Movie Booking Application

By Sabarish Iyer

Contents

SUMMARY

This project implements a Movie Ticket Booking System using a microservices architecture. The application is composed of five independently deployable modules: User Module (authentication and authorization), Movie Module (movie and theatre management), Tickets Module (ticket booking and order management), API Gateway (centralized routing and security), and Eureka Server (service discovery).

The system supports role-based access control using JWT-based stateless authentication. End users can register, log in, search and view movies, and book tickets, while administrators can manage movies and monitor booking status. All APIs are documented using Swagger / Java Docs, validated using Bean Validation, and protected using Spring Security.

The application follows clean separation of concerns using controller interfaces for API contracts and controller implementations for execution logic. Business logic resides in service layers with centralized exception handling and optional caching for improved performance.

The frontend is built using Angular (standalone architecture) and communicates with backend services through the API Gateway. Unit tests, Maven build reports, and static code analysis demonstrate code quality and maintainability.

This project satisfies the required business use cases while maintaining extensibility for future enhancements.

GitHub

Backend:

- Link: https://github.com/Sabarish-2/BookAMovie-Spring-Boot

To run;

```
mvn clean package

java -jar target/*.jar
```

Frontend:

- Link: https://github.com/Sabarish-2/MovieBookingApp_Angular

To run;

```
npm install

npm run build
```

Business requirements covered

- User Registration and Login with Role-based access (Admin / Customer)
- View and Search Movies by name and theatre
- Book Tickets for a selected Movie and theatre
- Persist booked tickets and allow users to view their bookings
- Admin management of Movies (Add / Delete) and its status updates
- Centralized API routing and Security handled by API Gateway
- Exception Handling and Security
- Swagger and Java Documentation

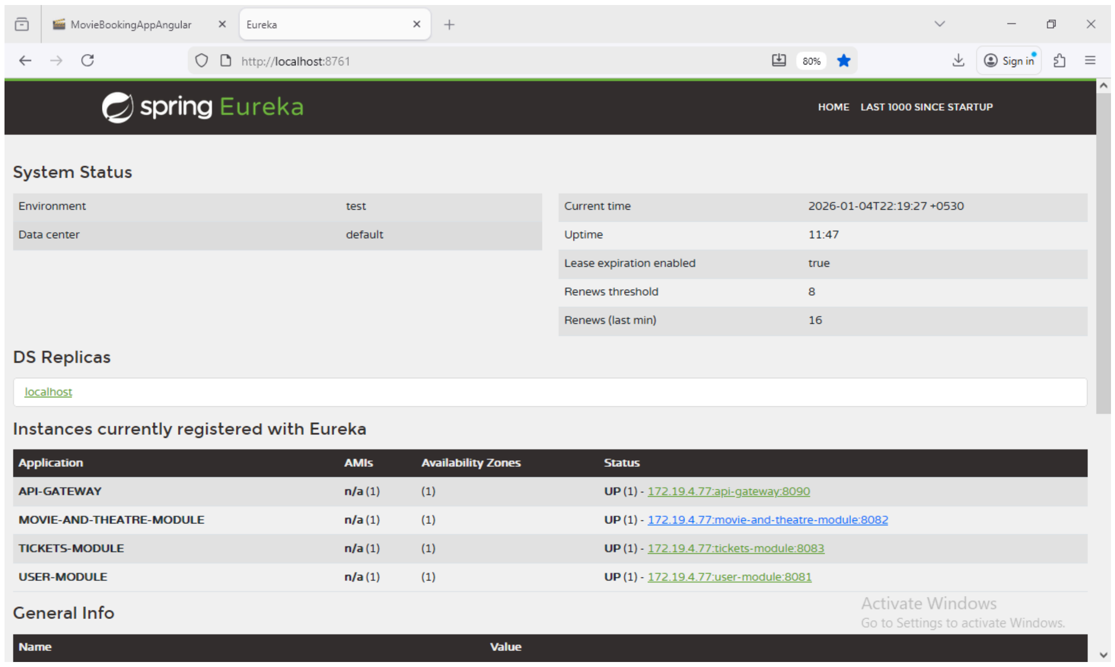
System Architecture

Micro Services List

1. Movie And Theatre Module - Handles All Movies Data and Movie CRUD Operation.
2. Tickets Module - Handles All Tickets and Ticket Booking Mechanism.
3. User Module - Handles Login, Register and Forgot Password Mechanism.
4. API Gateway - Routes All Requests to respective Micro-service.
5. Eureka Server - Stores and gives an instance of required Micro-service from load-balancer.

Eureka Server

Dashboard



System Status

Environment	test
Data center	default

Current time	2026-01-04T22:19:27 +0530
Uptime	11:47
Lease expiration enabled	true
Renews threshold	8
Renews (last min)	16

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 172.19.4.77:api-gateway:8090
MOVIE-AND-THEATRE-MODULE	n/a (1)	(1)	UP (1) - 172.19.4.77:movie-and-theatre-module:8082
TICKETS-MODULE	n/a (1)	(1)	UP (1) - 172.19.4.77:tickets-module:8083
USER-MODULE	n/a (1)	(1)	UP (1) - 172.19.4.77:user-module:8081

General Info

Name	Value

Activate Windows
Go to Settings to activate Windows.

API Gateway

All client requests are routed through the API Gateway, which handles centralized routing and security. The gateway forwards requests to MovieService, Tickets Service, and UserService based on configured routes, while Eureka Server enables service discovery.

Primary Flow of Admin Creating A Movie

Entities

Movie Entity:

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Movie {

   @EmbeddedId
   private MovieAndTheater movieAndTheatre;

   @Min(1)
   @Column(nullable = false)
   private int ticketsAllotted;

   @Enumerated(EnumType.STRING)
   private MovieStatus adminOverrideStatus;



   public Movie(MovieAndTheater movieAndTheatre, int ticketsAllotted) {
      this.ticketsAllotted = ticketsAllotted;
      this.movieAndTheatre = movieAndTheatre;
   }
}
```

Movie And Theatre Entity:

- Composite Key Requires Embeddable class

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Embeddable
```

```
public class MovieAndTheater {
    @NonNull
    private String movieName;
    @NonNull
    private String theatreName;
}
```

Data Transfer Object (DTO)

Movie DTO:

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MovieDTO {

    @NotBlank(message = "{com.moviebookingapp.movie_and_theatre_module.dtos.movieName.invalid}")
    private String movieName;

    @NotBlank(message = "{com.moviebookingapp.movie_and_theatre_module.dtos.theatreName.invalid}")
    private String theatreName;

    @Min(value = 1L, message = "{com.moviebookingapp.movie_and_theatre_module.dtos.ticketsAllotted.invalid}")
    private int ticketsAllotted;

    private Integer ticketsAvailable;

    private MovieStatus movieStatus;

    public MovieDTO(String movieName, String theatreName, int ticketsAllotted) {
        this.movieName = movieName;
        this.theatreName = theatreName;
        this.ticketsAllotted = ticketsAllotted;
    }
```

Movie Controller

Create Movie Interface:

```
@Operation(summary = "Create A New Movie")
@ApiResponse(responseCode = "201", description = "Movie Created Successfully")
@ApiResponse(responseCode = "400", description = "Validation Error in Movie Details Provided")
@ApiResponse(responseCode = "409", description = "Movie Already Exists")
ResponseEntity<MovieDTO> createMovie(@Valid @RequestBody MovieDTO movieDTO);
```

Create Movie Method Implementation:

```
/**
 * Creates a new movie.
 *
 * @param movieDTO The movie details to create.
 * @return A response entity containing the created movie.
 */
@Override
@PostMapping("create")
@PreAuthorize("hasRole('ADMIN')")
public ResponseEntity<MovieDTO> createMovie(MovieDTO movieDTO) {
    return new ResponseEntity<>(movieService.addMovie(movieDTO), HttpStatus.CREATED);
}
```

Movie Service

Add Movie Method Implementation:

```
/**
 * Adds a new movie to the system.
 *
 * @param movieDTO Data transfer object containing movie details.
 * @return The added movie as a DTO.
 * @throws MovieAlreadyExistsException if the movie already exists.
 */
@Override
public MovieDTO addMovie(MovieDTO movieDTO) {
        Movie newMovie = mapper.map(movieDTO);


        if (movieRepository.findById(newMovie.getMovieAndTheatre()).isPresent()) {
            throw new MovieAlreadyExistsException(
                    "Movie " + movieDTO.getMovieName() + " at " + movieDTO.getTheatreName() + " Already Exists!");
        }


        Movie savedMovie = movieRepository.save(newMovie);
        return mapper.map(savedMovie);
    }
```

Movie Repository

```
@Repository
public interface MovieRepository extends JpaRepository<Movie, MovieAndTheater>, JpaSpecificationExecutor<Movie> {
}
```

Custom Exception

Exception for Same Movie Details

```
public class MovieAlreadyExistsException extends CustomException {


    @Serial
    private static final long serialVersionUID = 9L;


    public MovieAlreadyExistsException(String message) {
        super(serialVersionUID, HttpStatus.CONFLICT, message);
    }
}
```

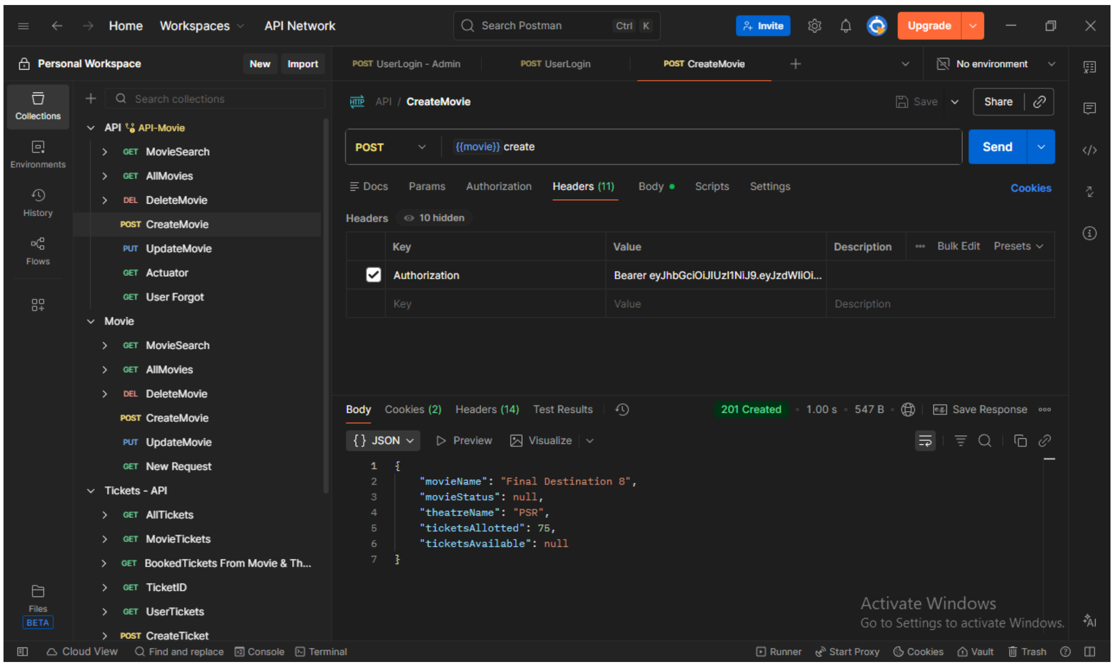
API Gateway Property

Pass all Movies related URLs to Movie Module

```
        - id: movie-and-theatre-module
          uri: lb://MOVIE-AND-THEATRE-MODULE
          predicates:
            - Path=/api/v1.0/moviebooking/movies/**
```

Postman Screenshots for Add Movie

Admin Adding Movie:



POST {{movie}} create

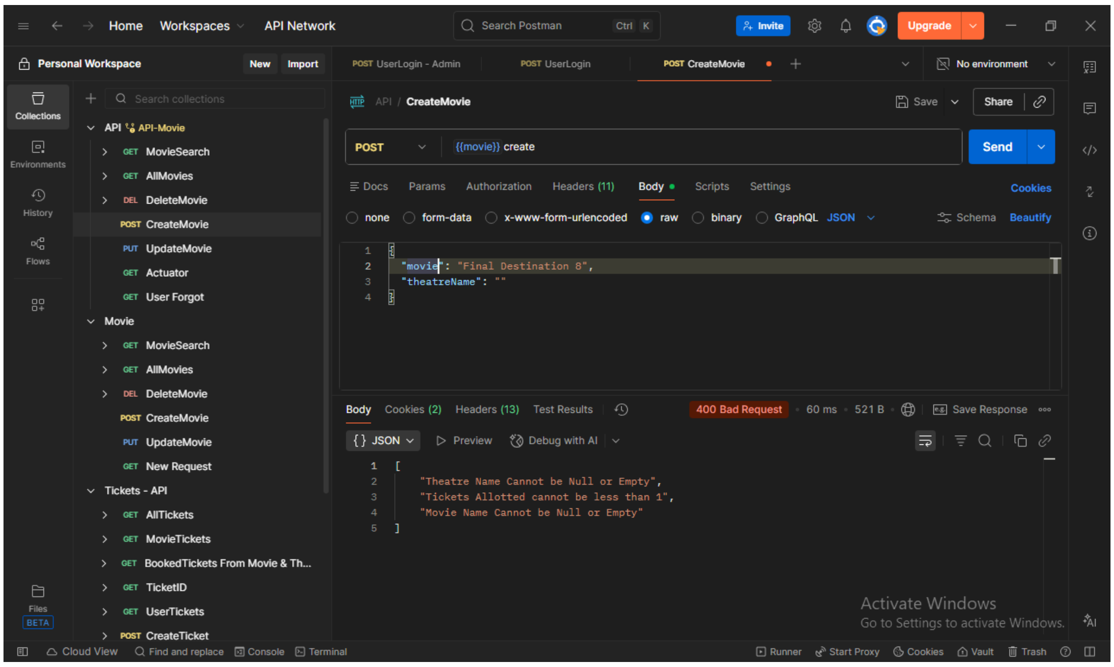
Headers (11)

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOi...	

201 Created · 1.00 s · 547 B

```
{
    "movieName": "Final Destination 8",
    "movieStatus": null,
    "theatreName": "PSR",
    "ticketsAllotted": 75,
    "ticketsAvailable": null
}
```

Admin Adding Movie with incomplete Information:



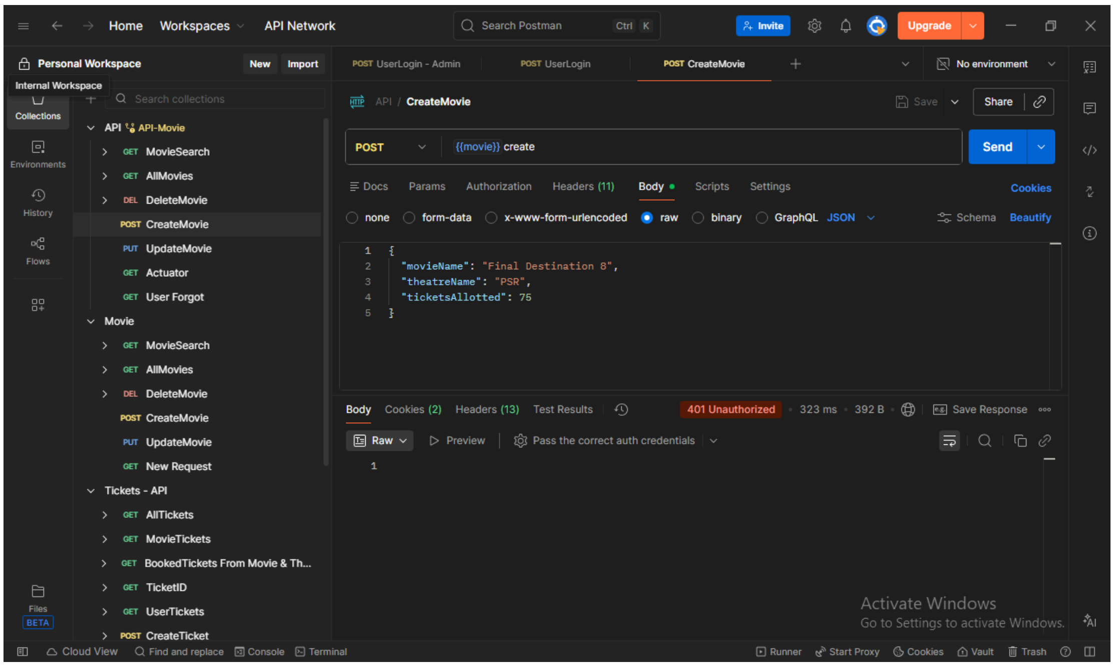
POST {{movie}} create

```
{
"movie": "Final Destination 8",
"theatreName": ""
}
```

400 Bad Request · 60 ms · 521 B

```
[
    "Theatre Name Cannot be Null or Empty",
    "Tickets Allotted cannot be less than 1",
    "Movie Name Cannot be Null or Empty"
]
```

User trying to Add Movie:


POST {{movie}} create

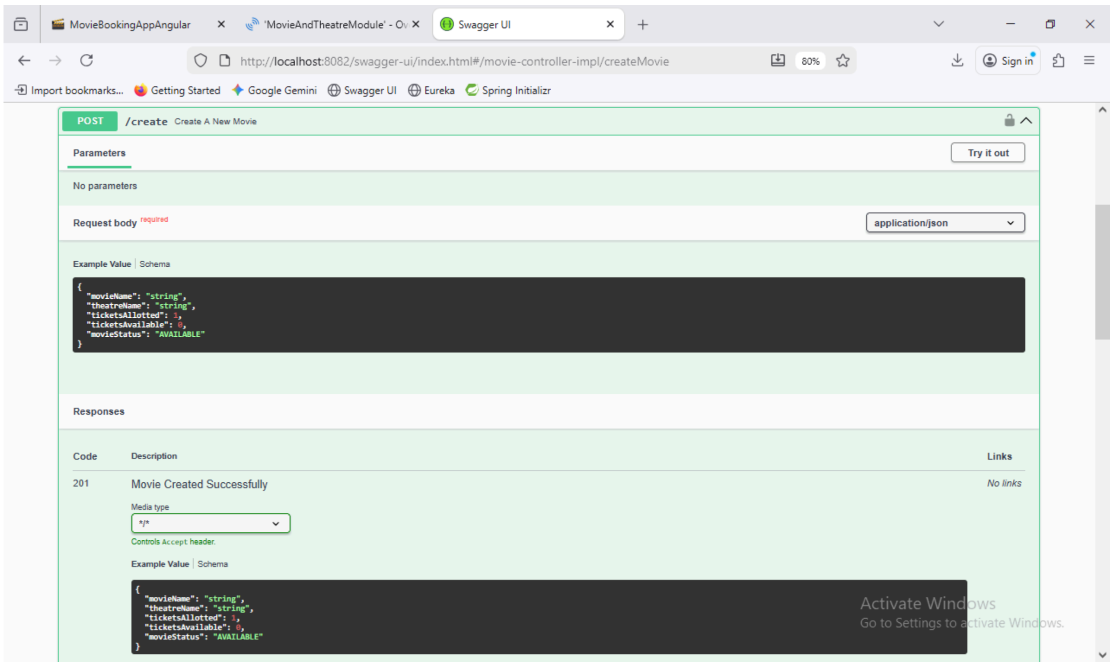
```
{
  "movieName": "Final Destination 8",
  "theatreName": "PSR",
  "ticketsAllotted": 75
}
```

401 Unauthorized · 323 ms · 392 B

Pass the correct auth credentials

Swagger Screenshots for Add Movie

Create Movie Endpoint


http://localhost:8082/swagger-ui/index.html#/movie-controller-impl/createMovie

POST /create Create A New Movie

Parameters

No parameters

Request body required — application/json

Example Value | Schema

```
{
  "movieName": "string",
  "theatreName": "string",
  "ticketsAllotted": 1,
  "ticketsAvailable": 0,
  "movieStatus": "AVAILABLE"
}
```

Responses

Code	Description	Links
201	Movie Created Successfully	No links

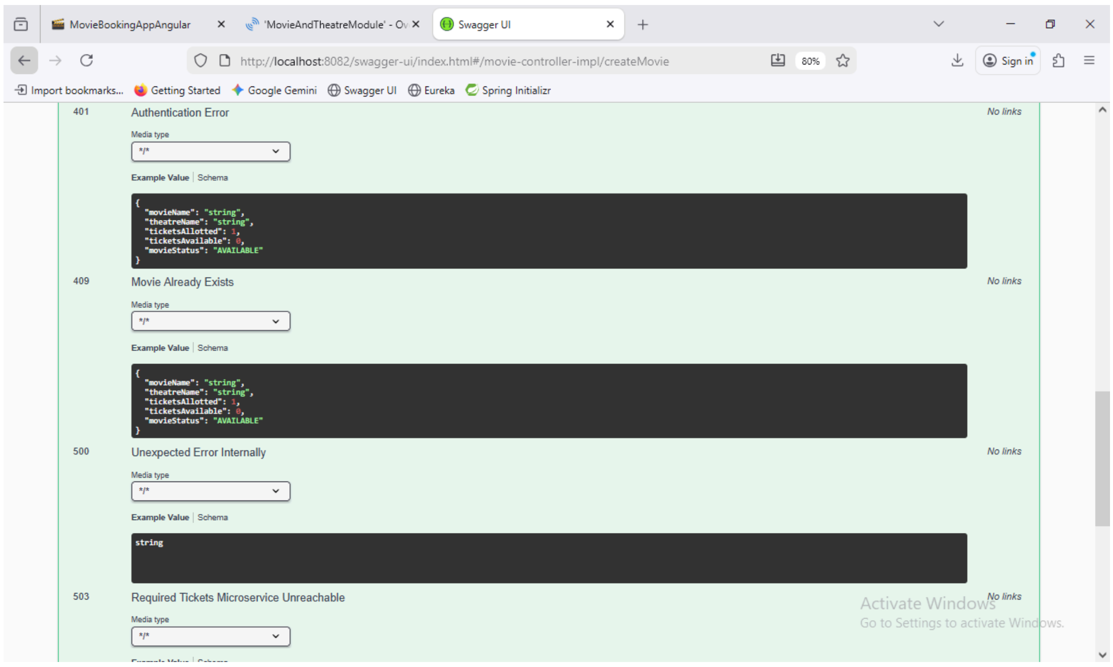
Media type: */*

Controls Accept header.

Example Value | Schema

```
{
  "movieName": "string",
  "theatreName": "string",
  "ticketsAllotted": 1,
  "ticketsAvailable": 0,
  "movieStatus": "AVAILABLE"
}
```

Responses by Create Movie



401 Authentication Error

Media type
/

Example Value | Schema

```
{
  "movieName": "string",
  "theatreName": "string",
  "ticketsAllotted": 1,
  "ticketsAvailable": 0,
  "movieStatus": "AVAILABLE"
}
```

No links

409 Movie Already Exists

Media type
/

Example Value | Schema

```
{
  "movieName": "string",
  "theatreName": "string",
  "ticketsAllotted": 1,
  "ticketsAvailable": 0,
  "movieStatus": "AVAILABLE"
}
```

No links

500 Unexpected Error Internally

Media type
/

Example Value | Schema

```
string
```

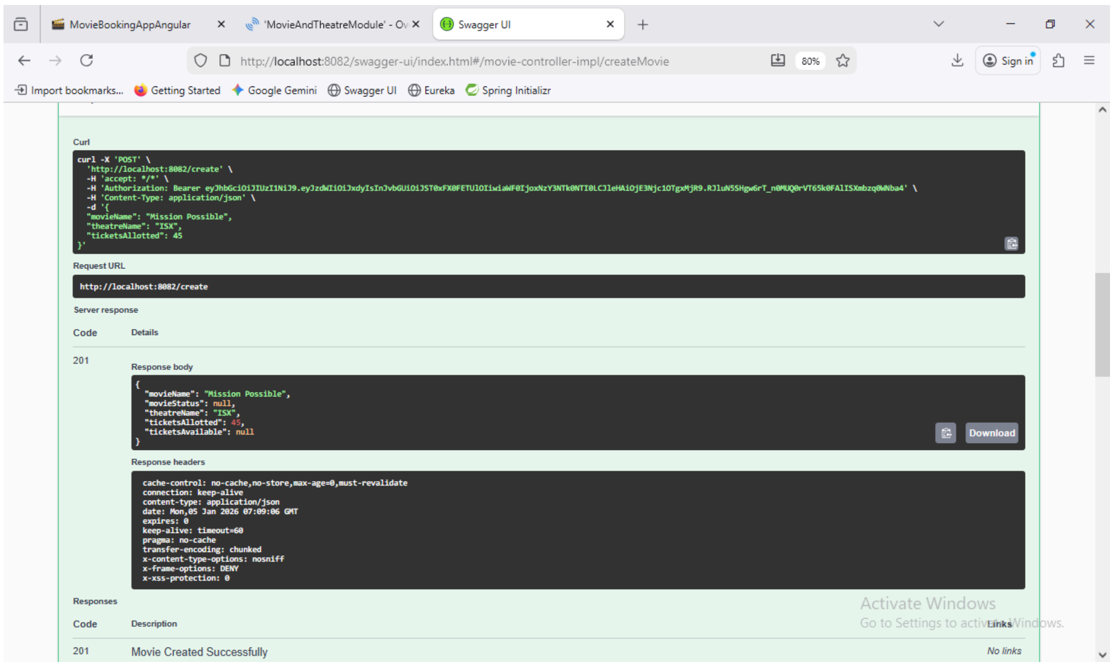
No links

503 Required Tickets Microservice Unreachable

Media type
/

No links

Successfully Created a Movie



Curl

```
curl -X 'POST' \
  'http://localhost:8082/create' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJxdyIsInJvbGUiOiJST0xFX0FETUlOIiwiaWF0IjoxNzY3NTk0NTI0LCJleHAiOjE3Njc1OTgxMjR9.RJluN5SHgw6rT_n0MUQ0rVT65k0FAlISXmbzq0WNba4' \
  -H 'Content-Type: application/json' \
  -d '{
  "movieName": "Mission Possible",
  "theatreName": "ISX",
  "ticketsAllotted": 45
}'
```

Request URL

```
http://localhost:8082/create
```

Server response

Code Details

201

Response body

```
{
  "movieName": "Mission Possible",
  "movieStatus": null,
  "theatreName": "ISX",
  "ticketsAllotted": 45,
  "ticketsAvailable": null
}
```

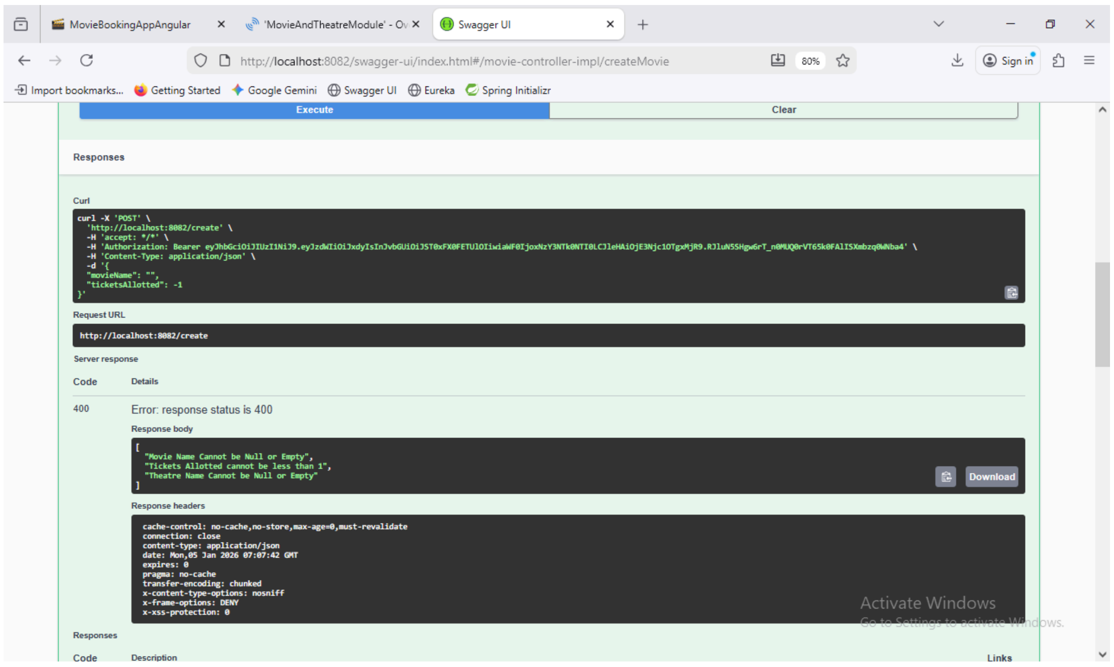
Response headers

```
 cache-control: no-cache,no-store,max-age=0,must-revalidate
 connection: keep-alive
 content-type: application/json
 date: Mon,05 Jan 2026 07:09:06 GMT
 expires: 0
 keep-alive: timeout=60
 pragma: no-cache
 transfer-encoding: chunked
 x-content-type-options: nosniff
 x-frame-options: DENY
 x-xss-protection: 0
```

Responses

Code	Description	Links
201	Movie Created Successfully	No links

Incomplete Details Provided



MovieBookingAppAngular
'MovieAndTheatreModule' - O
Swagger UI
http://localhost:8082/swagger-ui/index.html#/movie-controller-impl/createMovie
Execute
Clear
Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8082/create' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJxdyIsInJvbGUiOiJST0xFX0FETUlOIiwiaWF0IjoxNzY3NTk0NTI0LCJleHAiOjE3Njc1OTgxMjR9.RJluN5SHgw6rT_n0MUQ0rVT65k0FAlISXmbzq0WNba4' \
  -H 'Content-Type: application/json' \
  -d '{
  "movieName": "",
  "ticketsAllotted": -1
}'
```

Request URL

```
http://localhost:8082/create
```

Server response

Code 400 — Error: response status is 400

Response body

```
[
  "Movie Name Cannot be Null or Empty",
  "Tickets Allotted cannot be less than 1",
  "Theatre Name Cannot be Null or Empty"
]
```

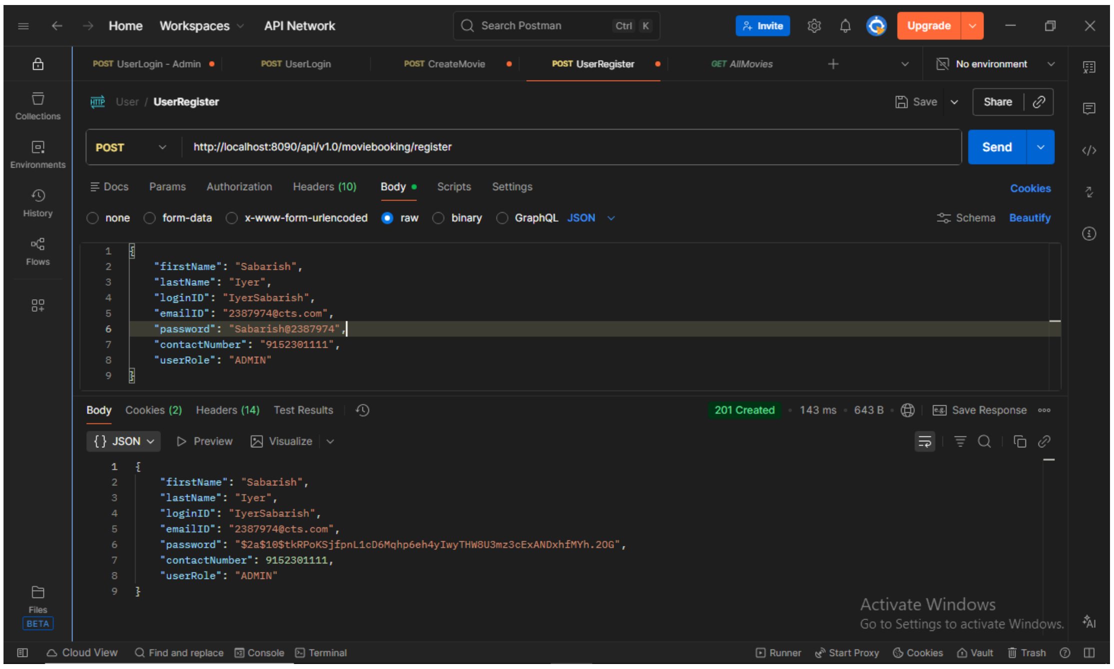
Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: close
content-type: application/json
date: Mon,05 Jan 2026 07:07:42 GMT
expires: 0
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```

Postman Screenshots for other Endpoints

Register User

Registering New User



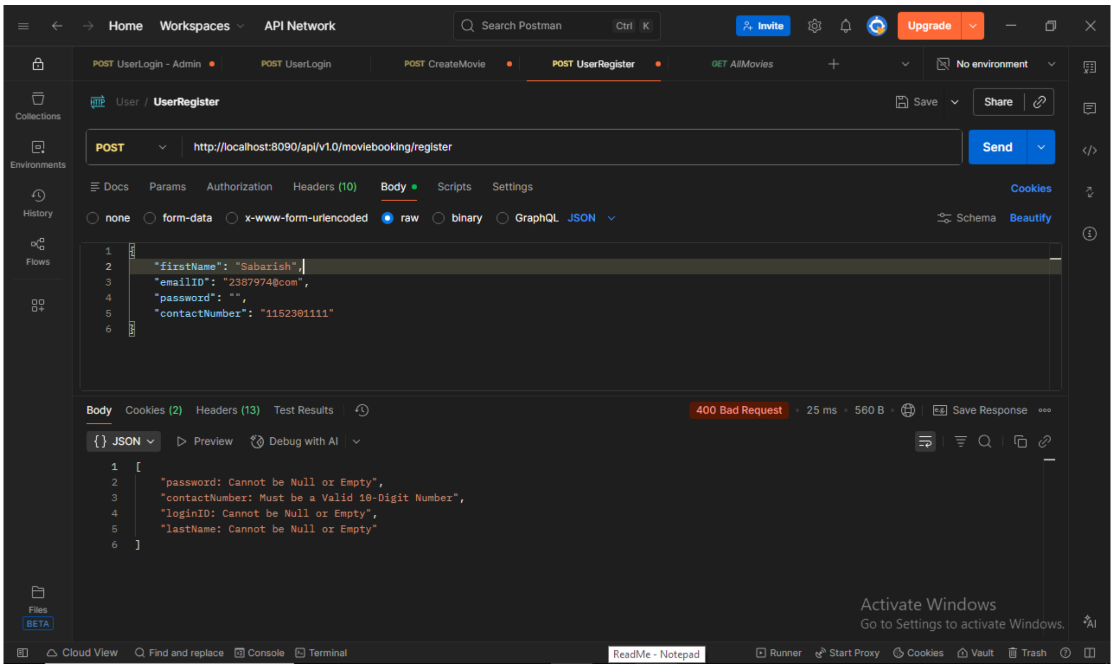
POST http://localhost:8090/api/v1.0/moviebooking/register

```
{
    "firstName": "Sabarish",
    "lastName": "Iyer",
    "loginID": "IyerSabarish",
    "emailID": "2387974@cts.com",
    "password": "Sabarish@2387974",
    "contactNumber": "9152301111",
    "userRole": "ADMIN"
}
```

Body — 201 Created · 143 ms · 643 B

```
{
    "firstName": "Sabarish",
    "lastName": "Iyer",
    "loginID": "IyerSabarish",
    "emailID": "2387974@cts.com",
    "password": "$2a$10$tkRPoKSjfpnL1cD6Mqhp6eh4yIwyTHW8U3mz3cExANDxhfMYh.2OG",
    "contactNumber": 9152301111,
    "userRole": "ADMIN"
}
```

Register Validation Check



POST http://localhost:8090/api/v1.0/moviebooking/register

Body: raw JSON

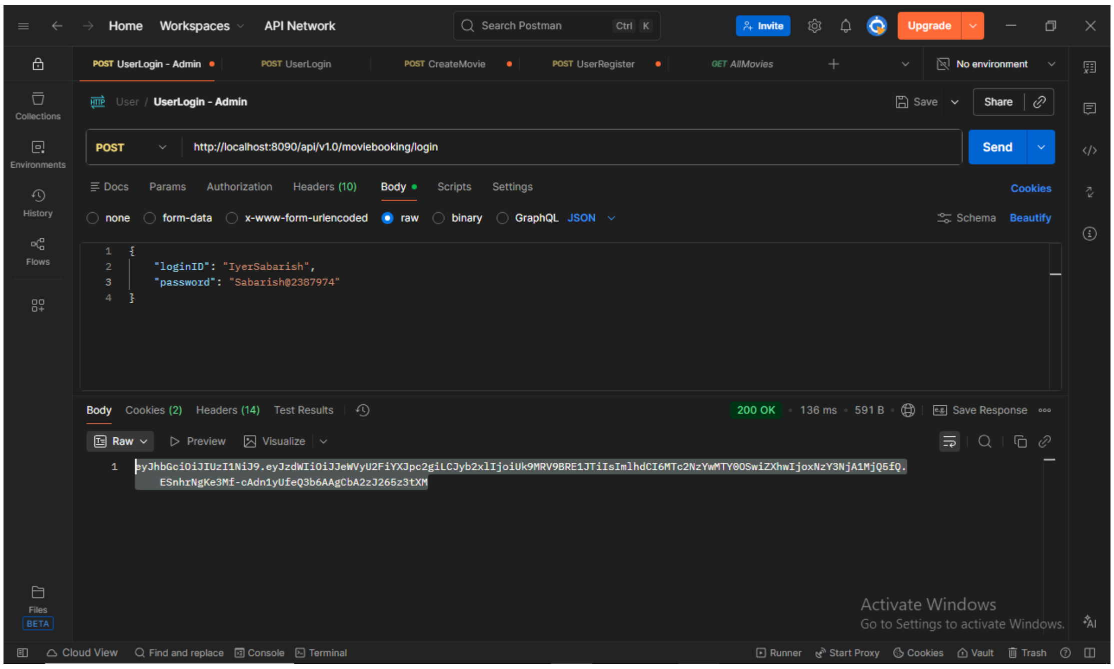
```
{
    "firstName": "Sabarish",
    "emailID": "2387974@com",
    "password": "",
    "contactNumber": "1152301111"
}
```

400 Bad Request · 25 ms · 560 B

```
[
    "password: Cannot be Null or Empty",
    "contactNumber: Must be a Valid 10-Digit Number",
    "loginID: Cannot be Null or Empty",
    "lastName: Cannot be Null or Empty"
]
```

Login User

Logging an Existing User In.



POST http://localhost:8090/api/v1.0/moviebooking/login

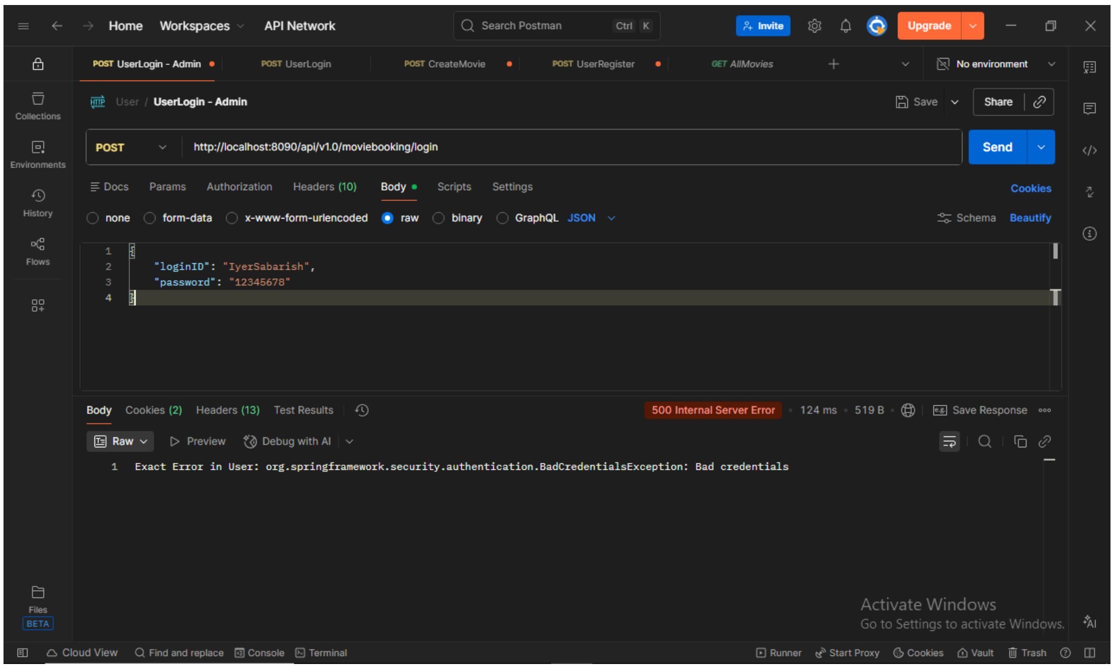
Body: raw JSON

```
{
    "loginID": "IyerSabarish",
    "password": "Sabarish@2387974"
}
```

200 OK · 136 ms · 591 B

```
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJJeWVyU2FiYXJpc2giLCJyb2xlIjoiUk9MRV9BRE1JTiIsImlhdCI6MTc2NzYwMTY0OSwiZXhwIjoxNzY3NjA1MjQ5fQ.
    ESnhrNgKe3Mf-cAdn1yUfeQ3b6AAgCbA2zJ265z3tXM
```

Login User with wrong Password:



POST http://localhost:8090/api/v1.0/moviebooking/login

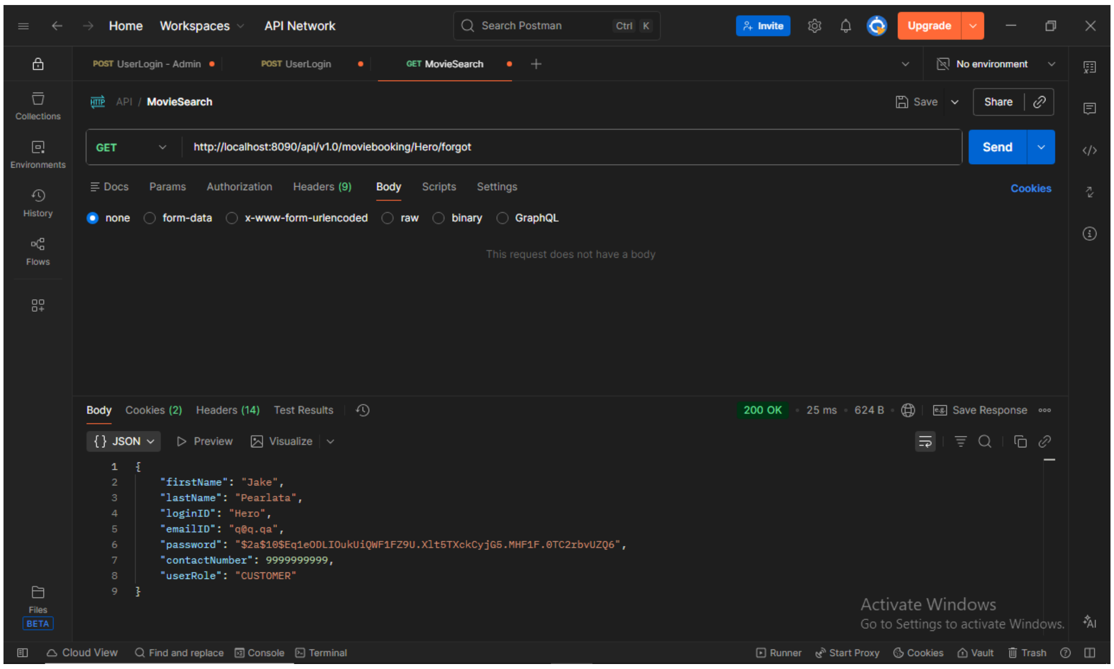
Body: raw [✓] JSON

```
{
    "loginID": "IyerSabarish",
    "password": "12345678"
}
```

```
Exact Error in User: org.springframework.security.authentication.BadCredentialsException: Bad credentials
```

Forgot Password

Forgot password Check



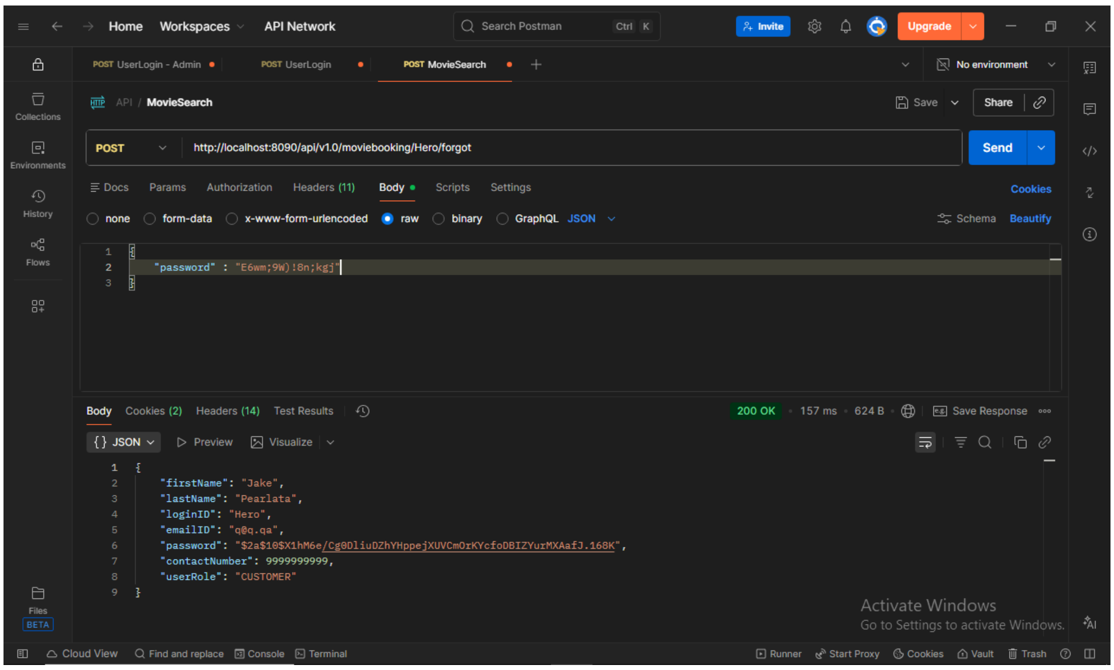
GET http://localhost:8090/api/v1.0/moviebooking/Hero/forgot

Body: [✓ none]

This request does not have a body

```
{
    "firstName": "Jake",
    "lastName": "Pearlata",
    "loginID": "Hero",
    "emailID": "q@q.qa",
    "password": "$2a$10$Eq1eODLIOukUiQWF1FZ9U.Xlt5TXckCyjG5.MHF1F.0TC2rbvUZQ6",
    "contactNumber": 9999999999,
    "userRole": "CUSTOMER"
}
```

Password Reset



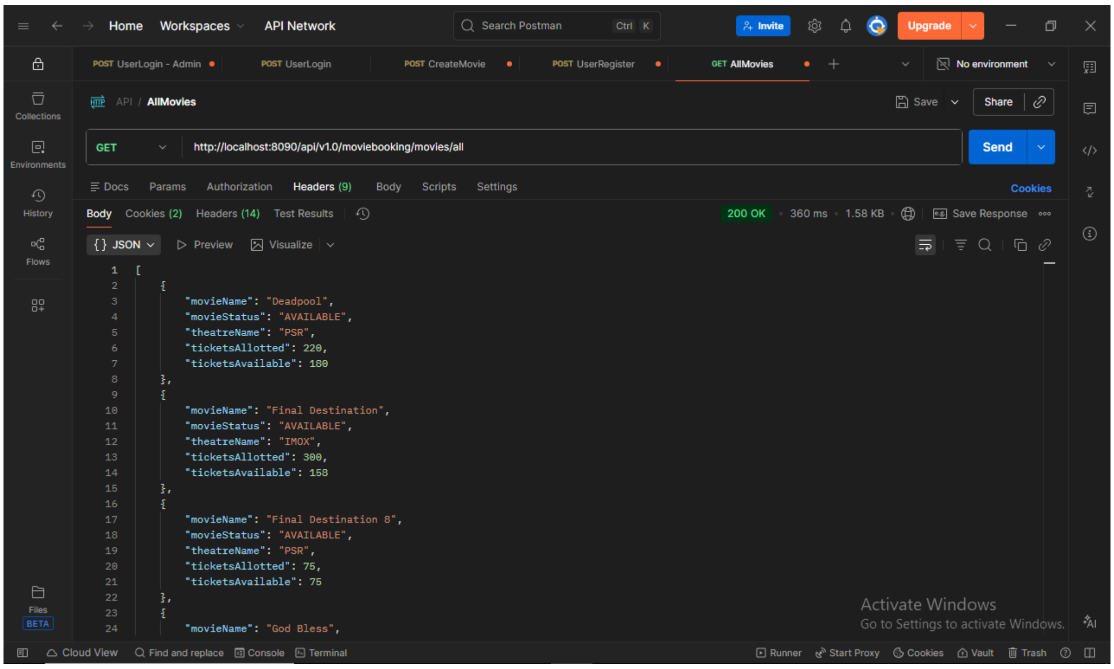
POST http://localhost:8090/api/v1.0/moviebooking/Hero/forgot

```
{
    "password" : "E6wm;9W)!8n;kgj"
}
```

200 OK · 157 ms · 624 B

```
{
    "firstName": "Jake",
    "lastName": "Pearlata",
    "loginID": "Hero",
    "emailID": "q@q.qa",
    "password": "$2a$10$X1hM6e/Cg0DliuDZhYHppejXUVCmOrKYcfoDBIZYurMXAafJ.168K",
    "contactNumber": 9999999999,
    "userRole": "CUSTOMER"
}
```

Get All Movies

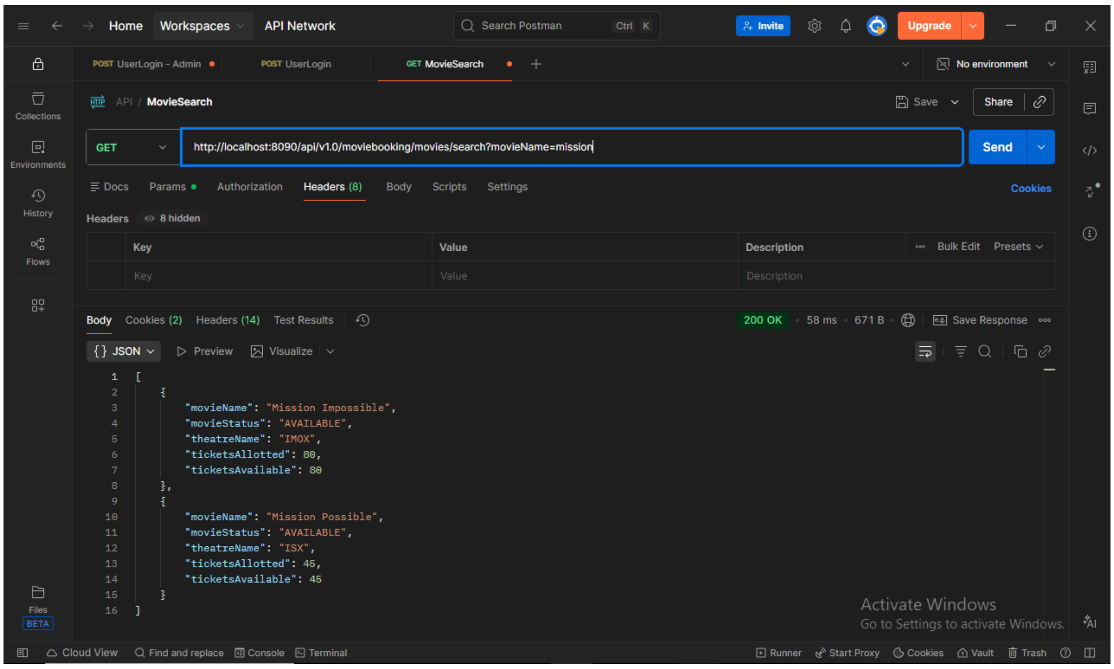


GET http://localhost:8090/api/v1.0/moviebooking/movies/all

200 OK · 360 ms · 1.58 KB

```
[
    {
        "movieName": "Deadpool",
        "movieStatus": "AVAILABLE",
        "theatreName": "PSR",
        "ticketsAllotted": 220,
        "ticketsAvailable": 180
    },
    {
        "movieName": "Final Destination",
        "movieStatus": "AVAILABLE",
        "theatreName": "IMOX",
        "ticketsAllotted": 300,
        "ticketsAvailable": 158
    },
    {
        "movieName": "Final Destination 8",
        "movieStatus": "AVAILABLE",
        "theatreName": "PSR",
        "ticketsAllotted": 75,
        "ticketsAvailable": 75
    },
    {
        "movieName": "God Bless",
```

Search for a Movie

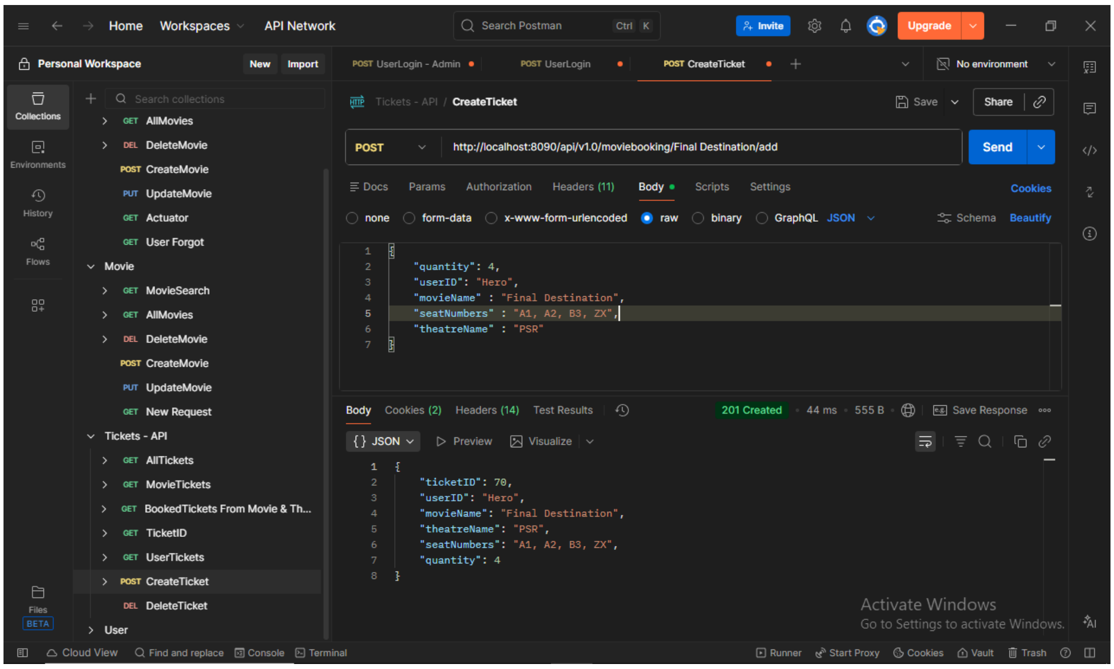


GET http://localhost:8090/api/v1.0/moviebooking/movies/search?movieName=mission

200 OK • 58 ms • 671 B

```json
[
    {
        "movieName": "Mission Impossible",
        "movieStatus": "AVAILABLE",
        "theatreName": "IMOX",
        "ticketsAllotted": 80,
        "ticketsAvailable": 80
    },
    {
        "movieName": "Mission Possible",
        "movieStatus": "AVAILABLE",
        "theatreName": "ISX",
        "ticketsAllotted": 45,
        "ticketsAvailable": 45
    }
]
```

Book Ticket for Movie



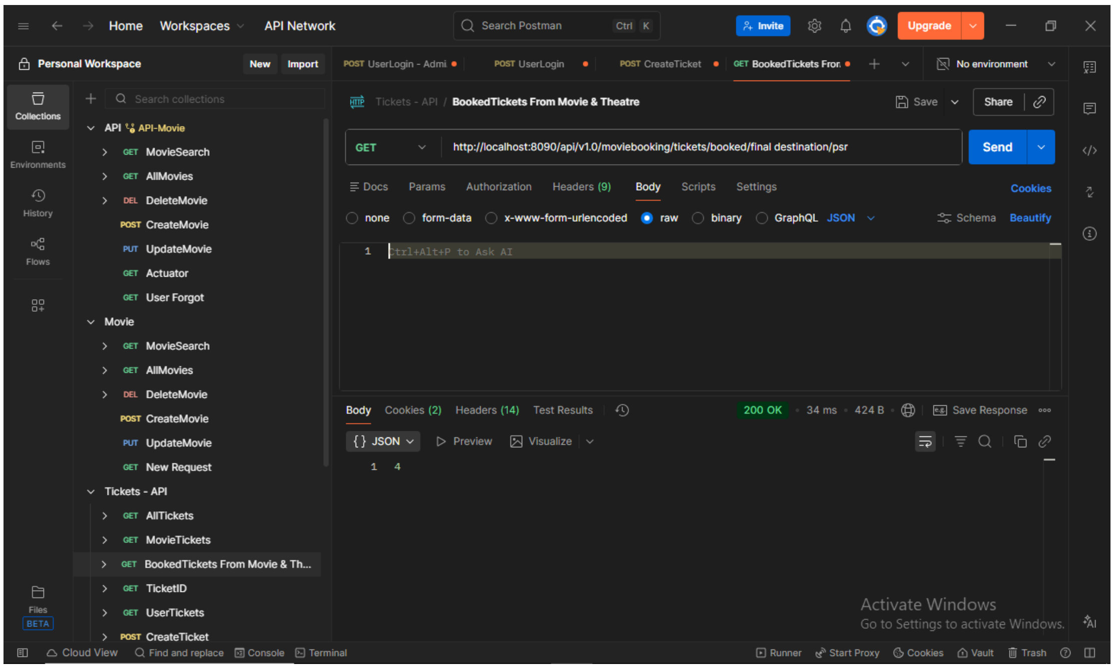
POST http://localhost:8090/api/v1.0/moviebooking/Final Destination/add

```json
{
    "quantity": 4,
    "userID": "Hero",
    "movieName" : "Final Destination",
    "seatNumbers" : "A1, A2, B3, ZX",
    "theatreName" : "PSR"
}
```

201 Created • 44 ms • 555 B

```json
{
    "ticketID": 70,
    "userID": "Hero",
    "movieName": "Final Destination",
    "theatreName": "PSR",
    "seatNumbers": "A1, A2, B3, ZX",
    "quantity": 4
}
```

Get All Tickets for Movie



Tickets - API / BookedTickets From Movie & Theatre

GET http://localhost:8090/api/v1.0/moviebooking/tickets/booked/final destination/psr

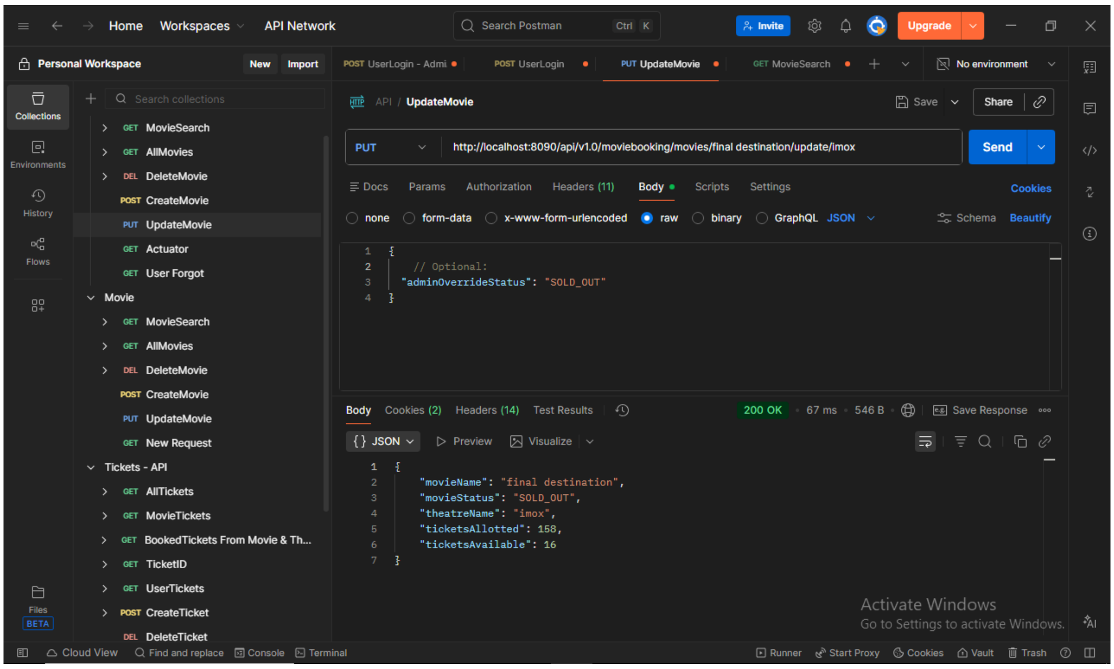
Body: raw JSON

```
Ctrl+Alt+P to Ask AI
```

Body Cookies (2) Headers (14) Test Results — 200 OK · 34 ms · 424 B

```
4
```

Movie Status Change:



API / UpdateMovie

PUT http://localhost:8090/api/v1.0/moviebooking/movies/final destination/update/imox

Body: raw JSON

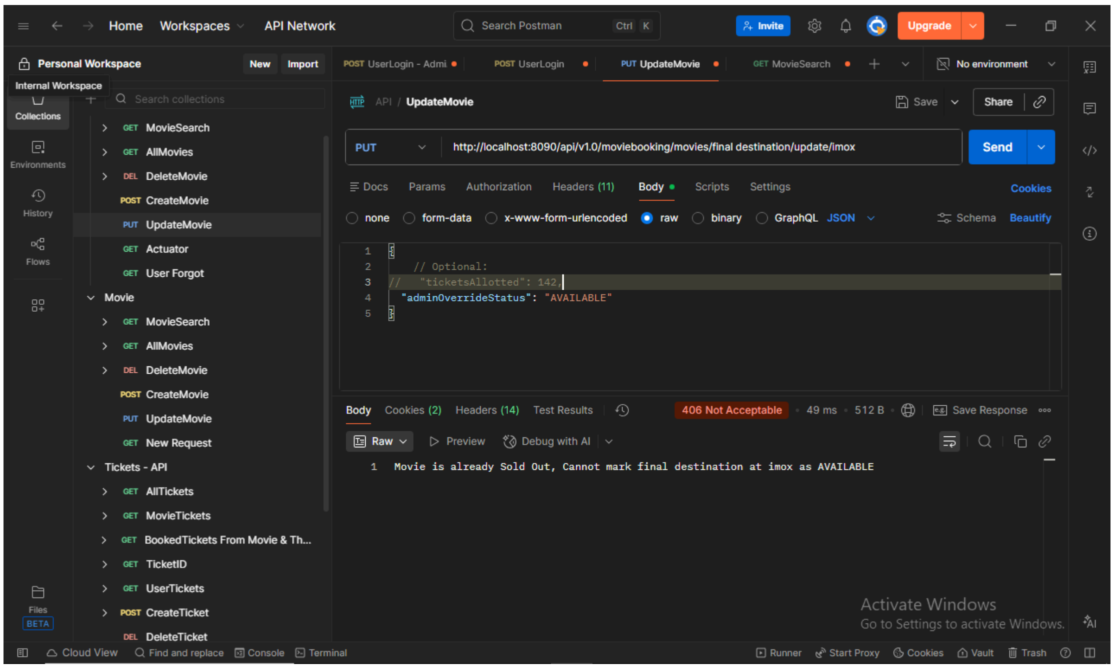
```
{
   // Optional:
  "adminOverrideStatus": "SOLD_OUT"
}
```

Body Cookies (2) Headers (14) Test Results — 200 OK · 67 ms · 546 B

```
{
    "movieName": "final destination",
    "movieStatus": "SOLD_OUT",
    "theatreName": "imox",
    "ticketsAllotted": 158,
    "ticketsAvailable": 16
}
```

Invalid Movie Update

Invalid Status Change



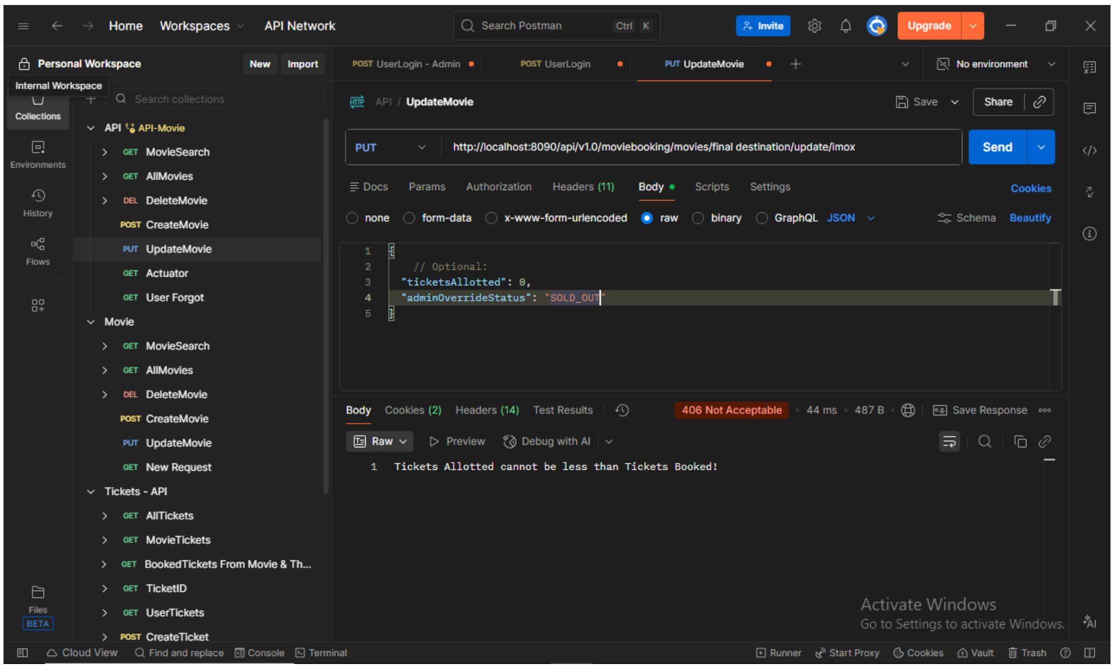
PUT http://localhost:8090/api/v1.0/moviebooking/movies/final destination/update/imox

```
{
    // Optional:
//    "ticketsAllotted": 142,
  "adminOverrideStatus": "AVAILABLE"
}
```

406 Not Acceptable · 49 ms · 512 B

```
Movie is already Sold Out, Cannot mark final destination at imox as AVAILABLE
```

Invalid Number of Tickets Allotted



PUT http://localhost:8090/api/v1.0/moviebooking/movies/final destination/update/imox

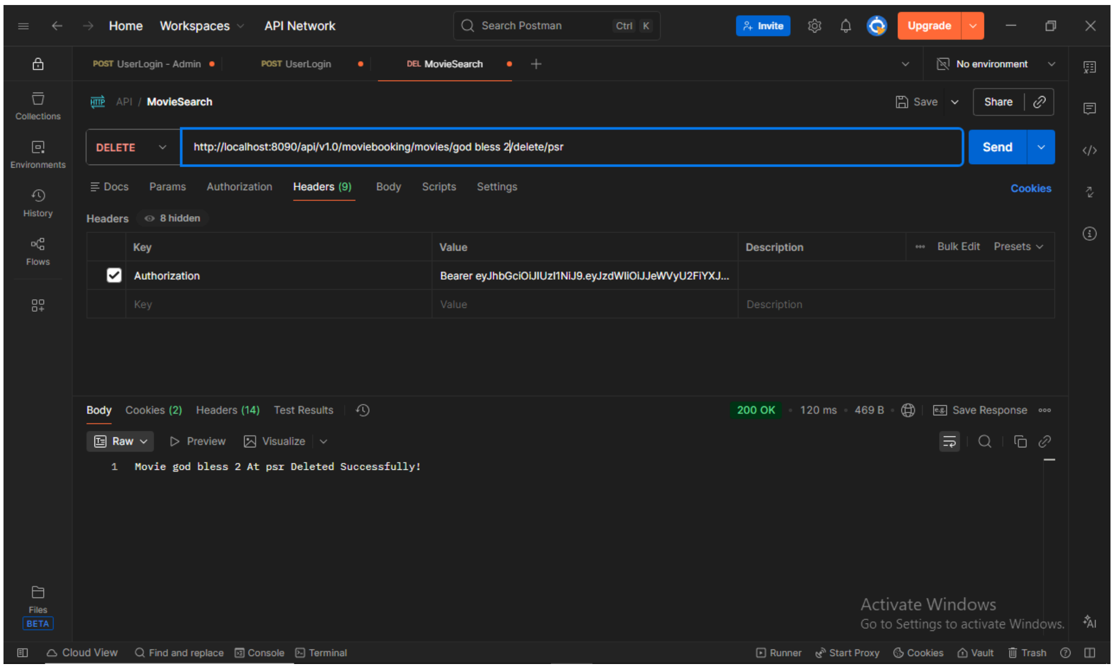
```
{
    // Optional:
  "ticketsAllotted": 0,
  "adminOverrideStatus": "SOLD_OUT"
}
```

406 Not Acceptable · 44 ms · 487 B

```
Tickets Allotted cannot be less than Tickets Booked!
```

Delete Movie

By Admin



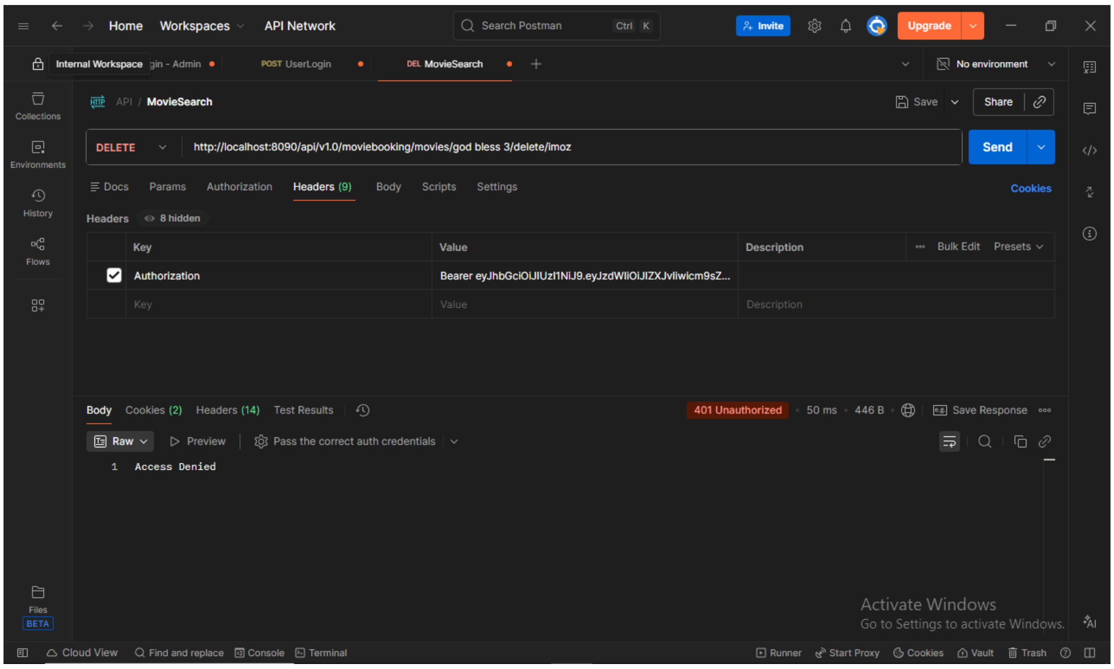
DELETE http://localhost:8090/api/v1.0/moviebooking/movies/god bless 2/delete/psr

	Key	Value	Description
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJJeWVyU2FiYXJ...	
	Key	Value	Description

200 OK · 120 ms · 469 B

```
Movie god bless 2 At psr Deleted Successfully!
```

By User



DELETE http://localhost:8090/api/v1.0/moviebooking/movies/god bless 3/delete/imoz

	Key	Value	Description
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJIZXJvliwicm9sZ...	
	Key	Value	Description

401 Unauthorized · 50 ms · 446 B

Pass the correct auth credentials

```
Access Denied
```

Caching and session management

Caching Implemented

- Tickets cannot be modified, hence can be cached for better performance.

```
/**
 * Retrieves a ticket by its ID.
 *
 * @param ticketID The ID of the ticket to retrieve.
 * @return The ticket as a DTO.
 * @throws TicketNotFoundException if the ticket is not found.
 */
@Override
@Cacheable("ticket")
public TicketDTO getTicketByID(Long ticketID) {
    Ticket ticket = ticketRepository.findById(ticketID)
            .orElseThrow(() -> new TicketNotFoundException(ticketID));
    return mapper.map(ticket);
}
```

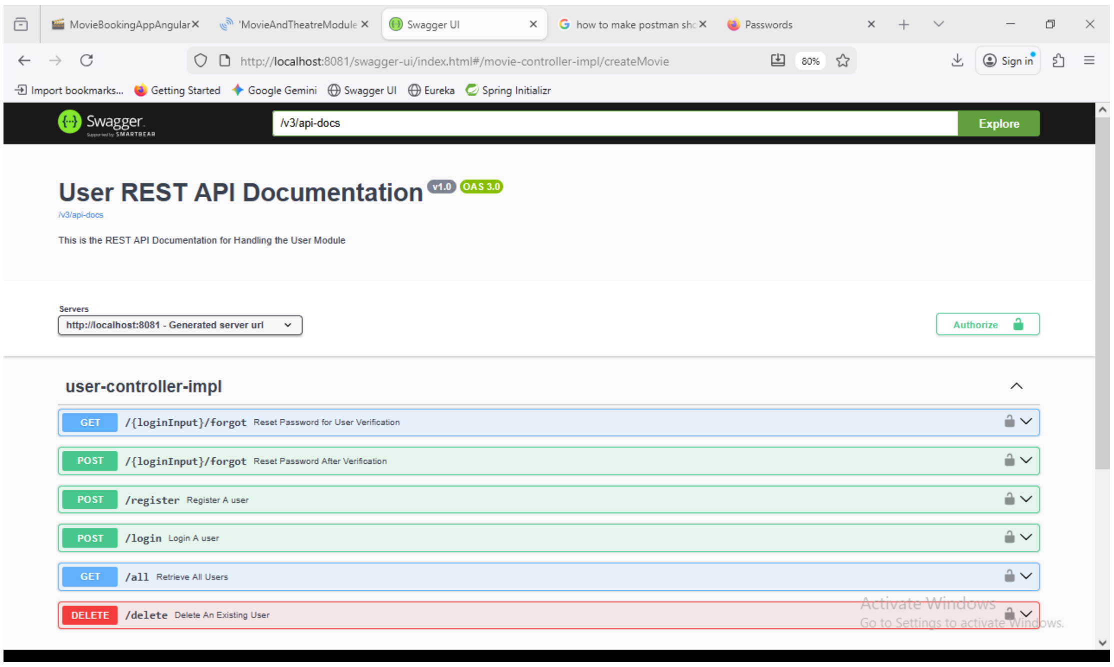
Session Management

- Stateless Session Tokens are used in security.

```
httpSecurity
                .csrf(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(auth -> auth
                                                        .requestMatchers("/swagger-ui/**", "/swagger-ui.html",
"/v3/api-docs/**").permitAll()
                        .requestMatchers(HttpMethod.GET, "/tickets/booked/**").permitAll()
                        .anyRequest().authenticated()
                )
                                                                                .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
```

SWAGGER

User Documentation


http://localhost:8081/swagger-ui/index.html#/movie-controller-impl/createMovie

/v3/api-docs

User REST API Documentation v1.0 OAS 3.0

/v3/api-docs

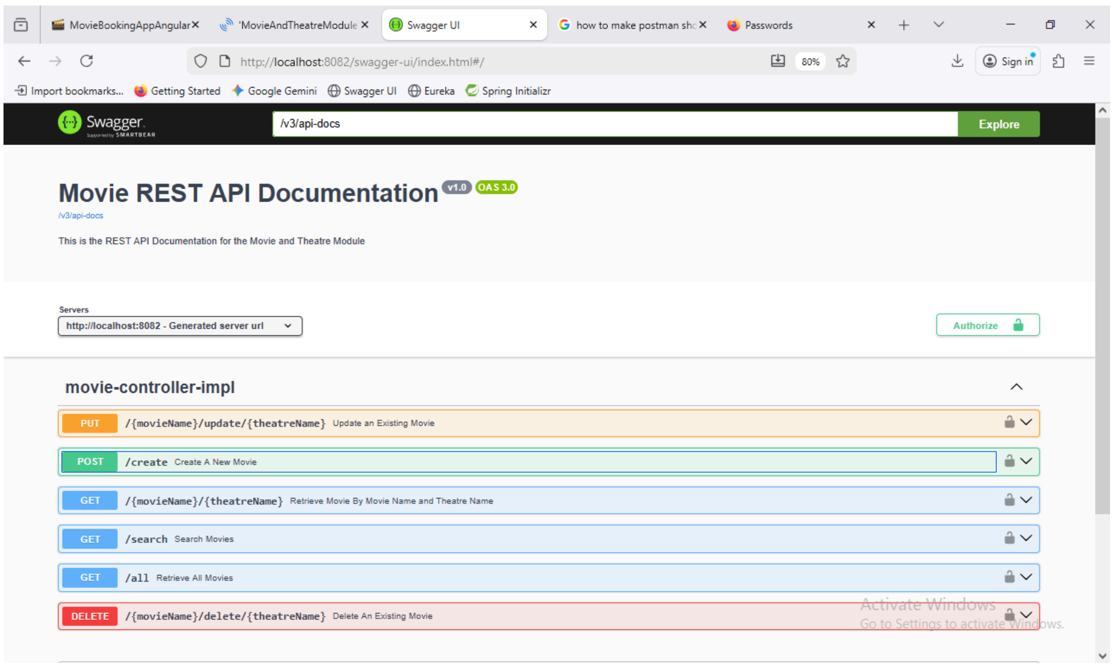
This is the REST API Documentation for Handling the User Module

Servers: http://localhost:8081 - Generated server url

user-controller-impl

- GET /{loginInput}/forgot Reset Password for User Verification
- POST /{loginInput}/forgot Reset Password After Verification
- POST /register Register A user
- POST /login Login A user
- GET /all Retrieve All Users
- DELETE /delete Delete An Existing User

Movies Documentation


http://localhost:8082/swagger-ui/index.html#/

/v3/api-docs

Movie REST API Documentation v1.0 OAS 3.0

/v3/api-docs

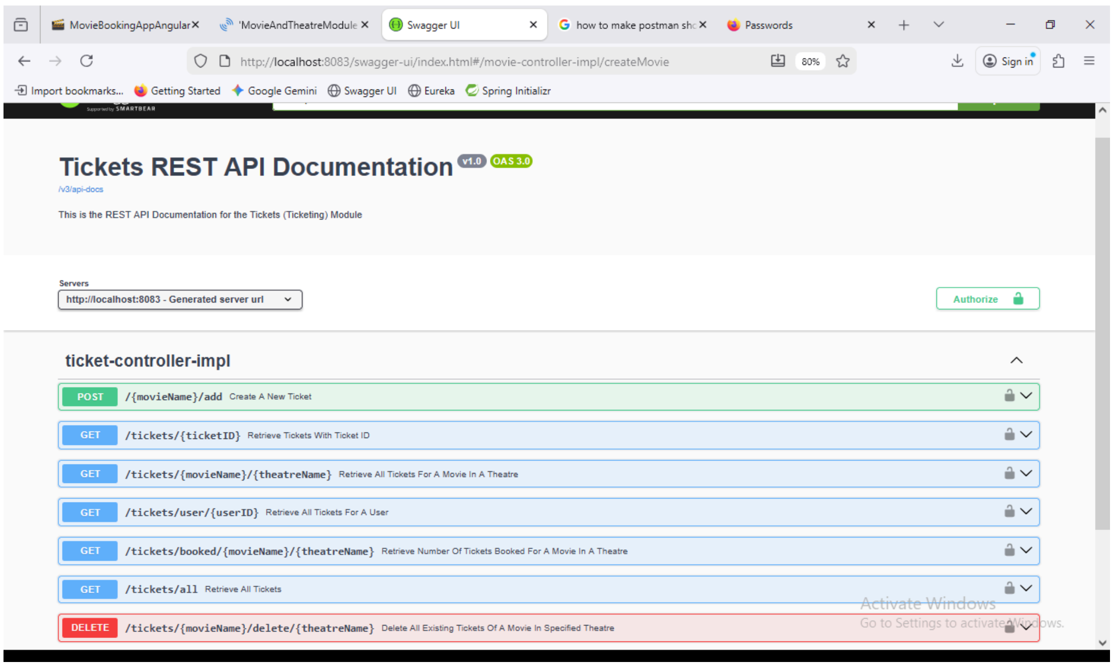
This is the REST API Documentation for the Movie and Theatre Module

Servers: http://localhost:8082 - Generated server url

movie-controller-impl

- PUT /{movieName}/update/{theatreName} Update an Existing Movie
- POST /create Create A New Movie
- GET /{movieName}/{theatreName} Retrieve Movie By Movie Name and Theatre Name
- GET /search Search Movies
- GET /all Retrieve All Movies
- DELETE /{movieName}/delete/{theatreName} Delete An Existing Movie

Tickets Documentation



Tickets REST API Documentation v1.0 OAS 3.0

/v3/api-docs

This is the REST API Documentation for the Tickets (Ticketing) Module

Servers
http://localhost:8083 - Generated server url

Authorize

ticket-controller-impl

```
POST    /{movieName}/add  Create A New Ticket
GET     /tickets/{ticketID}  Retrieve Tickets With Ticket ID
GET     /tickets/{movieName}/{theatreName}  Retrieve All Tickets For A Movie In A Theatre
GET     /tickets/user/{userID}  Retrieve All Tickets For A User
GET     /tickets/booked/{movieName}/{theatreName}  Retrieve Number Of Tickets Booked For A Movie In A Theatre
GET     /tickets/all  Retrieve All Tickets
DELETE  /tickets/{movieName}/delete/{theatreName}  Delete All Existing Tickets Of A Movie In Specified Theatre
```

Exception - Custom & Handling

Exception User Already Exists:

```
public class UserAlreadyExistsException extends CustomException {

  @Serial
  private static final long serialVersionUID = 9L;

  public UserAlreadyExistsException(String message) {
    super(serialVersionUID, HttpStatus.CONFLICT, message);
  }
}
```

Custom Exception Super Class for all Custom Exceptions:

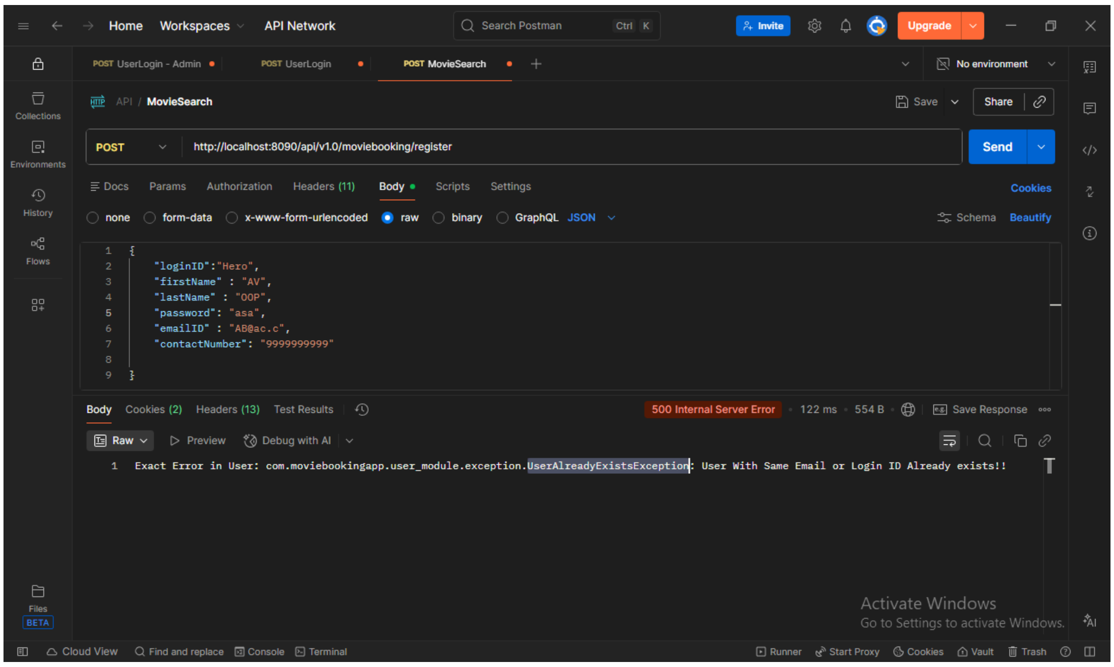
```
@Getter
public abstract class CustomException extends RuntimeException {
    private final long serialVersionUIDPerException;

    private final HttpStatus status;

    private final String message;

     protected CustomException(long serialVersionUIDPerException, HttpStatus status, String message)
{

        this.serialVersionUIDPerException = serialVersionUIDPerException;

        this.status = status;

        this.message = message;

    }

}
```

Global Exception Handler

- To handle all Exceptions and show Logs instead of throwing errors

```
    // Handle Custom Errors

    @ExceptionHandler(CustomException.class)

    public ResponseEntity<String> handleCustomExceptions(CustomException ex) {

        return new ResponseEntity<>(ex.getMessage(), ex.getStatus());

    }


    // Handle All other Errors

    @ExceptionHandler(Exception.class)

    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)

    public String handleOtherExceptions(Exception ex) {

        return "Exact Error in Movie: " + ex;

    }
```

Postman Showing Custom Exception



POST http://localhost:8090/api/v1.0/moviebooking/register

```
{
    "loginID":"Hero",
    "firstName" : "AV",
    "lastName" : "OOP",
    "password": "asa",
    "emailID" : "AB@ac.c",
    "contactNumber": "9999999999"

}
```

500 Internal Server Error

```
Exact Error in User: com.moviebookingapp.user_module.exception.UserAlreadyExistsException: User With Same Email or Login ID Already exists!!
```

Testing and Code Quality

Service Test

Get By ID

```
@Test
@DisplayName("GetMovieByID-Positive")
void test_GetMovieByID_positive() {
    when(movieRepository.findById(movieAndTheater)).thenReturn(Optional.of(movie));
    when(mapper.map(movie)).thenReturn(movieDTO);
    when(ticketsClient.getBookedTickets(any(), any())).thenReturn(ResponseEntity.ok(1L));

    MovieDTO actualMovie = movieService.getMovieByID(movieName, theatreName);

    assertEquals(movieDTO, actualMovie);
}

@Test
@DisplayName("GetMovieByID-Negative-MovieNotFound")
void test_GetMovieByID_negative_movieNotFound() {
    when(movieRepository.findById(movieAndTheater)).thenReturn(Optional.empty());
```

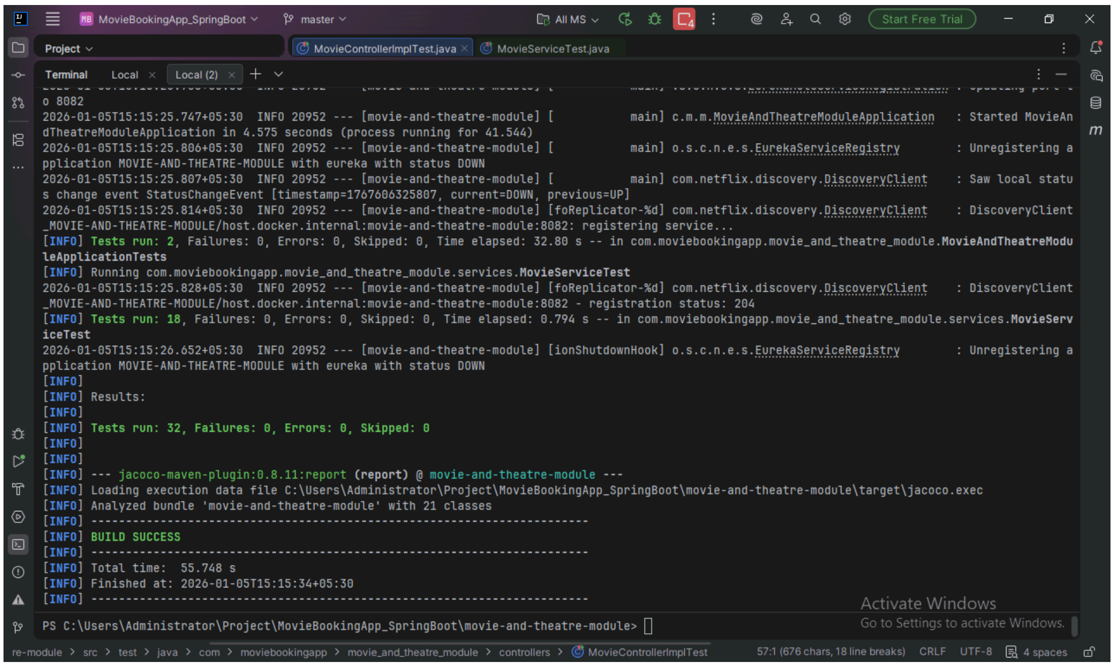
```
        assertThrows(MovieNotFoundException.class, () -> movieService.getMovieByID(movieName,
theatreName));
    }

    @Test
    @DisplayName("GetMovieByID-Negative-FeignRuntimeInPrivateMethod")
    void test_GetMovieByID_negative_feignRuntimeInPrivateMethod() {
        when(movieRepository.findById(movieAndTheater)).thenReturn(Optional.of(movie));
        when(mapper.map(movie)).thenReturn(movieDTO);
        when(ticketsClient.getBookedTickets(movieName, theatreName)).thenReturn(null);

        assertThrows(RuntimeException.class, () -> movieService.getMovieByID(movieName,
theatreName));
    }
```

Controller Test

View All:

```
    @Test
    @DisplayName("ViewAllMovies-Positive")
    void viewAllMovies_positive() {
        when(movieService.getAllMovies()).thenReturn(List.of(movieDTO));

        ResponseEntity<List<MovieDTO>> response = movieController.viewAllMovies();

        assertEquals(HttpStatus.OK, response.getStatusCode());
        assertEquals(List.of(movieDTO), response.getBody());
    }

    @Test
    @DisplayName("ViewAllMovies-Negative-MovieNotFound")
    void viewAllMovies_negative_movieNotFound() {
        when(movieService.getAllMovies()).thenThrow(new MovieNotFoundException());

        assertThrows(MovieNotFoundException.class, () -> movieController.viewAllMovies());
    }
```

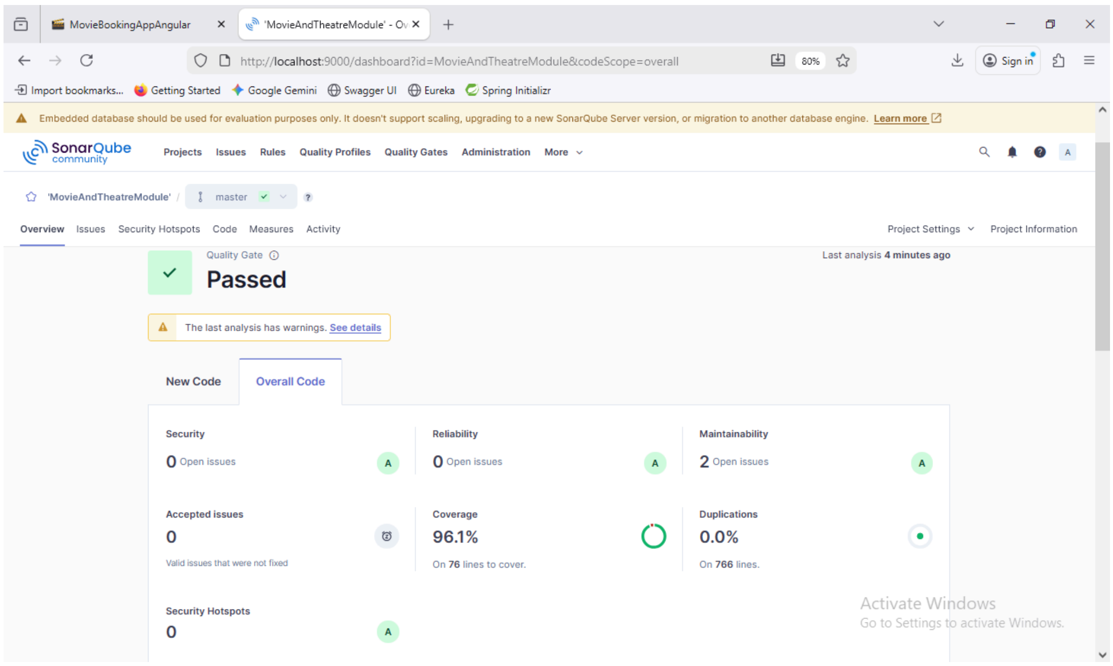
Maven Test:



```
o 8082
2026-01-05T15:15:25.747+05:30  INFO 20952 --- [movie-and-theatre-module] [           main] c.m.m.MovieAndTheatreModuleApplication   : Started MovieAn
dTheatreModuleApplication in 4.575 seconds (process running for 41.544)
2026-01-05T15:15:25.806+05:30  INFO 20952 --- [movie-and-theatre-module] [           main] o.s.c.n.e.s.EurekaServiceRegistry        : Unregistering a
pplication MOVIE-AND-THEATRE-MODULE with eureka with status DOWN
2026-01-05T15:15:25.807+05:30  INFO 20952 --- [movie-and-theatre-module] [           main] com.netflix.discovery.DiscoveryClient    : Saw local statu
s change event StatusChangeEvent [timestamp=1767606325807, current=DOWN, previous=UP]
2026-01-05T15:15:25.814+05:30  INFO 20952 --- [movie-and-theatre-module] [foReplicator-%d] com.netflix.discovery.DiscoveryClient    : DiscoveryClient
_MOVIE-AND-THEATRE-MODULE/host.docker.internal:movie-and-theatre-module:8082: registering service...
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 32.80 s -- in com.moviebookingapp.movie_and_theatre_module.MovieAndTheatreModu
leApplicationTests
[INFO] Running com.moviebookingapp.movie_and_theatre_module.services.MovieServiceTest
2026-01-05T15:15:25.828+05:30  INFO 20952 --- [movie-and-theatre-module] [foReplicator-%d] com.netflix.discovery.DiscoveryClient    : DiscoveryClient
_MOVIE-AND-THEATRE-MODULE/host.docker.internal:movie-and-theatre-module:8082 - registration status: 204
[INFO] Tests run: 18, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.794 s -- in com.moviebookingapp.movie_and_theatre_module.services.MovieServ
iceTest
2026-01-05T15:15:26.652+05:30  INFO 20952 --- [movie-and-theatre-module] [ionShutdownHook] o.s.c.n.e.s.EurekaServiceRegistry        : Unregistering a
pplication MOVIE-AND-THEATRE-MODULE with eureka with status DOWN
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 32, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.11:report (report) @ movie-and-theatre-module ---
[INFO] Loading execution data file C:\Users\Administrator\Project\MovieBookingApp_SpringBoot\movie-and-theatre-module\target\jacoco.exec
[INFO] Analyzed bundle 'movie-and-theatre-module' with 21 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  55.748 s
[INFO] Finished at: 2026-01-05T15:15:34+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\Administrator\Project\MovieBookingApp_SpringBoot\movie-and-theatre-module>
```

MovieBookingApp_SpringBoot
master
All MS
Start Free Trial
Project
MovieControllerImplTest.java
MovieServiceTest.java
Terminal
Local
Local (2)
Activate Windows
Go to Settings to activate Windows.
re-module > src > test > java > com > moviebookingapp > movie_and_theatre_module > controllers > MovieControllerImplTest
57:1 (676 chars, 18 line breaks) CRLF UTF-8 4 spaces

Sonar Qube:



http://localhost:9000/dashboard?id=MovieAndTheatreModule&codeScope=overall

Embedded database should be used for evaluation purposes only. It doesn't support scaling, upgrading to a new SonarQube Server version, or migration to another database engine. Learn more

'MovieAndTheatreModule' master

Quality Gate
Passed

Last analysis 4 minutes ago

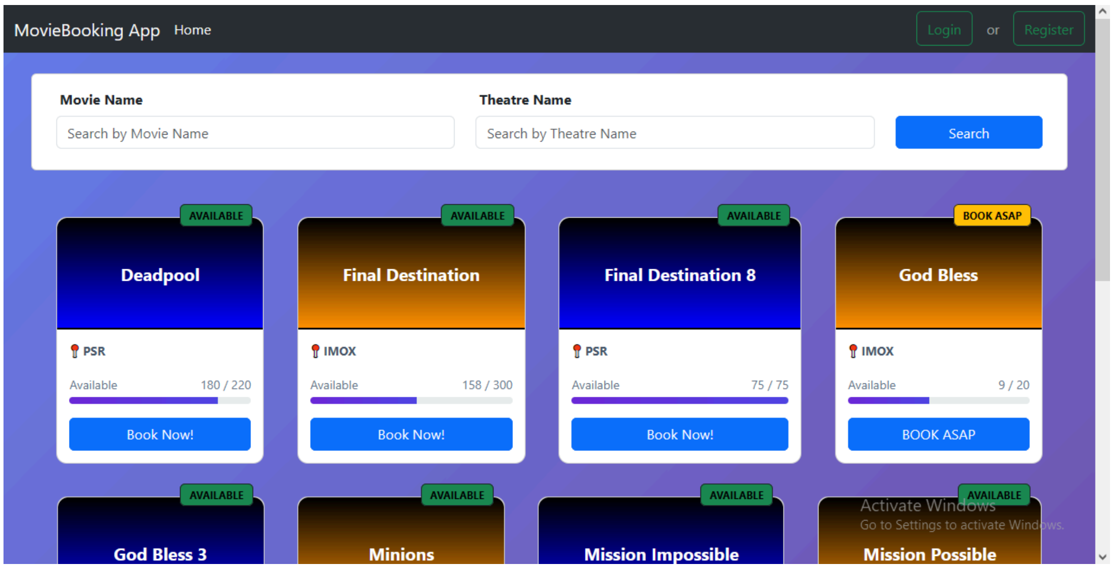
The last analysis has warnings. See details

New Code | Overall Code

Security: 0 Open issues — A
Reliability: 0 Open issues — A
Maintainability: 2 Open issues — A
Accepted issues: 0 — Valid issues that were not fixed
Coverage: 96.1% — On 76 lines to cover.
Duplications: 0.0% — On 766 lines.
Security Hotspots: 0 — A

Front-End Screenshots

Home Page



MovieBooking App Home Login or Register

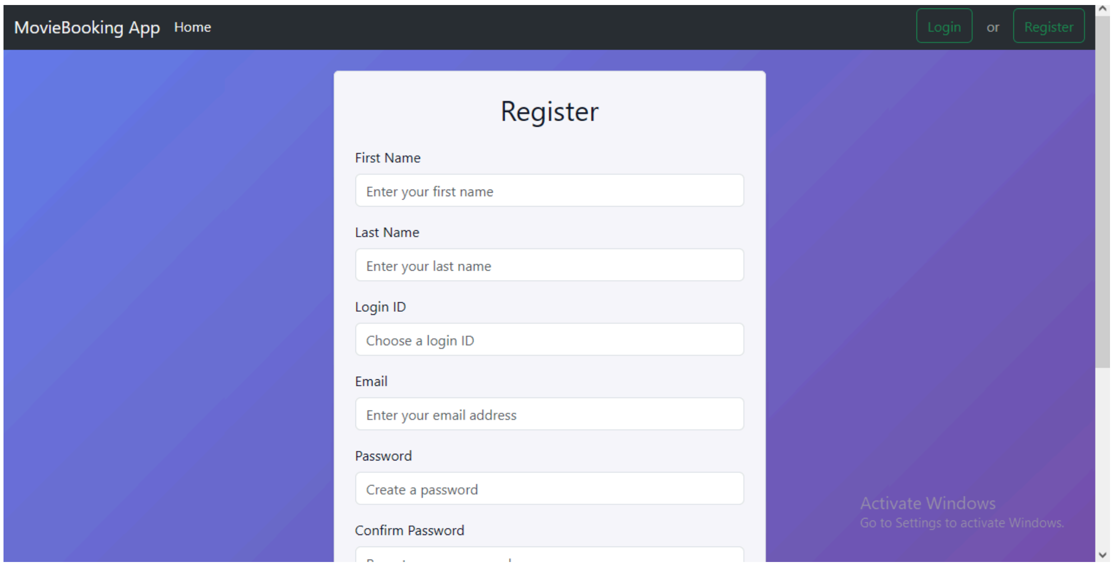
Movie Name: Search by Movie Name
Theatre Name: Search by Theatre Name
Search

Movie	Status	Theatre	Available	Button
Deadpool	AVAILABLE	PSR	180 / 220	Book Now!
Final Destination	AVAILABLE	IMOX	158 / 300	Book Now!
Final Destination 8	AVAILABLE	PSR	75 / 75	Book Now!
God Bless	BOOK ASAP	IMOX	9 / 20	BOOK ASAP
God Bless 3	AVAILABLE			
Minions	AVAILABLE			
Mission Impossible	AVAILABLE			
Mission Possible	AVAILABLE			

Activate Windows
Go to Settings to activate Windows.

Register

Register Form



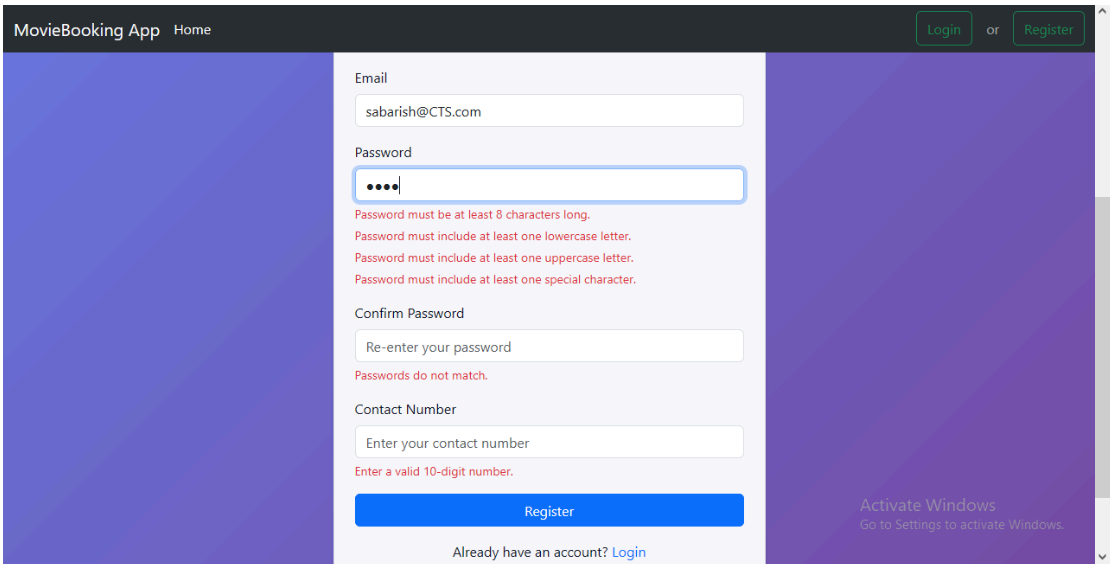
MovieBooking App Home Login or Register

Register

First Name: Enter your first name
Last Name: Enter your last name
Login ID: Choose a login ID
Email: Enter your email address
Password: Create a password
Confirm Password

Activate Windows
Go to Settings to activate Windows.

Register Validation



Email

sabarish@CTS.com

Password

••••

Password must be at least 8 characters long.

Password must include at least one lowercase letter.

Password must include at least one uppercase letter.

Password must include at least one special character.

Confirm Password

Re-enter your password

Passwords do not match.

Contact Number

Enter your contact number

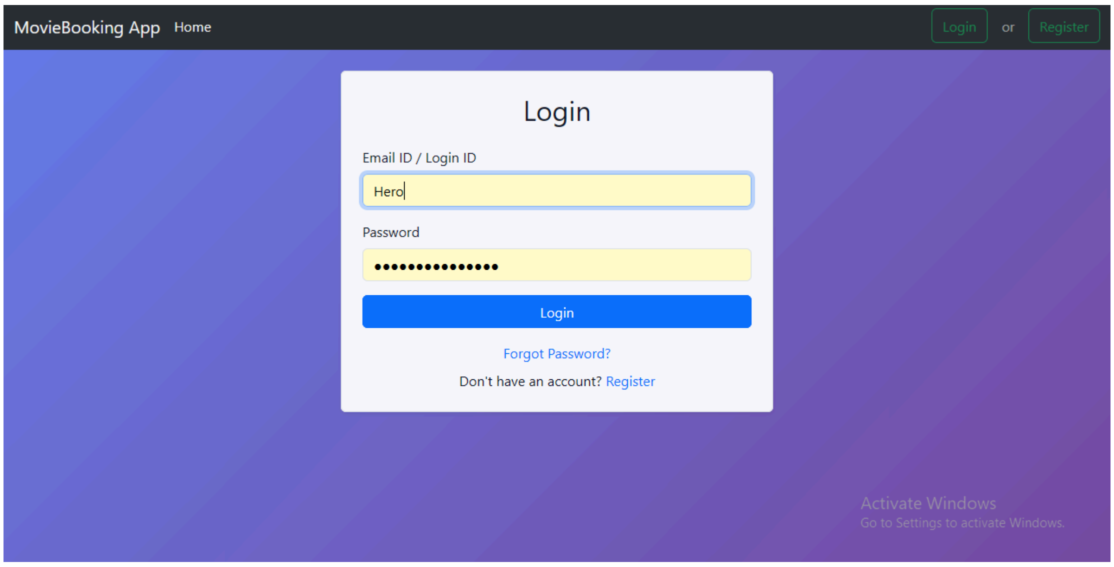
Enter a valid 10-digit number.

Register

Already have an account? Login

Activate Windows
Go to Settings to activate Windows.

Login Page



Login

Email ID / Login ID

Hero

Password

•••••••••••••••

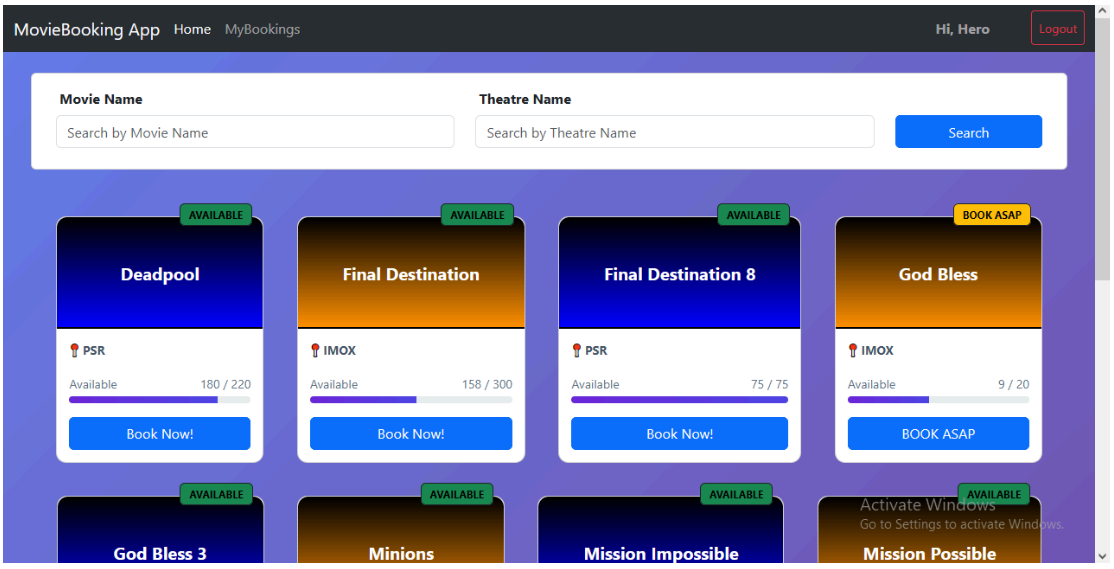
Login

Forgot Password?

Don't have an account? Register

Activate Windows
Go to Settings to activate Windows.

Homepage for Logged-In User



MovieBooking App Home MyBookings Hi, Hero Logout

Movie Name: Search by Movie Name
Theatre Name: Search by Theatre Name
Search

AVAILABLE
Deadpool
PSR
Available 180 / 220
Book Now!

AVAILABLE
Final Destination
IMOX
Available 158 / 300
Book Now!

AVAILABLE
Final Destination 8
PSR
Available 75 / 75
Book Now!

BOOK ASAP
God Bless
IMOX
Available 9 / 20
BOOK ASAP

AVAILABLE
God Bless 3

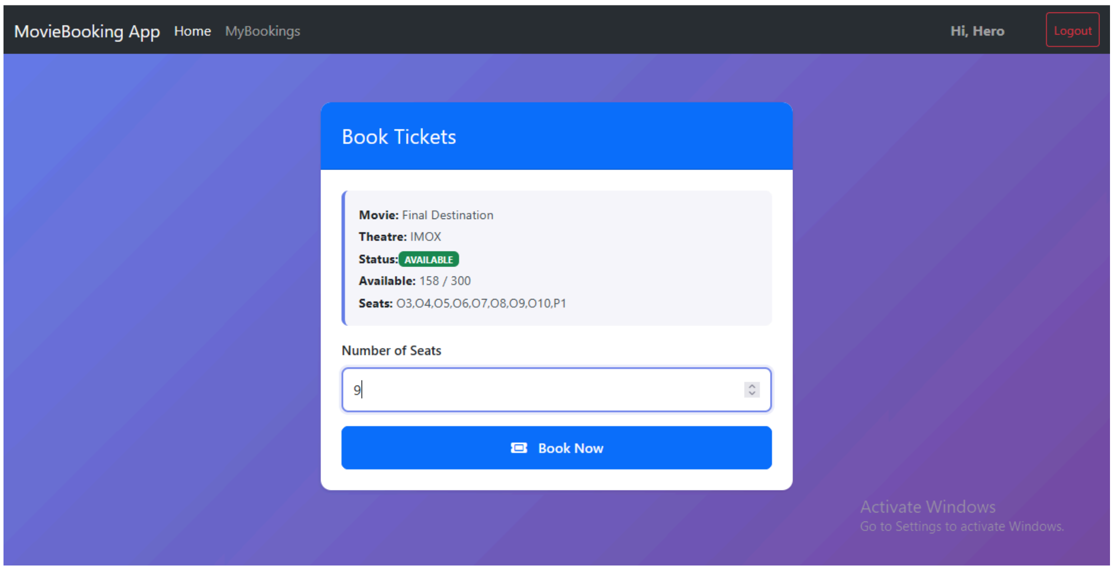
AVAILABLE
Minions

AVAILABLE
Mission Impossible

AVAILABLE
Mission Possible

Activate Windows
Go to Settings to activate Windows.

Movie Booking Page



MovieBooking App Home MyBookings Hi, Hero Logout

Book Tickets

Movie: Final Destination
Theatre: IMOX
Status: AVAILABLE
Available: 158 / 300
Seats: O3,O4,O5,O6,O7,O8,O9,O10,P1

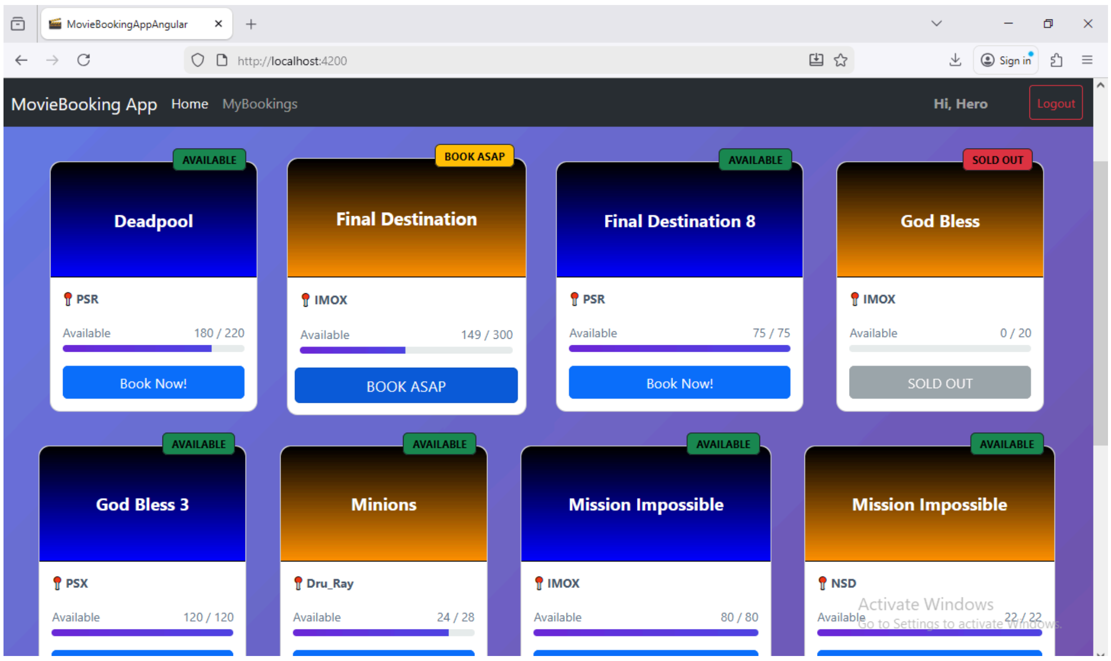
Number of Seats
9

Book Now

Activate Windows
Go to Settings to activate Windows.

Home Page

Status and Value change after Tickets booked



AVAILABLE
Deadpool
PSR
Available 180 / 220
Book Now!

BOOK ASAP
Final Destination
IMOX
Available 149 / 300
BOOK ASAP

AVAILABLE
Final Destination 8
PSR
Available 75 / 75
Book Now!

SOLD OUT
God Bless
IMOX
Available 0 / 20
SOLD OUT

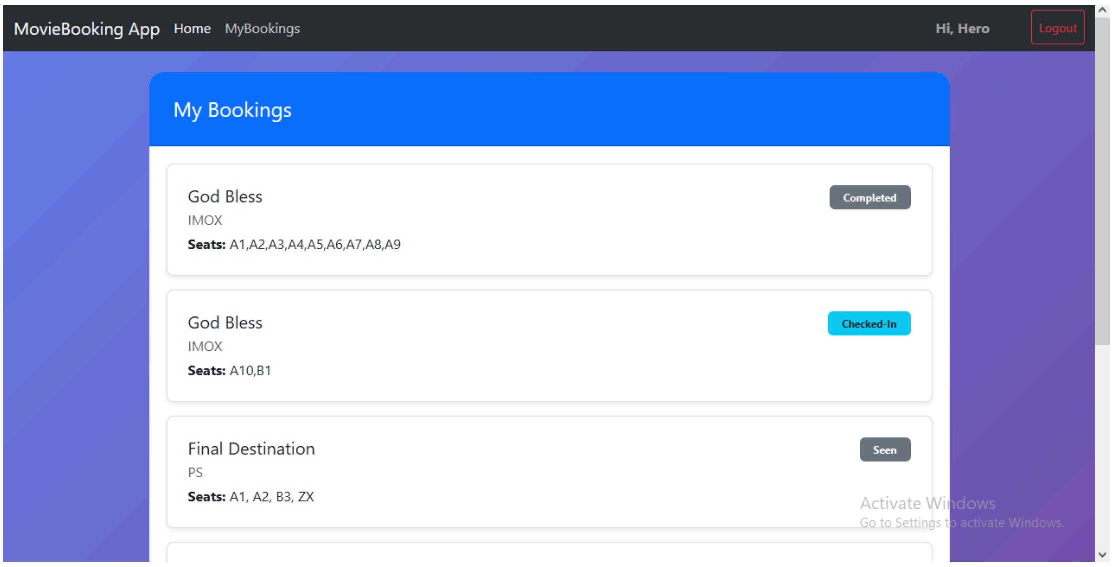
AVAILABLE
God Bless 3
PSX
Available 120 / 120

AVAILABLE
Minions
Dru_Ray
Available 24 / 28

AVAILABLE
Mission Impossible
IMOX
Available 80 / 80

AVAILABLE
Mission Impossible
NSD
Available 22 / 22

My Bookings



My Bookings

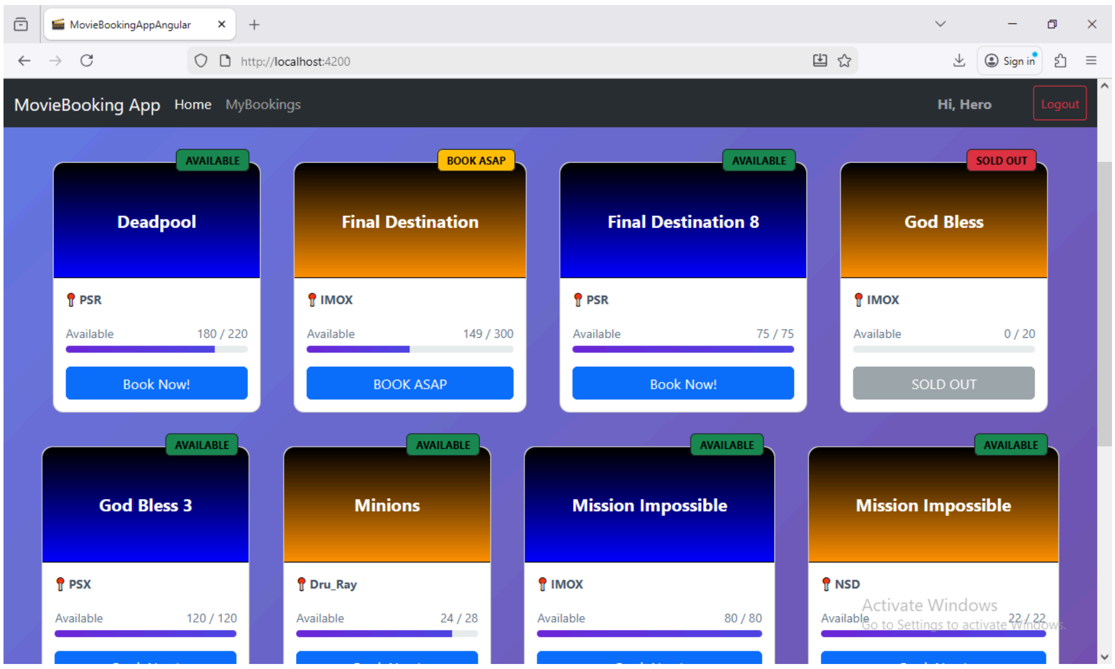
God Bless
IMOX
Seats: A1,A2,A3,A4,A5,A6,A7,A8,A9
Completed

God Bless
IMOX
Seats: A10,B1
Checked-In

Final Destination
PS
Seats: A1, A2, B3, ZX
Seen

Sold Out Movie

Blocks Ticket Booking



AVAILABLE
Deadpool
PSR
Available 180 / 220
Book Now!

BOOK ASAP
Final Destination
IMOX
Available 149 / 300
BOOK ASAP

AVAILABLE
Final Destination 8
PSR
Available 75 / 75
Book Now!

SOLD OUT
God Bless
IMOX
Available 0 / 20
SOLD OUT

AVAILABLE
God Bless 3
PSX
Available 120 / 120

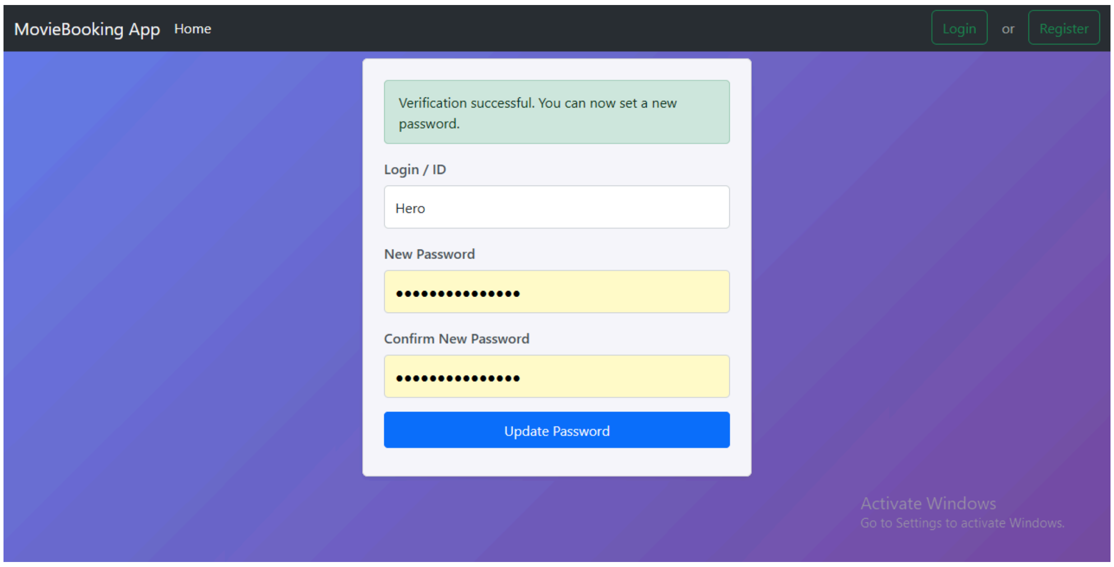
AVAILABLE
Minions
Dru_Ray
Available 24 / 28

AVAILABLE
Mission Impossible
IMOX
Available 80 / 80

AVAILABLE
Mission Impossible
NSD
Available 22 / 22

Forgot Password

Verify If User exists



Verification successful. You can now set a new password.

Login / ID

Hero

New Password

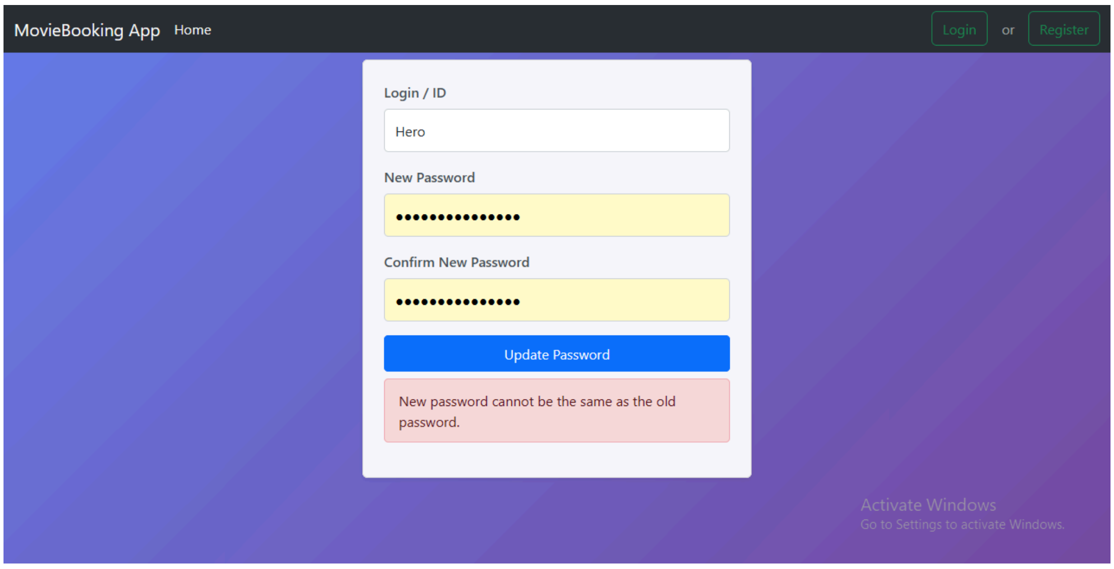
•••••••••••••••

Confirm New Password

•••••••••••••••

Update Password

Check for Old Password



Login / ID

Hero

New Password

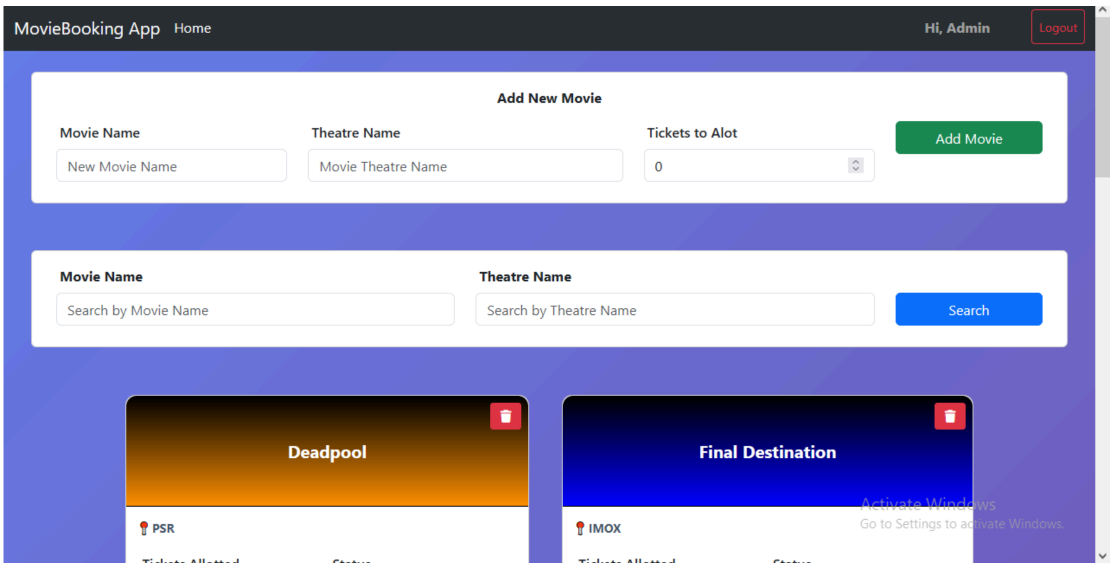
•••••••••••••••

Confirm New Password

•••••••••••••••

New password cannot be the same as the old password.

Admin Home Page:



Add New Movie

Movie Name	Theatre Name	Tickets to Alot
New Movie Name	Movie Theatre Name	0

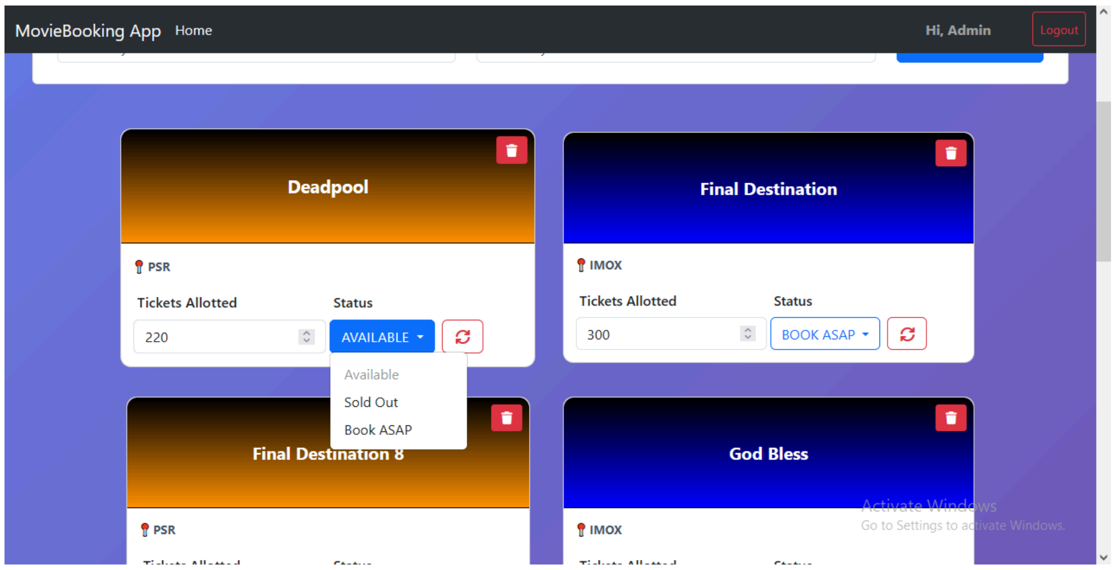
Movie Name	Theatre Name
Search by Movie Name	Search by Theatre Name

PSR

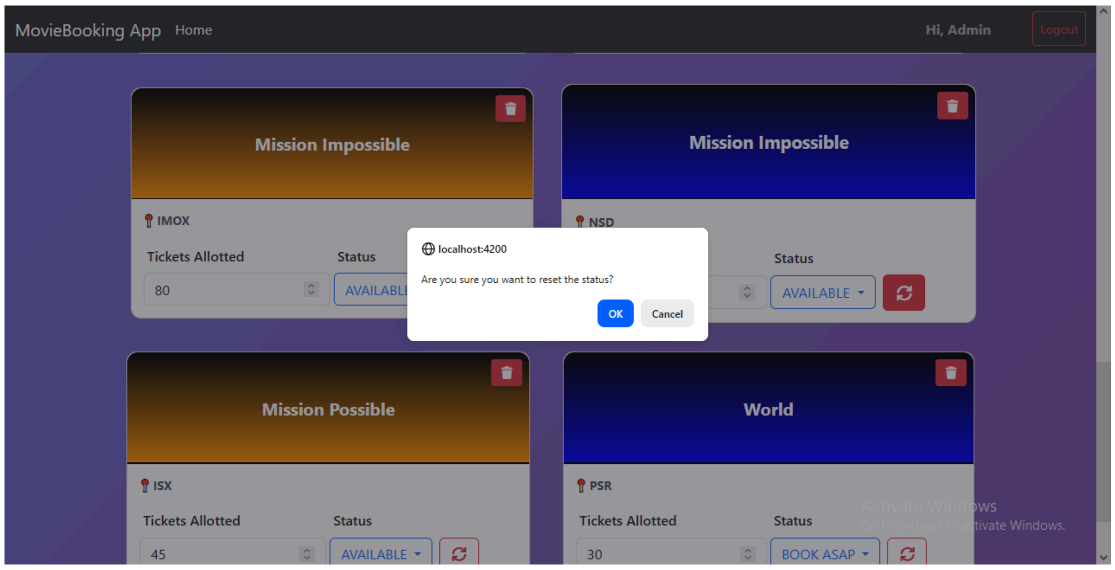
IMOX

Update Options

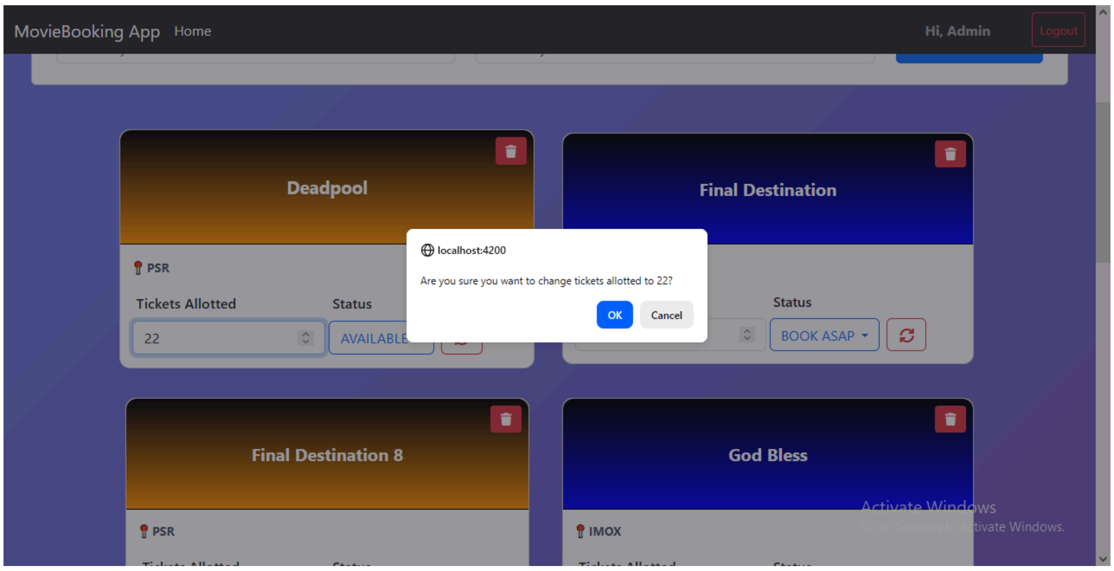
Movie Status Update



Button to Reset Movie Status



Tickets Allotted Update



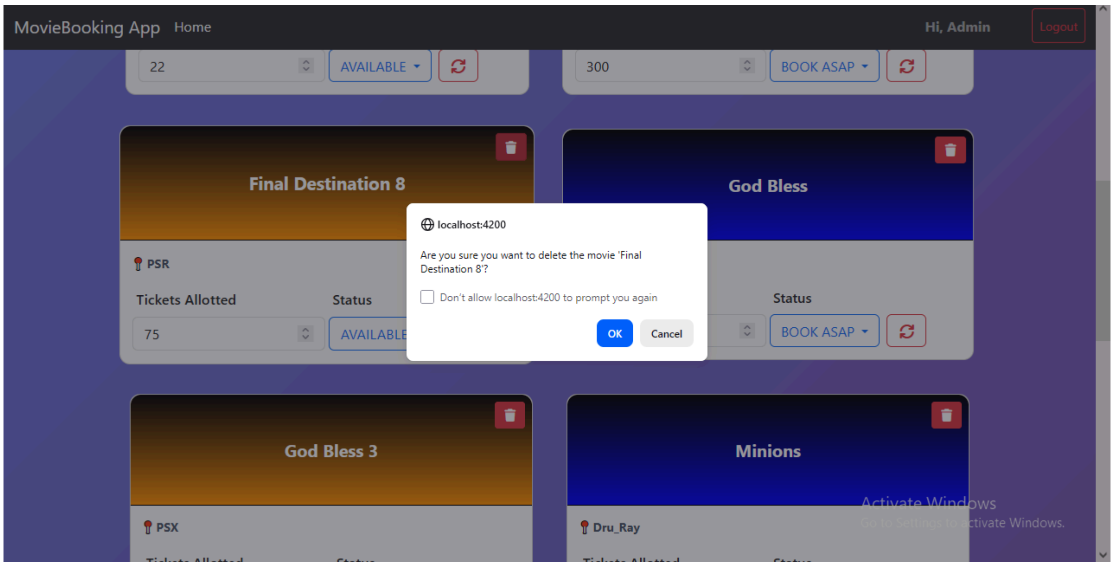
localhost:4200

Are you sure you want to change tickets allotted to 22?

OK Cancel

Movie Deletion

Confirm Before Deletion



localhost:4200

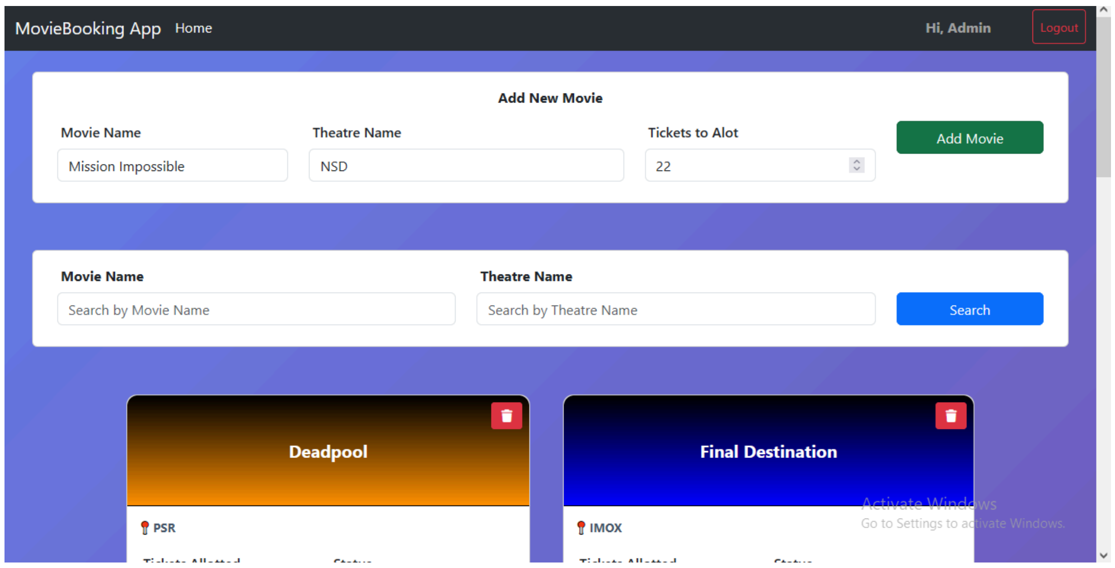
Are you sure you want to delete the movie 'Final Destination 8'?

Don't allow localhost:4200 to prompt you again

OK Cancel

New Movie

Creating Using New Movie Card



Conclusion

This project successfully implements a Movie Ticket Booking application using a microservices-based architecture. All required business functionalities—including user authentication, movie management, ticket booking, and administrative controls—have been implemented with proper validation, exception handling, and role-based security. The system leverages an API Gateway for centralized routing and security, Eureka Server for service discovery, and Swagger for API documentation. Unit testing, caching, and stateless session management ensure maintainability, performance, and scalability. Overall, the application meets the specified requirements and demonstrates clean architectural design and industry-standard development practices.