Backend Documentation of Movie Booking App

Project Overview

- **Project Name**: Booking System
- **Description:** A web-based application that allows users to register/login, book movies currently streaming in the theatre, see popular movies, cancel bookings if the show has not yet started, and modify user information (change password, email, or phone number).
- **Technology Stack:** Java, PostgreSQL, Tomcat, Jakarta, JAX-RS
- **Version: 1.0-SNAPSHOT**

System Architecture

1. **Backend Framework**
    - **Language:** Java 21
    - **Framework:** Jakatra EE with JAX RS for RESTful API
    - **Server:** Apache Tomcat
    - **Database:** PostgreSQL

USER FUNCTION

This API manages user-related operation

- Register New User
- Login Existing User
- Change User Details Like (email, number, password)
- See Booking History

REGISTER API

ENDPOINT

1. Register User

URL: /register
Method: POST
Request-Header:

- Key: Content-type
- Value: application/JSON

Request-Body:

```
{
        “name”: “test”,
        “email”: “test@test.com”,
        “phone”: “8870716671”,
        “password”: “Test@123”
}
```

Responses

Status code	JSON Response
200 OK	{“message”: “Registration successful”}
409 CONFLICT	{"error": "Email or Phone Number already exists."}
404 BAD_REQUEST	{"error": "Require every field"}
500 INTERNAL_SERVER_ERROR	{"error": "Database error occurred."}

LOGIN API

2. Login User

URL: /login
Method: POST
Request-Header:

- Key: Content-type
- Value: application/JSON

Request-Body:

```
{
        “email”: “test@test.com”,
        “password”: “Test@123”
}
```

Responses

Status Code	JSON Response
200 OK	{"message": "Login Successful. Welcome, " + user.getName() + "}
400 BAD_REQUEST	{"error": "Email/Phone and Password are required"}
401 *UNAUTHORIZED*	{"error": "Invalid email/phone or password"}
503 SERVICE_UNAVAILABLE	{"error": "Database is temporarily unavailable. Please try again later."}
500 INTERNAL_SERVER_ERROR	{"error": "Unexpected error occurred"}

User Dashboard API

Base URL

http://localhost:8080/BookingSystemAPI/api/users

Endpoints

1. Get Booking History

Endpoint:

Method: GET

URL: /users/history/{userId}

Description: Retrieves the booking history for a specific user.

Path Parameter:

Parameter	Type	Required	Description
userId	int	Yes	The ID of the user whose booking history is to be retrieved.

Response:

Status Code	JSON Response
200 OK	Returns a JSON array of booking history.
400 BAD_REQUEST	{\"error\": \"Invalid user ID format\"}
404 NOT_FOUND	{\"error\": \"No booking history found for this user\"}
500 INTERNAL_SERVER_ERROR	{\"error\": \"Unexpected error occurred\"}

Example Response:

```
[
  {
    "bookingId": 1,
    "userId": 10,
    "date": "2024-02-27",
    "status": "confirmed"
  }
]
```

Update User Information

Endpoint:

Method: PUT

URL: /users/update

Query Parameters:

Parameter	Type	Required	Description
userId	int	Yes	The ID of the user whose booking history is to be retrieved.
password	String	Yes	The current password for verification.

Request-Body:

```
{
   "userId": 123,
   "password": "OldPass123",
   "user": {
      "name": "John Doe",
      "email": "john.doe@example.com",
      "phone": "1234567890"
   }
}
```

Status Code	JSON Response
200 OK	{\"message\": \"User updated successfully\"}
400 Bad Request	{\"error\": \"User ID and password are required\"}
401 Unauthorized	{\"error\": \"Invalid password or user not found\"}
500 Internal Server Error	{\"error\": \"Unexpected error occurred\"}

Change User Password

Endpoint:

Method: PUT

URL: /users/change-password

Query Parameters:

Parameter	Type	Required	Description
userId	int	Yes	The ID of the user whose booking history is to be retrieved.
oldpassword	String	Yes	The current password for verification.
newpassword	String	Yes	The new password to be set.

Request-Body:

```
{
   "userId": 123,
   "oldPassword": "OldPass123",
   "newPassword": "NewPass456"
}
```

Response:

Status Code	JSON Response
200 OK	{\"message\": \"Password changed successfully\"}
400 Bad Request	{\"error\": \"User ID, old password, and new password are required\"}
401 Unauthorized	{\"error\": \"Incorrect old password or user not found\"}
500 Internal Server Error	{\"error\": \"Unexpected error occurred\"}

Logout

Endpoint:

Method: POST

URL: /users/logout

Request Headers

- Cookie: The session ID stored in cookies, required for identifying the user session.

Response

STATUS CODE	JSON RESPONSE
200 OK	{\"message\": \"User logged out successfully\"}

Booking API

Endpoint:

URL: /booking

Method: POST

Parameter:

Parameter	Type	Required	Description
userId	int	yes	Id of user who books the movie
movieShowId	int	yes	Id of the show they book
seatIds	Array	yes	Id of seats to be booked

Request-Body

```
{
  "userId": 8,
  "movieShowId": 2,
  "seatIds": [3, 4, 5]
}
```

Response:

Status Code	JSON Response
201 CREATED	{\"message\": \"Ticket booked successfully\", \"bookingId\": " + bookingId + "}
400 Bad Request	{\"error\": \"Booking validation failed or seat unavailable\"}
500 Internal Server Error	{\"error\": \"Unexpected error occurred\"}
400 BAD REQUEST	{ "error": "Invalid booking data" }

CANCEL BOOKING API

ENDPOINT:

URL: /booking/cancel/{Id}

Method: DELETE

Path Parameter:

Parameter	Type	Required	Description
Id	int	Yes	ID of the booking to be canceled.
User_id	int	Yes	ID of the user requesting the cancellation.

Response:

Status Code	JSON Response
200 OK	{ "message": "Booking canceled successfully" }
404 NOT FOUND	{ "error": "Cancellation failed. Booking doesn't exist or show has started." }
500 Internal Server Error	{ "error": "Internal server error occurred" }

Movie API

Base URL: /movies

Endpoints

1. Get Currently Playing Movies

- **URL: /movies/playing**
- **Method: GET**

Response

Status Code	JSON Response

200 OK	{
	"id": 1,
	"title": "Movie Name",
	“certificate”: ‘ A ’,
	“language”:’Engilsh’,
	“movie_cast”: ’Actor’,
	“duration”: ‘120’,
	“release_date”: ‘20/10/2024’,
	"genre": "Action"
	}

2. Get Top Watched Movies

- **URL: /movies/top-watched**
- **Method: GET**

Response

Status Code	JSON Response
200 OK	{ "id": 1, "title": "Movie Name", “certificate”: ‘ A ’, “language”:’Engilsh’, “movie_cast”: ’Actor’, “duration”: ‘120’, “release_date”: ‘20/10/2024’,

	"genre": "Action" }
404 NOT FOUND	{ "error": "No movies found" }

3. Get Movies by Address

- **URL: /movies/by-address/{address_id}**
- **Method: GET**

Path Parameter

Parameter	Type	Required	Description
Address_id	int	Yes	ID of the address (theater) to fetch movies from.

Response

Status Code	JSON Response
200 OK	{ "id": 1, "title": "Movie Name", “certificate”: ‘ A ’, “language”:’Engilsh’, “movie_cast”: ’Actor’, “duration”: ‘120’, “release_date”: ‘20/10/2024’, "genre": "Action" }
404 NOT FOUND	{ "error": "No movies found" }

Get Available Movie Shows

- **URL: /available-movie-shows**
- **Method: GET**
- **Description: Fetches a list of available movie shows based on the provided theater and movie IDs.**

Query Parameter

Parameter	Type	Required	Description
theater_id	Interger	no	The ID of the theater to filter movie shows.
movie_id	Integer	no	The ID of the movie to filter movie shows.

Response

Status Code	JSON Response
200 OK	{ "id": 1, "startTime": "2025-03-15T14:00:00", "screenId": 3, "movieId": 12, "theaterId": 5 }
500 INTERNAL SERVER ERROR	{ "error": "Could not fetch movie shows" }