# System Design Document

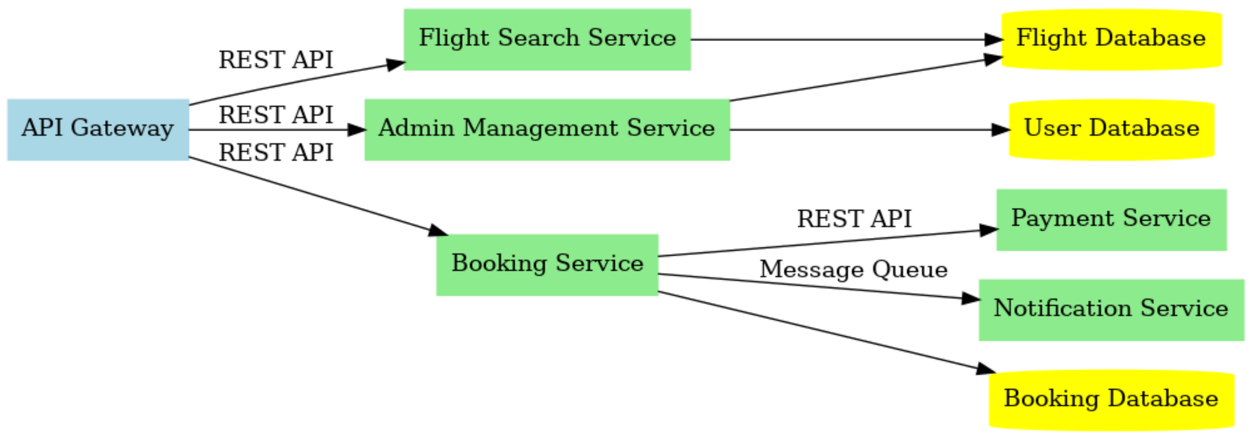
## 1. Architecture Overview

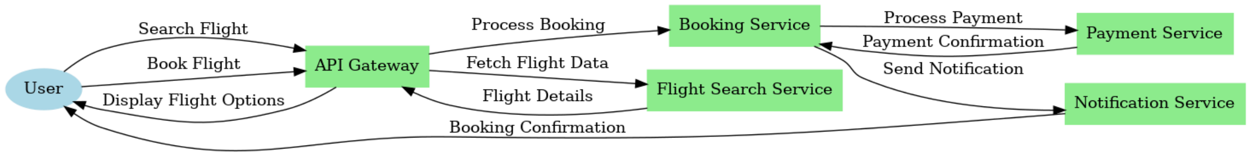
The system is designed as a microservices-based architecture to ensure modularity, scalability, and ease of maintenance. Below are the key components:

- **API Gateway**:
  - Acts as the entry point for all client requests.
  - Routes requests to the appropriate microservices based on the endpoint.
  - Performs security checks (authentication/authorization) and request throttling.
- **Flight Search Service**:
  - Manages flight schedules, prices, and availability.
  - Handles queries for searching flights based on user input (e.g., source, destination, dates).
- **Booking Service**:
  - Manages flight bookings.
  - Communicates with the Flight Search Service to validate seat availability.
  - Interacts with the Payment Service to confirm bookings.
- **Payment Service**:
  - Processes payments using external gateways.
  - Ensures payment security via PCI DSS compliance.
- **Notification Service**:
  - Sends email and SMS notifications for booking confirmations, updates, and cancellations.
- **Admin Management Service**:
  - Allows administrators to manage flights (CRUD operations for schedules, pricing, etc.).
  - Provides reporting and analytics capabilities.

## 2. Component Diagram

A detailed diagram shows the API Gateway, microservices, and their interactions via REST APIs and message queues.





## 3. Database Design

Each microservice has its own dedicated database to maintain data separation and ensure loose coupling. Below are the key databases:

- **Flight Database**:
  - Contains flight schedules, prices, and availability.
  - Key tables: Flights, Airlines, Airports.
- **Booking Database**:
  - Stores details of all bookings, including passenger information.
  - Key tables: Bookings, Passengers, BookingStatus.
- **User Database**:
  - Maintains user profiles, including booking history.
  - Key tables: Users, UserPreferences, UserBookings.

## 4. Communication

- **Synchronous**: REST APIs for client interactions.
- **Asynchronous**: RabbitMQ/Kafka for inter-service communication.

## 5. Deployment

- Dockerized services deployed on Kubernetes.
- Hosted on Oracle Cloud with CI/CD pipelines.

## 6. Monitoring and Logging

- Centralized logging with ELK stack.
- Performance monitoring with Prometheus and Grafana.