23
Hotel Reservation System

Hotel reservation system is a system that enables customers to book hotel rooms online. It allows customers to view available rooms, make reservations, and manage their bookings.

Step 1 - Understand the Problem and Establish Design Scope

Step 1 - Understand the Problem and Establish Design Scope

For a hotel reservation system, it takes months for a customer to book a room. We can, for example, book a room 1 year in advance or 1 day in advance. Requirements are different for different use cases. We need to ask questions to understand the requirements. Here are some sample questions:

candidate: Can customers make a reservation for more than one room?
interviewer: Yes, customers can book up to 5 rooms.
candidate: How many people can a room accommodate?
interviewer: Each room can accommodate up to 3 people.
candidate: How long can customers book a room in advance?
interviewer: Customers can book a room up to 1 year in advance.
candidate: Can customers cancel a reservation?
interviewer: Yes, customers can cancel a reservation.
candidate: Do customers need to pay for a reservation?
interviewer: Customers can pay for a reservation when they check in.
candidate: Can customers modify a reservation?
interviewer: Yes, customers can modify a reservation.
candidate: What types of payment do you support?
interviewer: We support credit card payment.
candidate: Do we need to send notifications to customers?
interviewer: Yes, we need to send notifications to customers when they make a reservation, modify a reservation, or cancel a reservation.

Functional requirements

- Customers can book a room.
- Customers can view available rooms.
- Customers can modify a reservation.
- Customers can cancel a reservation.
- Customers can pay for a reservation when they check in.
- Customers can receive notifications when they make a reservation, modify a reservation, or cancel a reservation.

Non-functional requirements

- Support 10 hotels. Assume 1000 rooms per hotel. Assume 2 people per room.
- Support 5 million users.
- Support 1 million reservations per month.
- Support 100 transactions per second.

Back of the envelope estimation

- Assume 10% of the rooms are reserved on a daily basis.
- Assume 1 million reservations per month.
- Assume 10 million users.
- Assume 100 transactions per second.
- Assume 10% of the users are active on a daily basis.
- Assume 10 million users. Assume 1 million reservations per month. Assume 10% of the users are active on a daily basis. Assume 100 transactions per second.

Step 2 - Propose High-Level Design and Get Buy-In

- Hotel Service
- Payment Service
- Notification Service

API design

We use RESTful API to expose the service to client. For our service, we will use JSON to transfer data between client and server.

Hotel Service

POST /v1/hotels	Add a hotel.
GET /v1/hotels/{id}	Get details about a hotel.
POST /v1/hotels/{id}/rates	Add a rate for a hotel.
GET /v1/hotels/{id}/rates?startDate=yyyyMMdd&endDate=yyyyMMdd	Get available rates for a hotel.
POST /v1/hotels/{id}/reservations	Reserve a room.
GET /v1/reservations?firstName=xxx&lastName=xxx	Get reservations.
DELETE /v1/reservations/{id}	Cancel a reservation.
PUT /v1/reservations/{id}	Update a reservation.

Rate Service API

POST /v1/rates	Add a rate.
PUT /v1/rates/{id}	Update a rate.
GET /v1/rates?hotelId=xxx&roomTypeId=xxx	Get rates.

Payment Service API

POST /v1/payments	Execute a payment.
PUT /v1/payments/{id}	Update a payment.
GET /v1/payments?orderId=xxx	Get payment status by order.

Notice that we use plural nouns in API URLs. For example, we use /v1/hotels instead of /v1/hotel. Here is an example of the request and response for the API to add a hotel:

```
POST /v1/hotels
{
  "name": "Radisson",
  "address": "1 Main St",
  "city": "Chicago",
  "country": "USA",
  "type": "Resort",
  "rating": 5
}
```

```
HTTP/1.1 201 Created
Location: /v1/hotels/123
```

Data model

Data access patterns are important when we design a data model. We need to ask questions such as what kind of queries, how much data, and how often the data will be accessed.

Hotel Table

Hotel Rate Table

Room Type Table

High-level design

The high-level design is shown in the diagram below. Hotel, rate, and room type services are responsible for CRUD operations for hotel, rate, and room type. The payment service is responsible for payment processing. The reservation service is responsible for room reservation.

Hotel, rate, and room type services are responsible for CRUD operations for hotel, rate, and room type. The payment service is responsible for payment processing. The reservation service is responsible for room reservation. Notification service is responsible for sending notifications to customers. Hotel, rate, and room type databases are relational databases. Payment and reservation databases are also relational databases. We use a message queue to decouple the payment service from the reservation service. The payment service publishes payment events to the message queue, and the reservation service subscribes to payment events from the message queue. We use third-party payment gateways to process payments. We use third-party email gateways and third-party SMS gateways to send notifications to customers.

Hotel and Rate Service

Hotel and rate services are responsible for CRUD operations for hotel and rate. We need to ask questions such as what kind of queries, how much data, and how often the data will be accessed.

- Hotel and rate services are read-heavy services. Hotel and rate data are not updated frequently. We can use a cache to improve read performance.
- Hotel and rate services are internal services. We use gRPC to expose the services to other services. gRPC is a high-performance, open-source universal RPC framework. It is a binary protocol and is more efficient than JSON. It supports bidirectional streaming and multiplexing. It supports strong typing and code generation. It supports authentication and encryption. It supports load balancing and service discovery.
- Hotel and rate services are stateless services. We can scale them horizontally.
- Hotel and rate services are CRUD services. We can use a relational database to store hotel and rate data.

Here is the diagram for the hotel and rate services:

Hotel and rate services are read-heavy services. We can use a cache to improve read performance. We can use a cache-aside pattern to load data into the cache. We can use a write-through cache to update the cache when we update the database.

Reservation Service

Reservation service is responsible for room reservation. We need to ask questions such as what kind of queries, how much data, and how often the data will be accessed.

- Reservation service is a read-heavy service. We can use a cache to improve read performance.
- Reservation service is a stateless service. We can scale it horizontally.
- Reservation service is a CRUD service. We can use a relational database to store reservation data.
- Reservation service is a transaction service. We need to ensure data consistency.

Here is the diagram for the reservation service:

Reservation service is a read-heavy service. We can use a cache to improve read performance. We can use a cache-aside pattern to load data into the cache. We can use a write-through cache to update the cache when we update the database.

Payment Service

Payment service is responsible for payment processing. We need to ask questions such as what kind of queries, how much data, and how often the data will be accessed.

- Payment service is a write-heavy service. We can use a relational database to store payment data.
- Payment service is a stateless service. We can scale it horizontally.
- Payment service is a transaction service. We need to ensure data consistency.
- Payment service is a third-party service. We need to use third-party payment gateways to process payments.
- Payment service is a CRUD service. We can use a relational database to store payment data.
- Payment service is a third-party service. We need to use third-party payment gateways to process payments.

Here is the diagram for the payment service:

Payment service is a write-heavy service. We can use a relational database to store payment data. We need to use third-party payment gateways to process payments.

Notification Service

Notification service is responsible for sending notifications to customers. We need to ask questions such as what kind of queries, how much data, and how often the data will be accessed.

- Notification service is a write-heavy service. We can use a message queue to store notification data.
- Notification service is a stateless service. We can scale it horizontally.
- Notification service is a third-party service. We need to use third-party email gateways and third-party SMS gateways to send notifications to customers.

Here is the diagram for the notification service:

Notification service is a write-heavy service. We can use a message queue to store notification data. We need to use third-party email gateways and third-party SMS gateways to send notifications to customers.

Step 3 - Design Deep Dive

- Scalability
- Hotel search
- Concurrency
- Caching
- Monitoring
- Logging
- Security
- Rate limiting

Scalability

Hotel and rate services are read-heavy services. We can use a cache to improve read performance. We can use a cache-aside pattern to load data into the cache. We can use a write-through cache to update the cache when we update the database.

Reservation service is a read-heavy service. We can use a cache to improve read performance. We can use a cache-aside pattern to load data into the cache. We can use a write-through cache to update the cache when we update the database.

Payment service is a write-heavy service. We can use a relational database to store payment data. We need to use third-party payment gateways to process payments.

Notification service is a write-heavy service. We can use a message queue to store notification data. We need to use third-party email gateways and third-party SMS gateways to send notifications to customers.

Hotel search

Hotel search is a read-heavy service. We can use a cache to improve read performance. We can use a cache-aside pattern to load data into the cache. We can use a write-through cache to update the cache when we update the database.

Hotel search is a stateless service. We can scale it horizontally.

Hotel search is a CRUD service. We can use a relational database to store hotel data.

Hotel search is a third-party service. We need to use third-party search engines to process hotel search.

Hotel search is a third-party service. We need to use third-party search engines to process hotel search.

Concurrency

Concurrency is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Concurrency is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Caching

Caching is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Caching is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Monitoring

Monitoring is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Monitoring is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Logging

Logging is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Logging is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Security

Security is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Security is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Rate limiting

Rate limiting is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Rate limiting is a key issue in a hotel reservation system. We need to ensure data consistency. We need to use third-party payment gateways to process payments.

Step 4 - Wrap Up

- Hotel Service
- Payment Service
- Notification Service

Chapter Summary

Reference Material

- Hotel reservation system
- Design a hotel reservation system
- Hotel reservation system
- Hotel reservation system
-