11-06-2025

DSA Project

Hostel Management System

Submitted By: Qasim Munir FA23-BCS-252

UMAIR AHMAD FA23-BCS-244

Submitted To: Sir Imran Shehzad

Jinnah Hostel Management System Documentation

1. Introduction

The **Jinnah Hostel Management System** is a web-based application designed to manage hostel bookings, guest records, and various data structures for efficient operations. It includes features such as room booking, guest record management, and data structure visualizations for room service requests, check-in queues, maintenance tasks, and guest connections. In this we use Data Structure i.e: Heap, Queue, Stack and LinkedList.

---

2. System Features

2.1. Booking Management

- Add new bookings with guest details (name, address, phone, room type, days).
- Calculate total fare based on room type and duration.

2.2. Records Management

- View all guest records in a table format.
- Search for specific guests by name, room number, or phone.
- Edit or delete existing guest records.

2.3. Data Structures Integration

- **Priority Queue (Heap):** Manages room service requests based on priority (High, Medium, Low).
- **Queue:** Handles guest check-in/check-out in a FIFO (First-In-First-Out) manner.
- **Stack:** Tracks room maintenance tasks in a LIFO (Last-In-First-Out) order.
- **Linked List:** Maintains connections between guests (e.g., shared bookings or groups).

2.4. Reports

- Displays statistics such as total guests, revenue, and room type distribution.

---

3. Data Structures and Their Functions

3.1. Priority Queue (Heap) for Room Service Requests

Purpose: Ensures high-priority requests (e.g., urgent repairs) are processed first.

Functions:

- `enqueue(item, priority)` – Adds a new request with a priority level.
- `dequeue()` – Removes and returns the highest-priority request.
- `bubbleUp()` **&** `sinkDown()` – Maintains the heap structure.

Usage in System:

- `addServiceRequest()` – Adds a new request to the heap.
- `processServiceRequest()` – Processes the highest-priority request.

3.2. Queue for Check-in/Check-out

Purpose: Manages guest check-ins in the order they arrive (FIFO).

Functions:

- `enqueue(element)` – Adds a guest to the queue.
- `dequeue()` – Removes the first guest from the queue.

Usage in System:

- `enqueueGuest()` – Adds a guest to the check-in queue.
- `dequeueGuest()` – Checks in the next guest.

3.3. Stack for Room Maintenance

Purpose: Tracks maintenance tasks, where the most recent task is handled first (LIFO).

Functions:

- `push(element)` – Adds a maintenance task to the stack.
- `pop()` – Removes and returns the latest task.

Usage in System:

- `pushMaintenance()` – Logs a new maintenance issue.
- `popMaintenance()` – Marks the latest task as completed.

3.4. Linked List for Guest Connections

Purpose: Maintains relationships between guests (e.g., shared bookings).

Functions:

- `addConnection(name, connectedTo)` – Links two guests.
- `findConnections(name)` – Retrieves all connections of a guest.

Usage in System:

- `addConnection()` – Establishes a connection between two guests.
- `findConnections()` – Displays all connections for a given guest.

---

4. Backend API Integration

The system interacts with a backend server (`http://localhost:3000`) for:

- **Adding/Updating/Deleting** guest records (`/api/addGuest`, `/api/guest/:roomNo`).
- **Fetching reports** (`/api/reports`).
- **Managing room service requests** (`/api/room-service-requests`).

---

5. User Interface (UI) Components

- **Tab-based navigation** (Booking, Records, Edit, Data Structures, Reports).
- **Interactive alerts** (success, error, info).
- **Visualizations** for data structures (Heap, Queue, Stack, Linked List).

---

6. Conclusion

The **Jinnah Hostel Management System** efficiently manages hostel operations using modern web technologies and data structures. It ensures smooth handling of bookings, guest records, and service requests while providing real-time reports and visualizations.

Future Enhancements

- **Real-time updates** (WebSocket integration).
- **Automated billing & notifications**.
- **Enhanced security** (user authentication).

Appendix: Code Structure

- **HTML:** Defines the UI layout.
- **CSS:** Styling for a clean, responsive interface.
- **JavaScript:** Implements data structures and API interactions.