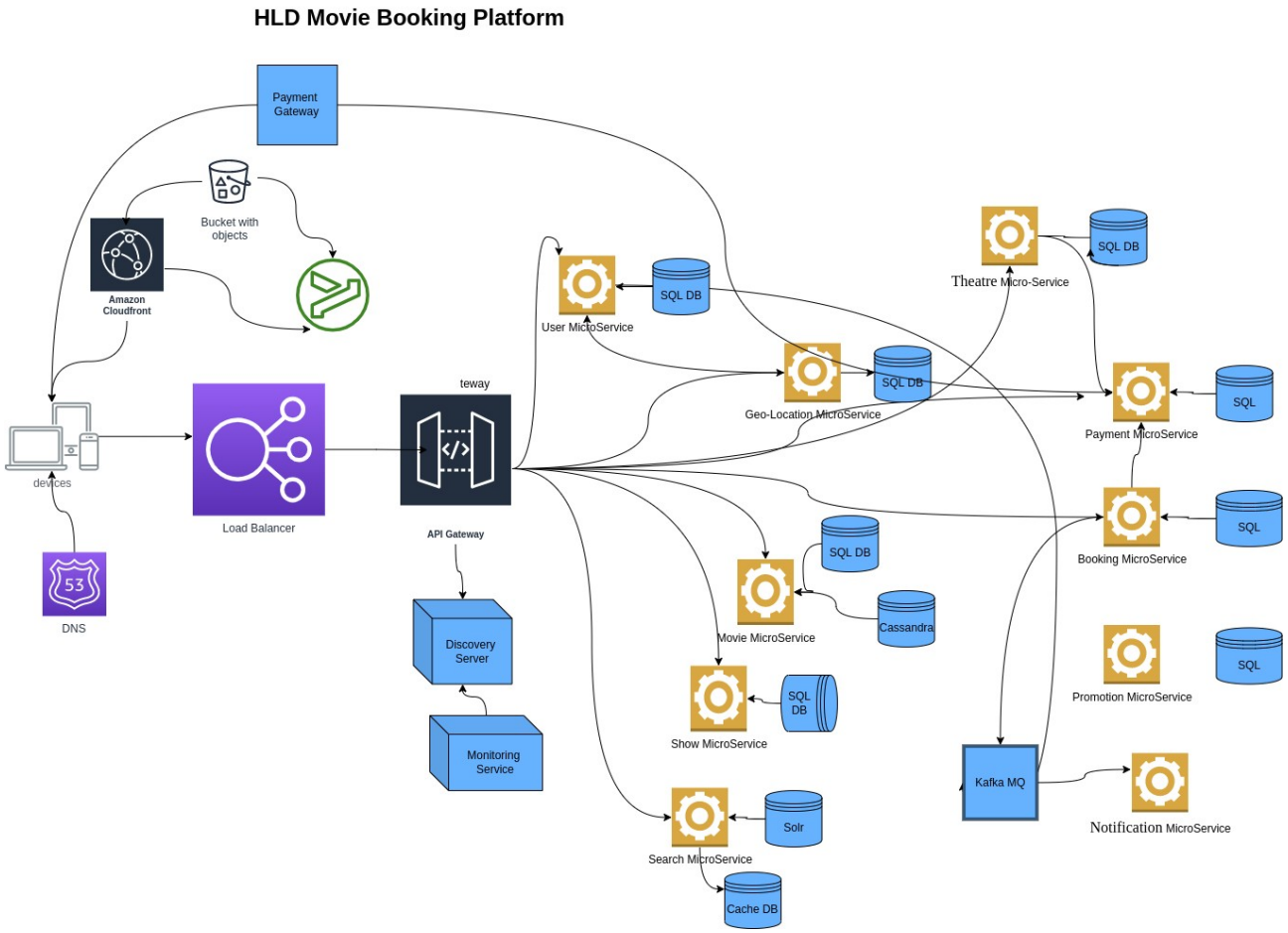
Movie Booking Platform

High Level Design



CDN: To serve Static Content & Media
DNS: To resolve Website Domain Name
LoadBalancer: To load Balance the traffic
API Gateway: To route, load balance, rate-limit the traffic to respective service, authentication & authorization
Discovery Server: For Service Discovery
Monitoring Service: To monitor the Microservice

Low Level Design

 # Classes

- User
- Admin extends User
- Customer extends User
- TheatreOwner extends User
- City [id, name, state]
- Movie [id, name, description, actors, ratings]
- Theatre [id, name, User owner, cityId]
- Screen[Theatre, List<ScreenSeat>, List<Show>]
- ScreenSeat[ScreenId, RowId, SeatType, SeatId]
- enum SeatType{Platinum, Gold, Silver}
- Show[Id, MovieId, StartTime, EndTime, Type (Morn/Eve/Noon)]
- ShowPrice[ShowId, SeatType, BookingPrice],
- Promotion [TheatreId, minTicketCount, discountType[Percentage/Value], discountValue, PromotionStrategy],
- PromotionStrategy {NTicketPurchaseDiscountStrategy, AfternoonDiscountStrategy}
- BookingOrder [orderId, orderAmount, discount, netAmount, List<ScreenSeat> bookedSeats, PaymentType]

 # SQL Tables

- mbs_theatre
- mbs_movie
- mbs_user
- mbs_city
- mbs_screen [screenId, theatreId]
- mbs_screen_seat[screenId, seatId, seatType, RowId]
- mbs_show [screenId, movieId, showStartTime, showEndTime]
- mbs_movie_theatre
- mbs_movie_theatre_show [movie-Theatre FK, start-time, End-time]
- mbs_ticket[orderid, screenId, customerId, showId, totalAmount, promoDiscountAmount, netAmount]
- mbs_ticket_seat [orderid, seatId]

 # Services involved

1. Theatre Service [Manage Theatre, Manage Screen, Manage ScreenSeat, Manage Theatre Movie]
2. Geo-Location Service [Manage City, Manage State, Manage ZipCode]
3. Movie Service [Manage Movie Info, Manage Artists, Manage Cast & Crew, Manage Review, Manage Ratings, Manage Trailers]
4. User Service [Update User details, Register Customer, Register Theatre Owner]
5. Booking Service[getBooking, cancelBooking, getBookingHistory,]
6. Search Service [getMovieSuggestion, getMovieDetail]
7. Notification Service [Notify of booking on Whatsapp, Email, SMS & Other Notifications]
8. Payment Service [Handle Payment of Booked Order]
9. Show Service [Manage Movie Show]
10. Theatre Service [Manage Theatre, Manage Screen, Manage ScreenSeat]
11. Promotion Service [manage promotions]

 # Set of important APIs

- getCityList() : GET (/api/location-service/cities/)
- getMovieListByCity(cityId): GET (/api/movie-service/cities/{cityId}/movies/)

- getTheatreByCity(cityId): GET (/api/theatre-service/cities/{cityId}/theatres/)
- getTheatreByCityAndMovie(cityId, movieId) : GET (/api/theatre-service/cities/{cityId}/theatres/)
- getShowByDate(date, theatreId, movieId)
- getAvailableSeats(showId)
- blockCustomerSelectedSeats()
- bookCustomerSelectedSeat()
- getPromotions

ServiceCommunication: REST API or Async [Kafka] or GraphQL

External Services: Payment gateway Service, SMS & Email Service, CDN

 # Persistence Layer:
SQL DB [To Store Master Data & Transactional data],
NoSQL DB : MongoDB/Cassandra for Detailed Movie Info such as movie description, actors, crew, comments, and reviews, ratings, trailers, gallery
Caching: Redis Cache [To cache movie information]
Full text Search: Solr/ElasticSearch [To Support Movie Search]

 # Application Tech Stack
Java, Spring Boot, Swagger, and Hibernate, React/Angular(UI)
Security: JWT/Oauth2
Deployment: Jenkins, Docker/VM

 # Class Structure

```
class User {
  String id;
  String name;
  String email;
  String phoneNo;
}

class Customer extends User {}

class Admin extends User {}

class TheatreOwner extends User {
    Theatre theatre;
}

class Theatre {
  String id;
  String name;
  ThreatreOwner owner;
  List<Screen> screens;

  String street;
  String city;
  String state;
  String country;
  String zipCode;

}

class Screen {
  String id;
  List<ScreenSeat> seats;
  List<Show> shows;
  String theatreId;
}

class ScreenSeat {
  int id;
  SeatType type;
  boolean isAvailable;
  double ticketCost;
}

enum SeatType {
  PLATINU, GOLD, SILVER
}
```

```
class Movie {
  int id;
  String name;
  Stirng description;
  String lang;
}

class Show {
  int id;
  Movie movie;
  Date StartTime;
  Date endTime;
  String screenId;
}

class Booking {
  String bookingId;
  List<ScreenSeat> bookedSeats;
  Show show;
  Customer user;
  BigDecimal totalAmount;
  BigDecimal promoDiscountAmount;
  BigDecimal netAmount; //totalAmount - promoDiscountAmount
  Date bookingDate;
}


class Payment {
  String paymentId;
  String bookingId;
  Date paymentDate;
  PaymentMode mode;
  PaymentStatus status;
  double totalAmount;
}

enum PaymentStatus {
  FAILED, DECLINED, SUCCESS
}

enum PaymentMode {
  UPI, CARD, WALLET
}
```

 # Design Patterns:

1. Factory Pattern: To get the user based on type
2. Strategy Pattern: To apply offer as per the applicable Promotion Strategy