SOFTWARE DESIGN & ARCHITECTURE

Project: Hotel Management System

NAME	MUHAMMAD ROHAAN KHAN
CLASS	BSSE
SECTION	B
ROLL NO	*BSSE / Sp-22 / 037*
TEACHER	SIR OMER AFTAB

❖ Description:

The hotel industry is a highly competitive market, and hoteliers must find ways to provide excellent customer service while keeping their costs low. One of the ways to achieve this is through the implementation of a hotel management system. A hotel management system is a software solution that can help hotels automate and streamline their daily operations. In this project the details are maintained like customer details, reservation details, booking details and billing details The reservation process of reserving rooms for the customers, canceling the reserved rooms, booking the rooms, vacating the rooms, the restaurant management, billing process etc. All are computerized and the management is done without any difficulty. The reports can be viewed completely and the head of the management daily or weekly or monthly can review it. For company auditing it will be more useful. This Proposed System will be interactive, faster and user-friendly for the end users. A hotel management system (HMS) typically contains of various parts that can be customized to meet the specific needs of the hotel. The modules can include reception management, housekeeping management, accounting, inventory management, and guest relationship management. The modules work together to provide a seamless and efficient hotel operation. Hotel management systems have been around for many years, but with advances in technology, they have become more experienced and customizable to fulfill the specific needs of individual hotels. Today, hotel management systems are important tools for hotel operators to manage their works and provide guests with a memorable & luxury experience.

❖ Types of Requirements:

1. Functional Requirements:

Room Management:

1. Add, update, or delete room details (type, price, availability).
2. Real-time room availability tracking.

Reservation System:

1. Book, modify, and cancel room reservations.
2. Check reservation status.
3. Automatic assignment of rooms during booking.

Guest Management:

1. Store guest details (name, contact info, ID proof).
2. Check-in and check-out functionality.
3. Generate invoices for guests.

Billing and Payments:

1. Generate bills for services (room charges, food, additional services).
2. Support multiple payment methods (credit card, UPI, cash).
3. Automatic tax calculation.

Employee Management:

1. Add, update, and remove employee records.
2. Assign tasks to employees (room service, cleaning, etc.).

Service Requests:

1. Allow guests to request services (room service, laundry, etc.) through the system.
2. Track and update the status of service requests.

2. Non-Functional Requirements:

Scalability:

1. The system should handle increased user loads during peak seasons.
2. Support multiple branches of the hotel.

Performance:

1. System should process queries (e.g., availability check) in under 2 seconds.
2. Handle concurrent access by multiple users without lag.

Security:

1. Protect guest data with encryption.
2. Role-based access for administrators, staff, and guests.
3. Secure payment processing using PCI-DSS standards.

Availability:

1. Ensure 99.9% uptime.
2. Provide automatic failover to backup servers in case of a system crash.

Usability:

1. Simple and intuitive UI for hotel staff, guests, and administrators.
2. Multi-language support.

3. Business Requirements:

Increase Revenue:

1. Automate bookings to reduce errors and overbookings.
2. Promote additional services like spas, tours, or dining.

Cost Efficiency:

1. Minimize operational costs by automating repetitive tasks.
2. Reduce paperwork by digitizing records.

Customer Satisfaction:

1. Provide a seamless booking and service experience.
2. Enable guests to access services easily through a mobile app or website.

Regulatory Compliance:

1. Adhere to data protection laws like GDPR or CCPA.
2. Maintain records for audits and government regulations.

4. User Requirements:

Guests:

1. Book rooms and services online or in person.
2. View room availability and pricing.
3. Access their stay history and invoices.

Hotel Staff:

1. Manage daily tasks such as check-ins, check-outs, and room allocations.
2. Monitor service requests and update their status.
3. Use dashboards to view current occupancy and pending tasks.

Administrators:

1. View reports on occupancy, revenue, and employee performance.
2. Manage hotel details, pricing strategies, and employee roles.
3. Ensure the system's security and maintenance.

❖ **Software Process Models:**

Implementation of Incremental Model:

1. **Increment 1**: Basic Room Reservation System
   - Features: Room availability checking, booking, and cancellations.
   - Users: Front desk staff and guests.
2. **Increment 2**: Guest Management and Check-in/Check-out
   - Features: Guest details, automated check-in/check-out processing, and key assignment.
3. **Increment 3**: Billing and Payment System

- Features: Invoice generation, payment gateway integration, and receipt management.

4. **Increment 4**: Employee Scheduling and Task Management
   - Features: Employee roster creation, task assignment, and attendance tracking.
5. **Increment 5**: Service Requests and Guest Feedback
   - Features: In-room service requests, housekeeping notifications, and guest feedback collection.
6. **Increment 6**: Advanced Features and Analytics
   - Features: Real-time occupancy analytics, dynamic pricing suggestions, and loyalty programs.

❖ **Software Architectural Style:**

Distributed Machine Architecture:

a. Microservices:

- **Room Reservation Service**: Manages room availability, bookings, and cancellations.
- **Guest Management Service**: Handles guest check-ins/outs, profiles, and histories.
- **Billing and Payment Service**: Handles billing, invoicing, and payment processing.
- **Employee Scheduling Service**: Manages employee shifts, schedules, and work assignments.
- **Service Request Service**: Handles requests from guests for services like room cleaning, maintenance, etc.

b. API Gateway:

An **API Gateway** can be used as a single-entry point for clients (e.g., mobile apps, web apps) to interact with the distributed microservices. The API Gateway handles routing, authentication, and load balancing.

c. Message Brokers:

Communication between microservices can be asynchronous using **message brokers** (e.g., RabbitMQ, Kafka). For example, if a guest requests a service, the Service Request Service can publish an event to a message queue that other services (e.g., the Room Reservation Service or Maintenance Service) listen to.

d. Data Storage:

Each microservice will have its own **database** (Database per Service pattern) to avoid tight coupling. For example:

- Room Reservation Service can use a **NoSQL** database for flexibility and quick access.
- Guest Management Service can use a **Relational Database** (e.g., MySQL) for structured guest information.
- Billing and Payment Service might use a **transactional database**.

This approach ensures that the failure of one service does not affect the others, making the system more fault-tolerant.

e. Load Balancing:

Each service can be replicated across multiple instances to handle varying loads. A **load balancer** ensures that requests are evenly distributed across these instances.

f. Service Discovery:

In a distributed environment, services must find each other. Tools like **Consul** or **Eureka** can be used for **service discovery** so that each microservice knows how to find the others in real-time.

❖ **Software Architectural Patterns:**

The Microservices Architecture is one of the best architectural styles for a Hotel Management System (HMS).

1. **Services Breakdown**:

- **Room Booking Service**: Handles availability, reservations, cancellations, and updates.
- **Guest Management Service**: Maintains guest profiles, preferences, and history.
- **Billing and Payment Service**: Manages invoices, payments, refunds, and taxes.
- **Employee Management Service**: Handles scheduling, payroll, and task assignments.
- **Housekeeping Service**: Tracks room cleaning schedules and maintenance requests.
- **Notification Service**: Sends emails or SMS for confirmations, reminders, and promotions.

2. **Communication**:

- Services communicate through APIs or an event-driven messaging system (e.g., RabbitMQ, Kafka).
- Example: When a room is booked, the "Room Booking Service" triggers events to update the "Billing Service" and "Housekeeping Service."

3. **Data Management**:

- Each service has its own database (database per service pattern), ensuring independence and scalability.
- Example: "Room Booking Service" might use an SQL database for relational data, while "Notification Service" uses a NoSQL database for managing messages.

4. **Deployment**:

- Services can be deployed as containers (e.g., using Docker) and orchestrated using Kubernetes, enabling smooth scaling and monitoring.

❖ **Software Quality Attributes:**

1. Scalability:

- **Explanation**: The system must be able to handle growth in terms of the number of guests, bookings, and data volume. As a hotel expands or experiences seasonal fluctuations in guest traffic, the system should scale without performance degradation.
- **Importance**: A scalable HMS can accommodate the needs of both small and large hotels without requiring a complete redesign.

2. Security:

- **Explanation**: The system must protect sensitive data, such as guest information, payment details, and internal hotel data, from unauthorized access or breaches. This includes encryption, secure login methods, and data access control mechanisms.
- **Importance**: Ensures the safety of guests' personal and payment data and helps comply with privacy laws and regulations (e.g., GDPR, PCI DSS).

3. Performance:

- **Explanation**: The system should provide fast response times and ensure smooth operations, even under heavy load, such as when many guests are trying to make reservations or request services simultaneously.
- **Importance**: A responsive system contributes to a better user experience, both for guests and hotel staff, and ensures critical tasks (e.g., booking, check-in) are performed efficiently.

4. Availability:

- **Explanation**: The HMS should be available at all times, especially during peak periods like check-in/check-out hours. High availability ensures the system is operational even in case of hardware or software failures.
- **Importance**: Guests and staff depend on the system to perform essential functions, and any downtime could lead to frustration or lost revenue.

5. Usability:

- **Explanation**: The system's user interface should be intuitive and easy to use for hotel staff and guests. This includes clear navigation, simple booking processes, and easy access to information for staff (e.g., room availability, guest requests).
- **Importance**: A user-friendly system reduces training time, increases efficiency for staff, and ensures that guests can easily make bookings or manage their reservations.

6. Reliability:

- **Explanation**: The system should consistently perform as expected without unexpected failures. It should be able to recover quickly from errors, with minimal disruption to operations.
- **Importance**: Reliability is critical to maintain trust with users and ensure smooth day-to-day operations.

7. Maintainability:

- **Explanation**: The system should be easy to maintain, with clear documentation, modular code, and the ability to fix bugs or add new features without affecting other parts of the system.
- **Importance**: A maintainable system helps ensure long-term viability and minimizes the cost and time spent on updates or fixes.

8. Interoperability:

- **Explanation**: The HMS should be able to communicate with other systems, such as payment gateways, third-party booking platforms, and internal accounting software.
- **Importance**: Ensures seamless integration and data sharing across platforms, improving the overall efficiency and effectiveness of hotel operations.

9. Extensibility:

- **Explanation**: The system should allow for easy addition of new features or modules, such as loyalty programs, additional room services, or integration with new technologies (e.g., AI-driven chatbots).
- **Importance**: Extensibility ensures that the system can adapt to changing business needs and incorporate future advancements in technology.

10. Data Integrity:

- **Explanation**: The system must ensure the accuracy and consistency of data, particularly with regards to reservations, payments, and guest details. This can include checks for duplicate records or ensuring that all data is updated in real-time.
- **Importance**: Data integrity is crucial for maintaining accurate information and preventing issues such as double-bookings or incorrect billing.

❖ **Software Trade-Off's:**

1. Scalability vs. Performance

- **Scalability:** Building the system to handle increasing numbers of guests, rooms, and transactions as the hotel grows.
- **Performance:** Ensuring fast response times for operations like bookings, check-ins, and billing, which can be challenging when the system is scaled up.

Trade-off:

To achieve scalability, you may need to introduce additional layers like load balancing or sharding the database. However, these solutions can introduce additional complexity, which may affect the performance and response times. Optimizing performance might require minimizing the system's ability to scale seamlessly without some trade-offs in real-time processing.

2. Security vs. Usability

- **Security:** Ensuring that guest information (personal details, payment data, etc.) is well protected through encryption, multi-factor authentication, and role-based access control.
- **Usability:** Ensuring the system is easy to use for hotel staff, administrators, and guests (e.g., intuitive UI/UX for booking).

Trade-off:

Higher security may result in more complex user interfaces (e.g., additional authentication steps), which can make the system harder to use. Conversely, prioritizing ease of use may leave certain security vulnerabilities if not carefully implemented.

3. Real-time Updates vs. System Load

- **Real-time Updates:** For seamless guest experience (e.g., showing real-time availability of rooms, processing instant bookings).

- **System Load:** Real-time systems often place a high load on the server infrastructure, especially when dealing with a large number of simultaneous users.

Trade-off:

To ensure real-time updates, you might need to implement more sophisticated architectures, like event-driven systems or use in-memory databases, but this increases the system load and complexity. If you opt for less real-time interaction, it may reduce load but compromise the user experience.

4. Monolithic vs. Microservices Architecture

- **Monolithic Architecture:** All components of the HMS (e.g., room bookings, billing, guest management) are tightly integrated into one application.
- **Microservices Architecture:** Components are split into smaller, independently deployable services.

Trade-off:

A monolithic approach is easier to develop and deploy initially but may become harder to maintain as the system grows. Microservices offer better scalability, resilience, and flexibility but come with additional complexity in managing and deploying distributed systems.

5. Cloud vs. On-premises Deployment

- **Cloud Deployment:** Scalable infrastructure managed by a cloud service provider (e.g., AWS, Azure).
- **On-premises Deployment:** Servers and infrastructure are managed by the hotel’s IT team.

Trade-off:

Cloud deployment offers scalability, easier maintenance, and cost-effectiveness for smaller hotels, but may involve ongoing operational costs. On-premises deployment provides more control over data but can lead to higher initial investment and more complex maintenance.

6. Comprehensive Features vs. Simplicity

- **Comprehensive Features:** Providing a wide range of features (room service requests, online payments, billing history, guest loyalty programs, etc.).
- **Simplicity:** Focusing on core features to avoid complexity and ensure a smooth user experience.

Trade-off:

Adding more features makes the system more complex, requiring more development time, testing, and maintenance. It could also result in a more cluttered user interface, making it harder for staff to use. On the other hand, limiting features may mean that the hotel misses out on some opportunities for growth or improved customer experience.

7. Data Consistency vs. Availability (CAP Theorem)

- **Data Consistency:** Ensuring that all parts of the system (e.g., room availability, reservations) reflect the same data in real-time.
- **Data Availability:** Ensuring the system is always available for operations, even during high traffic or system failures.

Trade-off:

Achieving strict consistency might cause delays or make the system more prone to downtime. On the other hand, prioritizing availability might result in slight inconsistencies in data (e.g., booking the same room by two different guests), though these can often be managed with techniques like eventual consistency or conflict resolution.

8. Cost of Development vs. Future Maintenance

- **Cost of Development:** Opting for faster, more cost-effective development using simpler technologies or frameworks.
- **Future Maintenance:** Building with a focus on future scalability, extensibility, and easier maintenance (e.g., using design patterns, clean code).

Trade-off:

Developing quickly with simpler solutions may save time and money upfront but could result in a more difficult-to-maintain system in the long run. A more sophisticated development process might increase costs but pay off by reducing maintenance time and future technical debt.

❖ **Software Design Principal:**

Single Responsibility Principle (SRP):

Applying SRP to the Hotel Management System:

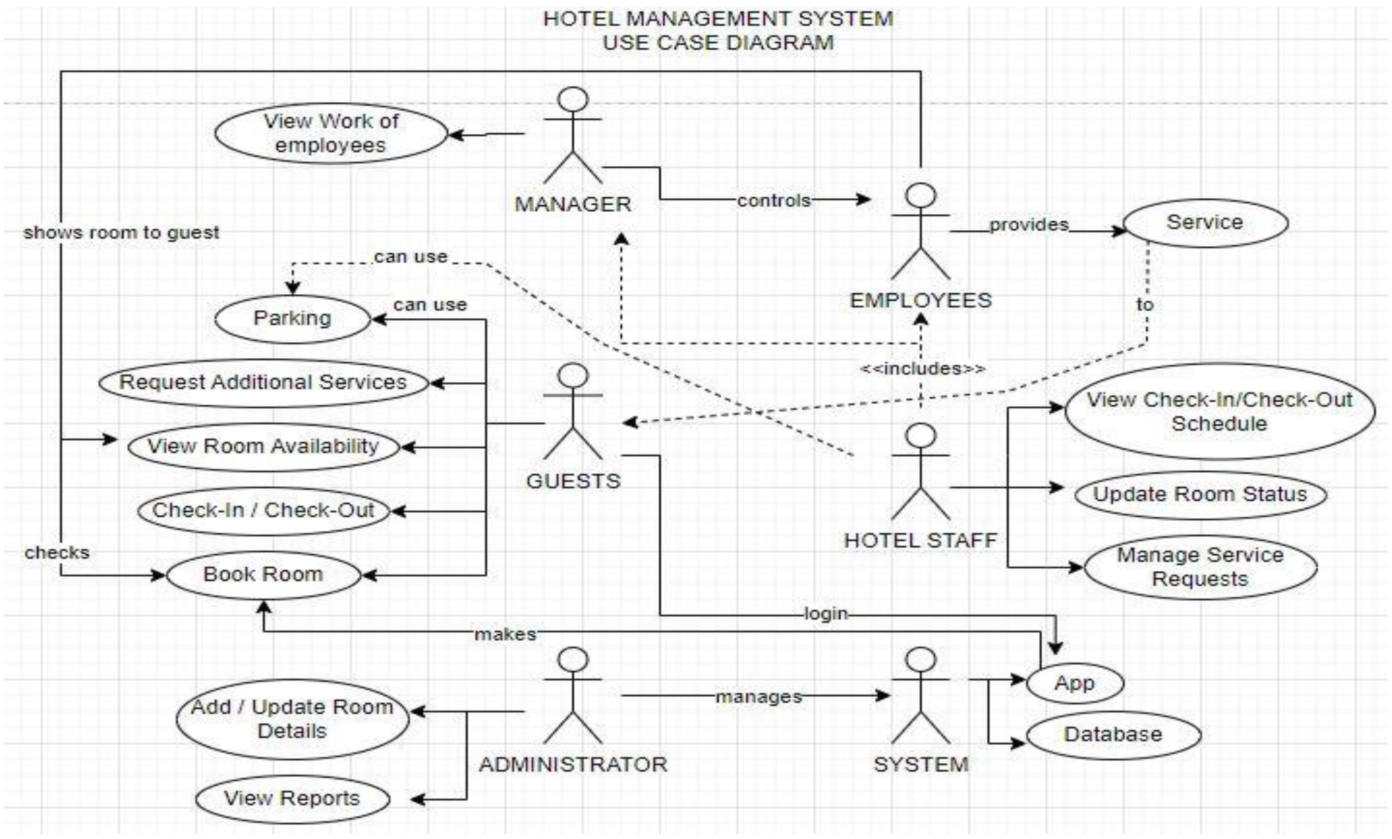
1. **Separation of Concerns**: Break down the HMS into smaller components, each with a single responsibility.
    a. **Room Reservation**: A class specifically handles the booking process, including availability checking and reservation creation.
    b. **Guest Management**: A class dedicated to managing guest details, such as check-in/check-out, personal information, etc.
    c. **Billing**: A class that takes care of invoicing, payments, and generating bills.
    d. **Employee Scheduling**: A class to manage employee shifts, scheduling, and payroll.

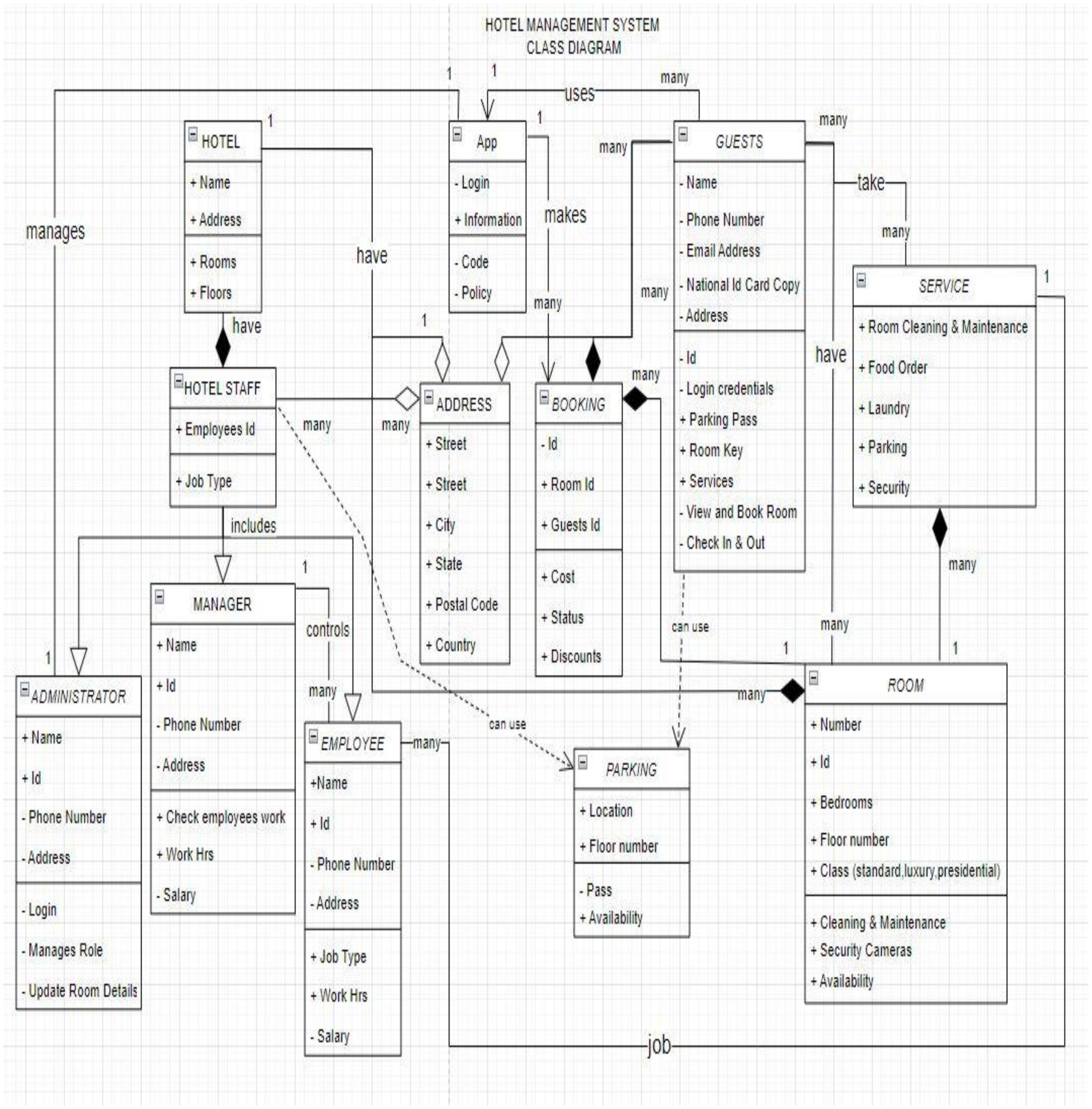
2. **Clear and Manageable Code**: By following SRP, each class is focused on a specific function of the HMS, making it easier to maintain, test, and extend.

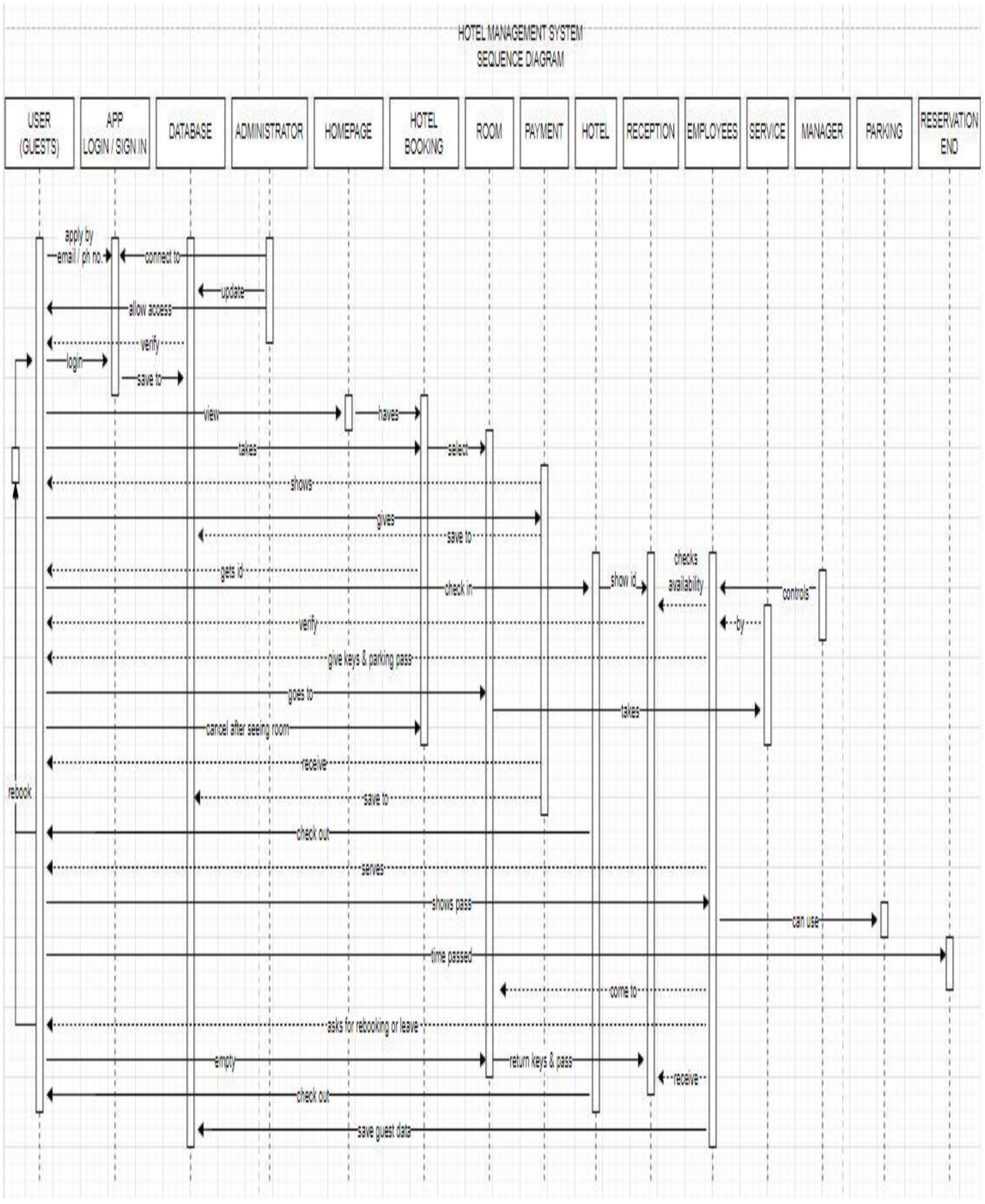
For instance, if you need to change the billing process, you only need to modify the Billing class, without worrying about affecting the reservation or guest management components.

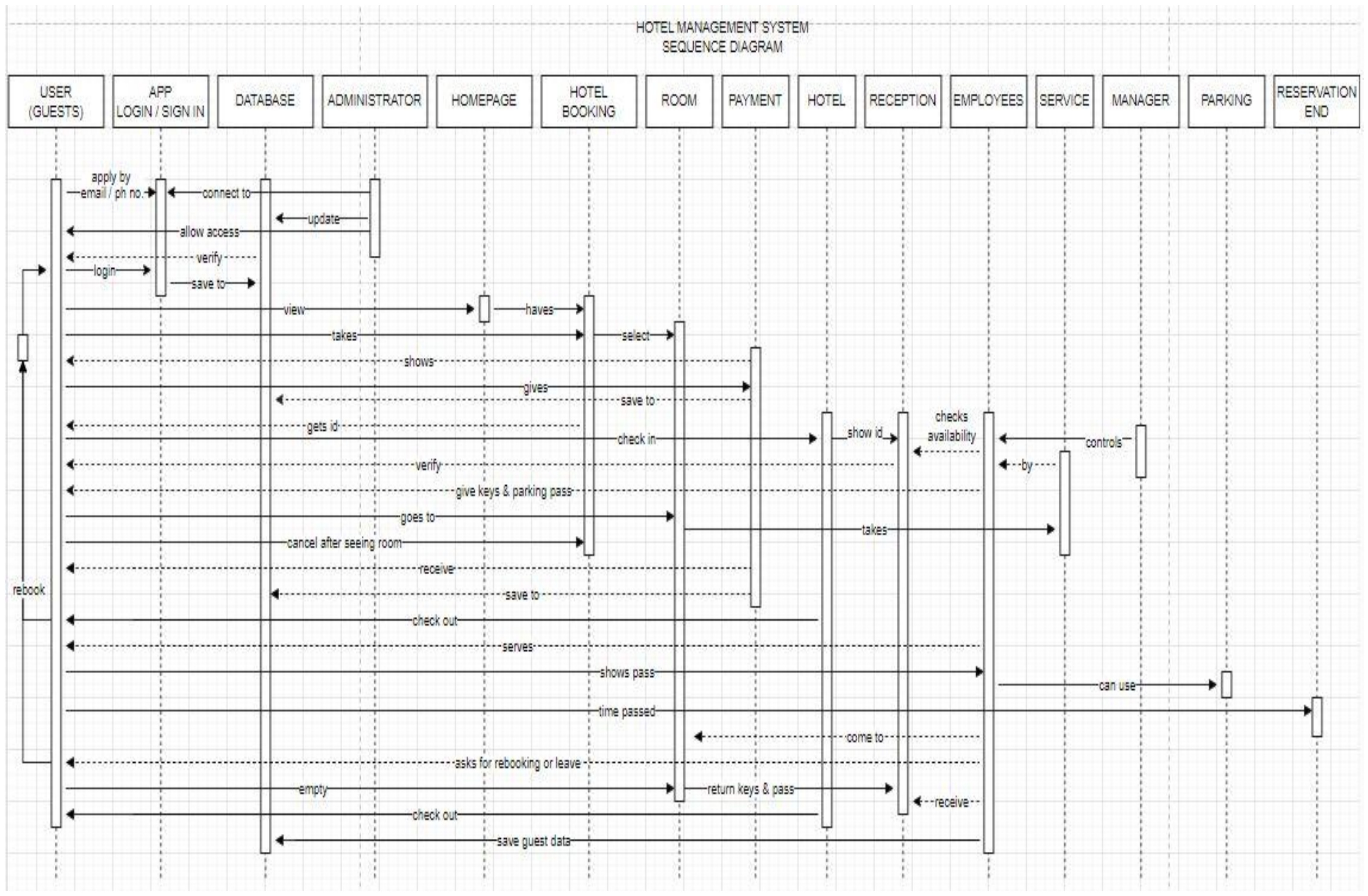
3. **Extensibility and Reusability**: It becomes much easier to reuse components in other parts of the system. For example, if the guest management module can be reused in a different context (say, in a mobile app), it will be simple because the class has only one responsibility.

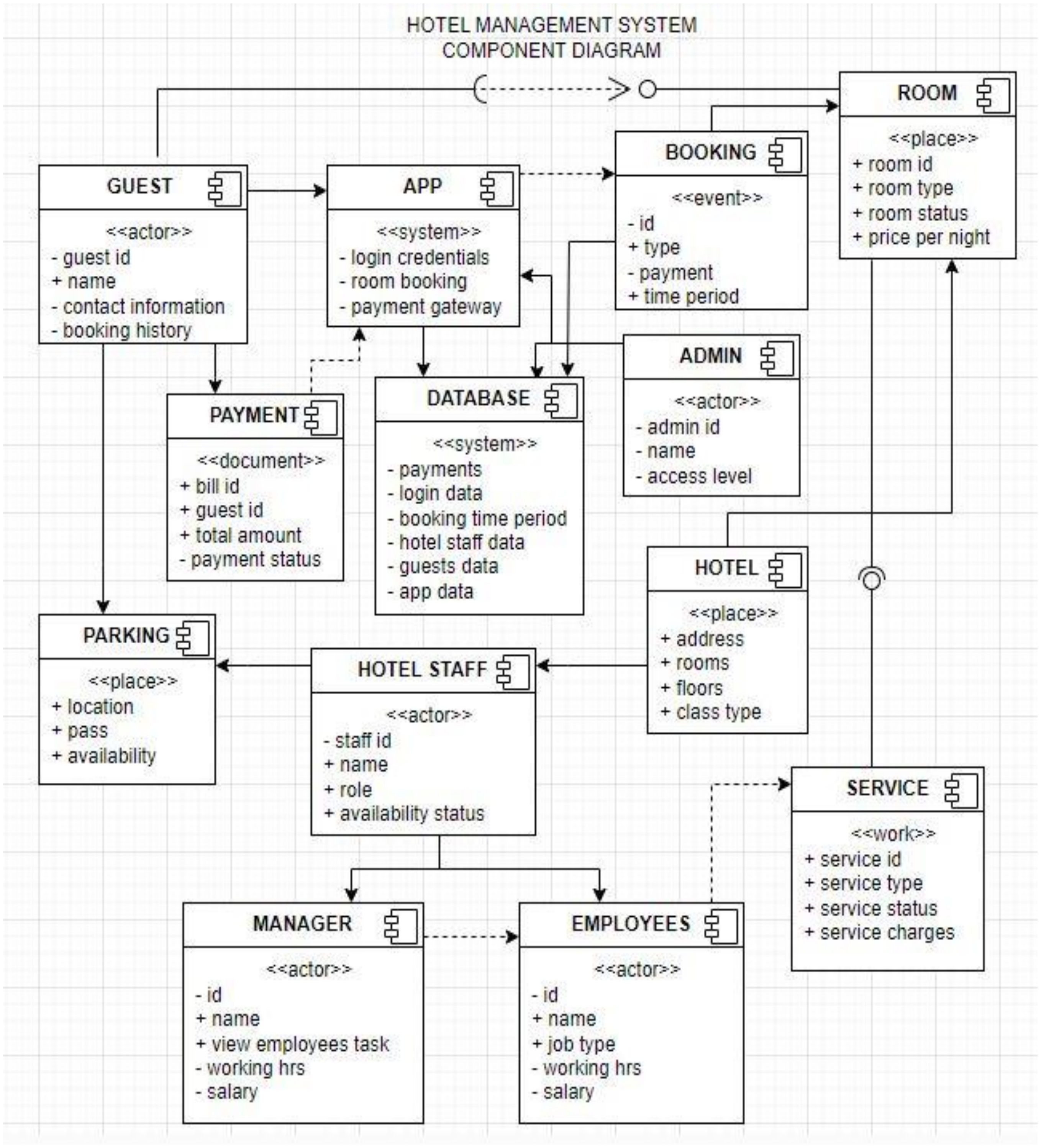
❖ **Software Diagrams:**

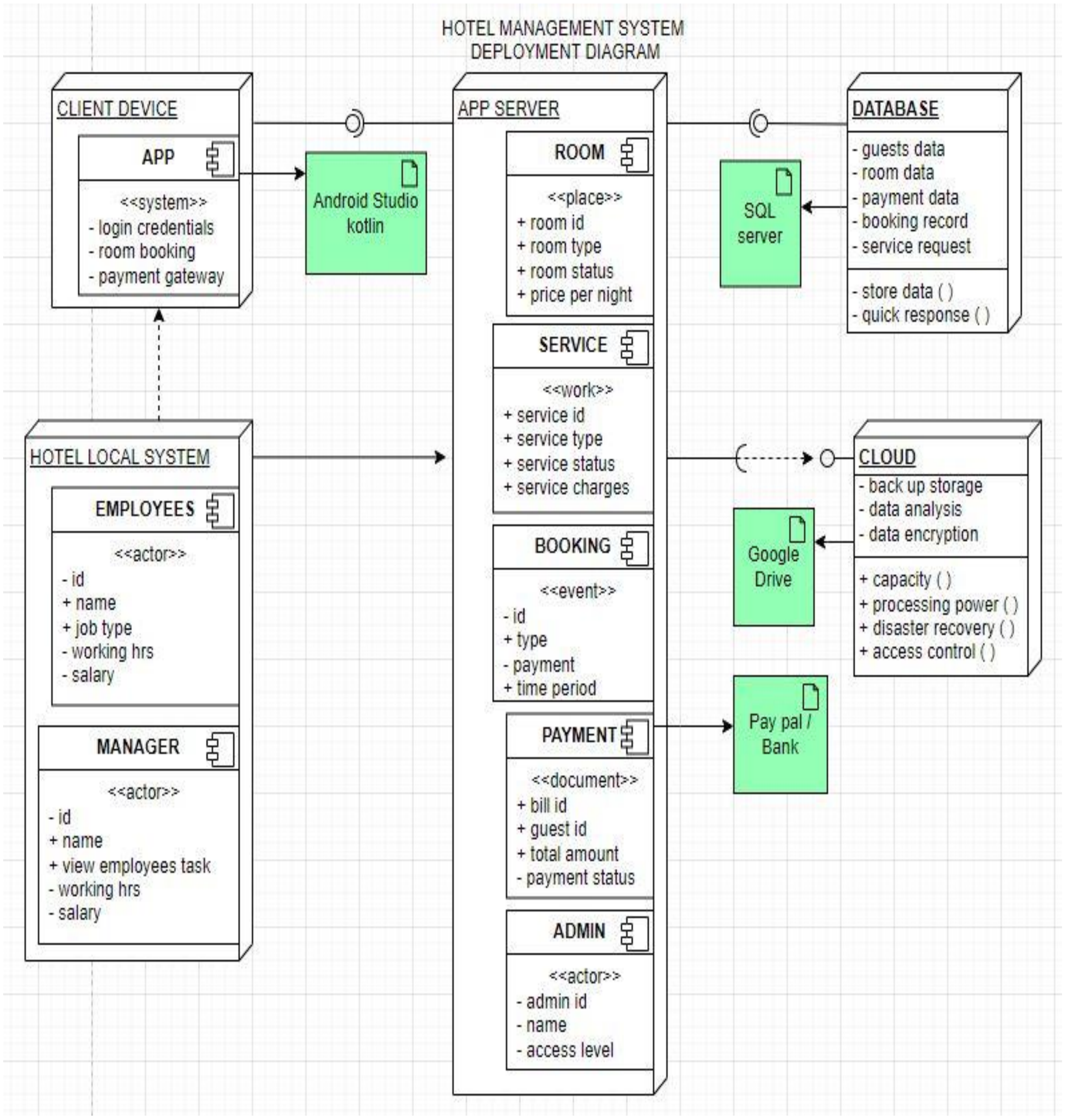


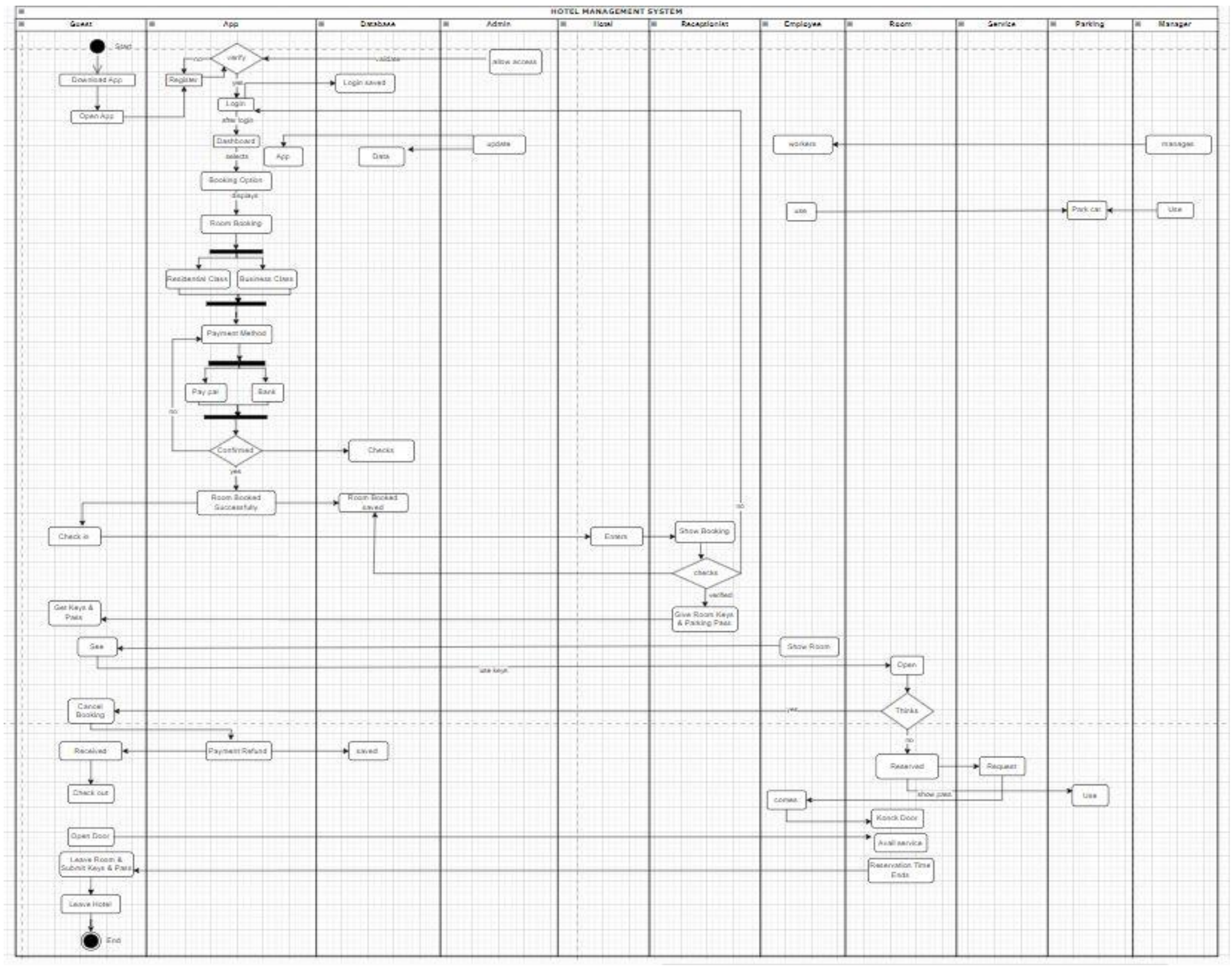

HOTEL MANAGEMENT SYSTEM
CLASS DIAGRAM
HOTEL
+ Name
+ Address
+ Rooms
+ Floors
App
- Login
+ Information
- Code
- Policy
GUESTS
- Name
- Phone Number
- Email Address
- National Id Card Copy
- Address
- Id
- Login credentials
+ Parking Pass
+ Room Key
+ Services
- View and Book Room
- Check In & Out
SERVICE
+ Room Cleaning & Maintenance
+ Food Order
+ Laundry
+ Parking
+ Security
HOTEL STAFF
+ Employees Id
+ Job Type
ADDRESS
+ Street
+ Street
+ City
+ State
+ Postal Code
+ Country
BOOKING
- Id
+ Room Id
+ Guests Id
+ Cost
+ Status
+ Discounts
MANAGER
+ Name
+ Id
- Phone Number
- Address
+ Check employees work
+ Work Hrs
- Salary
ADMINISTRATOR
+ Name
+ Id
- Phone Number
- Address
- Login
- Manages Role
- Update Room Details
EMPLOYEE
+Name
+ Id
- Phone Number
- Address
+ Job Type
+ Work Hrs
- Salary
PARKING
+ Location
+ Floor number
- Pass
+ Availability
ROOM
+ Number
+ Id
+ Bedrooms
+ Floor number
+ Class (standard,luxury,presidential)
+ Cleaning & Maintenance
+ Security Cameras
+ Availability
manages
uses
makes
have
have
take
have
includes
controls
can use
can use
job
1
many

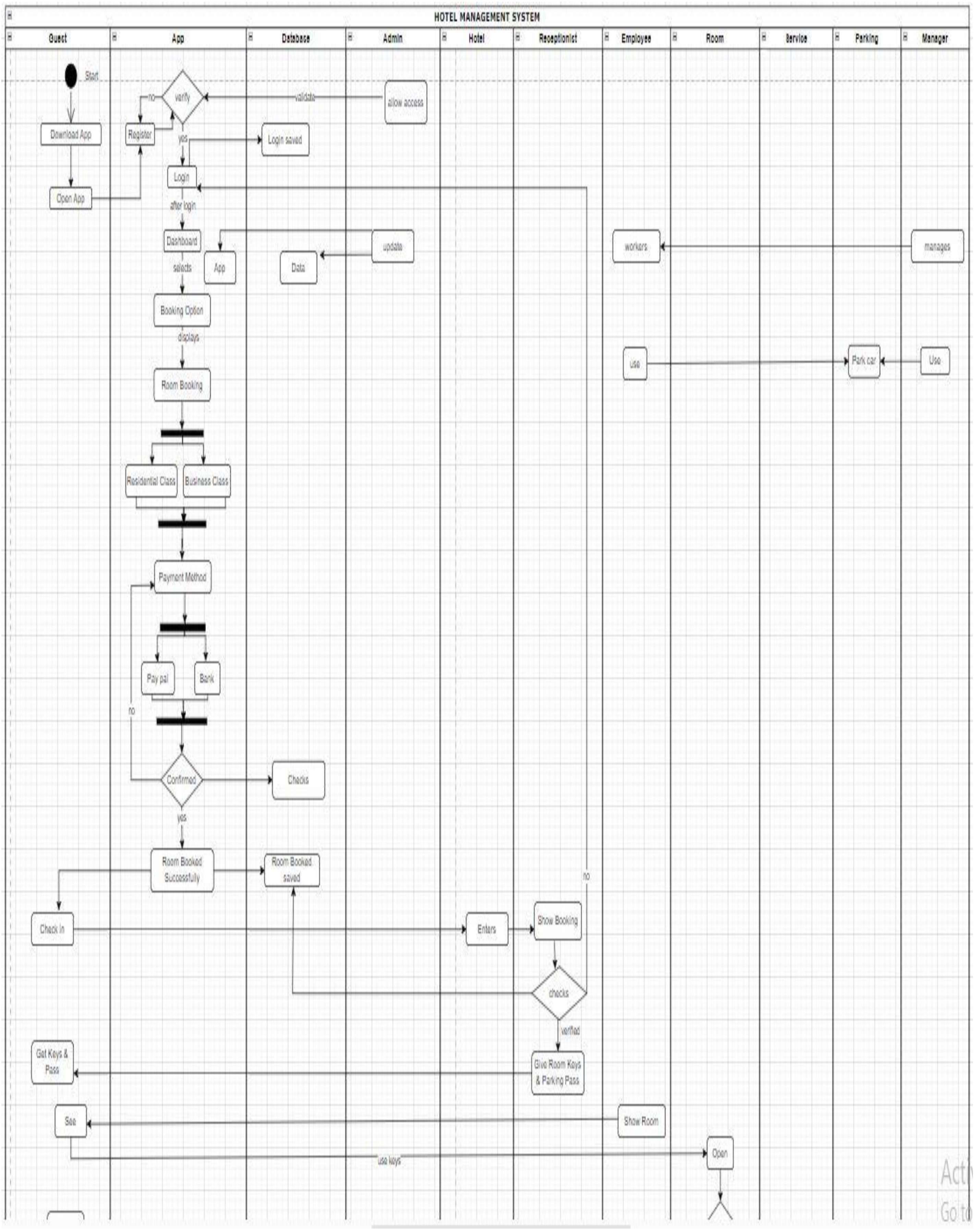

HOTEL MANAGEMENT SYSTEM
SEQUENCE DIAGRAM
USER (GUESTS)
APP LOGIN / SIGN IN
DATABASE
ADMINISTRATOR
HOMEPAGE
HOTEL BOOKING
ROOM
PAYMENT
HOTEL
RECEPTION
EMPLOYEES
SERVICE
MANAGER
PARKING
RESERVATION END
apply by email / ph no.
connect to
update
allow access
verify
login
save to
view
haves
takes
select
shows
gives
save to
gets id
check in
show id
checks availability
controls
verify
by
give keys & parking pass
goes to
takes
cancel after seeing room
receive
rebook
save to
check out
serves
shows pass
can use
time passed
come to
asks for rebooking or leave
empty
return keys & pass
receive
check out
save guest data

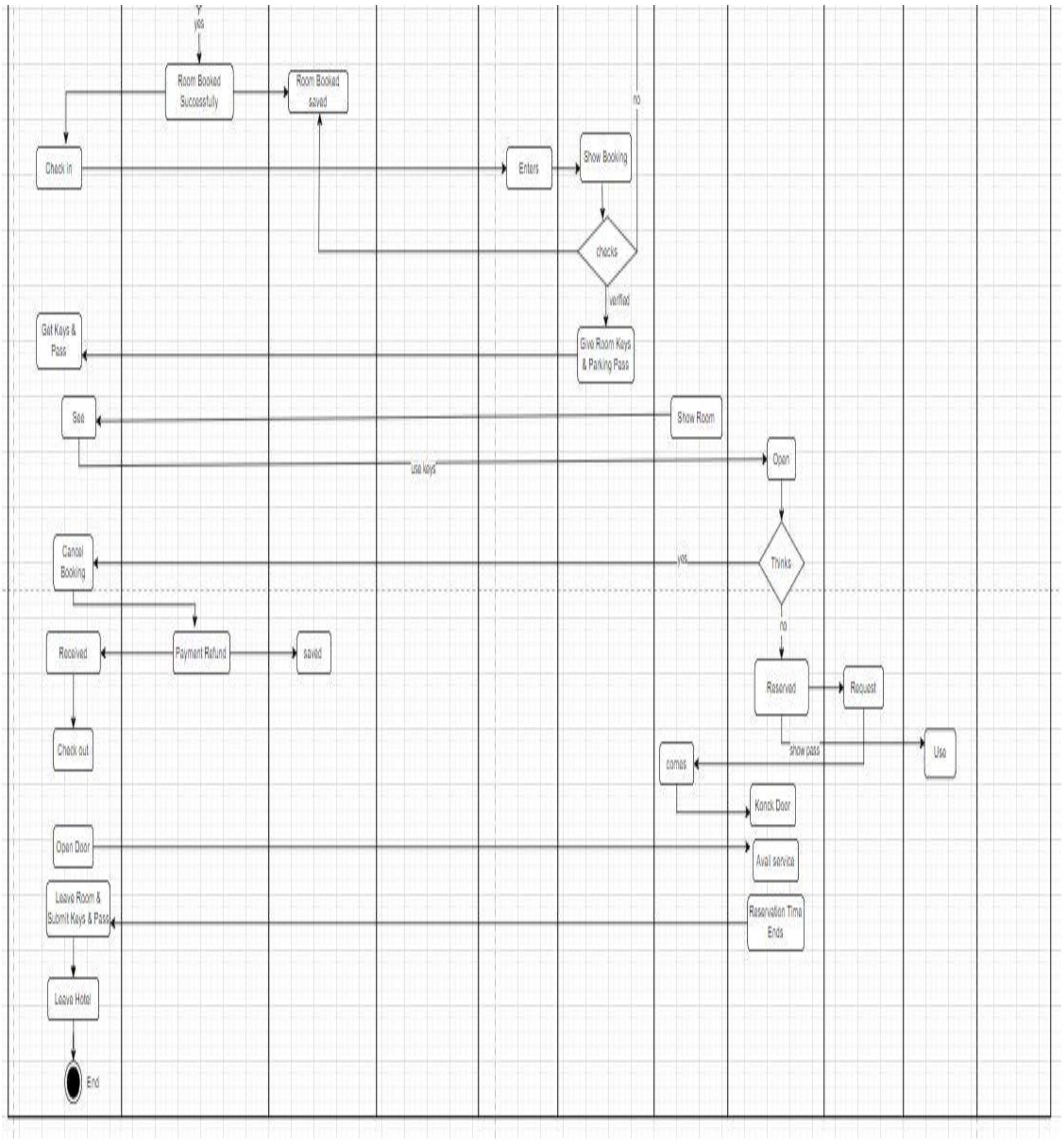

HOTEL MANAGEMENT SYSTEM
SEQUENCE DIAGRAM
USER (GUESTS)
APP LOGIN / SIGN IN
DATABASE
ADMINISTRATOR
HOMEPAGE
HOTEL BOOKING
ROOM
PAYMENT
HOTEL
RECEPTION
EMPLOYEES
SERVICE
MANAGER
PARKING
RESERVATION END
apply by email / ph no.
connect to
update
allow access
verify
login
save to
view
haves
takes
select
shows
gives
save to
gets id
check in
show id
checks availability
controls
verify
by
give keys & parking pass
goes to
takes
cancel after seeing room
receive
rebook
save to
check out
serves
shows pass
can use
time passed
come to
asks for rebooking or leave
empty
return keys & pass
receive
check out
save guest data


HOTEL MANAGEMENT SYSTEM
COMPONENT DIAGRAM
ROOM
<<place>>
+ room id
+ room type
+ room status
+ price per night
BOOKING
<<event>>
- id
+ type
- payment
+ time period
GUEST
<<actor>>
- guest id
+ name
- contact information
- booking history
APP
<<system>>
- login credentials
- room booking
- payment gateway
ADMIN
<<actor>>
- admin id
- name
- access level
PAYMENT
<<document>>
+ bill id
+ guest id
+ total amount
- payment status
DATABASE
<<system>>
- payments
- login data
- booking time period
- hotel staff data
- guests data
- app data
HOTEL
<<place>>
+ address
+ rooms
+ floors
+ class type
PARKING
<<place>>
+ location
+ pass
+ availability
HOTEL STAFF
<<actor>>
- staff id
+ name
+ role
+ availability status
SERVICE
<<work>>
+ service id
+ service type
+ service status
+ service charges
MANAGER
<<actor>>
- id
+ name
+ view employees task
- working hrs
- salary
EMPLOYEES
<<actor>>
- id
+ name
+ job type
- working hrs
- salary


HOTEL MANAGEMENT SYSTEM
DEPLOYMENT DIAGRAM
CLIENT DEVICE
APP
<<system>>
- login credentials
- room booking
- payment gateway
Android Studio kotlin
APP SERVER
ROOM
<<place>>
+ room id
+ room type
+ room status
+ price per night
SERVICE
<<work>>
+ service id
+ service type
+ service status
+ service charges
BOOKING
<<event>>
- id
+ type
- payment
+ time period
PAYMENT
<<document>>
+ bill id
+ guest id
+ total amount
- payment status
ADMIN
<<actor>>
- admin id
- name
- access level
DATABASE
- guests data
- room data
- payment data
- booking record
- service request
- store data ( )
- quick response ( )
SQL server
CLOUD
- back up storage
- data analysis
- data encryption
+ capacity ( )
+ processing power ( )
+ disaster recovery ( )
+ access control ( )
Google Drive
Pay pal / Bank
HOTEL LOCAL SYSTEM
EMPLOYEES
<<actor>>
- id
+ name
+ job type
- working hrs
- salary
MANAGER
<<actor>>
- id
+ name
+ view employees task
- working hrs
- salary


HOTEL MANAGEMENT SYSTEM
Guest
App
Database
Admin
Hotel
Receptionist
Employee
Room
Service
Parking
Manager
Start
Download App
Open App
verify
no
Register
yes
validate
allow access
Login saved
Login
after login
Dashboard
selects
App
update
Data
workers
manages
Booking Option
displays
Room Booking
use
Park car
Use
Residential Class
Business Class
Payment Method
Pay pal
Bank
no
Confirmed
Checks
yes
Room Booked Successfully
Room Booked saved
no
Check in
Enters
Show Booking
checks
verified
Give Room Keys & Parking Pass
Get Keys & Pass
See
Show Room
use keys
Open
Cancel Booking
yes
Thinks
no
Received
Payment Refund
saved
Reserved
Request
show pass
Use
Check out
comes
Knock Door
Open Door
Avail service
Leave Room & Submit Keys & Pass
Reservation Time Ends
Leave Hotel
End


HOTEL MANAGEMENT SYSTEM
Guest
App
Database
Admin
Hotel
Receptionist
Employee
Room
Service
Parking
Manager
Start
Download App
Open App
verify
no
Register
yes
validate
allow access
Login saved
Login
after login
Dashboard
selects
App
update
Data
workers
manages
Booking Option
displays
Room Booking
use
Park car
Use
Residential Class
Business Class
Payment Method
Pay pal
Bank
no
Confirmed
Checks
yes
Room Booked Successfully
Room Booked saved
no
Check in
Enters
Show Booking
checks
verified
Get Keys & Pass
Give Room Keys & Parking Pass
See
Show Room
use keys
Open



❖ Tools and Technologies:

1. Backend Development

- **Programming Language:**
    - **Java.**
    - **Python**
    - **Node.js**
- **Frameworks:**
    - **Spring Boot (Java)** – Ideal for building enterprise-level applications with robust security features and easy integration with databases.
    - **Django (Python)** – Useful if you prefer Python for backend development.
    - **Express.js (Node.js)** – Lightweight and flexible for building RESTful APIs.

2. Database

- **Relational Database Management System (RDBMS):**
    - **MySQL/PostgreSQL** – Suitable for structured data like reservations, billing, and employee management.

3. Frontend Development

- **Web Application:**
    - **HTML/CSS/JavaScript** for basic web design.
    - **React.js** or **Angular** – For dynamic, interactive user interfaces.
    - **Bootstrap** or **Material UI** – For responsive and modern UI components.
- **Mobile Application:**
    - **Android Studio (Java/Kotlin)** for Android development.
    - **React Native or Flutter** – For cross-platform mobile app development.

4. Authentication & Security

- **OAuth 2.0/JWT (JSON Web Tokens)** – For secure, token-based user authentication.

- **SSL/TLS Encryption** – To ensure secure data transfer between the client and server.
- **Spring Security (Java)** or **Passport.js (Node.js)** – For managing authentication and roles/permissions.

5. Payment Gateway Integration

- **Stripe** or **PayPal API** – To handle online payments for room bookings, bills, and other services.

6. Deployment & Hosting

- **Cloud Services:**
  - **AWS** or **Google Cloud** – For hosting and scalability. You can use AWS EC2, S3 for storage, RDS for database management, and CloudFront for content delivery.
- **Docker** – For containerizing your application to simplify deployment across different environments.
- **Kubernetes** – For orchestrating containers and managing scalability.

7. Analytics and Monitoring

- **Google Analytics** – For tracking user interactions and behavior on the hotel's website or app.
- **Prometheus & Grafana** – For real-time monitoring and alerting on system performance.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** – For logging and analyzing server data.

8. Real-Time Features

- **WebSocket** – For real-time communication, such as instant updates on booking status or service requests.
- **Firebase** – For real-time notifications and chat functionalities.

9. AI and Automation

- **Machine Learning (AI) Libraries:**
  - **TensorFlow** or **PyTorch** – For implementing AI-driven features, such as customer behavior prediction or automated service recommendations.
- **Chatbot**:
  - **Dialog Flow** or **Rasa** – For providing automated customer support.
- **IoT Integration:**
  - **MQTT** – For IoT devices that might be used for room automation (e.g., smart thermostats, lighting).

10. DevOps

- **CI/CD Tools:**
  - **Jenkins** or **GitLab CI** – For continuous integration and deployment.
  - **GitHub Actions** – For automating workflows.
- **Version Control:**
  - **Git** – For managing the source code repository.

11. Business Intelligence (BI)

- **Power BI** or **Tableau** – For generating reports and analyzing data related to bookings, revenue, and staff efficiency.

❖ **Conclusion:**

The system overall design was described the system framework design part of the system uses architecture analysis and elaboration. In the introduction to the design of the database, the system involved in the main table structure and storage processes were introduced. Finally, in the Online hotel management system, we have developed a secure, user-friendly Hotel Management System. This system is completely secure since every user is provided with a user ID and Password so there is no chance of any unauthorized access. Online Payment, Booking, and cancellation make it easier to use. So, using this system will help in reducing the labor and provide more facility for Customer to like Hotel and visit again and again.