C Project Report

Hotel Room Booking System

By CodeNAS
IE2025015 Nitheesh Kumar Reddy
IE2025008 Bayyapu Adhiraj Reddy
IE2025020 Kumar Sanket

Project Overview

The Hotel Room Booking System is a C-based application for managing hotel operations such as room bookings, guest data management, staff authentication, and guest feedback. The system supports two main roles—*hotel guests* and *staff*—and provides a clear, menu-driven command-line interface for each role.

Detailed Explanation of Each Function

- Room Booking: Guests can view available rooms and book them. The system confirms the room number, validates its availability, and records guest details.
- Guest Management: Tracks guest data and maintains booking history.
- Staff Interface: Staff must log in to access management features. They can view all bookings, check room statuses, manage checkouts, and access user feedback.
- Feedback System: Guests can provide feedback on their stay, which is stored and viewable by staff.
- Booking History: Both current and past bookings are stored persistently (using `.dat` files).
- Authentication: Staff login is enforced with a password for security.
- Data Persistence: All booking, guest, and feedback data are saved to disk to preserve state across sessions.

Implementation of each Functionality

- *Room Booking*: Implemented in `booking.c`. When a guest books a room, the system checks if the room is available, updates room status, generates a booking ID, and saves the booking both in memory and to disk via `bookings.dat`.
- *Checkout*: Guests (or staff) can cancel a booking, marking the room as available again. Cancelled bookings move to history storage.
- *Room Management*: In `room.c`, room records are initialized for multiple floors and types. The table lists status, guest name, beds, and AC type.
- *Authentication*: In `auth.c`, the `staffLogin()` function compares an entered password with the hardcoded staff password ("nas2025"). Only successful login allows staff to access their menu.
- *Feedback System*: Implemented in `feedback.c`, allowing users to submit feedback, which is stored in a linked list and viewable by staff.
- *History*: Upon startup and exit, booking data is loaded/saved to files for persistence.
- *Menu Navigation*: The main menu in `main.c` drives user and staff choices. Depending on their option, appropriate modules and sub-menus are invoked.

Team Members

- Nitheesh Kumar Reddy (IE2025015):
  Staff authentication (auth.c, auth.h), Feedback system (feedback.c, feedback.h)
- Bayyapu Adhiraj Reddy (IE2025008):
  Room management (room.c, room.h), Main program flow (main.c)
- Kumar Sanket (IE2025020):
  Booking management (booking.c, booking.h)

Functions used

booking.c / booking.h

- userMenu()
  - Displays the guest menu, allowing actions like search, book, checkout, and submit feedback.
  - *Inputs*: Username (via CLI), guests' option
  - *Outputs*: Interactive CLI responses
  - *Role*: Entry point for all guest operations.
- staffMenu()
  - Displays the staff menu, letting staff view room/guest status, bookings, manage checkouts.
  - *Inputs*: Staff password verification, options
  - *Outputs*: CLI staff dashboard
  - *Role*: Controls access to administrative actions and reports.
- loadBookingData()

    - Loads current booking records from data file into a linked list on program start.
    - *Inputs*: File name/path
    - *Outputs*: Populated booking linked list in memory
    - *Role*: Ensures continuity by reading existing bookings.
- saveBookingData()
    - Saves current bookings in memory to disk at program end (or changes).
    - *Inputs*: Linked list of bookings
    - *Outputs*: Updated data file
    - *Role*: Maintains persistent state for current bookings.
- loadPastBookingData()
    - Loads historical/cancelled booking records from history file to memory.
    - *Inputs*: File location
    - *Outputs*: Linked list of history bookings
    - *Role*: Enables viewing of historical data for reference.
- savePastBookingData()
    - Saves completed/cancelled bookings from memory to disk.
    - *Inputs*: Linked list of completed/cancelled bookings
    - *Outputs*: Persisted booking history file
    - *Role*: Ensures all booking actions are properly archived.
- **addBooking(const char username)*
    - Books an available room for a given user, generating booking ID and updating status.
    - *Inputs*: Username, room choice/input
    - *Outputs*: New booking record added to memory and disk
    - *Role*: Core reservation function for guests.
- **cancelBookingForUser(const char username)*
    - Cancels a booking for a user, marks room as available, moves record to history.
    - *Inputs*: Username, room ID
    - *Outputs*: Room status, booking history update
    - *Role*: Allows checkout by guest or staff.
- **moveBookingToHistory(Booking b)*
    - Moves a booking from current to history linked list upon checkout/cancellation.
    - *Inputs*: Booking pointer
    - *Outputs*: Updated booking lists
    - *Role*: Manages transfer between active and archived bookings.
- applyBookingToRooms()

    - Updates all room records in memory to reflect active bookings accurately.
    - *Inputs*: Room and booking lists
    - *Outputs*: Synced booking-room status
    - *Role*: Ensures allocations are up-to-date.
- searchRooms()
    - Lets users/staff filter rooms by type, status, beds, or AC for easier selection.
    - *Inputs*: Search criteria
    - *Outputs*: Filtered room list
    - *Role*: Search functionality to improve usability.
- **viewUserBookings(const char username)*
    - Lists current and past bookings for a specific user in readable format.
    - *Inputs*: Username
    - *Outputs*: Display to CLI
    - *Role*: Personal booking history for guests.
- viewUserBookingsStaff()
    - Allows staff to view all users' bookings and statuses.
    - *Inputs*: None
    - *Outputs*: Complete booking list
    - *Role*: Management overview for staff.
- viewAllCurrentGuests()
    - Displays all currently checked-in guests and their room details.
    - *Inputs*: None
    - *Outputs*: Table of active guests
    - *Role*: Quick monitoring tool for staff.
- generateBookingID()
    - Generates and returns a new, unique booking ID by incrementing the last-used ID. Reads/writes this value from a file to ensure IDs remain unique across sessions.
    - *Inputs*: None (manages internally with a file)
    - *Outputs*: Integer booking ID
    - *Role*: Ensures every booking operation receives a unique, consistent ID.
- getLastBookingID()
    - Fetches the most recently used booking ID from a file at program startup. Defaults to 1000 if the file does not exist or is empty.
    - *Inputs*: None (reads file)
    - *Outputs*: Integer last booking ID
    - *Role*: Initializes ID generation to avoid duplicate IDs in persistent workflows.
- saveLastBookingID(int lastId)

- Writes the latest booking ID to disk after operations that change the ID, preserving continuity for future sessions.
- *Inputs*: Integer booking ID
- *Outputs*: None (writes file)
- *Role*: Keeps booking ID management robust and persistent across program runs.

room.c / room.h

- initializeRooms()
    - Creates all room records with their attributes (number, beds, AC, status) at program start.
    - *Inputs*: None
    - *Outputs*: Linked list of room structs
    - *Role*: Establishes hotel room inventory.
- listRoomsTable()
    - Displays all rooms in a formatted table with occupancy, guest name, and features.
    - *Inputs*: Linked list of rooms
    - *Outputs*: CLI table
    - *Role*: Visualizes current hotel occupancy.
- checkoutRoomStaff()
    - Allows staff to manually check out a guest from a designated room.
    - *Inputs*: Room number
    - *Outputs*: Room status edited, booking updated
    - *Role*: Direct staff control over checkout.

feedback.c / feedback.h

- **addFeedback(const char username)*
    - Collects and stores feedback text from a guest for future review.
    - *Inputs*: Username, feedback string

    - *Outputs*: Feedback list updated
    - *Role*: Gathers guest evaluations for the hotel.
- listFeedbacks()
    - Shows a list of all feedback entries stored.
    - *Inputs*: None
    - *Outputs*: Table of feedback, with user info
    - *Role*: Feedback display for staff improvement.

auth.c / auth.h

- staffLogin()
    - Prompts for and verifies a staff password before authorizing admin access.
    - *Inputs*: Password string
    - *Outputs*: Success/failure flag
    - *Role*: Secures access to staff-only functions.