

# CASE STUDY- MOVIE BOOKING APP (FSE-1)

By- Ankush Kumar Singh

# CONTENTS

---

1	Problem Statement .....	3
2	Proposed Movie Booking App Wireframe .....	3
3	Application Architecture .....	4
4	Tool Chain .....	5
5	Development flow.....	5
6	Business-Requirement: .....	5
7	Rubrics/Expected Deliverables .....	7
7.1	Rest API (Products & Frameworks -> Compute & Integration): .....	7
7.2	Database (Products & Frameworks -> Database & Storage): .....	8
7.3	Maven (Tooling):.....	8
7.4	Unit Testing:.....	8
7.5	Debugging & Troubleshooting .....	8
7.6	Code quality & Code coverage.....	8
7.7	Good To Have:.....	8
8	Frontend.....	9
9	Expected Outcomes: .....	9
9.7.1	MovieBookingApp Home Page .....	10
9.7.2	Search By Movie Name .....	11
9.7.3	Registration Page Validation.....	11
9.7.4	Registration Page Validation (Password Validation criteria) .....	12
9.7.5	Registration Page Validation (Password Do not Match).....	13
9.7.6	Successful Registration.....	13
9.7.7	Forgot Password _Reset_option .....	14
9.7.8	Login Page .....	14
9.7.9	Home Page After Successful Login .....	15
9.7.10	Sold Out Option Disables Tickets Booking .....	16
9.7.11	Can View the Tickets Booked by Logged In User .....	16
9.7.12	Ticket Book Page .....	17
9.7.13	Logged In User Can reset the Password Using Change Password Option .....	18
9.7.14	Change Password Validation.....	18
9.7.15	Admin Login .....	19
9.7.16	Admin Dashboard After Login.....	19
9.7.17	Movie added By Admin (Movie Name-Sigma).....	20
9.7.18	Added Movie Reflected in Dashboard .....	21

9.7.19	Admin Can Manually Mark Movie As SOLD OUT .....	21
9.7.20	Admin Can Manually Mark Movie As BOOK ASAP.....	22
9.7.21	Through Refresh Availability Option Ticket Status will be Set again .....	22
9.7.22	Normal Logged in User Can see the newly added Movie and Can book the tickets <b>Error! Bookmark not defined.</b>	
9.7.23	Ticket Book Validation Functionality .....	23
9.7.24	Successful Ticket Booked .....	23
9.7.25	Now the Movie Ticket Booking Status Changes Based on Ticket Availability.....	<b>Error!</b>
	<b>Bookmark not defined.</b>	
9.7.26	Ticket Also reflected in Ticket List.....	<b>Error! Bookmark not defined.</b>
9.7.27	In Admin Dashboard Also, changes Reflected through Refresh Availability option .....	24
9.7.28	Admin Can Delete the Movie .....	24
9.7.29	Admin Dashboard After Deleting Sigma Movie .....	25
9.7.30	In User Home Page Movie Removed .....	25
9.7.31	The ticket was also Removed for the Respective Deleted Movie.....	26

# 1 PROBLEM STATEMENT

---

**Movie Booking App** is a full-stack web application that enables users to register, authenticate, and browse movies to book tickets for their preferred shows. The system allows administrators to manage movie inventory, theatre information, and ticket availability based on bookings. Guest users cannot perform any booking operations without authentication.

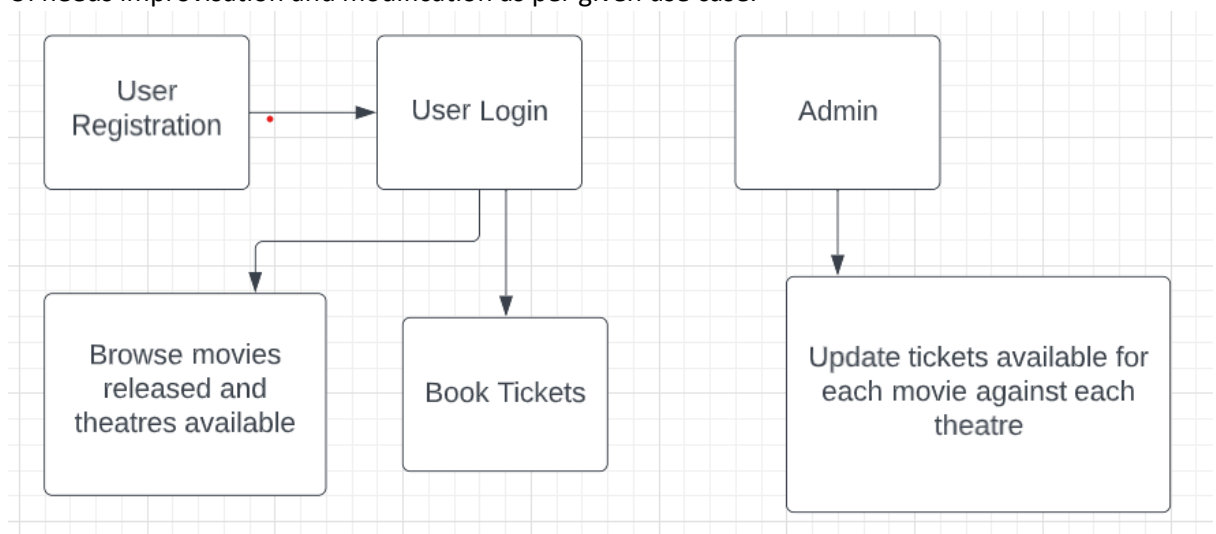
## 2 Key Objectives:

- Provide a secure user authentication mechanism for registration and login
- Enable users to search and browse available movies across different theatres
- Allow authenticated users to book tickets for selected movies and showtimes
- Provide administrators with tools to manage movies, theatres, and ticket inventory
- Maintain real-time ticket availability updates based on user bookings
- Ensure data security through BCrypt password hashing and role-based access control

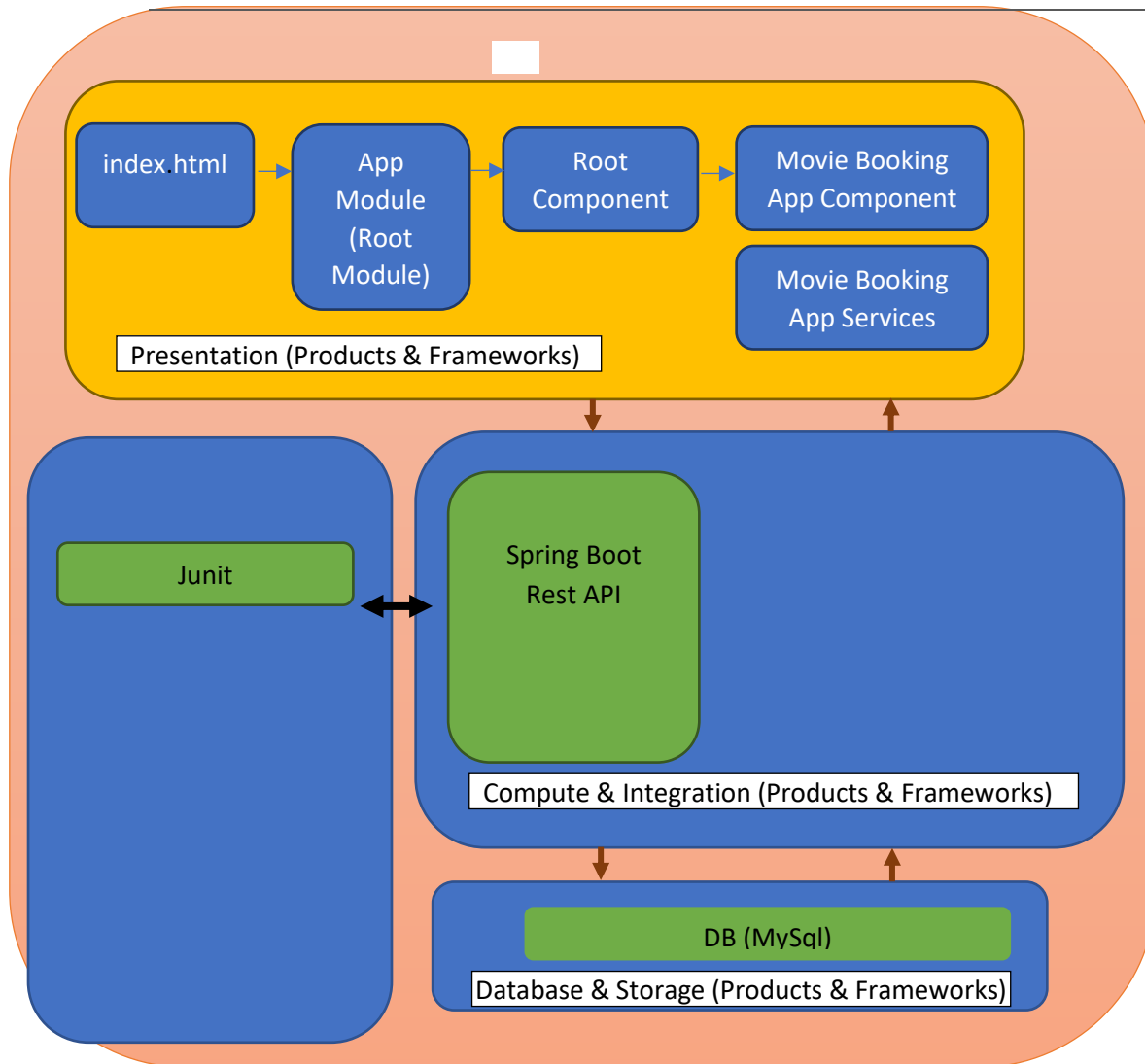
# 3 PROPOSED MOVIE BOOKING APP WIREFRAME

---

1. UI needs improvisation and modification as per given use case.



## 4 APPLICATION ARCHITECTURE



## 5 TOOL CHAIN

---

Competency	Skill	Skill Detail
Engineering Mindset	Networking and Content Delivery	
	Ways of Working	
	Consulting Mindset	
	DevOps	
Programming Languages	Application Language	Java
Products & Frameworks	Presentation	Angular 17
	Compute & Integration	Spring Boot
	Database & Storage	MySQL/SQL-Server
		Maven
		JUnit 5

## 6 DEVELOPMENT FLOW

---

No	MC	Competency	Section	Indicative Mechanism for Evaluation (Passing score of 60% in each MC)
<a href="#">Business Requirement</a>				
1	Backend	Rest API, Database, Testing, Debugging & Troubleshooting	-	Code Submission and Evaluation, Panel Presentation
2	Front End	Angular	-	Code Submission and Evaluation, Panel Presentation

---

## 7 BUSINESS-REQUIREMENT:

---

As an application developer, develop frontend, middleware and deploy the Movie Booking App with below guidelines:

**Pre-requisite before implementing the user story:**

---

Movie Booking App Case Study, UI/UX, Middleware, and DB with Best Practices.

Create a static database as “Movie” and add the below fields

- Movie name, Total number of tickets allotted and theatre name (Movie name and theatre name should be of composite primary key)

There should be a predefined set of tickets assigned to each theatre against each movie. Just create sample 2 movies with 2 theatres assigned. This will be used for the upcoming user story to fetch the data.

User Story #	User Story Name	User Story
US_01	Registration and Login	<p>As a user I should be able to login/Register in the movie booking application</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"><li>1. A logged-in user can reset their password so they can login, even if they forget their password.</li><li>2. A logged-in user:<ol style="list-style-type: none"><li>a. Cannot change their username.</li><li>b. Can logout from their account.</li></ol></li><li>3. As a customer I should be able to furnish following details at the time of registration<ol style="list-style-type: none"><li>a. First Name</li><li>b. Last Name</li><li>c. Email</li><li>d. Login Id</li><li>e. Password</li><li>f. Confirm Password</li><li>g. Contact Number</li></ol></li><li>4. All details fields must be mandatory</li><li>5. Login Id and Email must be unique</li><li>6. Password and Confirm Password must be same</li><li>7. If any constraint is not satisfied, validation message must be shown</li></ol>
US_02	View &Search Movies	<p>As a user I should be able to view all the recent movies opened for booking. User can search any particular movies as well</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"><li>a. User can view all the existing released movies.</li><li>b. User can search any particular movie based on the movie name</li></ol>
US_03	Book Tickets for a movie	<p>As a user I should be able to book tickets for a movie. Add this booking to a database table “Tickets”. Assign movie name and theatre name as a foreign key to database table “Movie” which is already created as a pre-requisite</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"><li>a. Book a movie ticket</li><li>b. Below are the details to be added<ul style="list-style-type: none"><li>• Movie Name</li><li>• Theatre name</li><li>• Number of tickets</li></ul></li></ol>

		<ul style="list-style-type: none"> <li>• Seat Number</li> </ul>
US_04	View booked tickets and Update ticket status	<p>As an admin I should be able to view booked tickets and update the balance tickets available for a movie</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"> <li>View number of booked tickets for a particular movie from table "Tickets"</li> <li>Check the tickets available from table "Movie" by calculating the total tickets booked</li> <li>If the quantity is 0, update the ticket status as 'SOLD OUT', else update as 'BOOK ASAP'</li> </ol>

## 8 RUBRICS/EXPECTED DELIVERABLES

### 8.1 REST API (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

- Use Spring Boot to version and implement the REST endpoints.
- Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

POST	/api/v1.0/moviebooking/register	Register as new user
GET	/api/v1.0/ moviebooking /login	Login
GET	/api/v1.0/ moviebooking /<username>/forgot	Forgot password
GET	/api/v1.0/ moviebooking /all	View all movies
GET	/api/v/1.0/moviebooking/movies/search/moviename*	Search by movie name
POST	/api/v1.0/ moviebooking /<moviename>/add	Book tickets for a movie
PUT	/api/v1.0/moviebooking /<moviename>/update/<ticket>	Update ticket status
DELETE	/api/v1.0/ moviebooking /<moviename>/delete/<id>	Delete movie

- \*Movie name may be partial or complete username
- Use necessary configuration in place for REST API in application. Properties or bootstrap. Properties or application.yml; whichever is applicable.
- Package Structure for Spring Boot Project will be like com. moviebookingapp. \* With proper naming conventions for package and beans.
- Use configuration class annotated with @Configuration and @Service for business layer.
- Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.
- Follow Spring Bean Naming Conventions

## 8.2 DATABASE (PRODUCTS & FRAMEWORKS -> DATABASE & STORAGE):

1. As an application developer:
  - a. Implement ORM with Spring Data MongoDB and MongoDB / JpaRepository for other databases like SQL server/ MySQL. For complex and custom queries, create custom methods and use @Query, Aggregations (Aggregation Operation, Match Operation, Aggregation Results), implementation of MongoTemplate etc as necessary.
  - b. Have the necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml OR Java based configuration; whichever is applicable.
  - c. Implement SQL procedures, triggers, and other performance-enhancing techniques.
  - d. Implement at least 2 role-based authentications/authorisations in your Spring Boot applications.
  - e. Implement cache and session management techniques using the Spring Boot API.

## 8.3 MAVEN (TOOLING):

1. As an application developer:
  - a. Create the spring boot project using Maven CLI
  - b. Generate Surefire test reports and share it as a part of deliverables
  - c. Using Maven CLI generate the project documentation, and share it as a part of the deliverables.

## 8.4 UNIT TESTING:

1. As an application developer:
  - a. Implement Unit testing using JUnit 5 for all the layers Exp: Controller layer, Service layer, and Repository layer.
  - b. Generate the test results with positive and negative test scripts.

## 8.5 DEBUGGING & TROUBLESHOOTING

1. Generate bug reports & error logs - Report must be linked with final deliverables, which should also suggest the resolution for the encountered bugs and errors.

## 8.6 CODE QUALITY & CODE COVERAGE

1. Ensure code quality with static analysis tools
2. Code coverage for API should be > 80%

## 8.7 GOOD TO HAVE:

1. Implement error/exception handling mechanisms
2. Implement logging and monitoring to detect and troubleshoot issues
3. Implement rate limiting to prevent abuse of API endpoints
4. Blogs should be fetched and displayed within 30 seconds
5. Use containerization to package and deploy the application

6. Improve maintainability with modular codebase, automated testing, CI/CD pipelines

## 9 FRONTEND

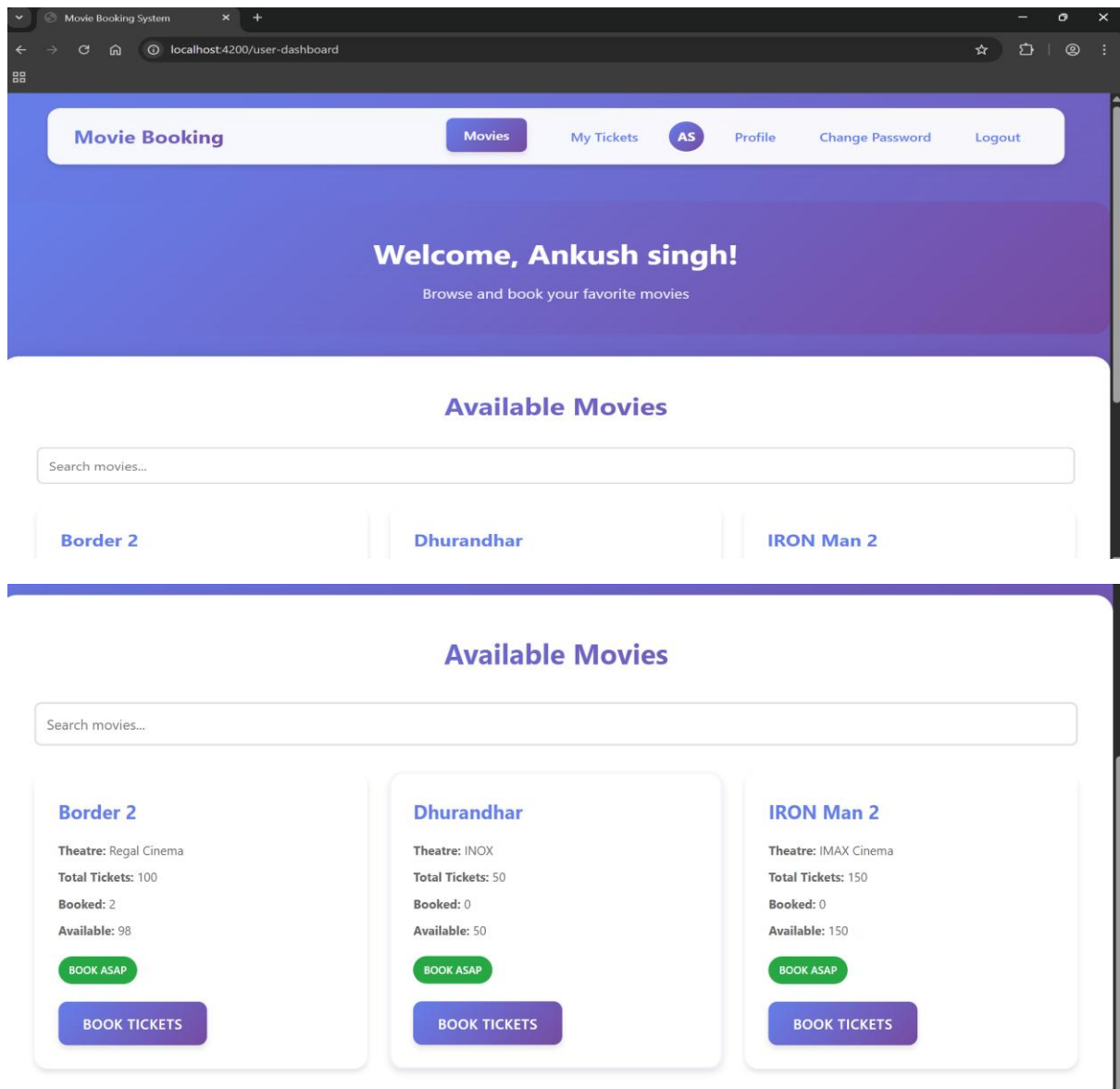
---

1. Develop the front end for all user stories.
2. Implement using Angular
3. Implement all the Front-End validation rules
4. Proper naming conventions and folder structures
5. Implement using proper SOLID design principles
6. Perform unit and integration testing for the front-end application

## 10 EXPECTED OUTCOMES:

---

### 10.1.1 MovieBookingApp Home Page



### 10.1.2 Search By Movie Name

### Available Movies

#### Border 2

Theatre: Regal Cinema  
Total Tickets: 100  
Booked: 2  
Available: 98

BOOK ASAP

BOOK TICKETS

### 10.1.3 Registration Page Validation(field can't be empty, Login Id must be unique)

LoginRegister

Create Account

FIRST NAME \*

firstName is required

LAST NAME \*

singh

EMAIL \*

akash@gmail.com

Invalid email format

LOGIN ID \*

loginid is required

Movie Booking System

Book your favorite movies with ease

The screenshot shows a web browser window with the title "Movie Booking System" and the address bar displaying "localhost:4200". The page features a registration form on the left and a purple sidebar on the right with the text "Movie Booking System" and "Book your favorite movies with ease". The form fields are: EMAIL \* (akash@gmail.com), LOGIN ID \* (ankush), PASSWORD \* (\*\*\*\*\*), CONFIRM PASSWORD \* (\*\*\*\*\*), and CONTACT NUMBER \* (7039709648). A red error message "Login ID already exists" is displayed below the form. A blue "REGISTER" button is at the bottom of the form.

(Here, login Id "ankush" already exists. so, registration unsuccessful)

#### 10.1.4 Registration Page Validation (Password Validation criteria- minimum 6 characters required)

The screenshot shows the same web browser window as before, but with a different registration attempt. The form fields are: LAST NAME \* (singh), EMAIL \* (akash@gmail.com), LOGIN ID \* (akash), PASSWORD \* (\*\*\*\*\*), CONFIRM PASSWORD \* (\*\*\*\*\*), and CONTACT NUMBER \* (7039709648). A red error message "Must be at least 6 characters" is displayed below the password field. A blue "REGISTER" button is at the bottom of the form.

### 10.1.5 Registration Page Validation (Password Do not Match)

The screenshot shows a web browser window with the address bar displaying 'localhost:4200'. The page title is 'Movie Booking System'. The registration form is on the left, and the system logo and tagline are on the right. The form fields and their validation status are as follows:

- EMAIL \***: Input 'akash@gmail.com'. Below the input is a red error message: 'Invalid email format'.
- LOGIN ID \***: Input 'ankush'.
- PASSWORD \***: Input field is empty. Below it is a red error message: 'password is required'.
- CONFIRM PASSWORD \***: Input field is empty. Below it is a red error message: 'Passwords must match'.
- CONTACT NUMBER \***: Input field is empty.
- REGISTER**: A blue button at the bottom of the form.

The right side of the page features the text 'Movie Booking System' and 'Book your favorite movies with ease'.

### 10.1.6 Successful Registration

The screenshot shows the same web browser window as the previous one, but with successful registration data and a success message. The form fields and their status are as follows:

- EMAIL \***: Input 'ankushS@gmail.com'.
- LOGIN ID \***: Input 'ankush'.
- PASSWORD \***: Input field contains six dots '\*\*\*\*\*'.
- CONFIRM PASSWORD \***: Input field contains six dots '\*\*\*\*\*'.
- CONTACT NUMBER \***: Input '7039709648'.
- Registration successful! Please login.**: A green message box below the contact number field.
- REGISTER**: A blue button at the bottom of the form.

The right side of the page remains the same with the text 'Movie Booking System' and 'Book your favorite movies with ease'.

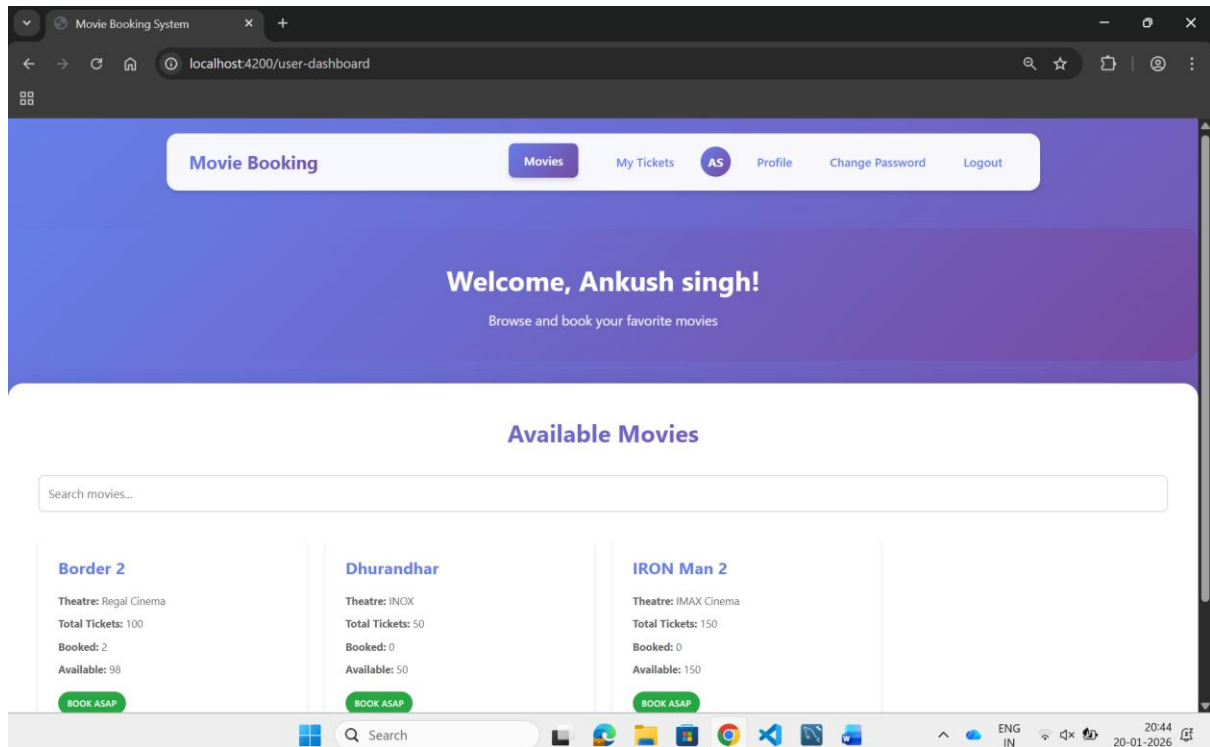
### 10.1.7Forgot Password \_Reset\_option

The screenshot shows the 'Change Password' form within the Movie Booking System. The form is centered on a purple gradient background. At the top, a navigation bar includes the 'Movie Booking' logo, links for 'Movies', 'My Tickets', a user profile icon labeled 'AS', a 'Profile' link, a 'Change Password' button, and a 'Logout' link. The form itself has a title 'Change Password' and three input fields: 'CURRENT PASSWORD \*', 'NEW PASSWORD \*', and 'CONFIRM NEW PASSWORD \*'. Each field contains masked text (dots). Below the fields is a blue 'CHANGE PASSWORD' button.

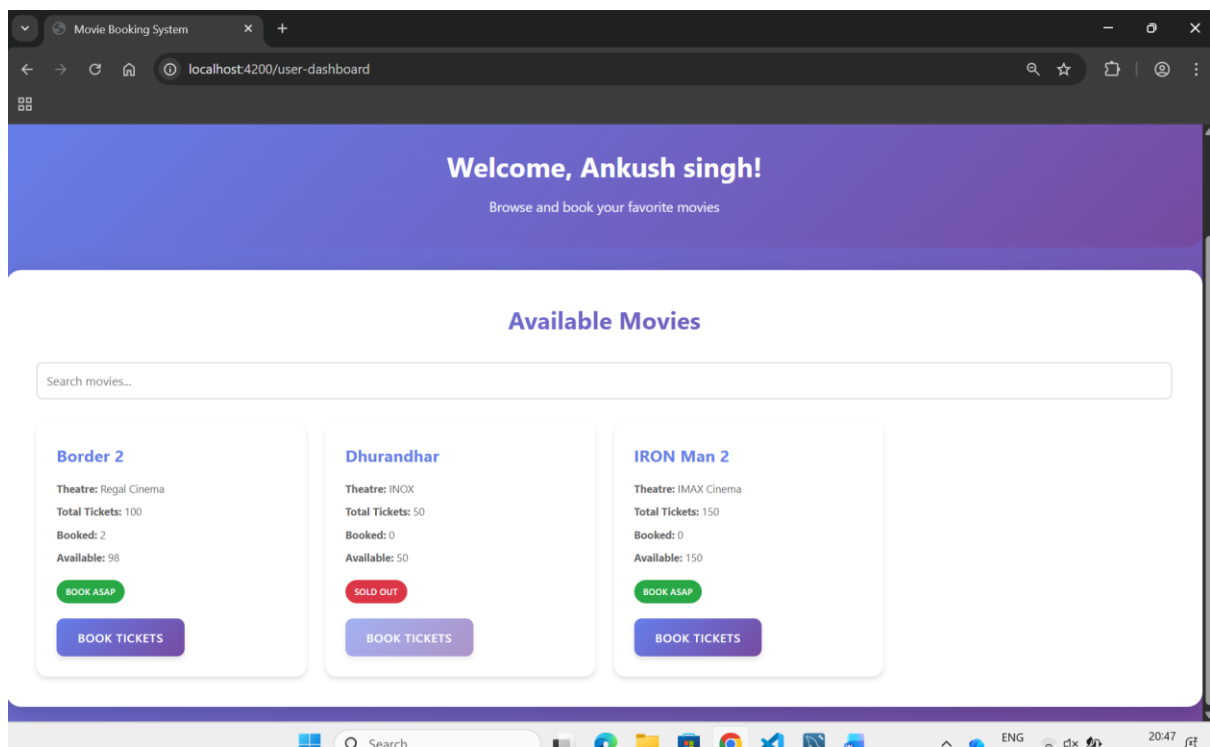
### 10.1.8Login Page

The screenshot shows the Login Page of the Movie Booking System. The page has a purple gradient background. On the left, a white login card contains a 'Login' tab (selected) and a 'Register' tab. Below the tabs is a 'Welcome Back' message. The card has two input fields: 'LOGIN ID \*' and 'PASSWORD \*'. A blue 'LOGIN' button is at the bottom of the card, with a 'Forgot Password?' link below it. To the right of the card, the text 'Movie Booking System' is displayed in large white letters, with the tagline 'Book your favorite movies with ease' underneath.

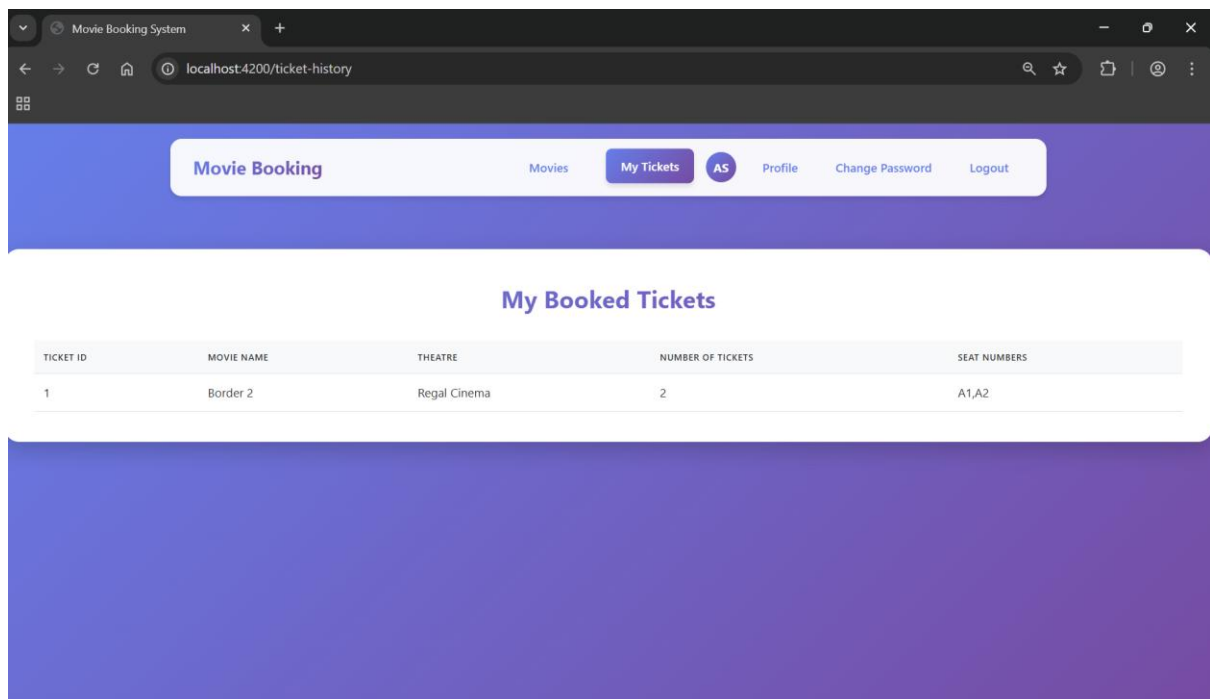
### 10.1.9 Home Page After Successful Login



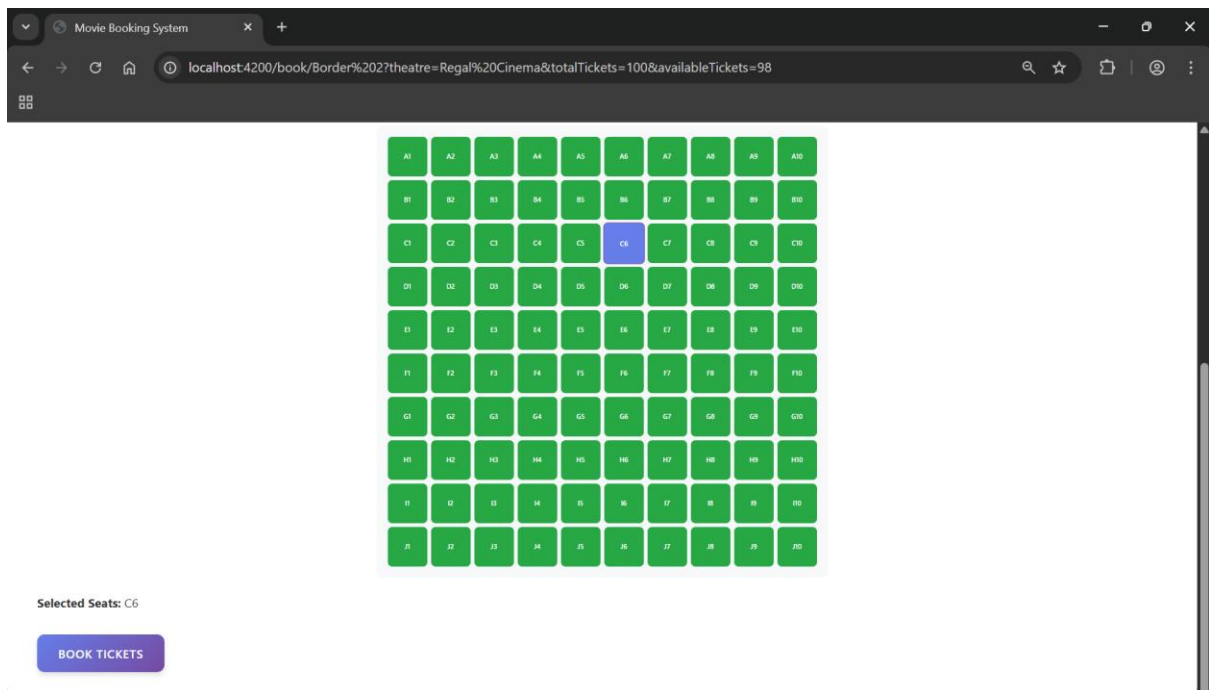
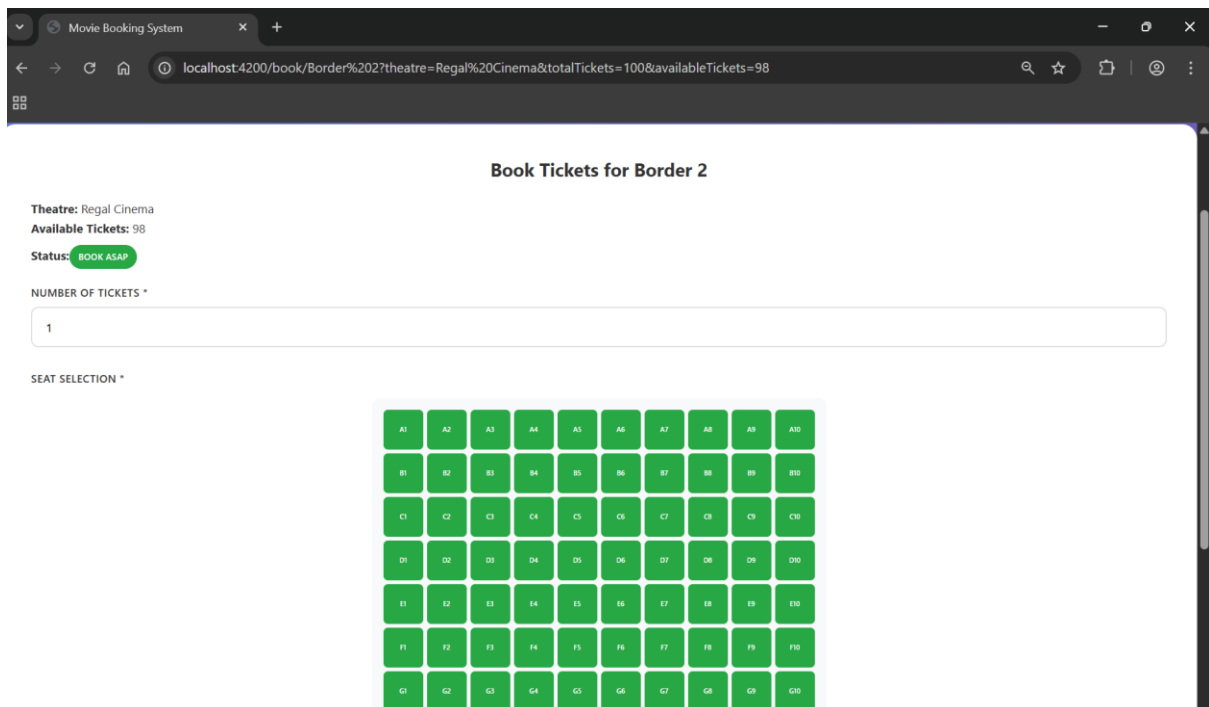
## 10.1.10 Sold Out Option Disables Tickets Booking



## 10.1.11 Can View the Tickets Booked by Logged In User



## 10.1.12 Ticket Book Page



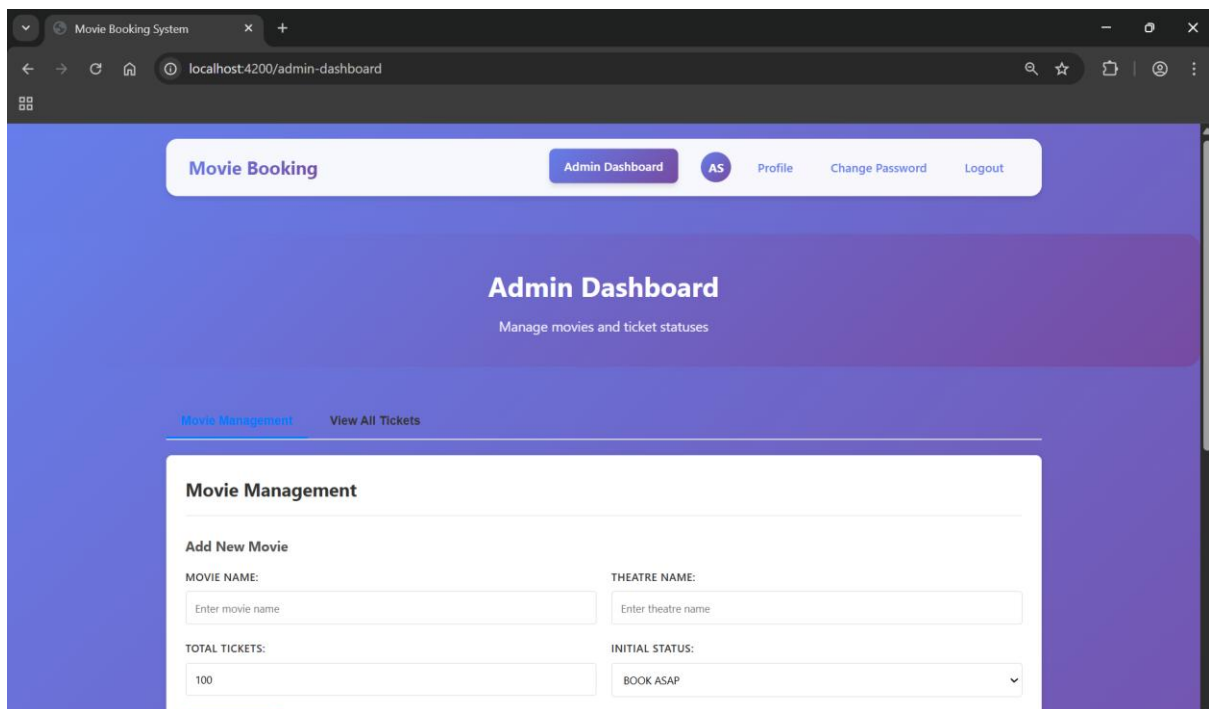
### 10.1.13 Logged In User Can reset the Password Using Change Password Option

The screenshot shows a web application interface for a 'Movie Booking' system. The header bar is purple and contains the 'Movie Booking' logo, navigation links for 'Movies', 'My Tickets', 'AS' (a user profile icon), 'Profile', 'Change Password' (a highlighted button), and 'Logout'. The main content area is white and titled 'Change Password'. It contains three input fields: 'CURRENT PASSWORD \*', 'NEW PASSWORD \*', and 'CONFIRM NEW PASSWORD \*'. Each field has a password mask (dots). Below the fields is a blue 'CHANGE PASSWORD' button.

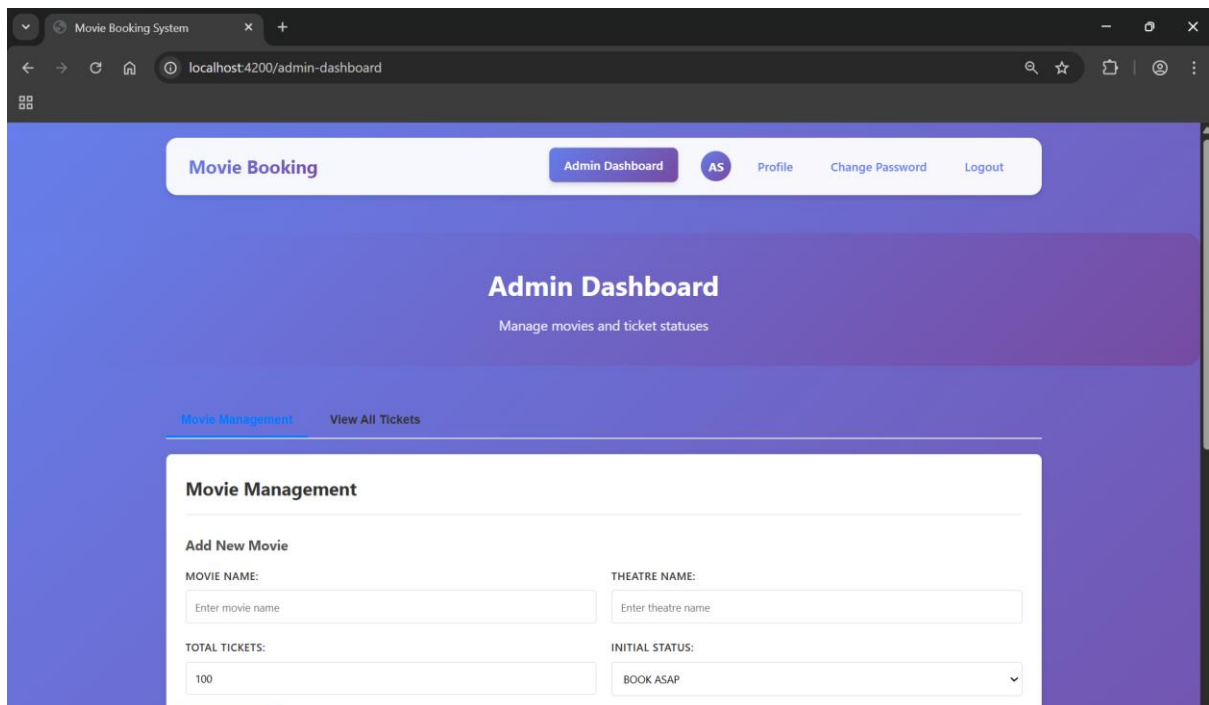
### 10.1.14 Change Password Validation

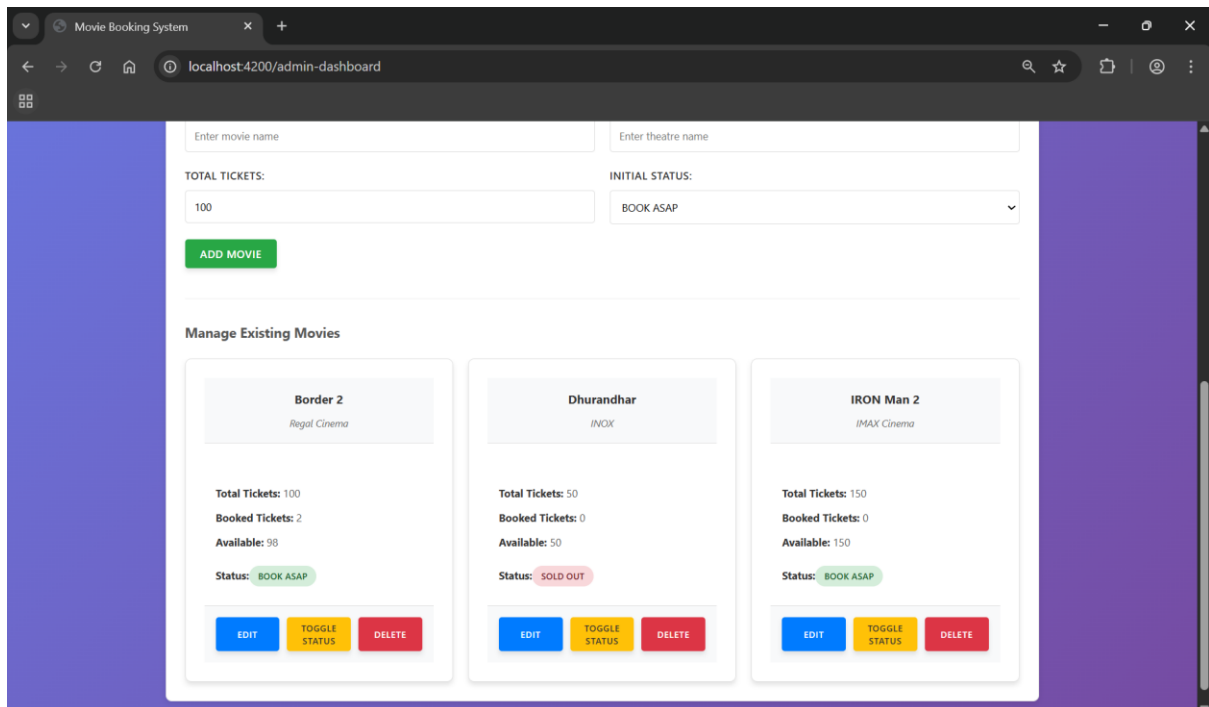
This screenshot shows the same 'Change Password' form as the previous one, but with validation errors. The browser's address bar shows 'localhost:4200/change-password'. The form fields are now empty, and each has a red border and a red error message below it: 'Current password is required' for the first field, 'newPassword is required' for the second, and 'Passwords must match' for the third. The 'CHANGE PASSWORD' button remains at the bottom.

## 10.1.15 Admin Login

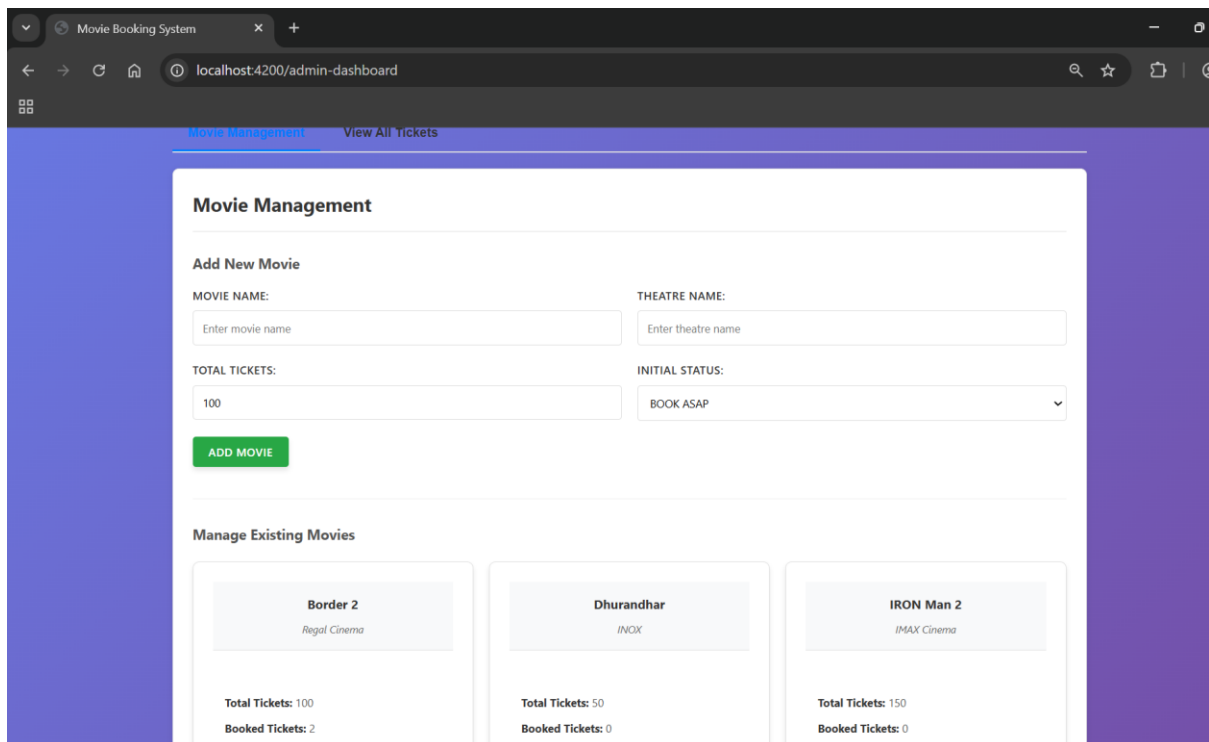


## 10.1.16 Admin Dashboard After Login

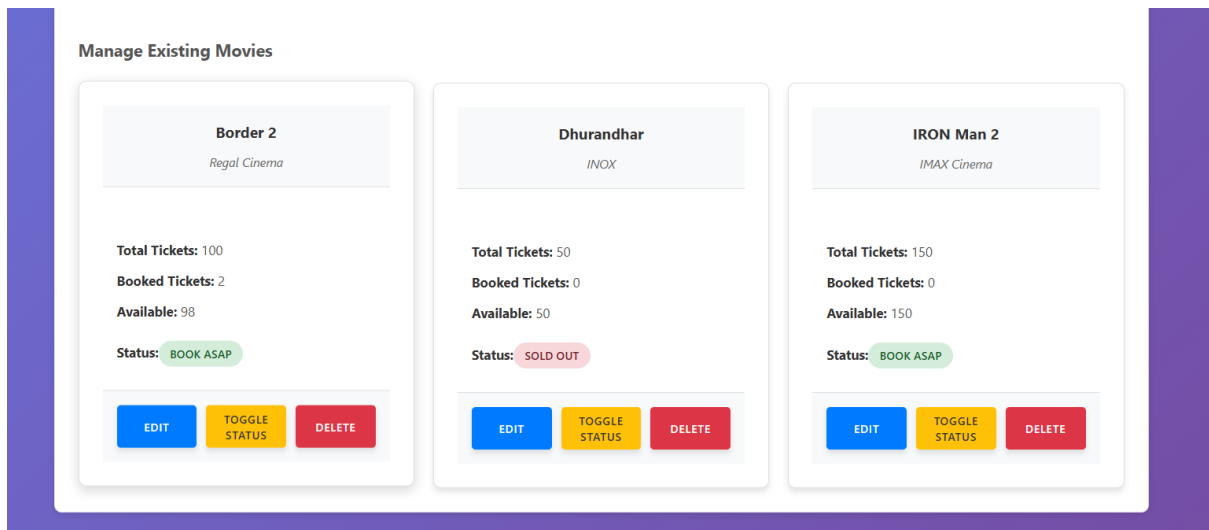




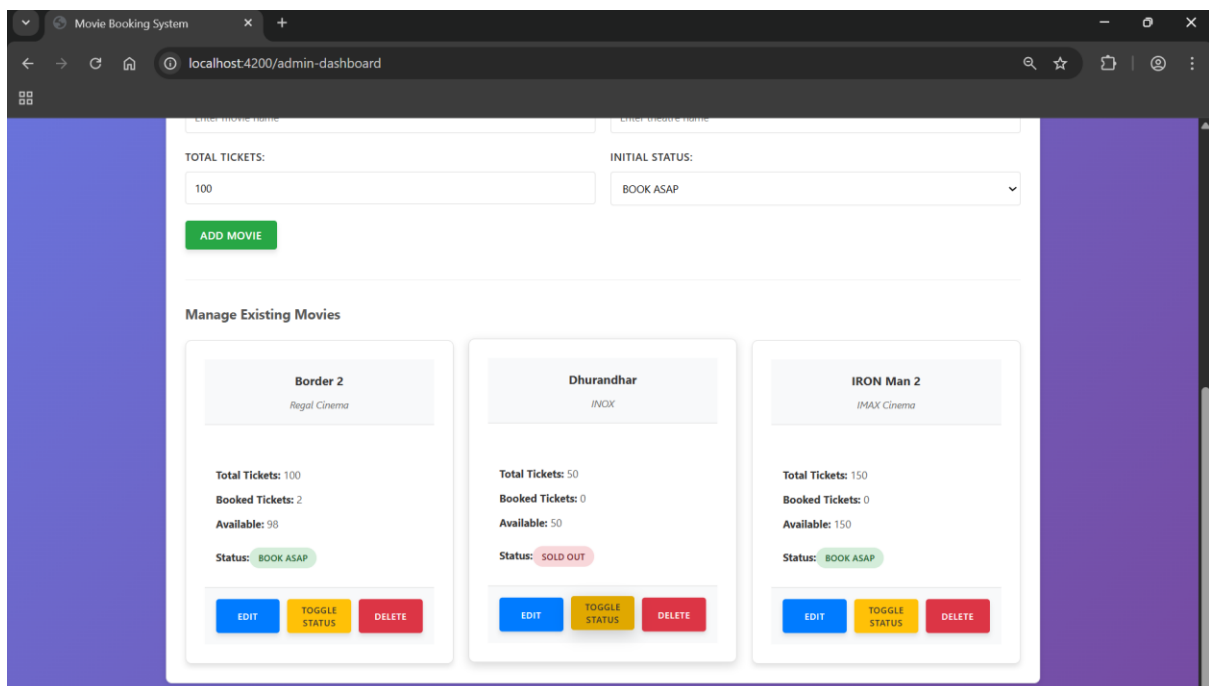
### 10.1.17 Movie added By Admin (Movie Name-Sigma)



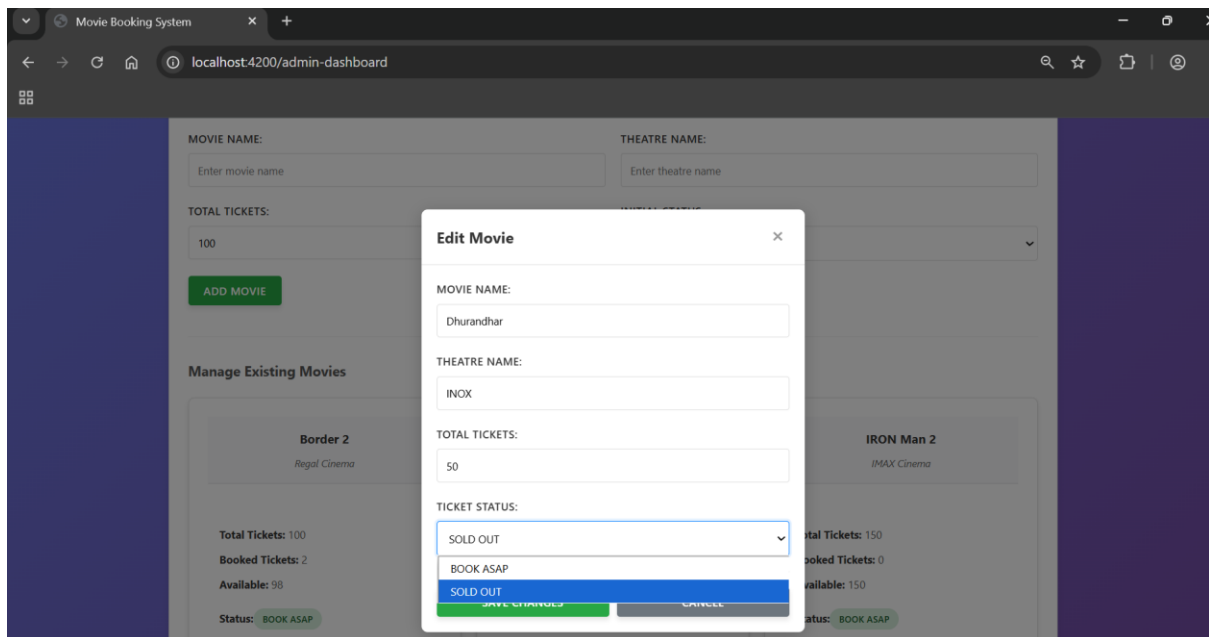
## 10.1.18 Added Movie Reflected in Dashboard



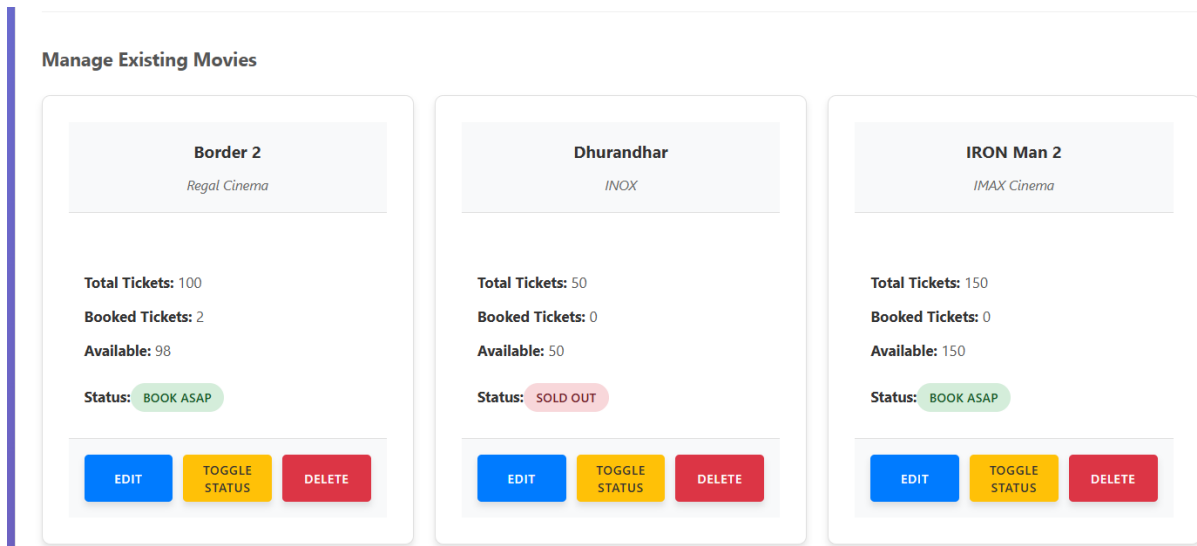
## 10.1.19 Admin Can Manually Mark Movie As SOLD OUT



## 10.1.20 Admin Can Manually Mark Movie As BOOK ASAP



## 10.1.21 Through Refresh Availability Option Ticket Status will be Set again (If Available tickets equal to Zero, Status Will be marked as SOLD OUT)



So here we are providing flexibility to admin, of Manually setting the status of Movie tickets booking and Automatically status changes based on ticket availability.

## 10.1.22 Ticket Book Validation Functionality

← → ↺ 🏠

localhost:4200/book/Dhurandhar?theatre=INOX&totalTickets=50&availableTickets=50

🔍 ☆ 📄 🗑

Book Tickets for Dhurandhar

Theatre: INOX

Available Tickets: 50

Status: BOOK ASAP

NUMBER OF TICKETS \*

5

SEAT SELECTION \*

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
H1	H2	H3	H4	H5	H6	H7	H8	H9	H10
I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
J1	J2	J3	J4	J5	J6	J7	J8	J9	J10

Selected Seats: A1, A2, B2, B1

Please select exactly 5 seat(s)

BOOK TICKETS

## 10.1.23 Successful Ticket Booked

SEAT SELECTION \*

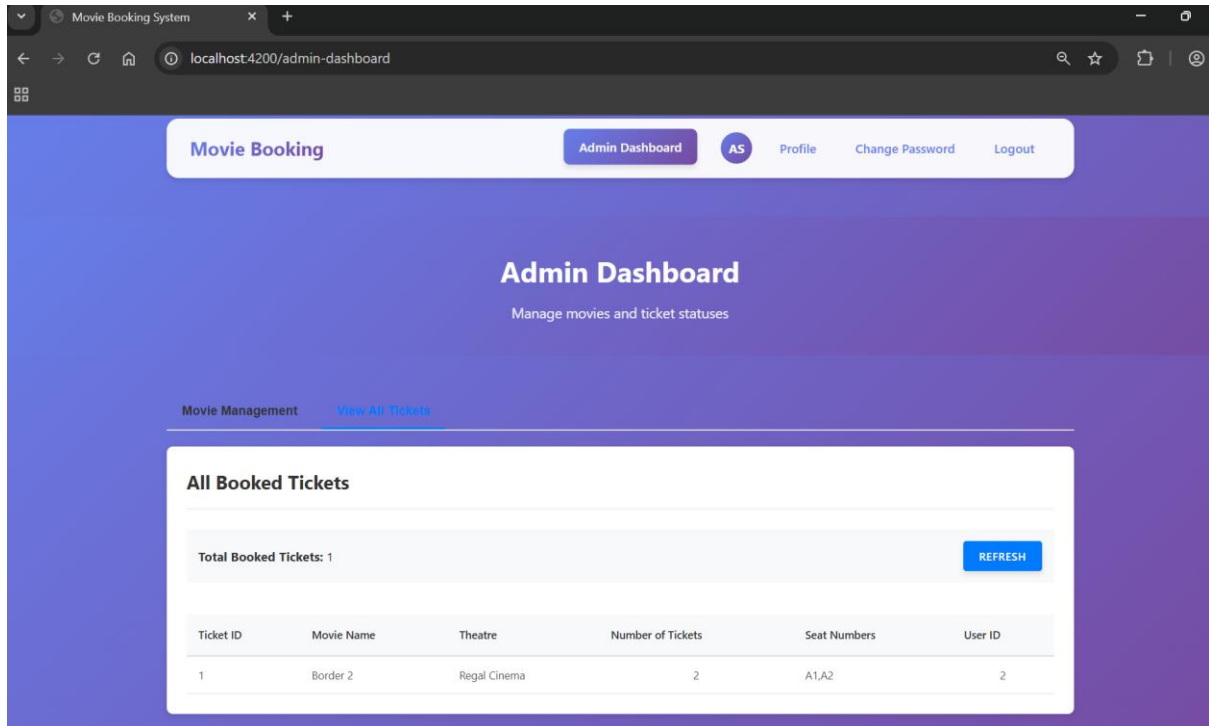
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
H1	H2	H3	H4	H5	H6	H7	H8	H9	H10
I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
J1	J2	J3	J4	J5	J6	J7	J8	J9	J10

Selected Seats: A1, B1, B2, A2, A3

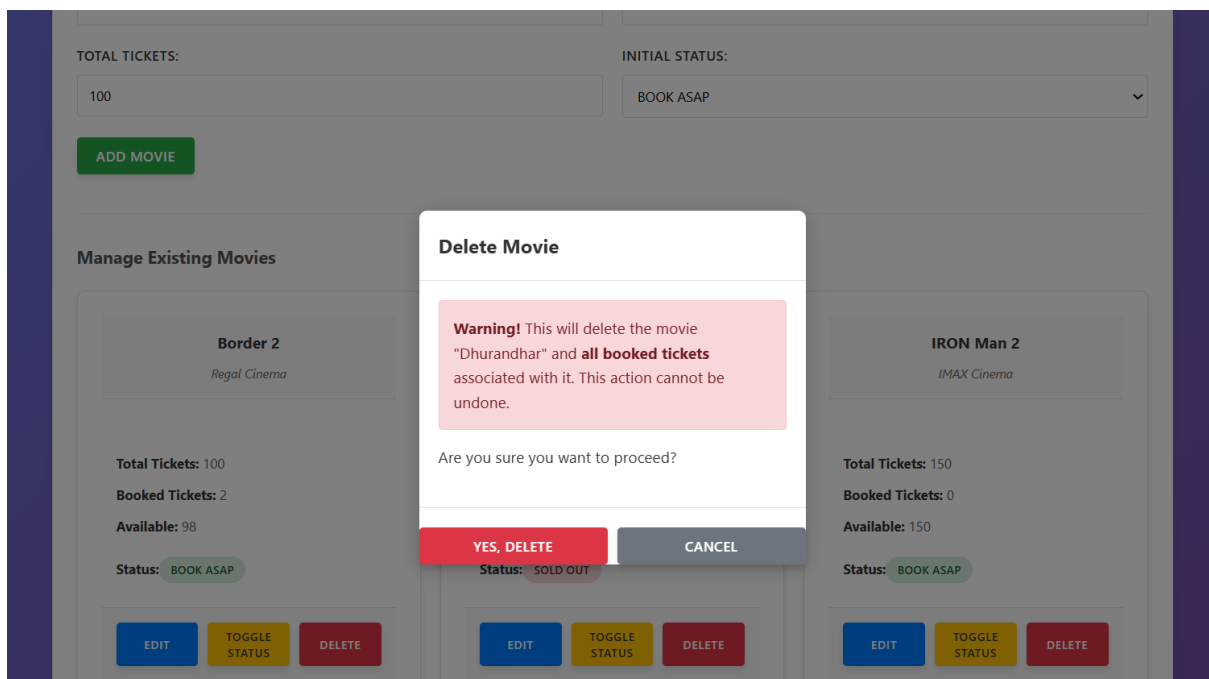
Tickets booked successfully! Redirecting...

BOOK TICKETS

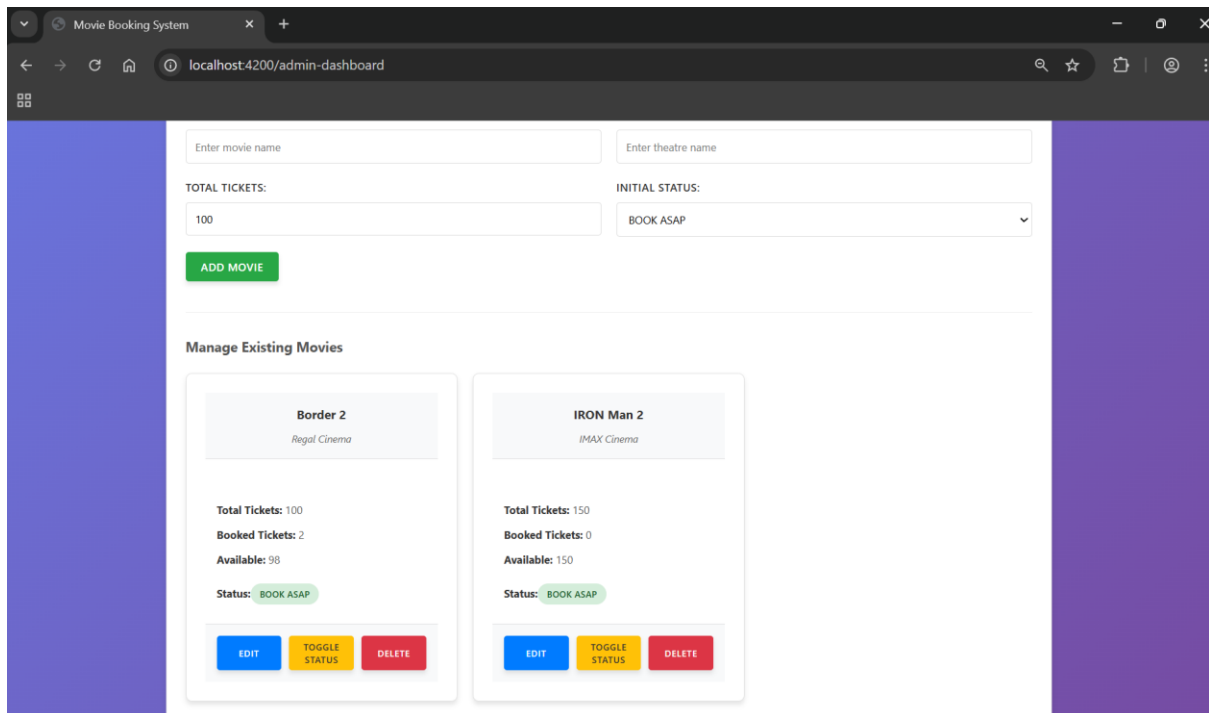
### 10.1.24 In Admin Dashboard Also, changes Reflected through Refresh Availability option



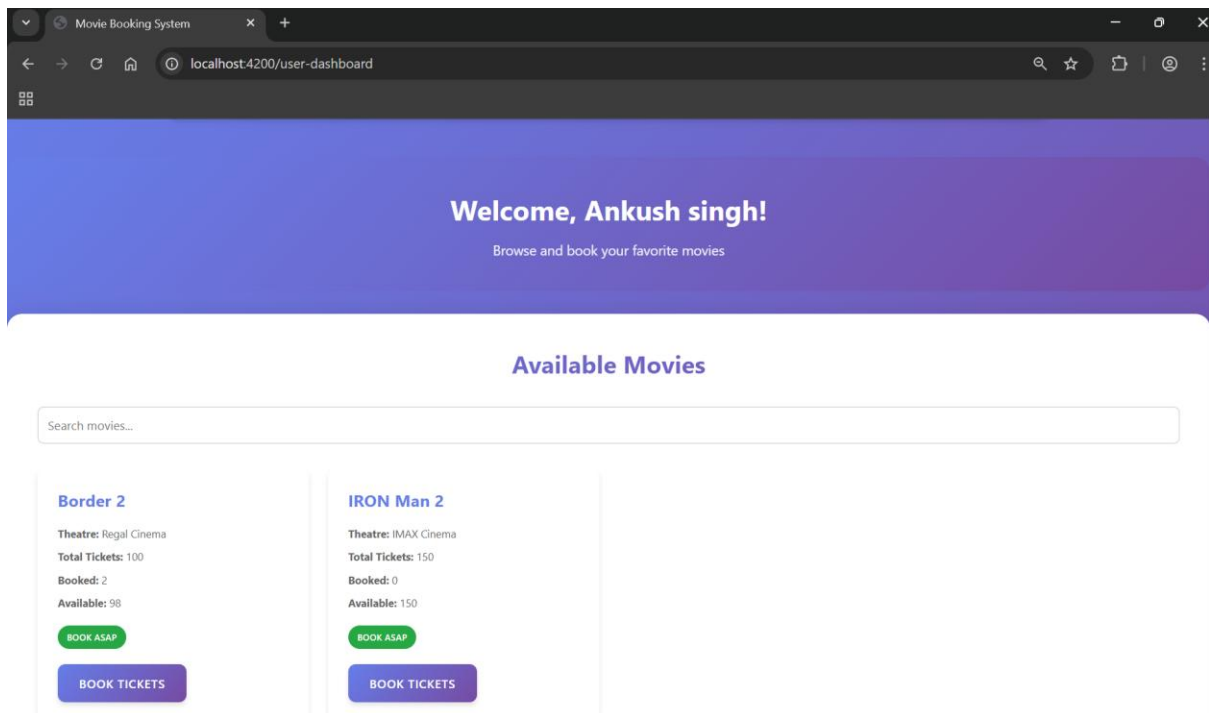
### 10.1.25 Admin Can Delete the Movie



### 10.1.26 Admin Dashboard After Deleting Dhurandhar Movie



### 10.1.27 In User Home Page Movie Removed



10.1.28      The ticket was also Removed for the Respective Deleted Movie

Before:

