Project Report

Hotel Management System

Submitted by: Manas Saini

Reg No:25BCE10183

Date: 25,November 2025

1. Abstract

The Hotel Management System is a console-based software application designed to assist hotel administrators in managing room bookings and check-outs. The project utilizes Python to create a user-friendly interface that handles data input, updates room statuses in real-time, and ensures data integrity through exception handling. This report documents the system's design, implementation, and functionality.

2. Introduction

In the modern era, automation is key to business efficiency. This project focuses on the core operations of a hotel: checking room availability, registering guests, and processing departures. By moving away from manual record-keeping, the system minimizes errors and optimizes the workflow at the reception desk.

3. System Analysis

3.1 Hardware Requirements

- **Processor:** Intel Core i3 or equivalent (Minimum).
- **RAM:** 2GB or higher.
- **Storage:** 100MB free space.

3.2 Software Requirements

- **Operating System:** Windows, macOS, or Linux.
- **Runtime Environment:** Python 3.6 or higher.
- **IDE/Editor:** VS Code, PyCharm, or standard IDLE.

4. System Design & Methodology

4.1 Data Structure

The project uses a Python **Dictionary** (hotel_rooms) as the primary database.

- **Keys:** Integers representing Room Numbers (e.g., 101, 202).
- **Values:** Strings representing status. If the string is "Available", the room is free. If it contains a name (e.g., "Alice"), the room is occupied.

Reasoning: Dictionaries provide O(1) time complexity for lookups, making the system extremely fast even if the number of rooms increases.

4.2 Algorithm / Logic Flow

1. **Start:** Initialize the dictionary with default values.
2. **Menu Loop:** Display options to the user.
3. **Booking Logic:**
    - Input Room Number.
    - Check if Room Number exists in Dictionary Keys.
    - Check if Value is "Available".
    - If True -> Update Value to Guest Name.
    - If False -> Display "Occupied" error.
4. **Check-Out Logic:**
    - Input Room Number.
    - Check if Room is occupied.
    - If True -> Reset Value to "Available".
5. **Exception Handling:** Wrap inputs in try-except blocks to catch non-integer inputs.

5. Implementation Details

The system is modularized into four main functions:

1. **display_rooms()**: Iterates through the dictionary items and prints the status of every room.
2. **book_room(room_num, guest_name)**: Accepts arguments, validates the room status, and updates the dictionary.

3. **check_out(room_num)**: Reverts the status of a specific key in the dictionary to the default state.
4. **main_menu()**: The driver function that contains the while True loop and processes user input.

6. Limitations & Future Scope

6.1 Limitations

- **Data Persistence:** Currently, data is stored in RAM. If the program closes, all booking data is lost.
- **Scalability:** Adding rooms requires editing the source code dictionary manually.

6.2 Future Scope

- **File Handling:** Implementing CSV or JSON storage to save booking data permanently.
- **Billing Module:** Adding room rates and calculating the total bill upon check-out.
- **Search:** Adding a function to search for a guest by name rather than room number.

7. Conclusion

The Hotel Management System successfully demonstrates the application of Python programming to solve a real-world logistical problem. It meets the primary objectives of efficiency, accuracy, and ease of use. While currently a console-based application, the logic implemented here serves as a solid backend structure for future web or GUI-based iterations.