requests = cursor.fetchall()
    except Exception as e:
        messagebox.showerror("Lỗi", f"Đã xảy ra lỗi: {e}")
    finally:
        conn.close()

    request_window = tk.Toplevel(root)
    request_window.title("Yêu cầu của người dùng")
    request_tree = ttk.Treeview(request_window,
                                columns=("ID", "Tên người dùng", "Nội dung
yêu cầu", "Ngày yêu cầu"),
                                show="headings")
    request_tree.heading("ID", text="ID")
    request_tree.heading("Tên người dùng", text="Tên người dùng")
    request_tree.heading("Nội dung yêu cầu", text="Nội dung yêu cầu")
    request_tree.heading("Ngày yêu cầu", text="Ngày yêu cầu")
    for request in requests:
        request_tree.insert("", "end", values=request)
    request_tree.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

# Navigation buttons in the left frame
ttk.Button(left_frame, text="Thư Viện", command=show_thu_vien).pack(fill='x')
ttk.Button(left_frame, text="Mượn trả sách",
```

```
command=show_muon_tra).pack(fill='x')
ttk.Button(left_frame, text="Yêu Cầu Người Dùng",
command=yeu_cau_nguoi_dung).pack(fill='x')

root.mainloop()
```