Assignment 9

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()
```

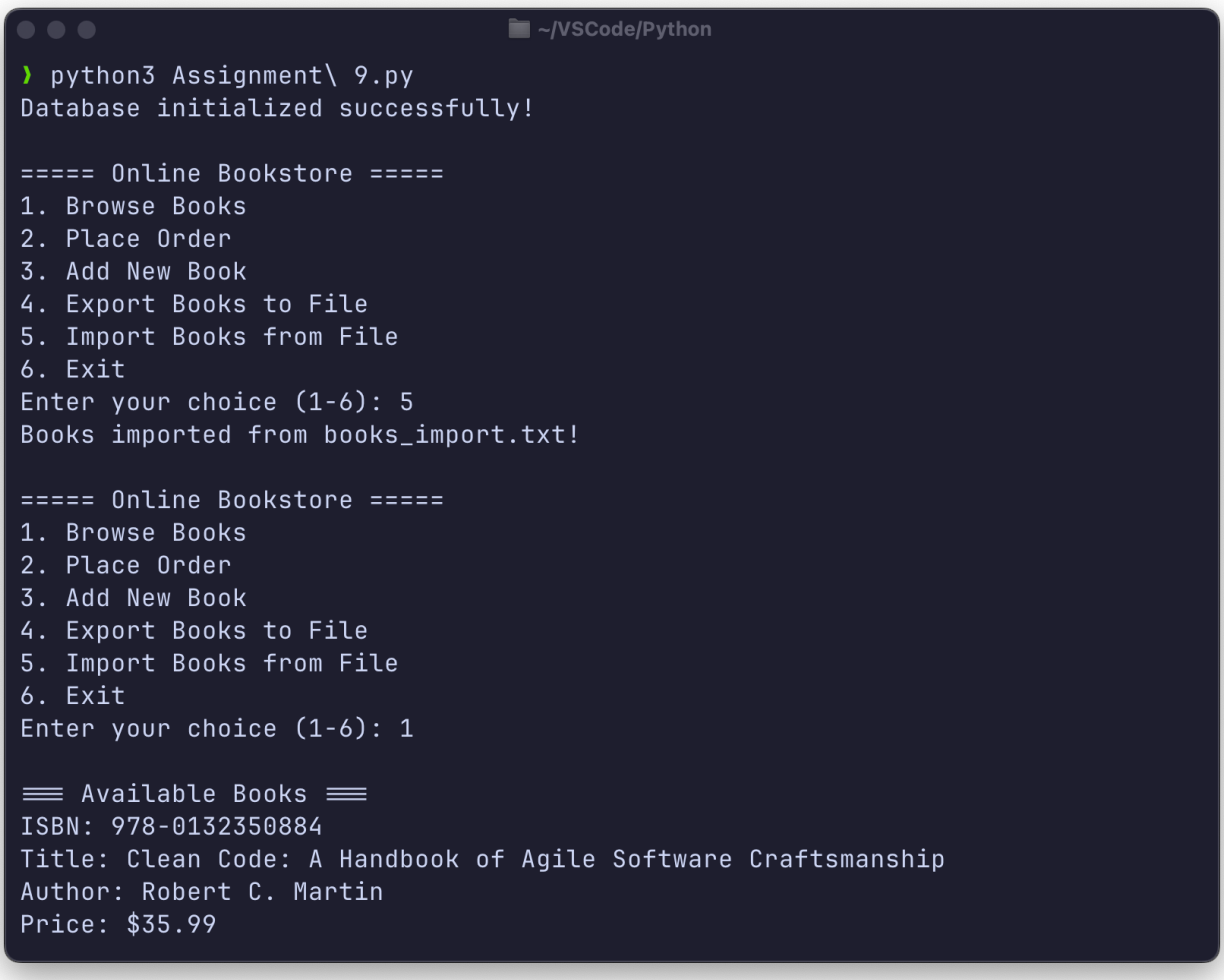
```
def import_books_from_file(filename="books_import.txt"):
    """Import books from a text file."""
    conn = create_connection()
    try:
        cursor = conn.cursor()
        with open(filename, 'r') as f:
            for line in f:
                data = line.strip().split('|')
                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.")
```

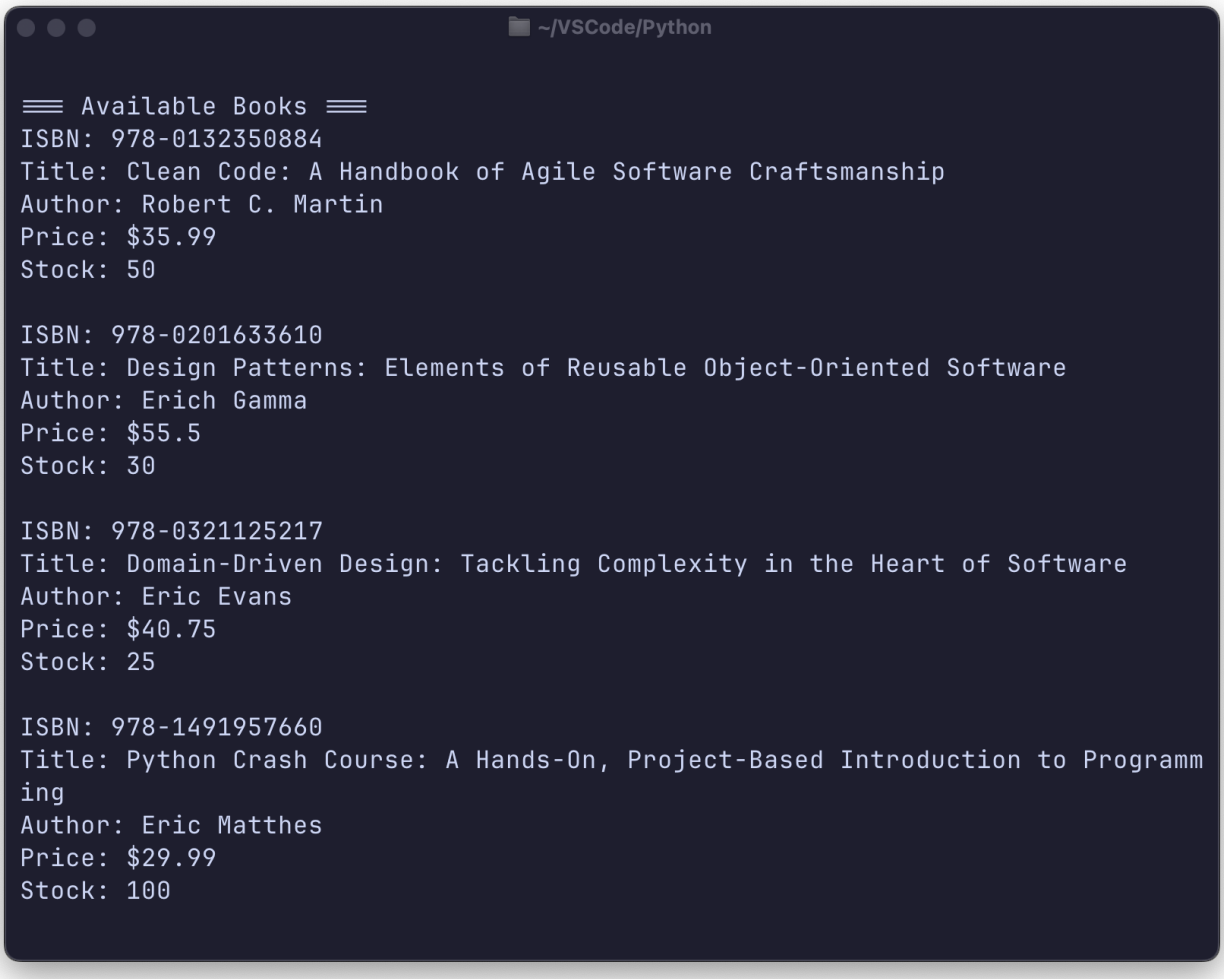
```
    except sqlite3.IntegrityError:
        print("Book with this ISBN already exists!")
    finally:
        conn.close()

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,))
```

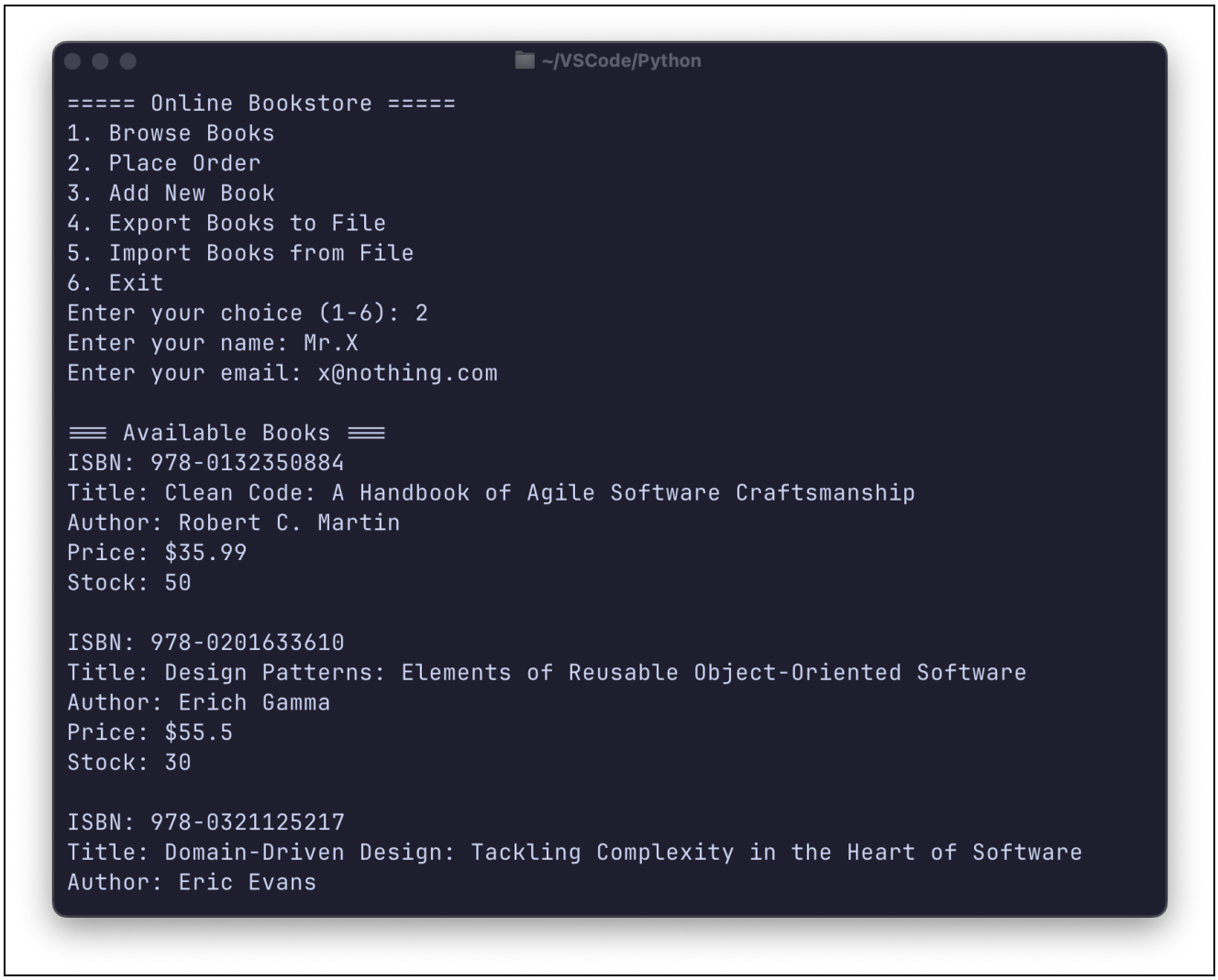
```
        stock = cursor.fetchone()[0]
        if stock < quantity:
            print("Insufficient stock!")
            return

        # 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':
```

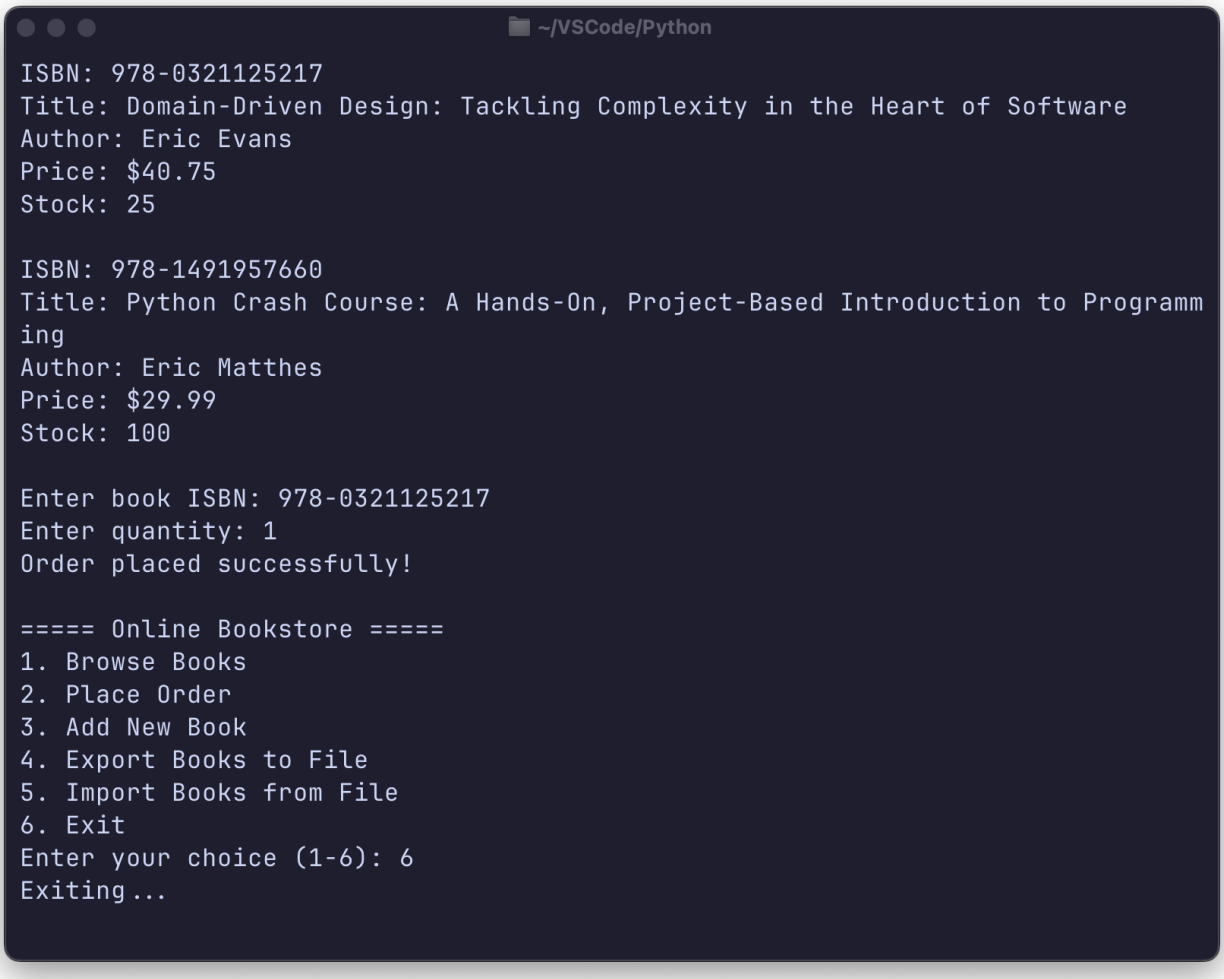
```
            add_book()
        elif choice == '4':
            export_books_to_file()
        elif choice == '5':
            import_books_from_file()
        elif choice == '6':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```

Output Screenshot:




~/VSCode/Python
Available Books
ISBN: 978-0132350884
Title: Clean Code: A Handbook of Agile Software Craftsmanship
Author: Robert C. Martin
Price: $35.99
Stock: 50
ISBN: 978-0201633610
Title: Design Patterns: Elements of Reusable Object-Oriented Software
Author: Erich Gamma
Price: $55.5
Stock: 30
ISBN: 978-0321125217
Title: Domain-Driven Design: Tackling Complexity in the Heart of Software
Author: Eric Evans
Price: $40.75
Stock: 25
ISBN: 978-1491957660
Title: Python Crash Course: A Hands-On, Project-Based Introduction to Programm
ing
Author: Eric Matthes
Price: $29.99
Stock: 100


~/VSCode/Python
===== Online Bookstore =====
1. Browse Books
2. Place Order
3. Add New Book
4. Export Books to File
5. Import Books from File
6. Exit
Enter your choice (1-6): 2
Enter your name: Mr.X
Enter your email: x@nothing.com
=== Available Books ===
ISBN: 978-0132350884
Title: Clean Code: A Handbook of Agile Software Craftsmanship
Author: Robert C. Martin
Price: $35.99
Stock: 50
ISBN: 978-0201633610
Title: Design Patterns: Elements of Reusable Object-Oriented Software
Author: Erich Gamma
Price: $55.5
Stock: 30
ISBN: 978-0321125217
Title: Domain-Driven Design: Tackling Complexity in the Heart of Software
Author: Eric Evans



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**