**BOMMARILLU HOTEL MANAGEMENT SYSTEM**

**High-Level and Low-Level Design**

**High-Level Design:**

1. **Database:**
   * Centralized database to store booking, room, menu, and payment information.
   * Database schema designed to support data integrity and relationships.
2. **Modules for User Interface (UI):**

* Booking
* Rooms Info
* Menu card
* Payment
* Record (Admin people only)
* Exit

1. **Security:**
   * Implement secure authentication mechanisms for admin access.

**Low-Level Design:**

1. **Booking:**

* Validate user inputs (name, date, etc.) against predefined patterns.
* Check database for room availability using SQL queries.
* Confirm the booking details are stored in the database.

**Error Handling:**

* Handle errors for invalid user input.
* Display appropriate messages for room unavailability.

1. **Rooms Info:**
   * Fetch room details from the database.
   * Display room information on the user interface.
   * Ensure the system updates room status in real-time.

**Error Handling:**

* Handle database connection errors.
* Display friendly messages for unexpected issues with room information.

1. **Menu Card:**

* Retrieve menu details from the database.
* Display menu items and prices.
* Implement functionality to add items to a virtual cart.

**Error Handling:**

* Implement error messages for failure to fetch menu details.
* Handle errors when adding items to the virtual cart.

1. **Payment:**

* Integrate payment gateways with proper error handling.
* Securely transmit payment information.
* Generate and store payment receipts in the database.

**Error Handling:**

* Implement robust error handling for payment gateway failures.
* Display clear error messages for invalid payment details.

1. **Record (Admin People Only):**
   * Implement authentication mechanisms for admin access.
   * Design a secure database schema for storing sensitive information.
   * Implement CRUD operations for record management.

**Error Handling:**

* Implement proper error messages for authentication failures.
* Handle errors during record retrieval and modification.

1. **Exit:**
   * Implement a graceful exit mechanism.
   * Save user data and system state before exiting.

**Error Handling:**

* Address potential errors during the saving of user data.
* Provide feedback to the user in case of unexpected issues during exit.