# *Hotel Reservation System*
# System Design Document

**Team**: Tech Travel Titans

**Team Members**: Muskanmeet Kaur (219633361)
Sahib Deep Singh (219170646)
Mithunan Sivagnanam (215770472)
Saba Aafreen (217945825)

## Table of Contents

# Introduction

## Overview:

The hotel industry has a lot of moving parts and functions that can be difficult to keep track of if the system is not robust and dynamic. In this document, we discuss the software architecture to be implemented on our Hotel Reservation System and the design goals the proposed architecture will fulfill.

## Software Architecture:

It is befitting that our Hotel Reservation System implements a three-tier closed architecture. This architecture allows for optimizing scalability, and supportability, while providing a seamless user experience for all stakeholders. The system needs a robust and scalable data access layer to handle and store the large amount of data that is constantly changing. The system would also need an effective user interface layer that focuses on giving the best experience for our users, and lastly our application layer will be needed to perform business logic and several processes. A three-tier closed architecture approach is appropriate for the system as each layer is responsible for major functions of the system as described previously. The closed architecture approach enables for layer isolation so that testing and maintainability becomes easier, making each layer independent which allows more changes to occur without affecting the other layers (i.e. lower coupling), thus enhancing flexibility.

## Design Goals:

The system with the three-tier architecture would provide high maintainability, scalability due to the high amount of guest reservations and cancellations, as well as room allocations. There would be the ease of booking a hotel without disruptions and providing excellent customer service with the help of real-time data processing (e.g., via messaging, real-time updates on available rooms, assistance in "walking the guest", keeping OTAs and hotel operators to stay coordinated, etc.).

## References:

This document follows the requirements outlined in the *Hotel Reservation System Requirements Analysis Document.*

## Current Software Architecture

There are several types of software architecture that are used in hotel reservation systems today to ensure efficiency and robustness, but they also come with a few disadvantages as well. Currently, most of the hotel reservation systems implement *microservices architecture* which divides the system into smaller, independent components that are responsible for specific services such as hotel management services or customer services like searching and booking, making payments, etc. Though this architecture pattern allows for smooth flows in usage of the system it has some drawbacks. Firstly, microservices increase design complexity and things such as the size of each microservice, and what frameworks to use for each microservice must be tackled. When it comes to testing, the more services that are added the more difficult it becomes to test the system in terms of coordination. In addition to testing, maintaining, and operating each microservice needs more focus for each microservice. Lastly, as more microservices are added, maintaining data consistency becomes a challenge which cannot be overlooked, especially in an industry that has several moving parts and changes in data. We must ensure that the data we store, and process stay consistent and available to all stakeholders such as our customers, OTAs, and hotel operators to ensure customer satisfaction and meet business goals.

Therefore, our *three-tier closed architecture* provides a less complex and easy to maintain solution that preserves data integrity and is flexible. Each layer has its own purpose and is independent, thus change will not affect the other layers. Each layer can implement different technologies that are best suited for each job, such as the presentation layer can be used on mobile applications or web browsers. This architecture allows for high scalability since the application layer can be expanded to several devices to distribute the data load of both incoming data and current data. We want to also emphasize the independence of each layer in our system and the ease of maintainability as it is very important to do so given that the data is always changing, and the system requires real-time data updates to our different stakeholders. Lastly, we’ve ensured independence by implementing a *closed* architecture where one layer only communicates with the one after it.

## Proposed Software Architecture

### Overview

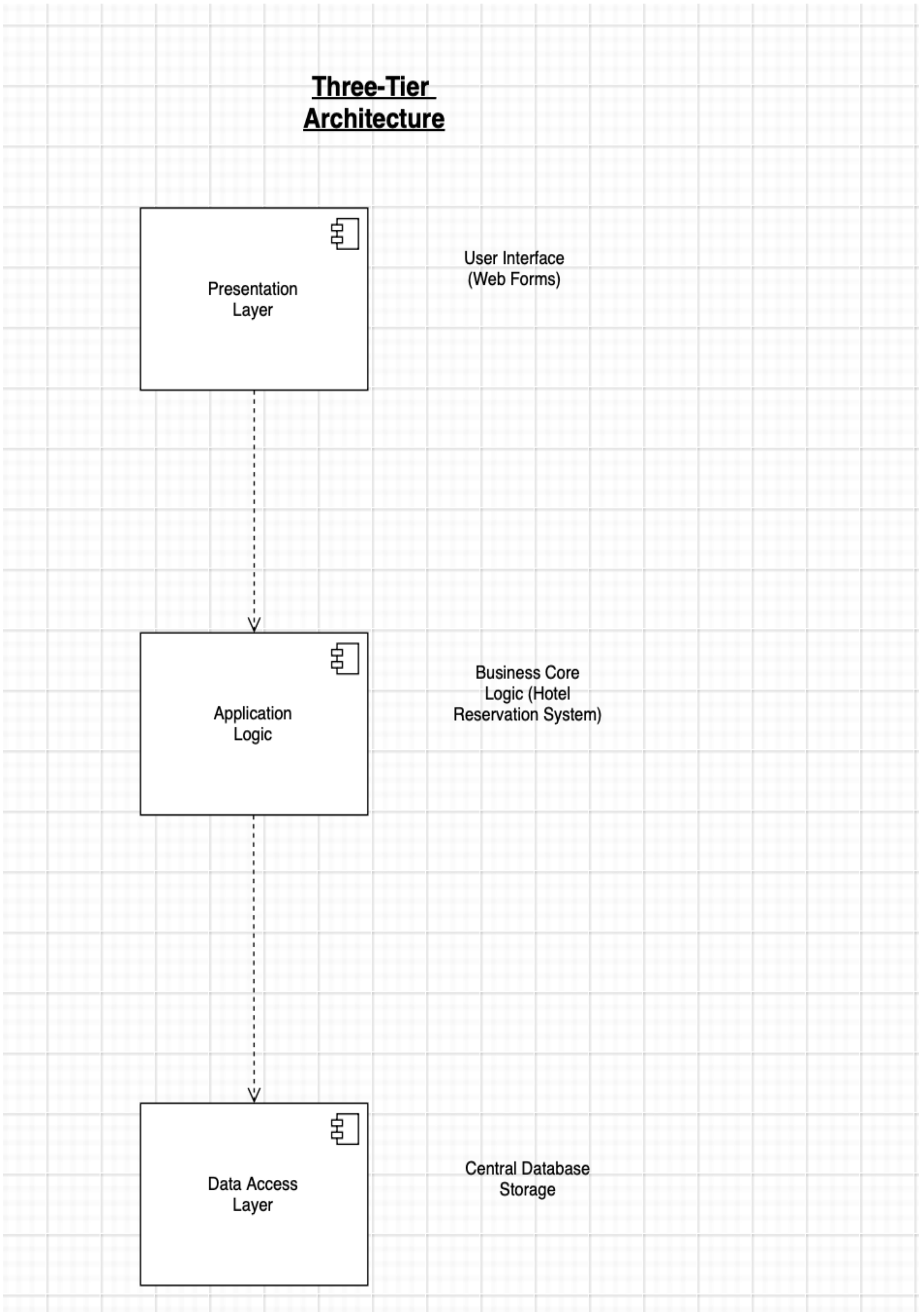
The proposed software architecture for the Hotel Reservation System is designed to provide a robust and scalable solution to manage various functionalities related to hotel bookings and management. The Hotel Reservation System is structured using a **three-tier** architecture, comprising the Presentation Layer, Application Layer, and Data Access Layer. This architecture ensures modularization, scalability, and maintainability of the system. Each layer is responsible for specific tasks, and interactions occur in a **closed** architecture manner, with each layer only interacting with the layer immediately below it.

**Three layers:**

- The Hotel Reservation system's interface layer includes all boundary objects that deal with the user, including windows, forms, web pages, and so on. The top-tier is the user-interface which runs on the client side of the browser.

- The Hotel Reservation system's application logic layer includes all control and entity objects, realizing the processing, rule checking, and notification required by the application. The middle-tier is the whole business logic encoded in the application software i.e. Hotel Reservation system itself which runs on the Web Server

- The Hotel Reservation system's storage layer realizes the storage, retrieval, and query of persistent objects. The third layer is the Database layer where Oracle Database Server stores and maintains all the data needs of the system.


Three-Tier Architecture
Presentation Layer
User Interface (Web Forms)
Application Logic
Business Core Logic (Hotel Reservation System)
Data Access Layer
Central Database Storage

The Hotel Reservation System consists of the following major subsystems on a high level:

1. **Account Management**
    a. Responsible for handling user accounts and authentication processes.
    b. Subsystems:
        i. **CreateAccountManagement**: Manages the creation of user accounts.
        ii. **LoginManagement**: Handles user login and authentication.
        iii. **AuthenticateUserManagement**: Verifies user credentials and manages role-specific dashboards.
        iv. **LogoutManagement**: Manages user logout and session termination.

2. **Reservation Management**
    a. Focuses on handling hotel reservations, booking, and related processes.
    b. Subsystems:
        i. **SearchHotelsManagement:** Manages the hotel search interface and search result processing.
        ii. **BookRoomManagement:** Handles room booking and reservation confirmation.
        iii. **MakePaymentManagement:** Manages the payment process for booked rooms.
        iv. **ModifyReservationManagement:** Allows users to modify existing reservations.
        v. **CancelReservationManagement:** Manages the cancellation of reservations.

3. **Check-In/Check-Out Management**
    a. Deals with the check-in and check-out processes, ensuring a smooth guest experience.
    b. Subsystems:
        i. **Check-InManagement**: Manages both online and front desk check-in processes.
        ii. **Check-OutManagement**: Handles the check-out process, including payment and feedback.

4. **Hotel Management**
    a. Focuses on the management of hotels, their details, and operational aspects.
    b. Subsystems:
        i. **ManageHotelsManagement:** Allows hotel staff to update information, upload images, and manage facilities, rooms, and rates.
        ii. **AccessForumManagement:** Enables interaction through an online forum.

        iii. **GenerateReportManagement:** Generates various reports on hotel performance and reservation data.

5. **User Interaction Management**
    a. Manages user interactions and settings within the system.
    b. Subsystems:
        i. **MessageUserManagement**: Handles user messaging.
        ii. **AccessUserSettingsManagement**: Allows users to customize their preferences, environmental options, and user information.
        iii. **RateExperienceManagement**: Manages the user feedback and rating process.

6. **Digital Key and Confirmation Management**
    a. Deals with the generation and confirmation processes related to reservations and access.
    b. Subsystems:
        i. **GenerateDigitalKeyCards/QRManagement**: Manages the generation of digital key cards or QR codes for guest access.
        ii. **SendConfirmationEmailManagement**: Handles the generation and sending of confirmation emails.

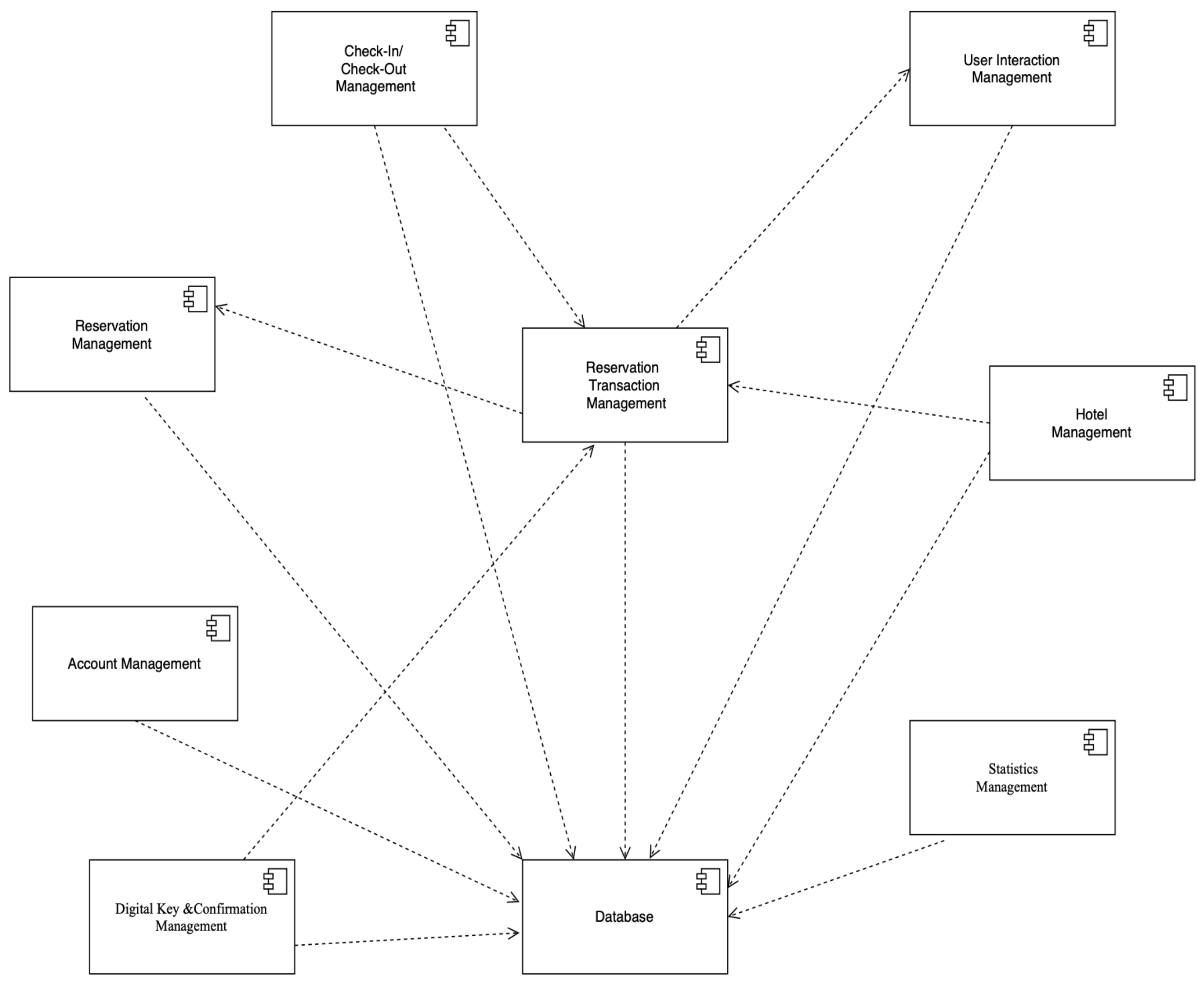
7. **Statistics Management**
    a. Provides real-time statistics and insights into the hotel's performance.
    b. Subsystems:
        i. **CheckRealTimeStatisticsManagement:** Displays real-time data on revenue, occupancy rates, and other key performance indicators.

8. **Reservation Transaction Management**
    a. Centralized subsystem dedicated to coordinating and managing transactions specifically related to hotel reservations.
    b. Ensures atomicity, consistency, isolation, and durability (ACID properties) for transactions that involve multiple subsystems.
    c. Subsystems:
        i. **ReservationTransactionManagement:** Specialized in handling transactions related to the reservation life cycle, ensuring the integrity and accuracy of reservation-related data.
        ii. **PaymentTransactionManagement:** Coordinates transactions related to payments for room bookings.
        iii. **UserAccountTransactionManagement:** Handles transactions related to user account creation, modification, and authentication.

iv. **HotelOperationTransactionManagement:** Manages transactions related to hotel operations, such as check-in, check-out, and room modifications.
v. **FeedbackTransactionManagement:** Coordinates transactions related to guest feedback and ratings.

Each subsystem is designed to handle specific functionalities, contributing to the overall efficiency and effectiveness of the Hotel Reservation System. The modular structure allows for easy maintenance, scalability, and future enhancements.

**System Component Diagram**


Check-In/ Check-Out Management
User Interaction Management
Reservation Management
Reservation Transaction Management
Hotel Management
Account Management
Statistics Management
Digital Key &Confirmation Management
Database

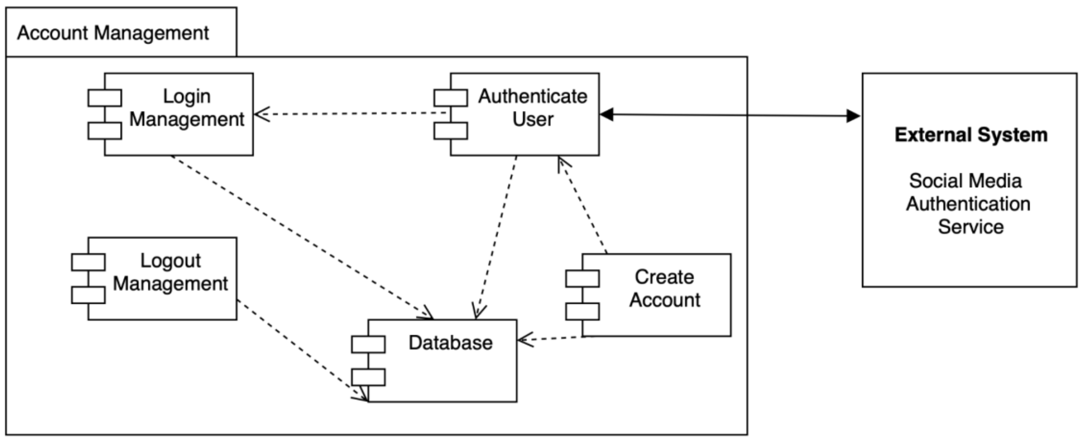
## Subsystem decomposition

Detailed subsystem decomposition for above system component diagram is as follows:

### Account Management

The Account Management subsystem is responsible for handling user accounts and authentication processes within the Hotel Reservation System. It includes several key components such as CreateAccountManagement, LoginManagement, AuthenticateUserManagement, and LogoutManagement. CreateAccountManagement facilitates the creation of new user accounts, while LoginManagement handles user login and authentication. AuthenticateUserManagement verifies user credentials and manages role-specific dashboards after successful authentication. LogoutManagement, on the other hand, manages user logout and session termination. Each of these components connects to the central database for user account information storage.

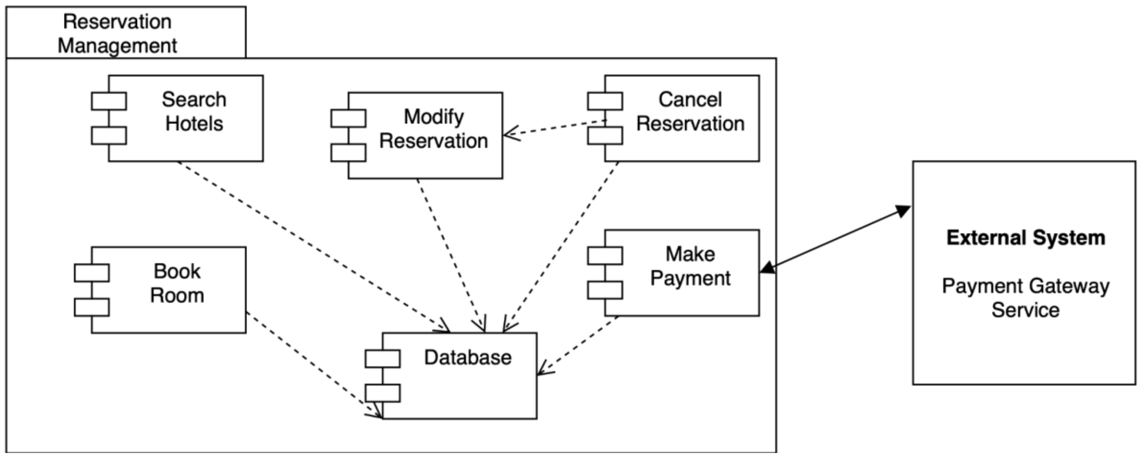
- **External System:** Social Media Authentication Service
  **Integration Purpose:** Allow users to authenticate using their social media credentials.



### Reservation Management

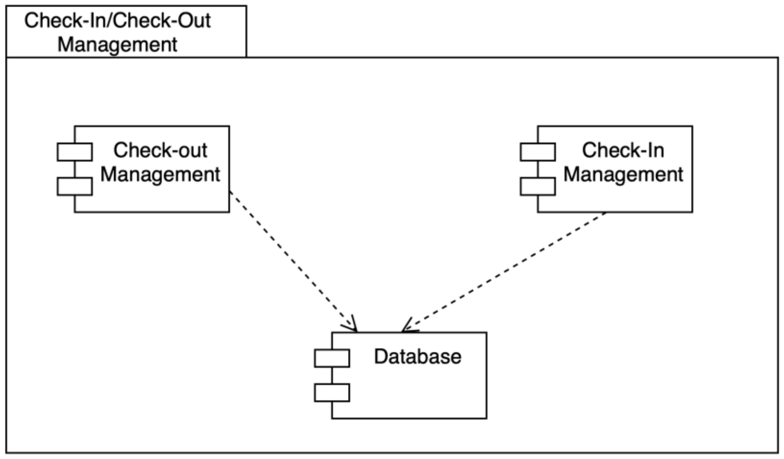
The Reservation Management subsystem focuses on handling hotel reservations, booking, and related processes. It comprises components such as SearchHotelsManagement, BookRoomManagement, MakePaymentManagement, ModifyReservationManagement, and CancelReservationManagement. SearchHotelsManagement is responsible for managing the hotel search interface and search result processing. BookRoomManagement handles the room booking and reservation confirmation, while MakePaymentManagement manages the payment process for booked rooms. ModifyReservationManagement allows users to update existing reservations, and CancelReservationManagement handles reservation cancellations. All these components interact with the central database to store and retrieve reservation-related information.

- **External System:** Payment Gateway Service
  **Integration Purpose:** Facilitate secure and efficient online payments for room bookings.



## Check-In/Check-Out Management Subsystem

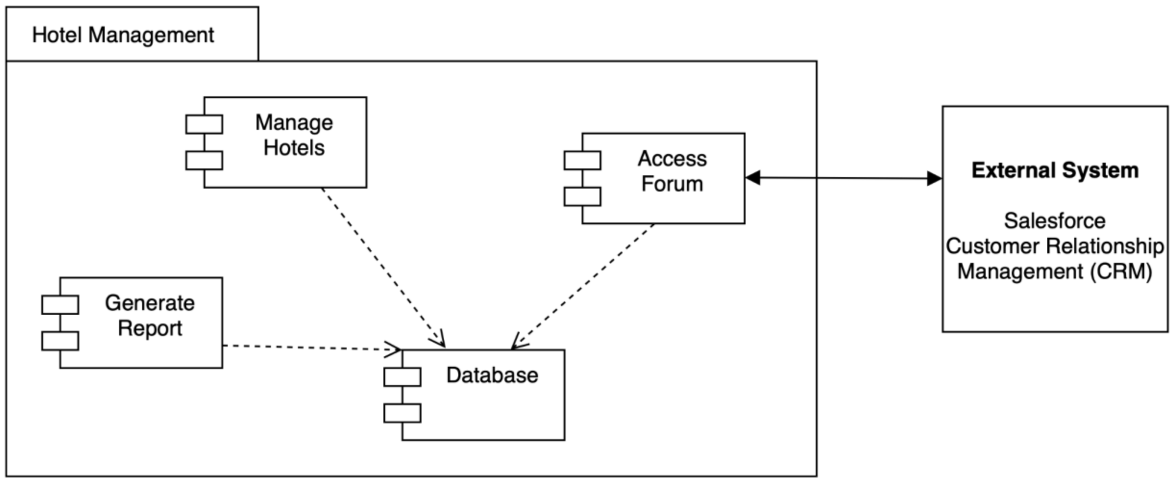
The Check-In/Check-Out Management subsystem ensures a smooth guest experience during check-in and check-out processes. It includes components like Check-InManagement and Check-OutManagement. Check-InManagement handles both online and front desk check-in processes, generating digital key cards or QR codes for room access. Check-OutManagement manages the check-out process, including payment and feedback. Both components connect to the central database for real-time room status updates, payment transactions, and guest feedback storage.



## Hotel Management Subsystem

The Hotel Management subsystem focuses on the management of hotels, including details and operational aspects. It includes components like ManageHotelsManagement, AccessForumManagement, and GenerateReportManagement. ManageHotelsManagement allows hotel staff to update information, upload images, and manage facilities, rooms, and rates. AccessForumManagement enables interaction through an online forum, and GenerateReportManagement generates various reports on hotel performance and reservation data. All these components connect to the central database for efficient hotel details management and data retrieval for reporting.

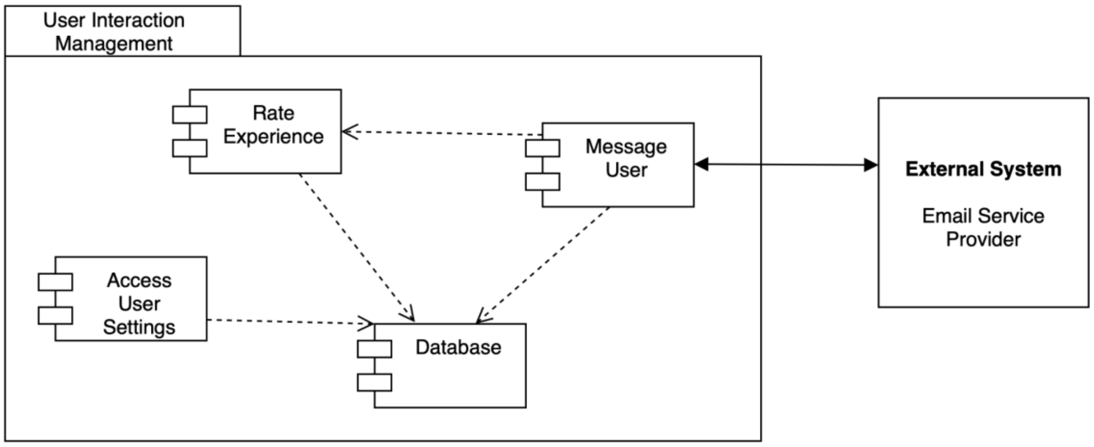
- **External System:** Salesforce Customer Relationship Management (CRM) System. **Integration Purpose:** Enhance customer interactions and maintain a centralized database of guest preferences and feedback.



## User Interaction Management Subsystem

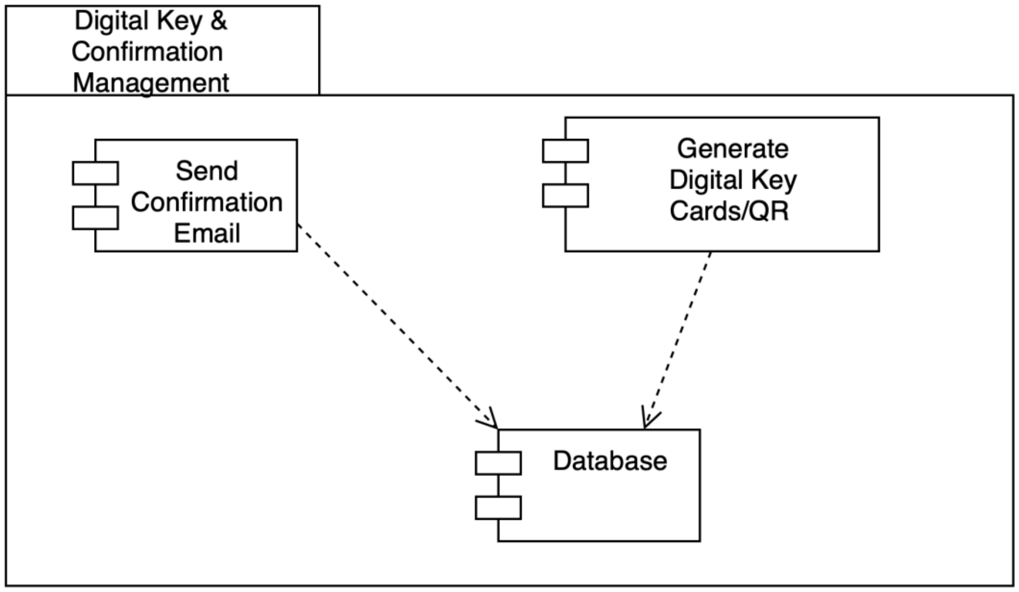
The User Interaction Management subsystem is responsible for managing user interactions and settings within the system. It includes components like MessageUserManagement, AccessUserSettingsManagement, and RateExperienceManagement. MessageUserManagement facilitates user messaging, AccessUserSettingsManagement allows users to customize preferences, and RateExperienceManagement manages the user feedback and rating process. All these components connect to the central database for storing user interactions, settings, and feedback.

- **External System:** Email Service Provider
  **Integration Purpose:** Ensure reliable and timely delivery of system-generated emails, such as confirmation emails and notifications.



## Digital Key and Confirmation Management Subsystem

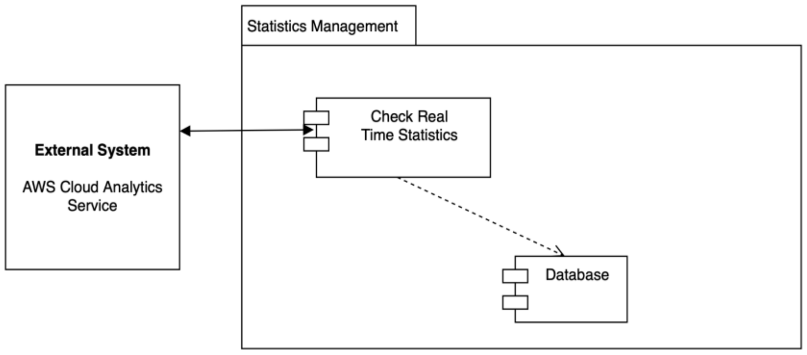
The Digital Key and Confirmation Management subsystem handle the generation and confirmation processes related to reservations and access. It includes components like GenerateDigitalKeyCards/QRManagement and SendConfirmationEmailManagement. GenerateDigitalKeyCards/QRManagement manages the generation of digital key cards or QR codes for guest access, while SendConfirmationEmailManagement handles the generation and sending of confirmation emails. Both components connect to the central database for storing digital key information and confirmation details.


Digital Key & Confirmation Management
Send Confirmation Email
Generate Digital Key Cards/QR
Database

## Statistics Management Subsystem

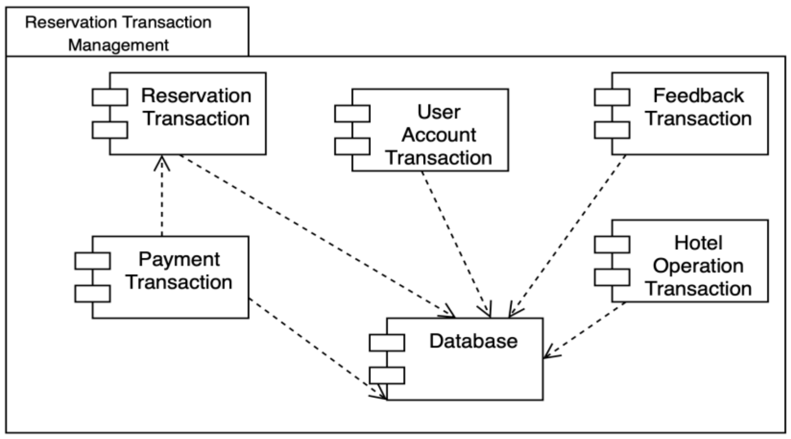
The Statistics Management subsystem provides real-time statistics and insights into the hotel's performance. It includes components like CheckRealTimeStatisticsManagement. This component connects to the central database to fetch real-time data on revenue, occupancy rates, and other key performance indicators, dynamically refreshing the statistics for user analysis.

- **External System:** AWS Cloud Analytics Service
  **Integration Purpose:** Enable advanced analytics and visualization tools for in-depth analysis of hotel performance.



## Reservation Transaction Management Subsystem

The Reservation Transaction Management subsystem serves as a centralized component coordinating and managing transactions across various modules. It includes components like ReservationTransactionManagement, PaymentTransactionManagement, UserAccountTransactionManagement, HotelOperationTransactionManagement, and FeedbackTransactionManagement. Each of these components connects to the central database to ensure atomicity, consistency, isolation, and durability (ACID properties) for transactions involving multiple subsystems.



## Hardware/software mapping

The Hotel Reservation System is designed with a three-tier architecture that facilitates a scalable, maintainable, and robust platform for hotel bookings and management. This architecture divides the system into three main layers: Presentation, Application, and Data Access Layers, each residing on designated hardware with specific software components.

**Hardware Configuration**

**Client Devices:** Users access the system via a web interface on various devices, including PCs, laptops, tablets, and smartphones. The system supports multiple operating systems (Windows, macOS, iOS, Android) and web browsers (Chrome, Firefox, Safari, Edge).

**Application Servers:** The business logic of the Hotel Reservation System operates on powerful, cloud-based virtual servers. These servers host the application developed in Java (for backend services) and React (for frontend services), ensuring high availability and scalability.

**Database Servers:** Oracle Database Servers are used for storing all system data, including user information, hotel data, reservations, and transactions. These servers ensure data integrity, performance, and reliable backup mechanisms.

**Software Components**

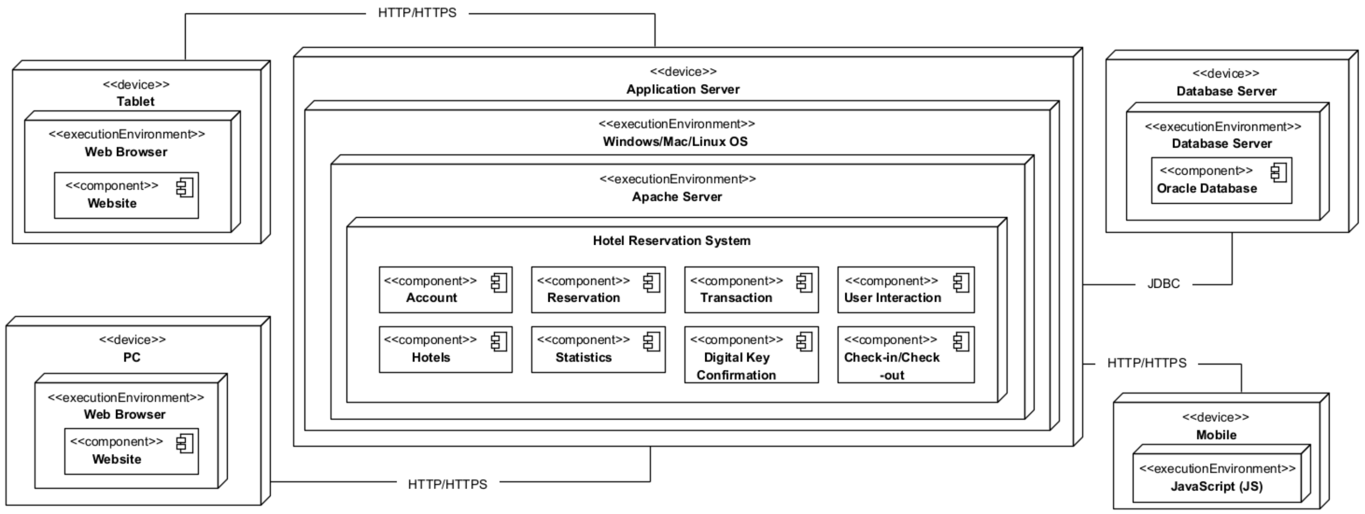
**Presentation/User Interface Layer:** Utilizes HTML5, CSS3, and JavaScript, with React for dynamic content rendering. This layer provides a seamless and responsive user experience across all client devices.

**Application Layer:** Developed in Java, this layer implements the core business logic, including account management, reservation processing, check-in/check-out operations, and integration with external services like payment gateways and Property Management Systems (PMS).

**Data Access Layer:** Manages data transactions with the Oracle Database, ensuring data consistency and reliability. It uses JDBC for database connectivity and SQL for data manipulation.

Deployment Diagram 1

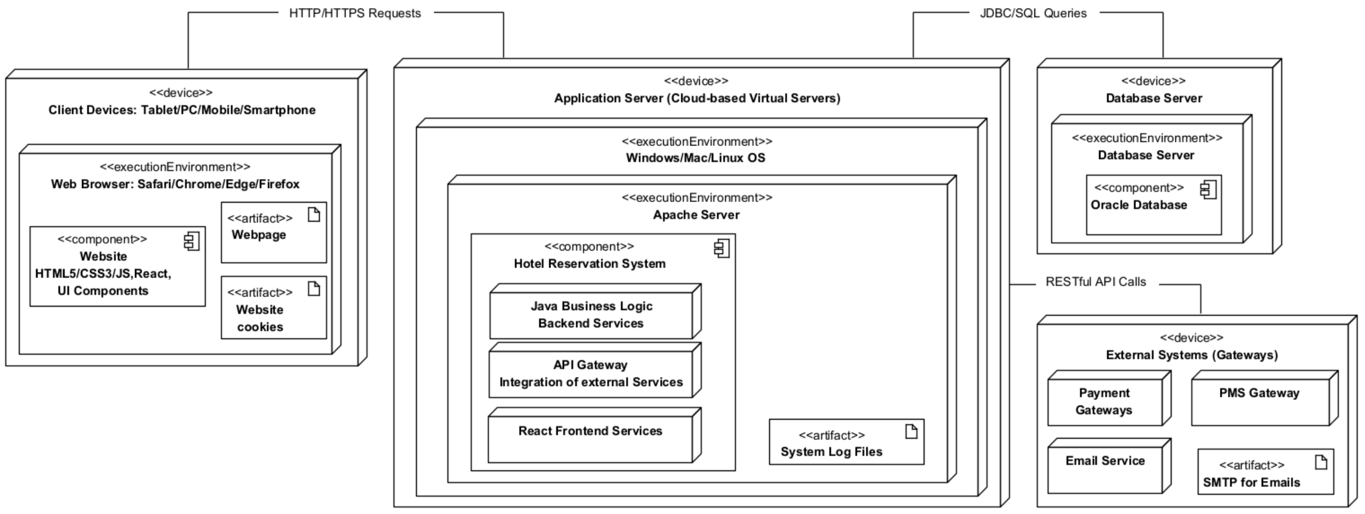
[Link to diagram]



HTTP/HTTPS
<<device>> Tablet
<<executionEnvironment>> Web Browser
<<component>> Website
<<device>> Application Server
<<executionEnvironment>> Windows/Mac/Linux OS
<<executionEnvironment>> Apache Server
Hotel Reservation System
<<component>> Account
<<component>> Reservation
<<component>> Transaction
<<component>> User Interaction
<<component>> Hotels
<<component>> Statistics
<<component>> Digital Key Confirmation
<<component>> Check-in/Check -out
<<device>> Database Server
<<executionEnvironment>> Database Server
<<component>> Oracle Database
JDBC
HTTP/HTTPS
<<device>> Mobile
<<executionEnvironment>> JavaScript (JS)
<<device>> PC
<<executionEnvironment>> Web Browser
<<component>> Website
HTTP/HTTPS

Deployment Diagram 2 (External Interactions)

[Link to diagram]



HTTP/HTTPS Requests
JDBC/SQL Queries
<<device>> Client Devices: Tablet/PC/Mobile/Smartphone
<<executionEnvironment>> Web Browser: Safari/Chrome/Edge/Firefox
<<component>> Website HTML5/CSS3/JS,React, UI Components
<<artifact>> Webpage
<<artifact>> Website cookies
<<device>> Application Server (Cloud-based Virtual Servers)
<<executionEnvironment>> Windows/Mac/Linux OS
<<executionEnvironment>> Apache Server
<<component>> Hotel Reservation System
Java Business Logic Backend Services
API Gateway Integration of external Services
React Frontend Services
<<artifact>> System Log Files
<<device>> Database Server
<<executionEnvironment>> Database Server
<<component>> Oracle Database
RESTful API Calls
<<device>> External Systems (Gateways)
Payment Gateways
PMS Gateway
Email Service
<<artifact>> SMTP for Emails

---

These deployment diagrams showcase the hardware and software mapping of the hotel reservation system. On the user side, web pages are accessed via browsers like Edge, Safari, or Chrome, depending on the client's device (PC, mobile, or tablet). The system's core business logic runs on the application server, handling and processing all client requests. The application server interfaces with an Oracle database to manage and fulfill client data requests efficiently. This architecture ensures seamless communication and operation between the client-side interface and the server-side application logic, enhancing overall system performance and user experience.

The possible issues with the multiple nodes are:

1. The complexity is increased when communication and flow of data is managed among numerous hardware nodes.
2. The failure in one node impacts the entire system functionality.
3. It gets challenging to integrate and ensure the compatibility between the off-shelf components of the software.

**Additional details:**
**Client Devices:** This is where the system's users, including hotel guests, staff, and online travel agents, interact with the Hotel Reservation System. They use a web browser to access the system's user interface. The UI is built using technologies like JavaScript, HTML, and CSS, with React providing a dynamic and responsive experience.

**Application Server:** The core logic of the Hotel Reservation System resides here, on a cloud-based virtual machine that ensures scalability and reliability. The server runs business logic implemented in Java, handling processes such as account management, reservation handling, and communication with the database. The API Gateway component, also developed in Java, facilitates secure and structured communication with external APIs, like payment gateways and property management systems. Additionally, the Application Server (Cloud VM) hosts Apache Server, which acts as the web server software responsible for serving the Hotel Reservation System's frontend and possibly some backend components. Apache Server efficiently handles HTTP/HTTPS requests from clients, ensuring smooth communication between users and the system. This setup allows for seamless interaction with the web interface across various client devices and browsers.

**Database Server (Oracle DB):** This server hosts the Oracle Database where all system data is stored. The database's design ensures data integrity, performance, and security, serving as the backbone for the system's data management.

**External Systems:** These are third-party services that the Hotel Reservation System integrates with to extend its functionalities. The Payment Gateway API allows for secure processing of financial transactions. The Email Service API enables the system to send confirmation emails and other notifications to users. Integration with Property Management Systems (PMS) through the PMS API allows for seamless synchronization of hotel operations and reservations data.

## Persistent data management

For the Hotel Reservation System, we have opted to store persistent data in a relational database management system due to its structured format and ability to handle complex relationships between data entities. To achieve this, we have chosen the most recent version of the Oracle Database Management System (DBMS). Oracle DBMS provides a robust relational structure, allowing data to be stored in tabular formats efficiently. The use of Oracle Database Server as our database platform provides scalability and performance to handle high-volume records efficiently, meeting the current and future needs of our Hotel Reservation System.

A relational database allows us to create, retrieve, update, and manage persistent data in a structured and organized manner. Persistent objects will be stored as tabular relation whilst each row in the table represents a record. Each entity object identified in our system will be mapped to corresponding tables in the Oracle database, ensuring a structured and organized storage approach. All many-to-many relationships will be converted to associative entities/tables/relations in relational schema. Primary-to-Foreign key constraints will be used in the database to describe one-to-many relationships between entities.

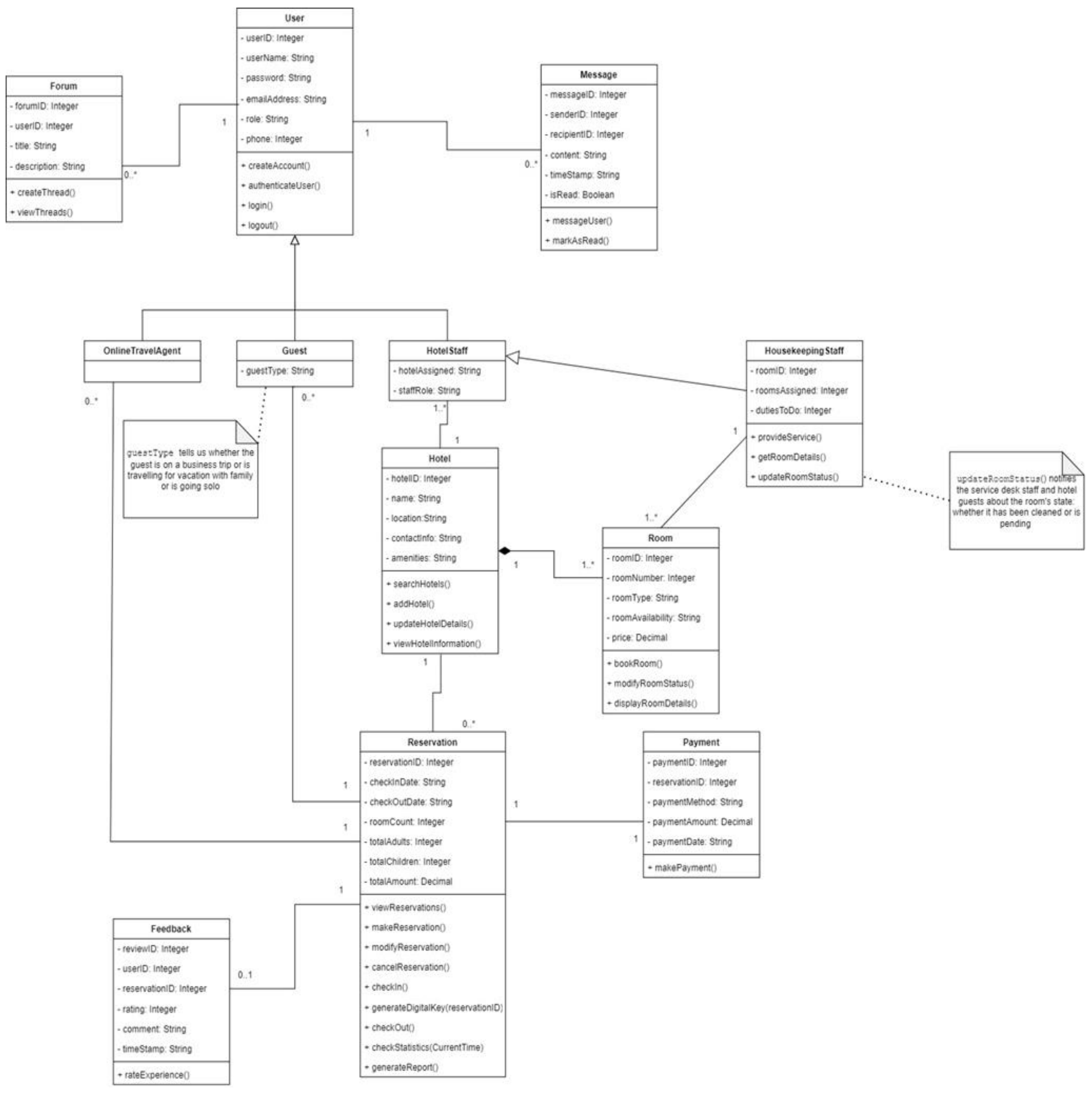
Persistent data will be backed up on a regular basis to avoid data loss caused by hardware failures, software problems, or other unanticipated circumstances. Regular backups, including complete, incremental, and snapshot-based backups, will be scheduled to maintain data integrity and availability. Additionally, a secondary Oracle database server has been established to serve as a backup server. This backup server operates in parallel with the primary database server, providing fault tolerance in the event of a main server failure. In our Hotel Reservation System, all entity objects and their data instances will reside in the Oracle database. The following entity objects constitute the core entities of our system:

- User (with all subtypes)
- Hotel
- Room
- Reservation
- Payment
- Feedback
- Message
- Forum

These fundamental entities for our Hotel Reservation System will be kept on the central Oracle database server. The visual representation of the system's structure is shown below, which includes the core system's classes, attributes, methods, and relationships between classes.

Hotel Reservation System Class Diagram

Link to the class diagram



## Access control and security

This table outlines the access control rules for different actors within the system, mapping them to the subsystems and their respective operations.

### Access Matrix

| **Actor** | **Subsystem** | **Operations** |
|---|---|---|
| Guest | AccountManagement | CreateAccount()<br>Login() |
| | ReservationManagement | SearchHotels()<br>BookRoom()<br>ModifyReservation()<br>CancelReservation() |
| | Check-In/Check-OutManagement | OnlineCheck-In()<br>OnlineCheck-Out() |
| | UserInteractionManagement | RateExperience()<br>AccessUserSettings() |
| | DigitalKey&ConfirmationManagement | GenerateDigitalKey() |
| HotelStaff | *InheritsGuest* | *All Guest Operations* |
| | StatisticsManagement | CheckStatistics() |
| | HotelManagement | ManageHotels()<br>GenerateReport()<br>AccessForums() |
| | UserInteractionManagement | MessageUser() |
| | DigitalKey&ConfirmationManagement | SendConfirmationEmail() |
| OnlineTravelAgency | AccountManagement | CreateAccount()<br>Login() |
| | ReservationManagement | SearchHotels()<br>BookRoom()<br>ModifyReservation()<br>CancelReservation() |

|  |  |  |
|---|---|---|
|  | DigitalKey&ConfirmationManagement | SendConfirmationEmail() |
|  | UserInteractionManagement | AccessUserSettings() |
| SystemAdministrator | AccountManagement | CreateAccount()<br>Login() |
|  | ReservationManagement | SearchHotels()<br>BookRoom()<br>ModifyReservation()<br>CancelReservation() |
|  | HotelManagement | ManageHotels()<br>GenerateReport()<br>AccessForums() |
|  | UserInteractionManagement | MessageUser()<br>AccessUserSettings() |
|  | DigitalKey&ConfirmationManagement | GenerateDigitalKey() |
|  | StatisticsManagement | CheckStatistics() |

## Access Control Rules

1. **Guest:**
    - Can create an account.
    - Can log in.
    - Can search for hotels.
    - Can view hotel information.
    - Can book a room.
    - Can check-in and check-out.
    - Can modify their reservation.
    - Can cancel their reservation.
    - Can rate their experience.
    - Can access user settings.
    - Can generate a digital key after check in.

2. **HotelStaff:**
    - All permissions of the Guest.
    - Can manage hotels (update information, upload images, manage facilities).

    - Can generate reports.
    - Can send messages to users (communication within the application).
    - Can check real-time stats.
    - Can send confirmation emails.
    - Can access forums.

3. **OnlineTravelAgency:**
    - Can create an account.
    - Can log in.
    - Can search for hotels.
    - Can view hotel information.
    - Can book a room.
    - Can modify the reservation.
    - Can cancel the reservation.
    - Can access user settings.
    - Can send confirmation email.

4. **SystemAdministrator:**
    - Can create an account.
    - Can log in.
    - Can search for hotels.
    - Can view hotel information.
    - Can book a room.
    - Can modify/cancel any reservation.
    - Can manage hotels (update information, upload images, manage facilities).
    - Can generate reports.
    - Can send messages to users (communication within the application).
    - Can access user settings.
    - Can generate a digital key after check in.
    - Can check real-time stats.
    - Can access forums.

## Security Measures

In addition to the access controls, two varying levels will manage the security controls.

- One will be on the entire system level, ensuring protection from hacking attempts, Denial of Service attacks, etc.
- Another level of security will be provided by subsystem for its functionality:
    - Subsystem <Account Management> is responsible for the password encryption, authorization, and authentication functionality for the user.

    - Subsystem <Reservation Management> is responsible for the secure payment gateway service by integrating with the external Payment Gateway Service.

**Additional Safety Features**

- Authentication Mechanism:
    - Robust authentication mechanism.
    - Multi-factor authentication for administrators.
- Encryption:
    - Use of encryption when transferring and storing sensitive data.
    - Use of HTTPS to communicate securely.
- Key Management:
    - Solid key control methods for encryption keys.
    - Continually refresh and cycle through encryption keys.
- Access Logging:
    - Record every single access attempt and notable system actions.
    - Examine entry recordings for security breaches on a weekly basis.
- User Permissions Review:
    - Based on roles and responsibilities, review, and update user permissions every three months.

## Global control flow

To achieve global software control in the system, an event-driven approach will be used. This approach enables various parts of the system to react independently to events, enhancing efficiency in handling requests. It also reduces the need for synchronization and effectively minimizes concurrency conflicts.

**Account Management Subsystem:**

- System Administrators or Database Administrators establish the first administrator (admin) accounts.
- These admin users have special permissions for managing the system.
- Admin Users can set up more admin users or regular users, like hotel employees or visitors.

**Guest Interaction**

- Guests access the booking system using their web browsers or mobile apps.
- They can search for available rooms based on specific requirements (e.g., dates, location, facilities).
- After selecting their preferred room, guests can submit a reservation request.

- The system processes the request and confirms the booking in real time.

**Event-Driven Control Flow:**

- The system responds to different activities, like user actions, outside API calls, or system alerts.
- Various parts of the system pay attention to specific events they want to handle by setting up listeners or handlers for those events.
- For instance, when a user tries to make a reservation, a "ReservationRequested" event is triggered, which is handled by Reservation Management subsystem.

**Request Initiation:**

- Users initiate requests (e.g., reservations, bookings, availability checks) that trigger events.
- These events activate designated subsystems (eg. Reservation Management) to process and respond to the requests.
- For instance, a reservation request triggers an event processed by the Reservation Management system, which then updates the database, as necessary.

**Subsystem Synchronization:**

- Even though subsystems function independently, there are situations where coordination is crucial, particularly when accessing shared data.
- To maintain data integrity and prevent conflicts, subsystems will use tools like locks, semaphores, and mutexes. These tools guarantee that only one subsystem can access shared resources at a time.
- For instance, when multiple subsystems need to update reservation information at the same time, a locking mechanism can be used to ensure that the data remains consistent.

**Concurrency Issues:**

- When many parts of a system try to use or change the same parts at the same time, problems with concurrency can happen.
- To stop this, the system is planned and worked together carefully.
- Event-driven systems, like ours, make it less likely that concurrency problems will happen because they let different parts of the system work separately and deal with events as they come in.
- When needed, to make sure that data is accurate and complete, the system also has the right ways to keep things in sync.

**Error Handling**

- Subsystems have built-in ways of dealing with errors and failures smoothly.

- Errors can come in various forms, like error events. These are handled by specific systems that record the errors, notify the responsible parties, and start processes to fix the issues.

**Data Access and Storage**

- The Data Access Layer manages storing, retrieving, and searching for data objects in the database, ensuring efficient data management.
- The Database Server keeps data secure and accessible, allowing for high-volume transactions and up-to-date information in real time.

## Boundary conditions

It is essential to set boundaries for starting, ending, and handling errors in the Hotel Reservation System to guarantee its stability and resilience.

**Start-up Behaviour**

- The Application Layer starts up first.
- The Data Access Layer and Presentation Layer initialize.
- The system connects to the Database Server.
- The system administrators and hotel staff can log in to use the administrative features.
- The system then sets up connections with OTAs to make sure reservation information is up-to-date.
- Finally, customers can use the front-end interfaces to look for and book rooms.

**Shutdown Behaviour**

- Users currently logged in are signed out.
- Connections to the database are closed safely.
- In-progress transactions are either finished or reversed to prevent data loss.
- Connections with other systems are ended.
- Resources are released correctly.
- A log is updated to record the shutdown and any problems that need attention.

**Error Behavior**

- Database Connection Errors: When the system cannot connect to the Database, it asks administrators to check database settings and restart services if needed.
- Reservation Processing Errors: If there are issues with reservations, like overbooking or room conflicts, the system alerts administrators and requires manual actions to resolve them.

- External Integration Errors: If the system encounters errors while connecting to external sources (e.g., OTAs), it retries the connection and records the failure for further investigation.
- User Authentication Errors: When users enter invalid login information, the system gives them guidance and lets them try again or change their passwords.
- Every part of the system has ways to handle errors that record and tag them, making sure they are dealt with and fixed as needed. When there are major errors, administrators are notified by email or system alerts so they can fix them and keep the system running smoothly.

## Glossary

- **User**: Represents individuals interacting with the hotel reservation system. Subtypes may include several types of users such as guests, hotel staff, and online travel agents, each with specific roles and permissions within the system. Hotel staff also includes several types of hotel staff such as cleaning staff, administrator, receptionist, manager and so on.
- **Hotel:** Represents an establishment available for reservations.
- **Room**: Represents individual units within a hotel available for reservation
- **Reservation**: Represents a booking made by a user for a specific room at a particular hotel for a defined period
- **Payment**: Represents transactions made to confirm and complete reservations
- **Feedback**: Represents ratings, reviews and feedback provided by guests after their stay at the hotel.
- **Message:** Represents communication exchanged between users (guests, online travel agents, hotel staff) within the system.
- **Forum**: Represents discussion boards or groups within the system where users can engage in conversations, share information, and ask questions related to hotel reservations and experiences.