BookUrMovie.com

Lights Camera & Action

TABLE OF CONTENTS

GOAL

The main objective of the project is to create an Online Movie Ticket Booking system that allows customers to know about new movies, their schedules, cinema locations, ticket price, ratings etc. The user will be able to search for movies, theatres and slots in order to book tickets. Users will be able to download tickets and invoice and will be notified through mail when the booking is confirmed. The user can rate the movies and theatres. This project will be able to recommend movies based on other users' preferences who have watched the given movie before.

This project will be using Kafka for the notification system. This project will be using Elasticsearch for finding movies accurately and efficiently.

The Administrator will have the option to add, update any theatre/movie. Likewise he/she can add/delete slots and can also view all movie ratings given by users.

TECHNOLOGIES USED

1. Java
2. Spring Boot
3. Database(mysql)
4. Kafka - sending email notifications
5. Messaging service
6. Sonarlint
7. Microservices
8. Elasticsearch - searching in movie titles and for movie recommendations
9. Security
10. Maven
11. Tomcat

USE CASES

USER PRIVILEGES

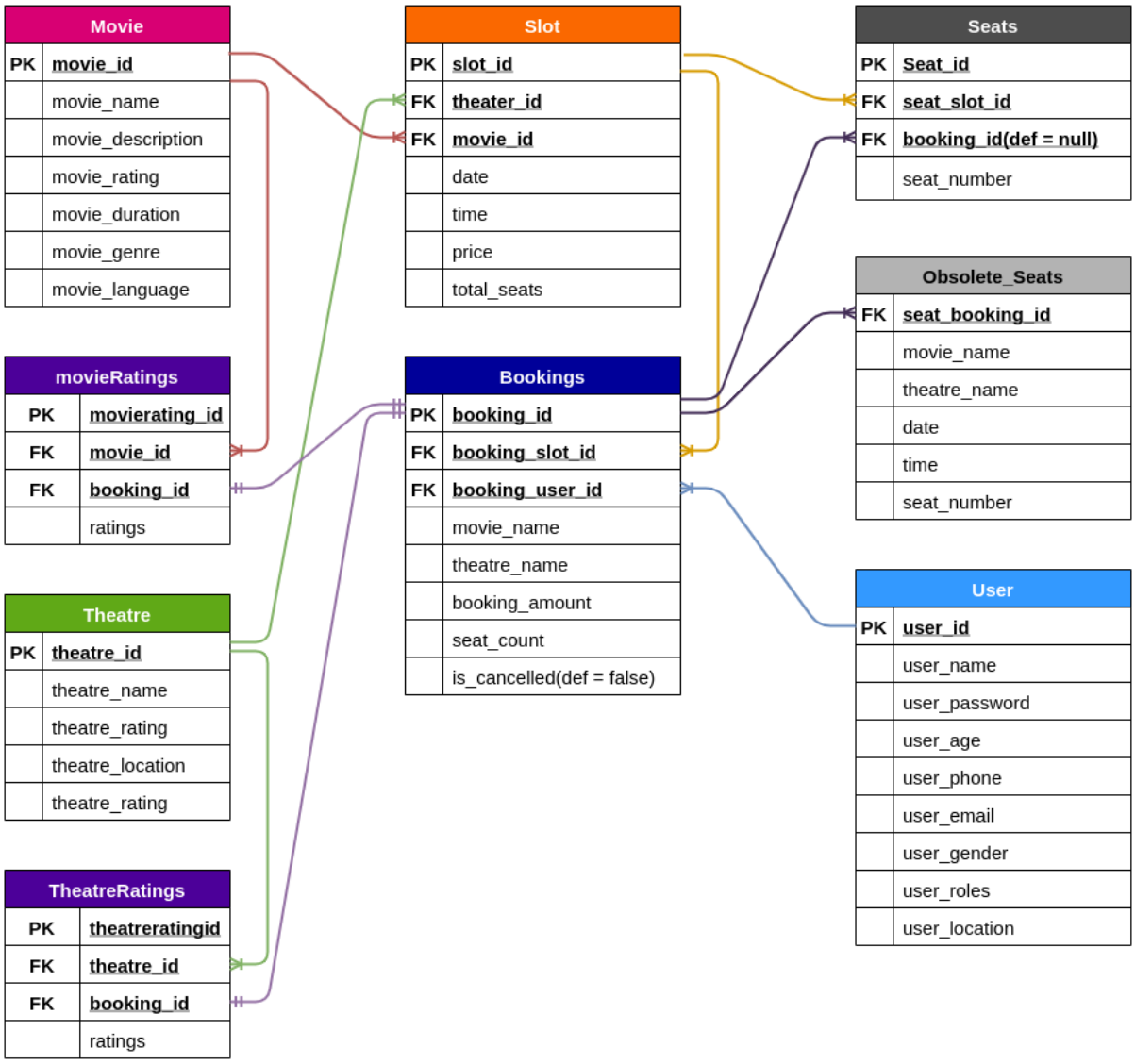
1. Sign up
2. Login
3. Update user details
4. List theatres
5. Find theatre
6. List movies and movie rating
7. Find movie
8. Find movie Slots
9. Find available Seats
10. Mail booked tickets
11. Generate invoice
12. View bookings
13. Select booking
14. Cancel booking
15. Rate experience

ADMIN PRIVILEGES

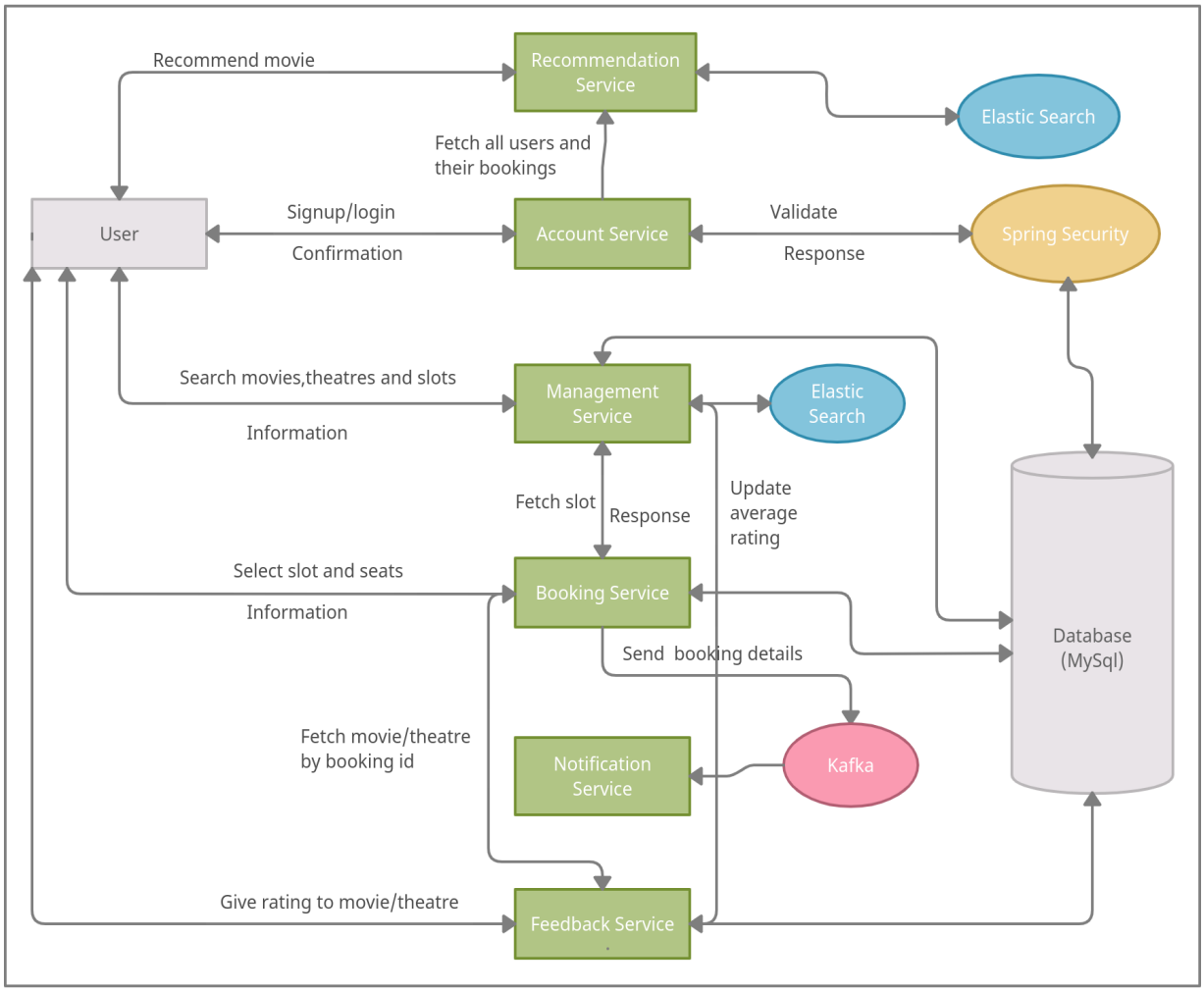
1. Login
2. Add theatres
3. List theatres
4. Update theatre
5. Add movies
6. List movies
7. Add movie slot in any theatre
8. Update movie slot in any theatre
9. Delete movie slots in any theatre and initiate refunds for users having a booking in that slot.
10. View all users
11. View all user-booking details
12. View aggregate rating of all movies.

DATABASE DESIGN AND MODELLING

ER DIAGRAM



ARCHITECTURAL FLOW DIAGRAM



PRE-REQUISITES

- **Run mysql in docker**
  - docker run -p 3306:3306 -d --name=mysql -e MYSQL_ROOT_PASSWORD=Password mysql/mysql-server:latest
  - docker exec -it mysql bash
  - mysql -u root -p
  - Password
  - ALTER USER 'root'@'localhost' IDENTIFIED BY 'Password';
  - CREATE USER 'usernameall'@'%' IDENTIFIED BY 'ThePassword';
  - grant all on *.* to 'usernameall'@'%';

- **Run elastic search in docker**
  - docker run -d --name es793 -p 9200:9200 -e "discovery.type=single-node" elasticsearch:7.9.3

- **Run kafka**
  - confluent local services start

API DESCRIPTION

1. Account Service

- [POST] localhost:8080/signup - Signup by providing user details.

```
{
   "username":"user12",
   "email":"user1@bym.com",
   "password":"User1@weee",
   "phone":"1000010000",
   "age":20,
   "gender":"male",
   "location":"Hyderabad"
}
```

- [GET](USER, ADMIN) localhost:8080/home - Homepage for BookUrMovie.com

- [PUT](USER) localhost:8080/users - Update User Details by providing user details

```
{
   "username":"admin",
   "email":"admin@bym.com",
   "password":"admin",
   "phone":"1000010000",
   "age":30,
   "gender":"male"
}
```

- [PUT] (USER) localhost:8080/users/location/{userLocation} - Update User Location by passing new location in the URL path variables

- [GET] (ADMIN) localhost:8080/users/{userId} - Find User bookings by user ID.

- [GET] (USER) localhost:8080/users/bookings - Find list of bookings for the current active user.

- [GET] (ADMIN) localhost:8080/users/{userId}/details - Fetch user details by user ID.

- [GET] (ADMIN) localhost:8080/users/profile - Fetch user summary by user ID.

- [GET] (ADMIN) localhost:8080/users/{userEmail} - Find user by Email

2. Management Service

2.1. Movie Service

- [GET](USER) localhost:8080/movies - Find list of all movies.
- [GET](USER) localhost:8080/movie/es/{movieName} - Find a movie using elastic search.
- [POST](ADMIN) localhost:8080/movies - Add a new movie by providing movie details.

```
{
   "movieName":"Any Movie Name",
   "movieDescription":"Dummy movie description. Please add a relevant movie
description of more than 50 characters.",
   "movieRating":0,
   "movieDuration": 3.5,
   "movieGenre":"Movie",
   "movieLanguage":"English"
}
```

- [PUT](ADMIN) localhost:8080/movies - Update a movie by providing movie details.

```
{
   "movieName":"Any Movie Name",
   "movieRating":0,
   "movieDuration": 3.5,
   "movieGenre":"Movie"
}
```

- [POST](ADMIN) localhost:8080/movies/es - Add existing movies to Elastic Search database.
- [PUT](USER) localhost:8080/movies/{movieId}/rating/{movieRating} - Update Average movie ratings by specifying movieId and the movie rating in the request URL.
- [GET](ADMIN) localhost:8080/movies/{movieId} - Find a movie by movie Id
- [GET](USER) localhost:8080/movies/{movieName}/details - Find a movie by movie name

2.2. Theatre Service

- [GET](ADMIN) localhost:8080/theatres/admin - Find list of all theatres.
- [GET](USER) localhost:8080/theatres - Find list of all theatres by user location.
- [POST](ADMIN) localhost:8080/theatres - Add a new theatre by providing theatre details.

```
{
"theatreName": "ITech",
"theatreLocation":"Bangalore"
}
```

- [PUT](ADMIN) localhost:8080/theatres - Update a theatre by providing theatre details.

```
{
   "theatreId":1,
   "theatreLocation":"Hyderabad"
}
```

- [PUT](USER) localhost:8080/theatres/{theatreId}/rating/{theatreRating} - Update Average theatre ratings by specifying theatre ID and the theatre rating in the request URL.
- [GET](ADMIN) localhost:8080/theatres/{theatreId} - Find a theatre by theatre Id
- [GET](ADMIN) localhost:8080/theatres/{theatreName}/details - Find a theatre by theatre name
- [GET](ADMIN) localhost:8080/theatres/{theatreName}/slots - Find theatre slots by theatre name

2.3. Slot Service

- [POST](ADMIN) localhost:8080/slots - Add a new slot by providing slot details.

```
{
   "slotTime": "12:00:00",
```

```
    "slotDate": "2021-01-25",
    "slotPrice":1000.00,
    "numberOfSeats": 5,
    "movieName": "MissionImpossible",
    "theatreName": "IMAX"
}
```

- [GET](ADMIN) localhost:8080/slots/admin - Find all slots.
- [GET](USER) localhost:8080/slots - Find all slots by user location.
- [DELETE] (ADMIN) localhost:8080/slots/{slotId} - Delete a slot by slot ID.
- [GET](USER) localhost:8080/slots/{slotId} - Find a slot by slot ID.
- [GET](USER) localhost:8080/slots/movies/{movieName} - Find all slots by movie name in user's location.
- [GET](ADMIN) localhost:8080/slots/{movieName}/{theatreName} - Find all slots with the given movie name and theatre name.
- [GET](ADMIN) localhost:8080/slots/find-movie/{slotId} - Find movie name for the given slot ID.
- [GET](ADMIN) localhost:8080/slots/find-theatre/{slotId} - Find theatre name for the given slot ID.
- [DELETE] (ADMIN) localhost:8080/slots/delete-outdated - Clear all seat data from outdated slots

2.4. Obsolete Seat Service

- [GET](ADMIN) localhost:8080/obsolete-seats/{seatId} - Find if an obsolete seat by seat ID

- [GET](ADMIN) localhost:8080/obsolete-seats/booking/{bookingId} - Find all obsolete seats by booking ID.

2.5. Seat Controller

- [GET](USER) localhost:8080/seats/{seatId} - Find a seat by the given seat ID in URL.
- [PUT](USER) localhost:8080/seats/{seatId} - Update seat's booking ID for the seat ID in th URL
- [GET](ADMIN) localhost:8080/seats/{seatId}/booking/available - Find if a seat is booked or not using the seat ID
- [DELETE](ADMIN) localhost:8080/seats/{seatId}/booking/available - Delete booking from seat using the seat ID. Make seats available for booking.

3. Booking Service

- [GET](USER) localhost:8080/bookings/{bookingId} - Find details of the booking with the given id.
- [GET](USER) localhost:8080/bookings - List all bookings for the user.
- [GET](ADMIN) localhost:8080/bookings/admin - List all bookings done till now.
- [POST](USER) localhost:8080/bookings - Book a ticket based on slot_id and seats list.

```
{
   "slotId" : 1,
   "seatNumberList": [3, 4, 5]
}
```

- [GET](ADMIN) localhost:8080/kafka/{booking_id} - Send booking id to kafka topic using a producer.

4. Notification Service

- [GET](ADMIN) localhost:8080/email/{booking_id} - Send booked tickets to the user.
- [GET](USER) localhost:8080/invoice/{booking_id} - Generate invoice.

5. Feedback Service

- [POST](USER) localhost:8080/movie-ratings - Rate the movie once you have watched it

```
{
```

```
   "bookingId" : 1,
   "rating" : 2
}
```

- [GET](ADMIN) localhost:8080/movie-ratings/{movie_rating_id} - Find movie rating by id
- [GET](ADMIN) localhost:8080/movie-ratings - Get all movie ratings
- [POST](USER) localhost:8080/theatre-ratings - Rate the theatre once you have visited it.

```
{
   "bookingId" : 1,
   "rating" : 4
}
```

- [GET](ADMIN) localhost:8080/theatre-ratings/{theatre_rating_id} - Find theatre rating by id
- [GET](ADMIN) localhost:8080/theatre-ratings - Get all theatre ratings.

6. Recommendation Service

- [POST](ADMIN) localhost:8080/create-index - Fetching user ratings from the database and classifying movies as liked / disliked by the user. Saving this classification based on users into a new elastic search index.
- [GET](USER) localhost:8080/recommendations/{movie_name} - Recommend movies based on other users preferences who have watched the given movie earlier.