Hotel Management System Using Streamlit and SQLite

Introduction

The **Hotel Management System** is a software application designed to streamline operations in hotel management, such as room booking, check-in, check-out, food ordering, bill generation, and records management. This project leverages **Streamlit**, a Python framework for creating interactive web applications, and **SQLite**, a lightweight database engine.

Objective

The primary goal of this project is to develop an easy-to-use interface for managing hotel operations while maintaining robust backend functionality. The system aims to:

1. Automate common hotel management tasks.
2. Ensure data consistency and integrity.
3. Provide a user-friendly GUI for managers and staff.

System Features

1. Room Management

Database Design:

- A table `rooms` stores details of each room, including:
  - `room_id`: Unique identifier for rooms.
  - `room_type`: Type of room (e.g., Single, Double, Suite).
  - `price_per_night`: Nightly rate.
  - `availability`: Status (`Available`, `Booked`, `Checked-in`).

Functionality: Users can query room types, prices, and availability. Available rooms are displayed with dynamic styling for improved readability.

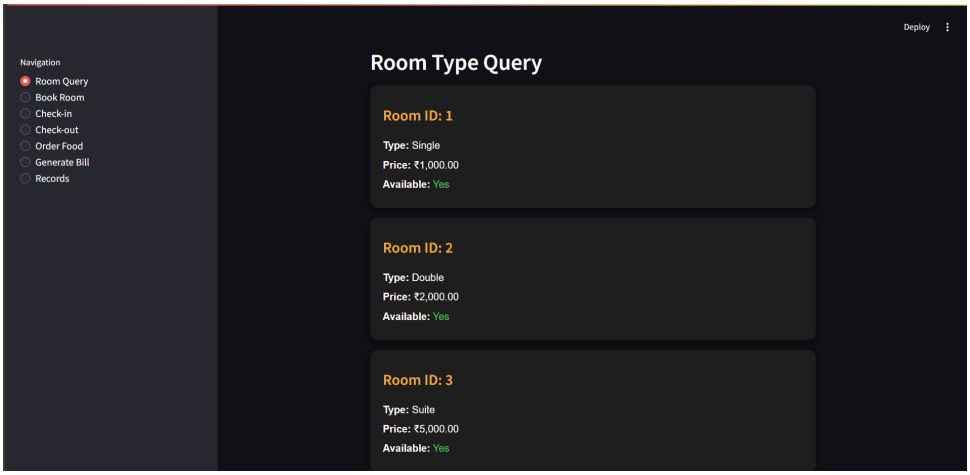

Figure 1: Room Query

2. Room Booking

Allows users to:

- Enter their name.
- Select an available room type.
- Specify check-in and check-out dates.

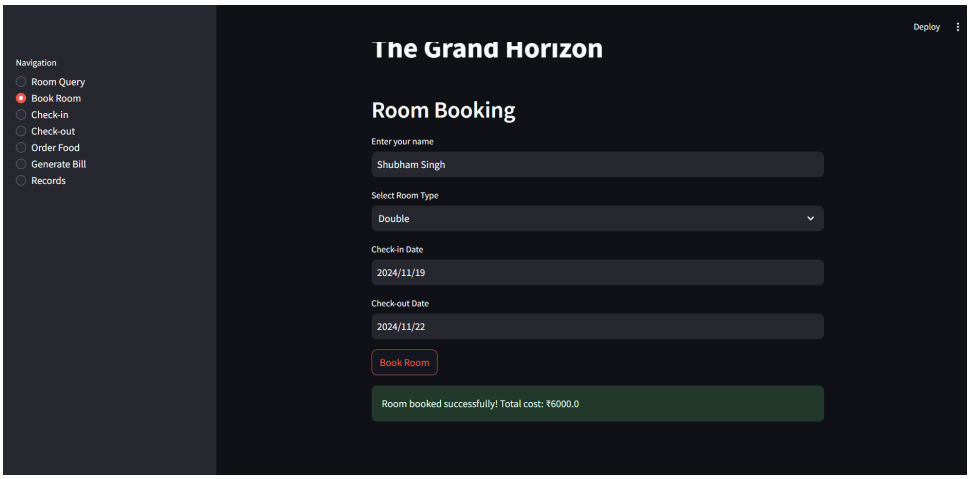

Figure 2: Room Booking

Key Features:

- Ensures logical date validation (check-out date must be after the check-in date).
- Automatically calculates the total cost of the stay.
- Updates room status to `Booked`.

3. Check-in/Check-out

Check-in:

- Users input their booking ID.
- The system validates the booking ID and updates room availability to `Checked-in`.

Check-out:

- Users input their booking ID.
- The system updates room availability to `Available`.

4. Food Ordering

- Users can select from predefined menu items (e.g., Burger, Pizza).
- Input their booking ID to place an order.

Key Features:

- Food prices are pre-defined in a dictionary.
- Orders are linked to bookings for later billing.

5. Bill Generation

The system generates a detailed bill for each booking ID, including:

- Customer details.
- Room type and cost.
- Food items ordered and their cost.
- Total amount due.

6. Records Management

- A comprehensive list of all bookings is displayed.
- Bookings include customer name, room type, check-in/out dates, and total cost.

Technical Implementation

Backend

SQLite Database:

- Tables: `rooms`, `bookings`, `food_orders`.
- Relationships:

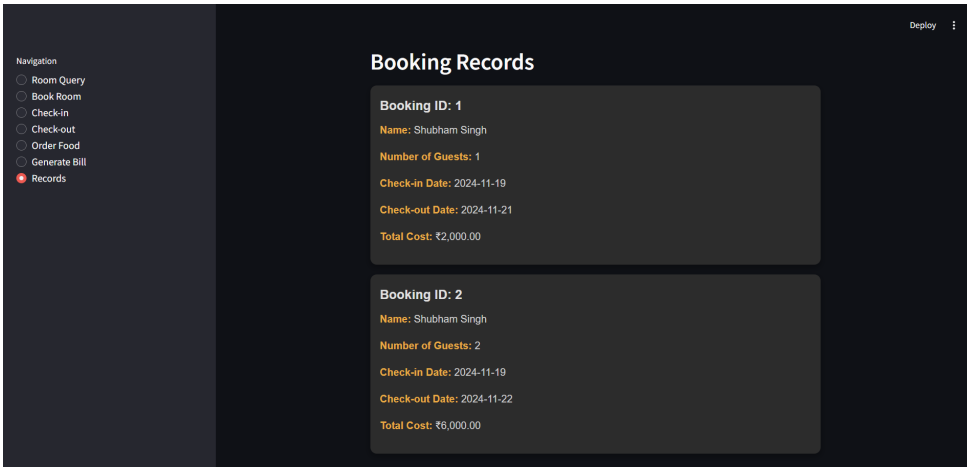

Figure 3: Records

- `rooms` → `bookings` (1-to-many).
- `bookings` → `food_orders` (1-to-many).

Database Initialization:

- The system creates tables if they do not exist.
- Sample data is populated into the `rooms` table.

Frontend

Streamlit Framework:

- Sidebar navigation for selecting features.
- Interactive forms and input fields for user data.
- Dynamic HTML and CSS for visual appeal.

Highlights

1. **Error Handling:** Prevents invalid dates for bookings and handles non-existent booking IDs gracefully.
2. **Dynamic Updates:** Real-time room availability status and automatic calculation of costs based on duration.
3. **Usability:** Clean and modern GUI with styled elements and intuitive workflows for all operations.

Future Enhancements

1. Integration with payment gateways for online payments during bill generation.
2. Room and menu management to allow dynamic updates by hotel staff.
3. Enhanced reporting to generate monthly revenue and occupancy reports.

Conclusion

This hotel management system provides an efficient solution for managing hotel operations through an interactive interface and robust database management. It successfully demonstrates the use of Python's **Streamlit** for building modern applications and **SQLite** for database management. While it fulfills most basic requirements, implementing the suggested enhancements can significantly improve its usability and scalability.