Capstone Project

|  |  |  |
| --- | --- | --- |
| **9 Online Bookstore System** | | |
| **Aim:**  Enhance the basic online bookstore system to include database integration, file handling,  exception handling, and an interactive interface.  Task Description:   Database Integration: Use SQLite to store and manage book inventory, customer  details, and order records.   File Operations: Provide functionality to export and import book inventory and  customer orders to/from text files.   Modules and Packages: Organize the system into reusable modules and packages for  database operations, file handling, and utility functions.   Exception Handling: Handle errors related to database operations, file handling, and  invalid user inputs gracefully.  Interactive Interface: Create a menu-driven program that allows users to browse  books, place orders, view order history, and manage inventory | | |
| **Code:**  import sqlite3  import os  from datetime import datetime  *# === Database Setup ===*  def create\_connection():  *"""Connect to SQLite database. Create if not exists."""*  try:  conn = sqlite3.connect('bookstore.db')  return conn  except sqlite3.Error as e:  print(f"Database error: {e}")  return None  def initialize\_database():  *"""Create tables for books, customers, and orders."""*  conn = create\_connection()  if conn:  try:  cursor = conn.cursor()  cursor.execute('''CREATE TABLE IF NOT EXISTS books (  isbn TEXT PRIMARY KEY,  title TEXT NOT NULL,  author TEXT,  price REAL,  quantity INTEGER  )''')    cursor.execute('''CREATE TABLE IF NOT EXISTS customers (  customer\_id INTEGER PRIMARY KEY AUTOINCREMENT,  name TEXT NOT NULL,  email TEXT UNIQUE  )''')    cursor.execute('''CREATE TABLE IF NOT EXISTS orders (  order\_id INTEGER PRIMARY KEY AUTOINCREMENT,  customer\_id INTEGER,  book\_isbn TEXT,  quantity INTEGER,  order\_date TEXT,  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),  FOREIGN KEY (book\_isbn) REFERENCES books(isbn)  )''')  conn.commit()  print("Database initialized successfully!")  except sqlite3.Error as e:  print(f"Table creation error: {e}")  finally:  conn.close()  *# === File Handling ===*  def export\_books\_to\_file(filename="books\_export.txt"):  *"""Export books to a text file."""*  conn = create\_connection()  try:  cursor = conn.cursor()  cursor.execute("SELECT \* FROM books")  books = cursor.fetchall()  with open(filename, 'w') as f:  for book in books:  f.write(f"{book[0]}|{book[1]}|{book[2]}|{book[3]}|{book[4]}\n")  print(f"Books exported to {filename}!")  except Exception as e:  print(f"Export error: {e}")  finally:  conn.close()  def import\_books\_from\_file(filename="books\_import.txt"):  *"""Import books from a text file."""*  conn = create\_connection()  try:  cursor = conn.cursor()  with open(filename, 'r') as f:  for line in f:  data = line.strip().split('|')  isbn, title, author, price, quantity = data  cursor.execute('''INSERT INTO books (isbn, title, author, price, quantity)  VALUES (?, ?, ?, ?, ?)''',  (isbn, title, author, float(price), int(quantity)))  conn.commit()  print(f"Books imported from {filename}!")  except FileNotFoundError:  print("File not found!")  except Exception as e:  print(f"Import error: {e}")  finally:  conn.close()  *# === Core Functions ===*  def add\_book():  *"""Add a new book to the inventory."""*  conn = create\_connection()  try:  isbn = input("Enter ISBN: ")  title = input("Enter title: ")  author = input("Enter author: ")  price = float(input("Enter price: "))  quantity = int(input("Enter quantity: "))    cursor = conn.cursor()  cursor.execute('''INSERT INTO books (isbn, title, author, price, quantity)  VALUES (?, ?, ?, ?, ?)''',  (isbn, title, author, price, quantity))  conn.commit()  print("Book added successfully!")  except ValueError:  print("Invalid input! Price/quantity must be numbers.")  except sqlite3.IntegrityError:  print("Book with this ISBN already exists!")  finally:  conn.close()  def browse\_books():  *"""Display all books in inventory."""*  conn = create\_connection()  try:  cursor = conn.cursor()  cursor.execute("SELECT \* FROM books")  books = cursor.fetchall()  print("\n=== Available Books ===")  for book in books:  print(f"ISBN: {book[0]}\nTitle: {book[1]}\nAuthor: {book[2]}\nPrice: ${book[3]}\nStock: {book[4]}\n")  except sqlite3.Error as e:  print(f"Database error: {e}")  finally:  conn.close()  def place\_order():  *"""Place an order and update inventory."""*  conn = create\_connection()  try:  *# Get customer details*  name = input("Enter your name: ")  email = input("Enter your email: ")    *# Add customer to database*  cursor = conn.cursor()  cursor.execute('''INSERT INTO customers (name, email)  VALUES (?, ?)''', (name, email))  customer\_id = cursor.lastrowid    *# Get book details*  browse\_books()  isbn = input("Enter book ISBN: ")  quantity = int(input("Enter quantity: "))    *# Check book availability*  cursor.execute("SELECT quantity FROM books WHERE isbn = ?", (isbn,))  stock = cursor.fetchone()[0]  if stock < quantity:  print("Insufficient stock!")  return    *# Update inventory and create order*  new\_stock = stock - quantity  cursor.execute('''UPDATE books SET quantity = ?  WHERE isbn = ?''', (new\_stock, isbn))  order\_date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  cursor.execute('''INSERT INTO orders (customer\_id, book\_isbn, quantity, order\_date)  VALUES (?, ?, ?, ?)''',  (customer\_id, isbn, quantity, order\_date))  conn.commit()  print("Order placed successfully!")  except ValueError:  print("Invalid input!")  except sqlite3.Error as e:  print(f"Order failed: {e}")  finally:  conn.close()  *# === Menu System ===*  def display\_menu():  print("\n===== Online Bookstore =====")  print("1. Browse Books")  print("2. Place Order")  print("3. Add New Book")  print("4. Export Books to File")  print("5. Import Books from File")  print("6. Exit")  def main():  initialize\_database()  while True:  display\_menu()  choice = input("Enter your choice (1-6): ")  if choice == '1':  browse\_books()  elif choice == '2':  place\_order()  elif choice == '3':  add\_book()  elif choice == '4':  export\_books\_to\_file()  elif choice == '5':  import\_books\_from\_file()  elif choice == '6':  print("Exiting...")  break  else:  print("Invalid choice. Try again.")  if \_\_name\_\_ == "\_\_main\_\_":  main()  **Output Screenshot:** | | |
| **Conclusion/Summary:**  Through the development of the Online Bookstore System, I successfully integrated core programming concepts to build a functional application. Here’s a summary of my accomplishments:  📚 Database Integration:  Designed SQLite tables for books, customers, and orders with proper relationships.  Executed CRUD operations to manage inventory and orders efficiently.  📂 File Handling:  Implemented export/import functionality using text files (books\_export.txt, books\_import.txt).  Resolved delimiter conflicts by using | to ensure smooth data parsing.  🛠 Error Handling:  Added try-except blocks to handle database errors, invalid inputs, and file issues.  Customized error messages for better user guidance (e.g., "Invalid price/quantity!").  💻 Interactive Interface:  Created a menu-driven console interface for easy navigation.  Enabled users to browse books, place orders, and manage inventory seamlessly.  🔧 Challenges & Learning:  Debugged the "too many values to unpack" error by switching delimiters.  Gained proficiency in SQLite, Python modules, and user input validation.  This project deepened my understanding of real-world application development, emphasizing the importance of structured code, error resilience, and user experience. It solidified my ability to combine databases, files, and interfaces into a cohesive system. | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |