|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | | | | | | | **Book My Movie - Design Document** | | | | | | |  | | |  |  | | | | Status: | Updated Draft with review comments | | | | Version: | Third | | | | Date: | **04 May 2020** | | | | | |
|  |  |  | |

Table of Contents

[1. Document Purpose 3](#_Toc25575484)

[2. Scope 3](#_Toc25575485)

[3. Solution Architecture 3](#_Toc25575486)

[4. Definitions 5](#_Toc25575487)

[5. Attachments 6](#_Toc25575488)

[6. Revision History 6](#_Toc25575489)

# Document Purpose

Online movie booking system provides online system to search for movies and theatres. It also has the functionality of booking the tickets or cancel them. The user will be able to view and update their details.

# Scope

* The aim is to build a system that can book movies for the user.
* Add/Edit/Delete/View movies can be done by the admin.
* Add/Edit/Delete/View theatres can be done by the admin.
* User can search for movies and can book tickets and even cancel them.
* Users can view details of movies, theatres, booking as well as their account information.
* Admin can generate reports like total bookings.

# Solution Architecture

A picture containing text

Description automatically generated

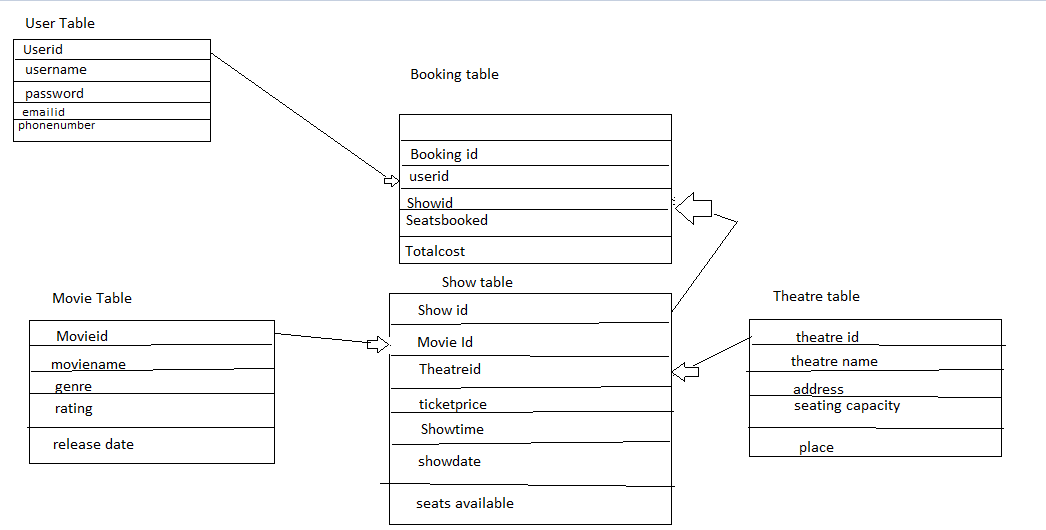
Figure 1: Architecture Diagram

* The Application has been implemented using the **Microservice Architecture**.
* The IDE used will be Spring Tool Suite 4-4.6.
* Technologies used: Java 1.8 SE.
* Spring boot Java framework will be used.
* **API Gateway** – Zuul server will be used as the API gateway and it routes all the client’s requests to the individual services.
* **Authorization** – The Authorization will be implemented using OAuth 2.0.
  + The user specific functionalities are restricted using roles (ADMIN and USER).
  + According the user the page will be redirected to either user page or admin page.
  + Grant Type – ‘password’ has been implemented.
* **Registry** – Eureka server has been used as the registry server.
* Please refer to [Attachments](#_Attachments) for **Swagger API documentation** of the services
* **User/Admin Service** – This service handles all the user whether it is admin or a normal user and according to the user the page will be redirected.
  + User can be login in if its already a user
  + Registration will be done for a new user.
  + Users can search movies and theatres.

Users can update or delete their account.

Users can view their account details.

* **Movie Service** – This service handles all the movie information like the movie name, price, rating etc.
  + User can search for a movie according to its name.
  + Admin can also search for movie.
  + Admin can add/delete/update/view a movie
  + Users can view movie related data.
* **Theatre Service** – This service handles all the theatre information like the theatre name, location etc.
  + User can search for a theatre according to its name.
  + Admin can also search for theatre.
  + Admin can add/delete/update/view a theatre
* **Booking Service** – This service handles all the booking related things like booking a ticket etc.
  + User can book the ticket according to the movie name or theatre name.
  + User can also cancel the tickets.
  + Admin can add/view/delete/update Show details.
* **DB Layer** – PostgreSQL database has been used. PFB the entity diagram for the same.
  + All the primary keys are **Auto Increment columns**



**FIG: Entity Relationship diagram**

# Definition

The following words, acronyms and abbreviations are referred to in this document.

|  |  |
| --- | --- |
| Term | Definition |
| API | Application Program Interface |