

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 Error! Bookmark not defined.	
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..... Error! Bookmark not defined.	
9.7.26	Ticket Also reflected in Ticket List..... Error! Bookmark not defined.	
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

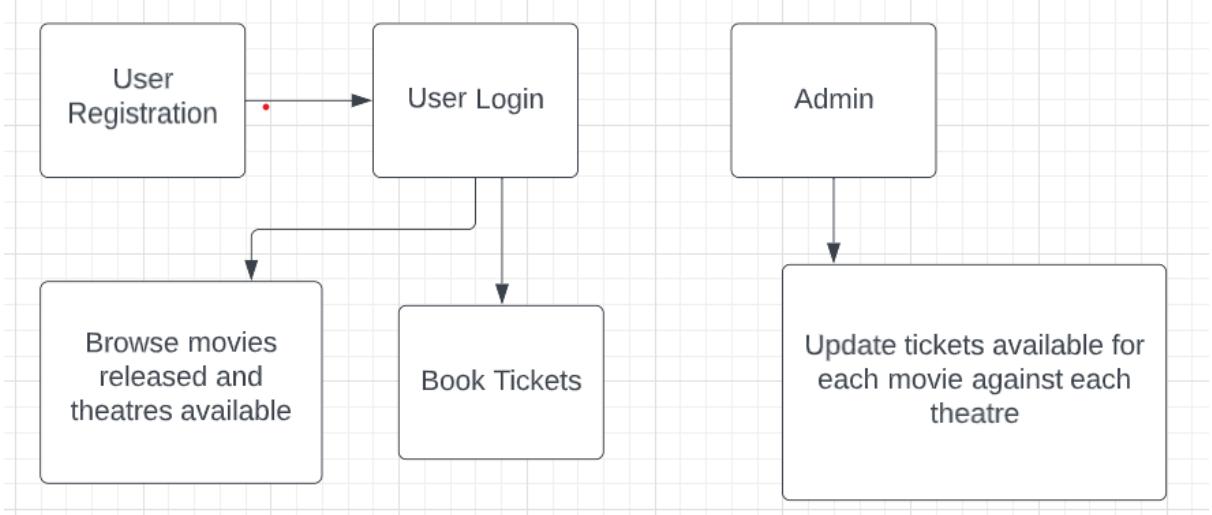
Movie Booking App is a full-stack web application that enables users to register, authenticate, and browse movies to book tickets for their preferred shows. The system allows administrators to manage movie inventory, theatre information, and ticket availability based on bookings. Guest users cannot perform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

2 Key Objectives:

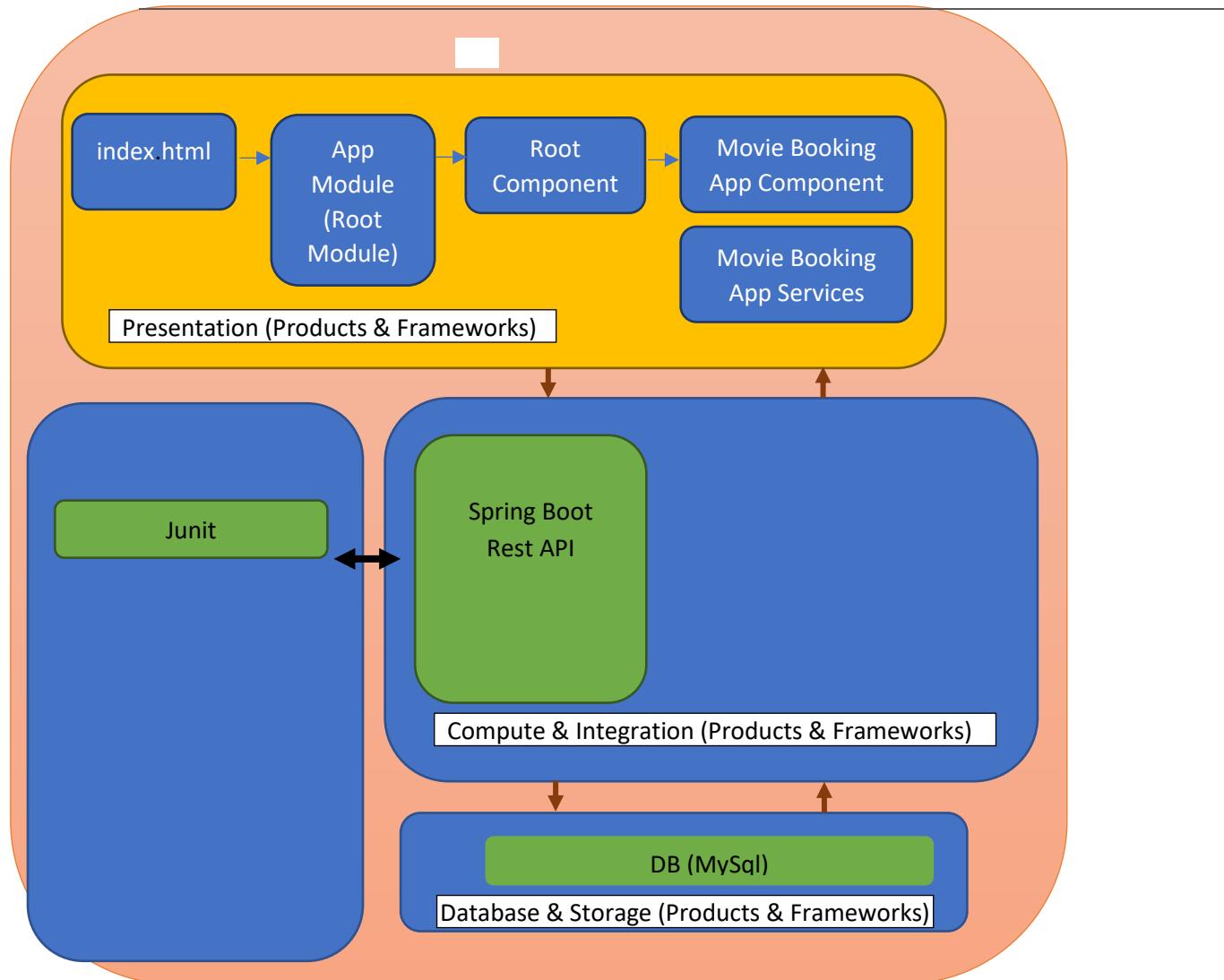
- Provide a secure user authentication mechanism for registration and login
- Enable users to search and browse available movies across different theatres
- Allow authenticated users to book tickets for selected movies and showtimes
- Provide administrators with tools to manage movies, theatres, and 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)
<u>Business Requirement</u>				
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:

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"> 1. A logged-in user can reset their password so they can login, even if they forget their password. 2. A logged-in user: <ol style="list-style-type: none"> a. Cannot change their username. b. Can logout from their account. 3. As a customer I should be able to furnish following details at the time of registration <ol style="list-style-type: none"> a. First Name b. Last Name c. Email d. Login Id e. Password f. Confirm Password g. Contact Number 4. All details fields must be mandatory 5. Login Id and Email must be unique 6. Password and Confirm Password must be same 7. If any constraint is not satisfied, validation message must be shown
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"> a. User can view all the existing released movies. b. User can search any particular movie based on the movie name
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"> a. Book a movie ticket b. Below are the details to be added <ul style="list-style-type: none"> • Movie Name • Theatre name • Number of tickets

		<ul style="list-style-type: none"> • Seat Number
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"> a. View number of booked tickets for a particular movie from table “Tickets” b. Check the tickets available from table “Movie” by calculating the total tickets booked c. If the quantity is 0, update the ticket status as ‘SOLD OUT’, else update as ‘BOOK ASAP’

8 RUBRICS/EXPECTED DELIVERABLES

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

- a. Use Spring Boot to version and implement the REST endpoints.
- b. 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/v1.0/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

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

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

1. As an application developer:
 - a. Implement ORM with Spring Data MongoRepository and MongoDB / JPARespository 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

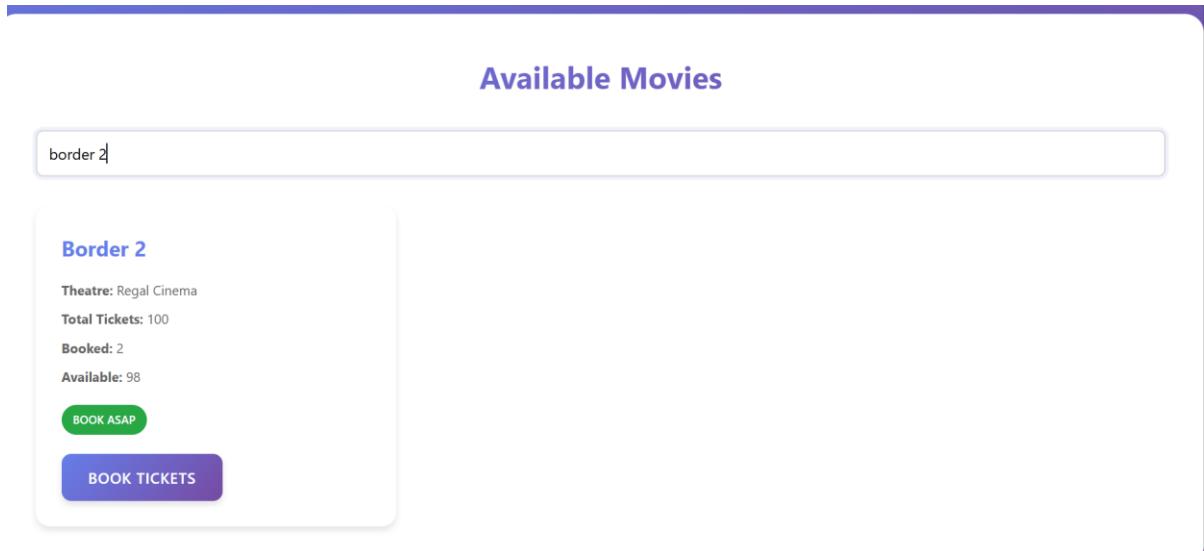
10 EXPECTED OUTCOMES:

10.1.1 MovieBookingApp Home Page

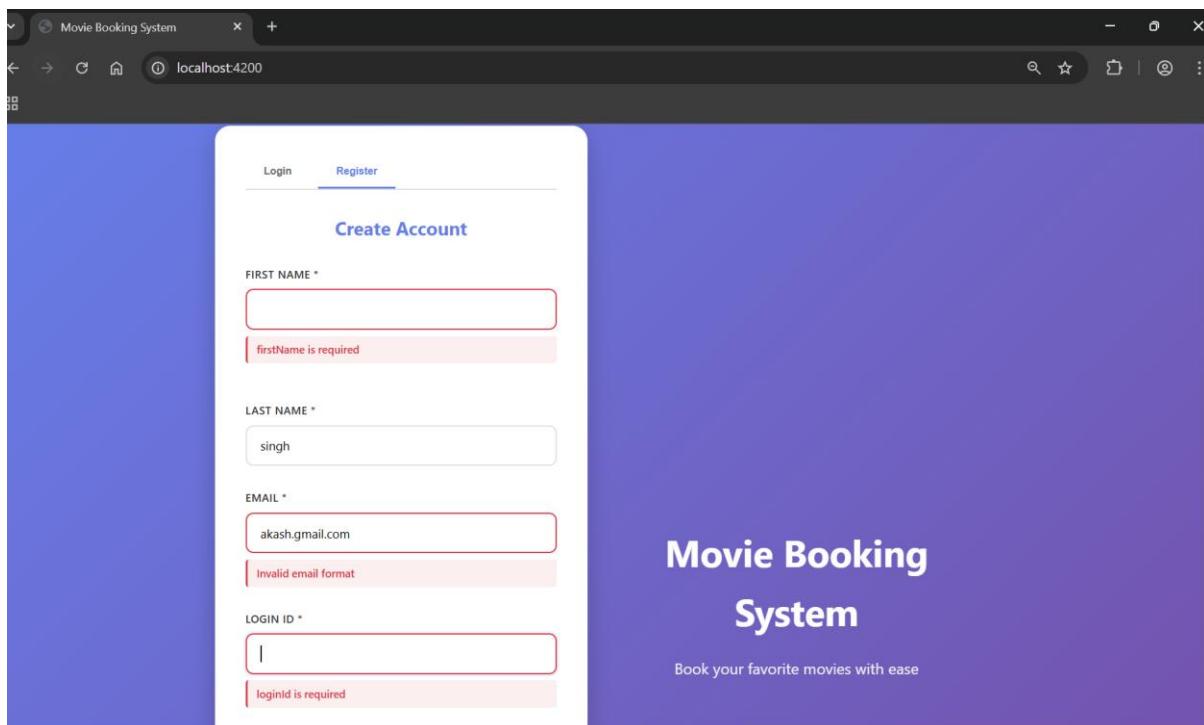
The screenshot shows the Movie Booking System home page. At the top, there is a navigation bar with links for 'Movies', 'My Tickets', 'Profile' (which has an 'AS' icon), 'Change Password', and 'Logout'. Below the navigation bar, a welcome message reads 'Welcome, Ankush singh!' followed by the sub-instruction 'Browse and book your favorite movies'. A search bar labeled 'Search movies...' is present. The main content area displays three movie cards:

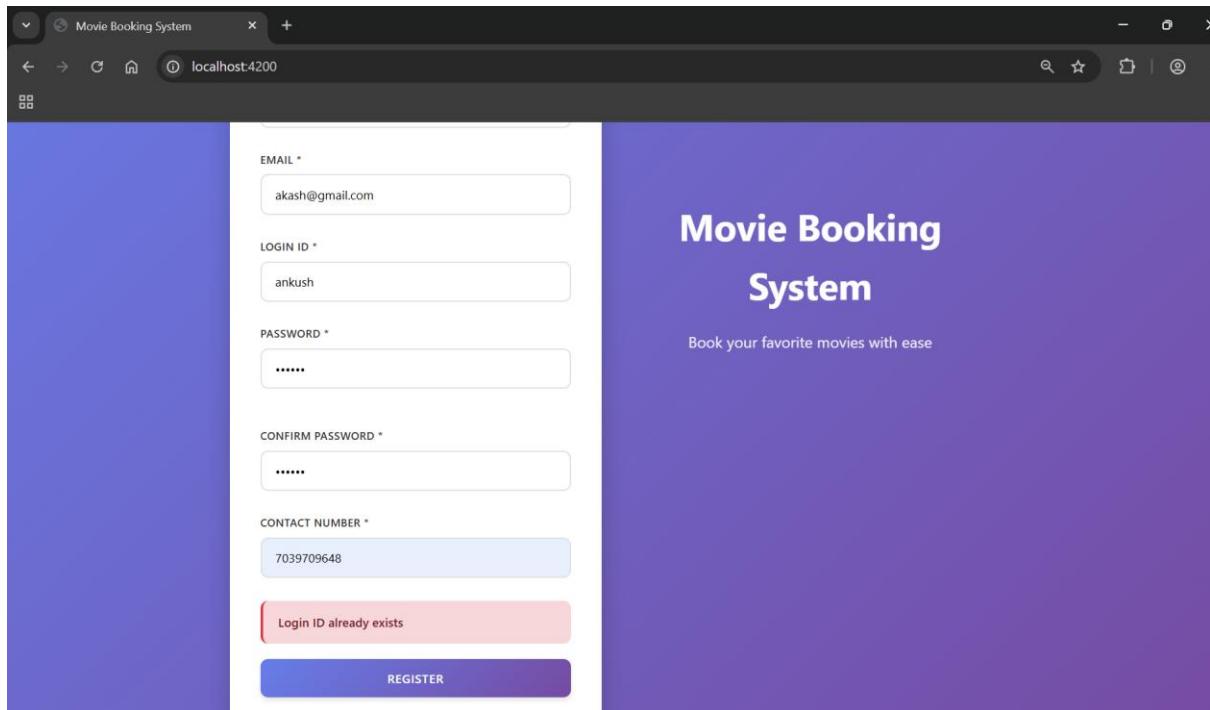
- Border 2**
Theatre: Regal Cinema
Total Tickets: 100
Booked: 2
Available: 98
[BOOK ASAP](#)
[BOOK TICKETS](#)
- Dhurandhar**
Theatre: INOX
Total Tickets: 50
Booked: 0
Available: 50
[BOOK ASAP](#)
[BOOK TICKETS](#)
- IRON Man 2**
Theatre: IMAX Cinema
Total Tickets: 150
Booked: 0
Available: 150
[BOOK ASAP](#)
[BOOK TICKETS](#)

10.1.2 Search By Movie Name



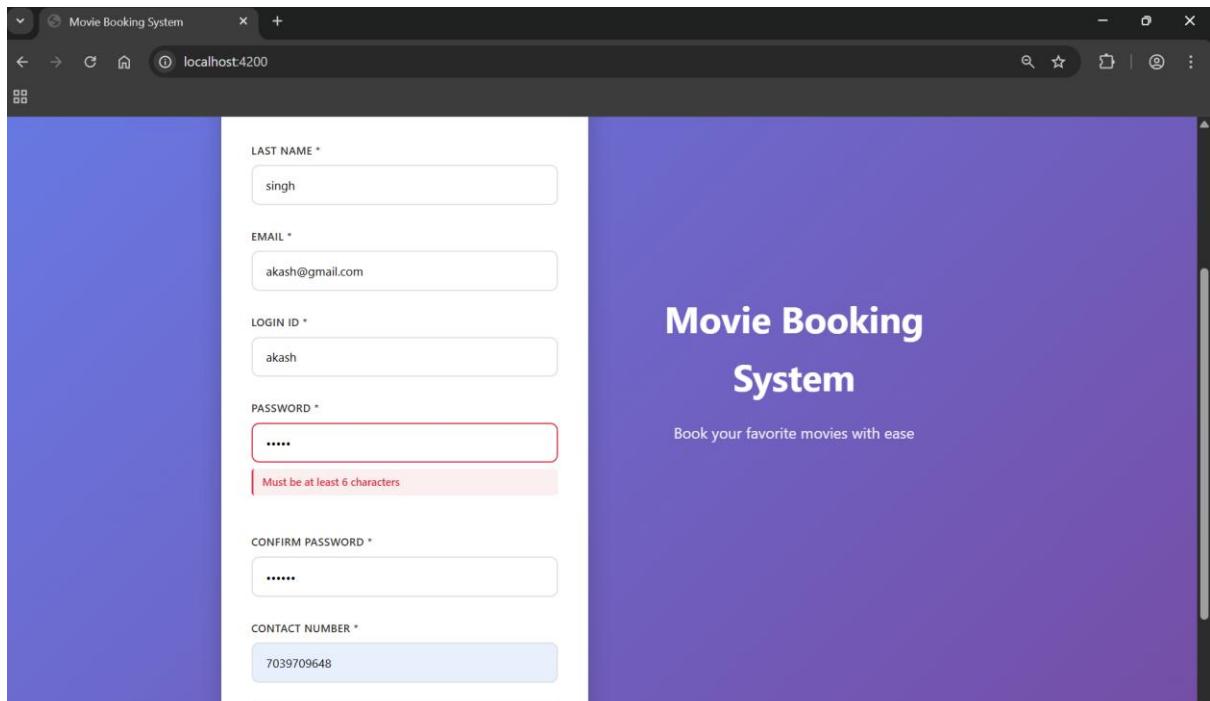
10.1.3 Registration Page Validation(field can't be empty, Login Id must be unique)



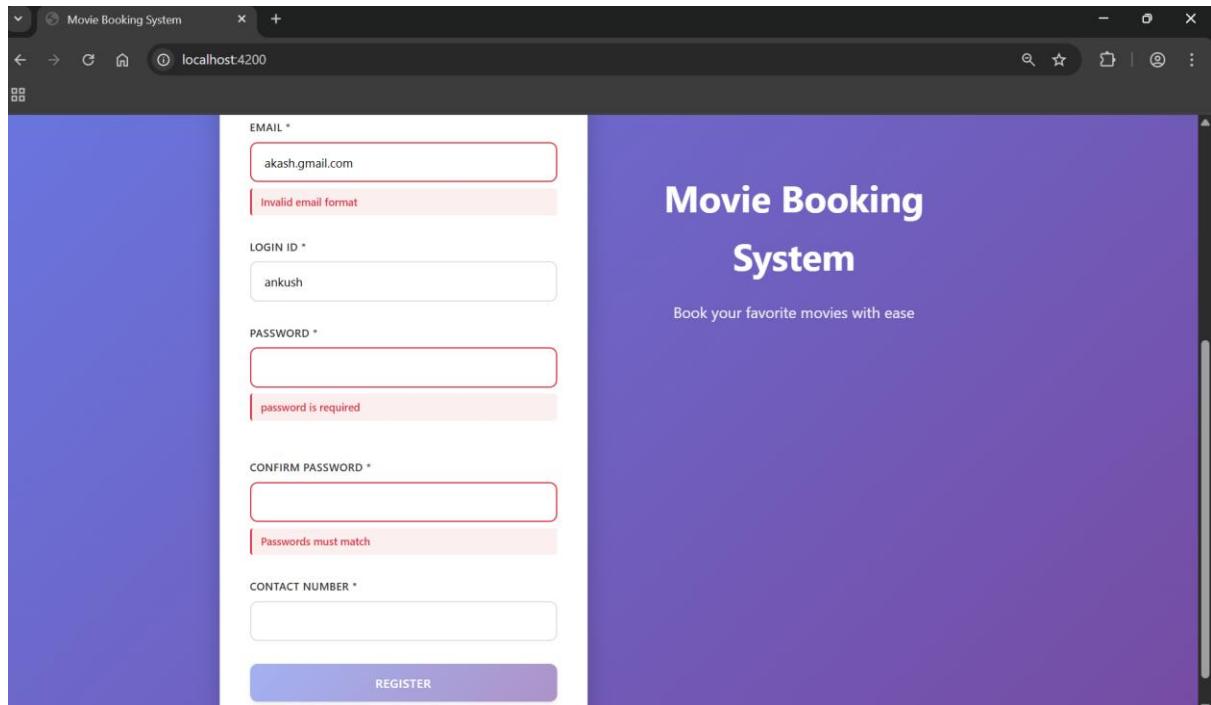


(Here, login Id “ankush” already exists. so, registration unsuccessful)

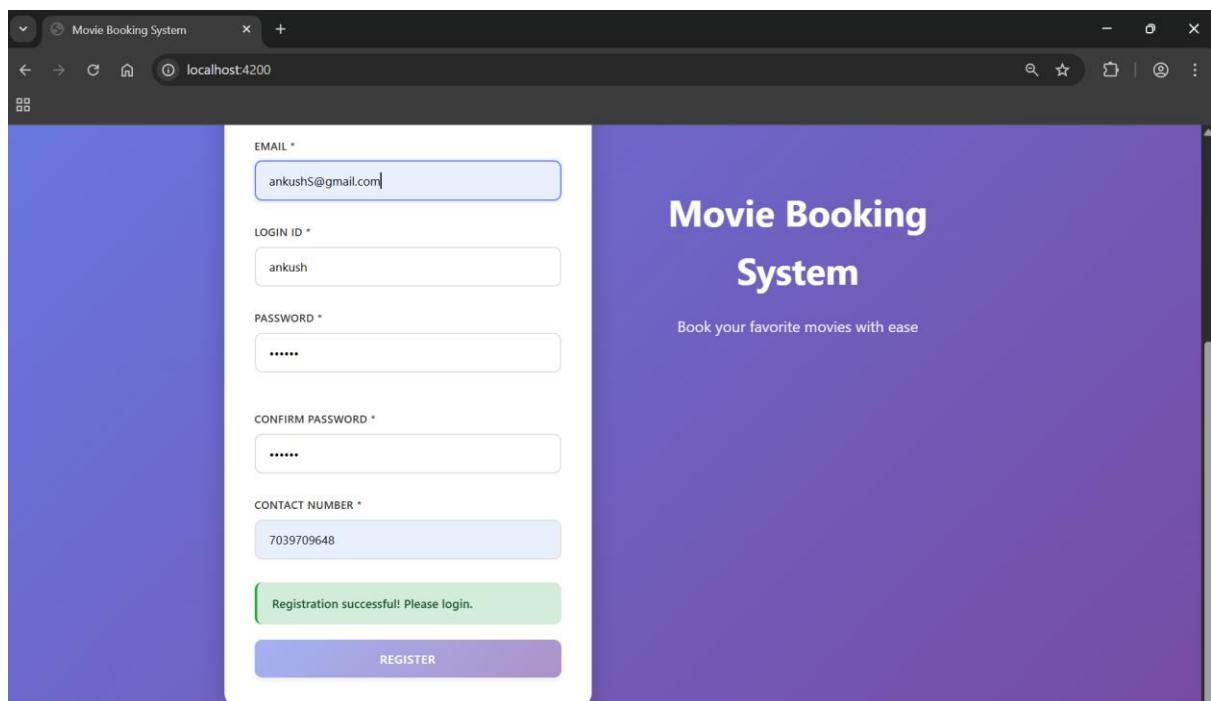
10.1.4 Registration Page Validation (Password Validation criteria- minimum 6 characters required)



10.1.5 Registration Page Validation (Password Do not Match)



10.1.6 Successful Registration



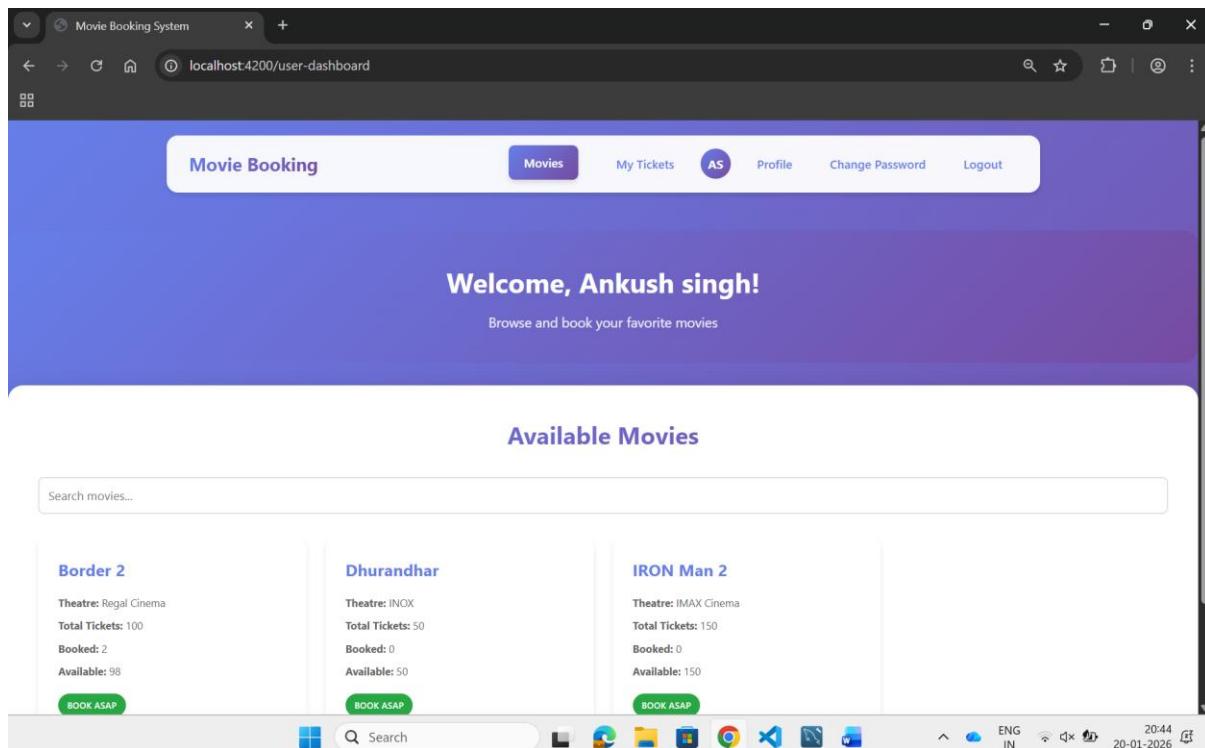
10.1.7Forgot Password _Reset_option

The screenshot shows a web application interface for 'Movie Booking'. At the top, there is a navigation bar with links for 'Movies', 'My Tickets', 'Profile' (which is currently selected, indicated by a blue background), 'Change Password', and 'Logout'. Below the navigation bar, the main content area has a title 'Change Password'. It contains three input fields: 'CURRENT PASSWORD *' with placeholder '.....', 'NEW PASSWORD *' with placeholder '.....', and 'CONFIRM NEW PASSWORD *' with placeholder '.....'. A blue button labeled 'CHANGE PASSWORD' is located below the input fields.

10.1.8Login Page

The screenshot shows a web browser window for 'Movie Booking System' at the URL 'localhost:4200'. The page features a large purple header with the text 'Movie Booking System' and the tagline 'Book your favorite movies with ease'. On the left side, there is a white login form with tabs for 'Login' (which is selected) and 'Register'. The form includes fields for 'LOGIN ID *' and 'PASSWORD *', both with placeholder text. A blue 'LOGIN' button is at the bottom, and a link for 'Forgot Password?' is below it. The overall design is clean and modern.

10.1.9 Home Page After Successful Login



10.1.10 Sold Out Option Disables Tickets Booking

Welcome, Ankush singh!

Browse and book your favorite movies

Available Movies

Search movies...

Movie	Theatre	Total Tickets	Booked	Available
Border 2	Regal Cinema	100	2	98
Dhurandhar	INOX	50	0	50
IRON Man 2	IMAX Cinema	150	0	150

BOOK ASAP

BOOK TICKETS

SOLD OUT

BOOK TICKETS

BOOK ASAP

BOOK TICKETS

10.1.11 Can View the Tickets Booked by Logged In User

Movie Booking

Movies My Tickets AS Profile Change Password Logout

My Booked Tickets

TICKET ID	MOVIE NAME	THEATRE	NUMBER OF TICKETS	SEAT NUMBERS
1	Border 2	Regal Cinema	2	A1,A2

10.1.12 Ticket Book Page

Movie Booking System

localhost:4200/book/Border%202?theatre=Regal%20Cinema&totalTickets=100&availableTickets=98

Book Tickets for Border 2

Theatre: Regal Cinema
Available Tickets: 98
Status: **BOOK ASAP**

NUMBER OF TICKETS *

1

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

Movie Booking System

localhost:4200/book/Border%202?theatre=Regal%20Cinema&totalTickets=100&availableTickets=98

Book Tickets for Border 2

Theatre: Regal Cinema
Available Tickets: 98
Status: **BOOK ASAP**

NUMBER OF TICKETS *

1

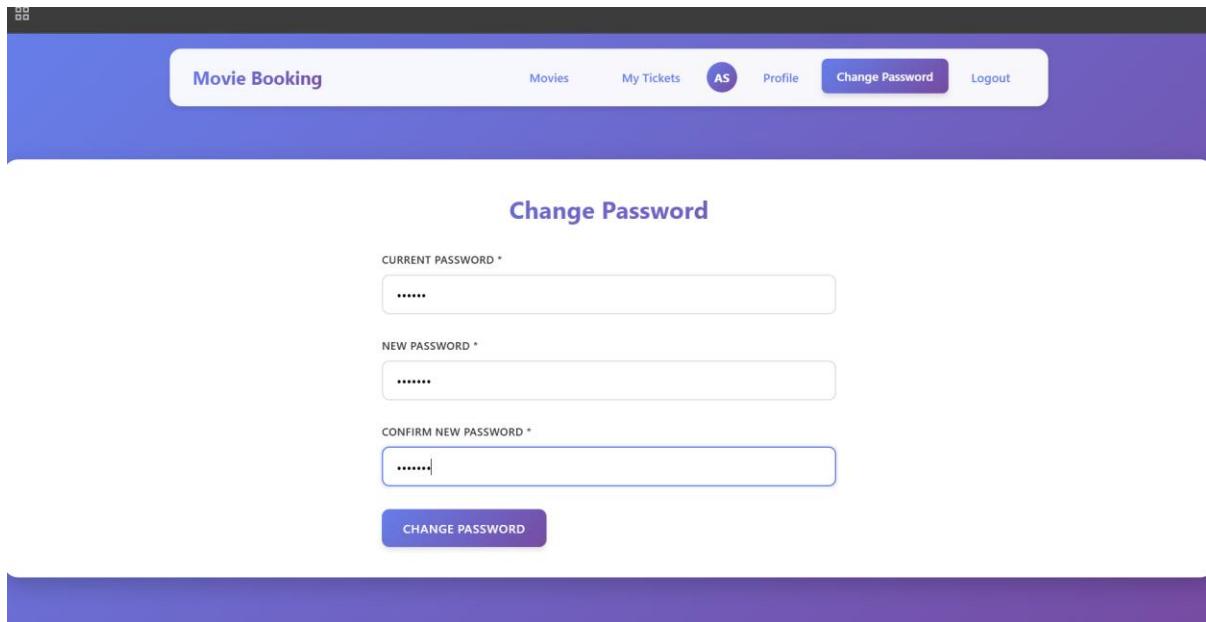
SEAT SELECTION *

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: C6

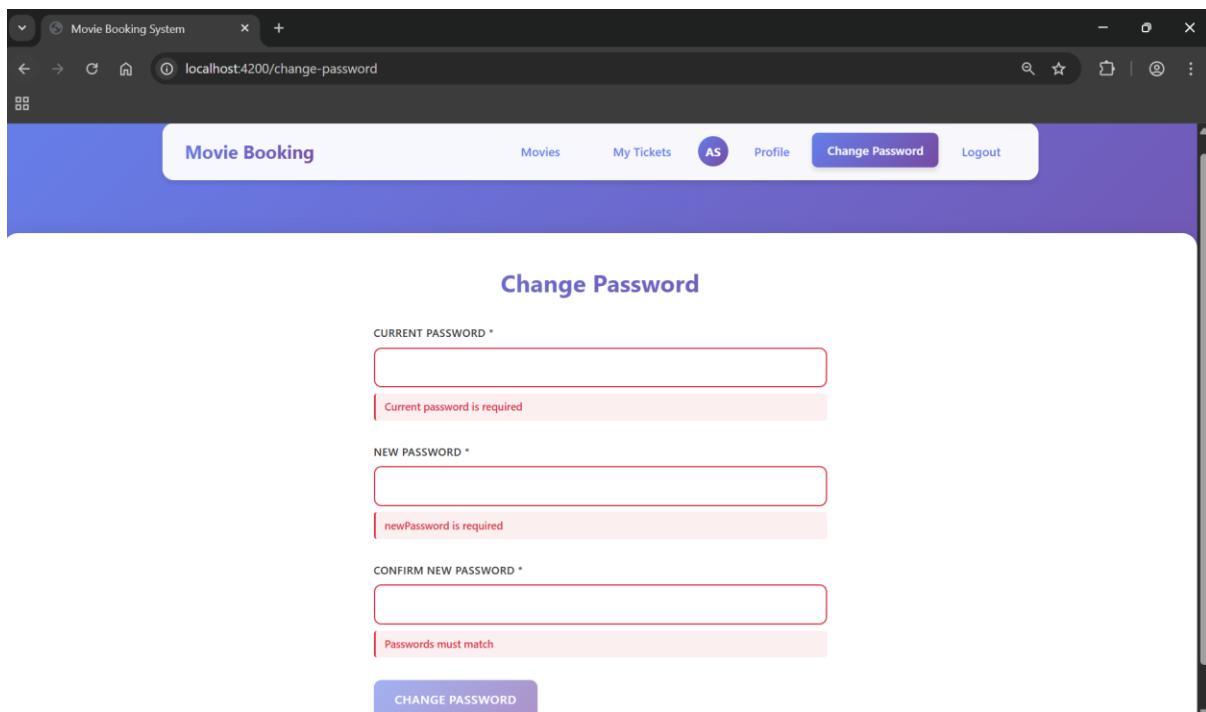
BOOK TICKETS

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



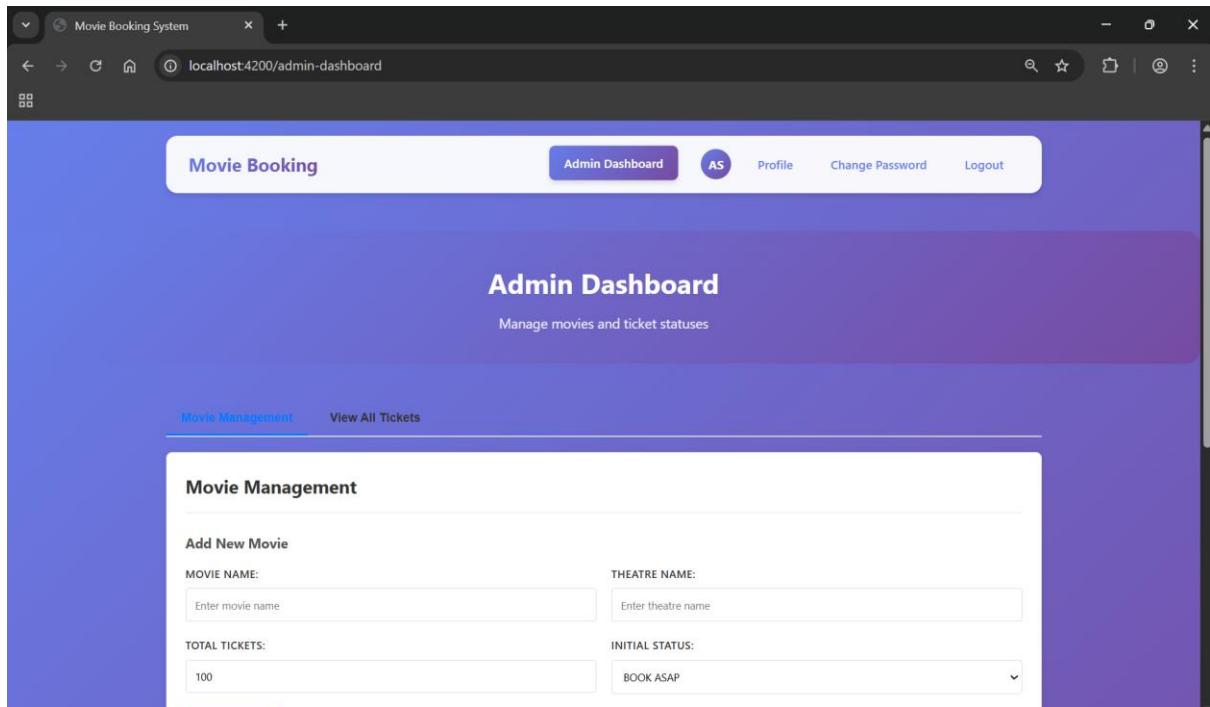
The screenshot shows the 'Change Password' page of the Movie Booking System. At the top, there is a navigation bar with links for 'Movies', 'My Tickets', 'Profile' (which is currently selected, indicated by a blue background), 'Change Password' (also in blue), and 'Logout'. Below the navigation bar, the main content area has a title 'Change Password'. It contains three input fields: 'CURRENT PASSWORD *' with placeholder '.....', 'NEW PASSWORD *' with placeholder '.....', and 'CONFIRM NEW PASSWORD *' with placeholder '.....'. A blue 'CHANGE PASSWORD' button is located at the bottom of the form.

10.1.14 Change Password Validation

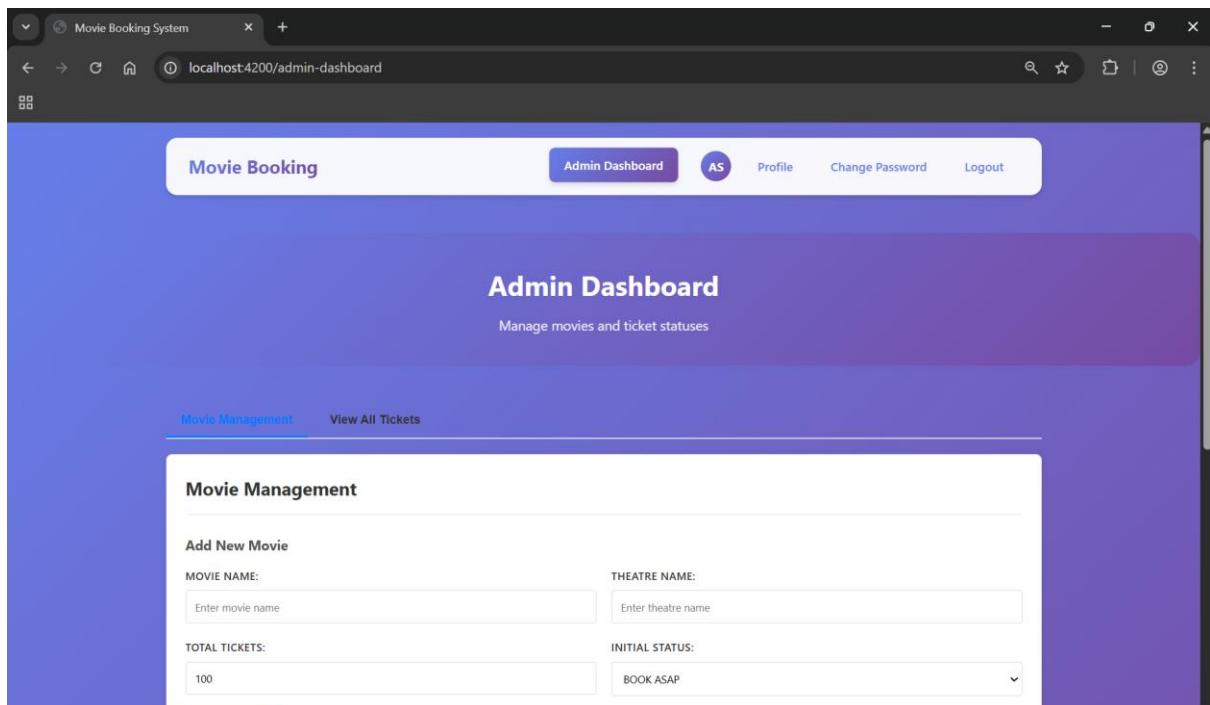


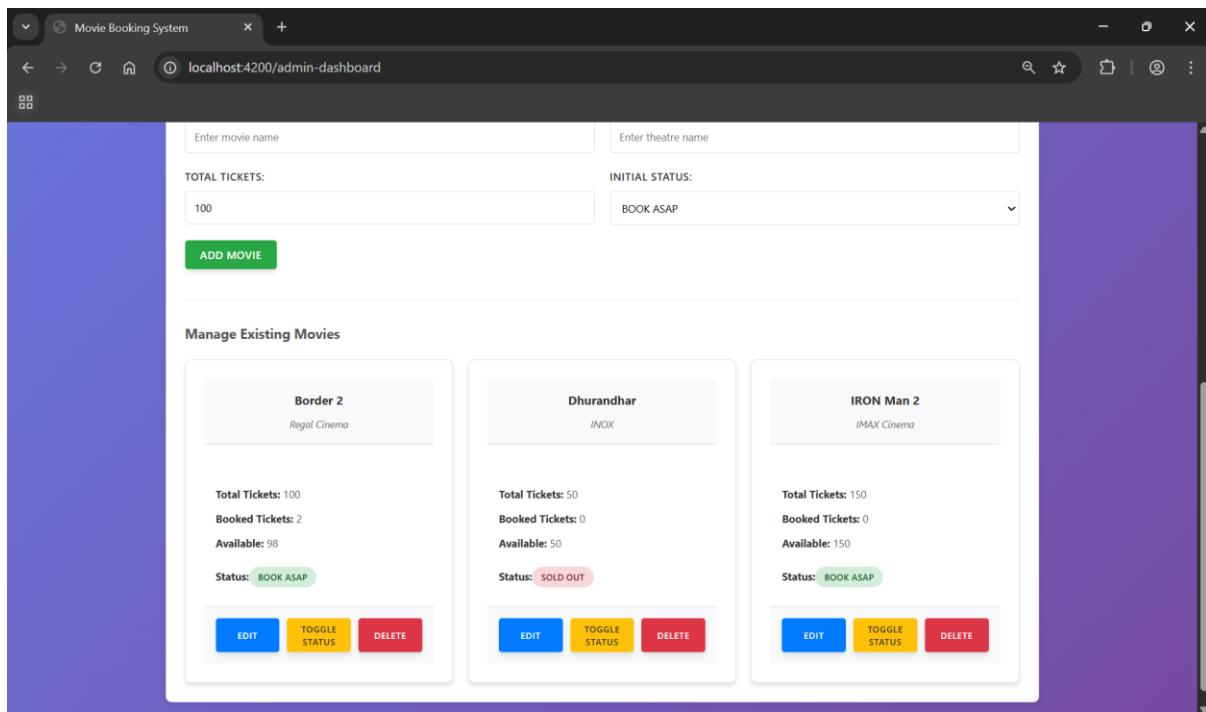
The screenshot shows the 'Change Password' page with validation errors. The browser's address bar indicates the URL is 'localhost:4200/change-password'. The validation errors are: 'CURRENT PASSWORD *' field is highlighted in red with the message 'Current password is required'. The 'NEW PASSWORD *' field is also highlighted in red with the message 'newPassword is required'. The 'CONFIRM NEW PASSWORD *' field is highlighted in red with the message 'Passwords must match'. The 'CHANGE PASSWORD' button is visible at the bottom.

10.1.15 Admin Login

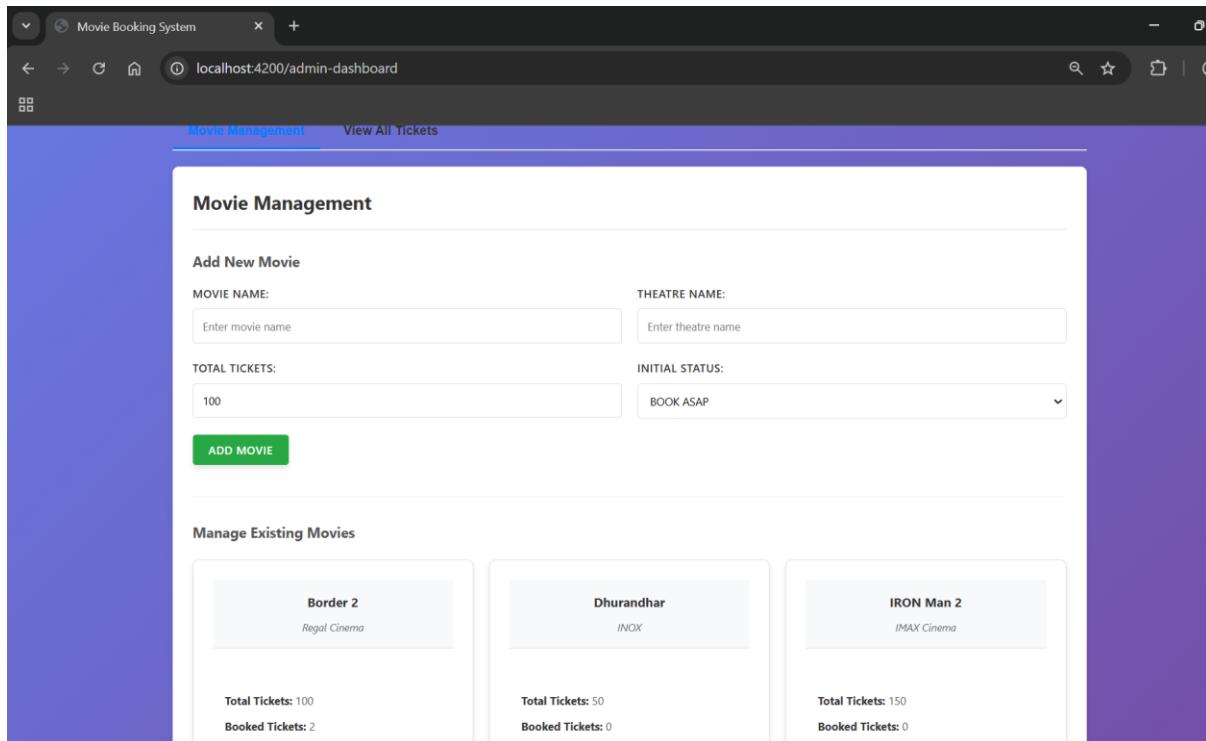


10.1.16 Admin Dashboard After Login

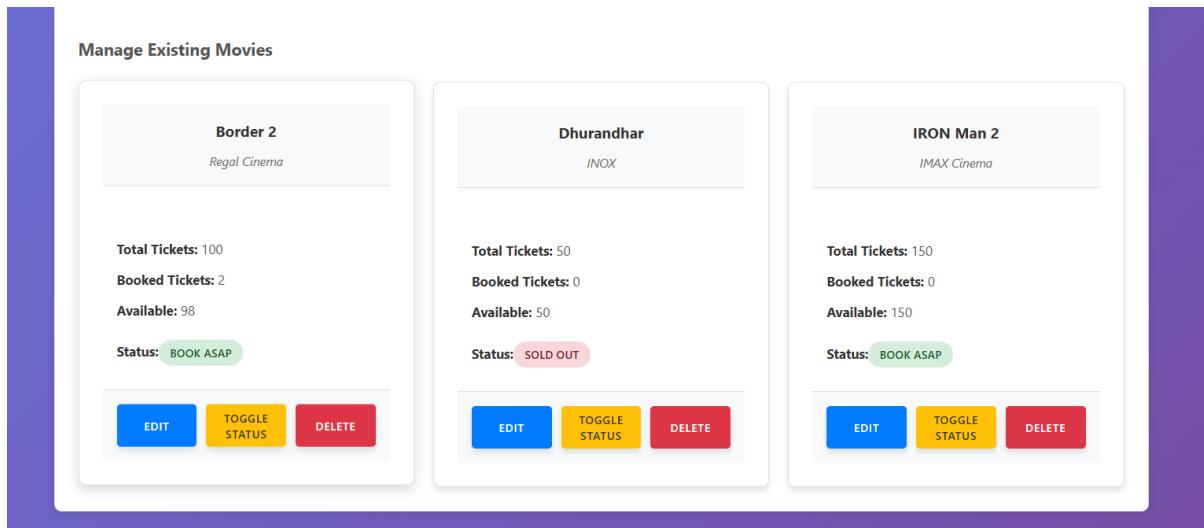




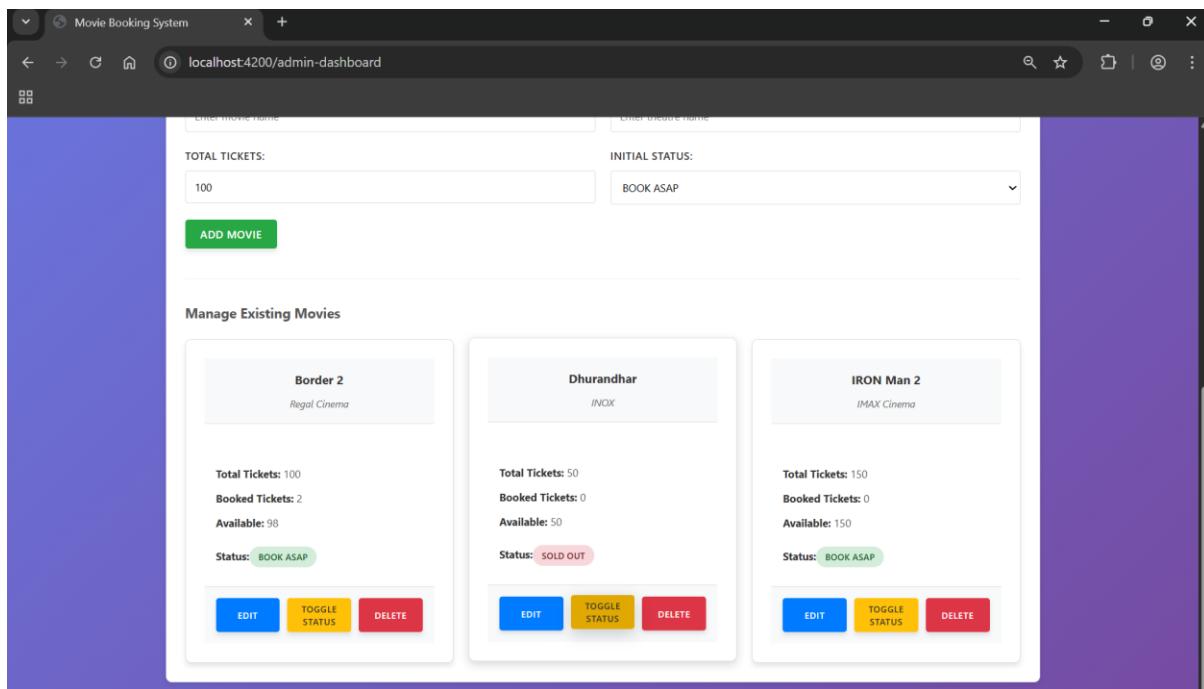
10.1.17 Movie added By Admin (Movie Name-Sigma)



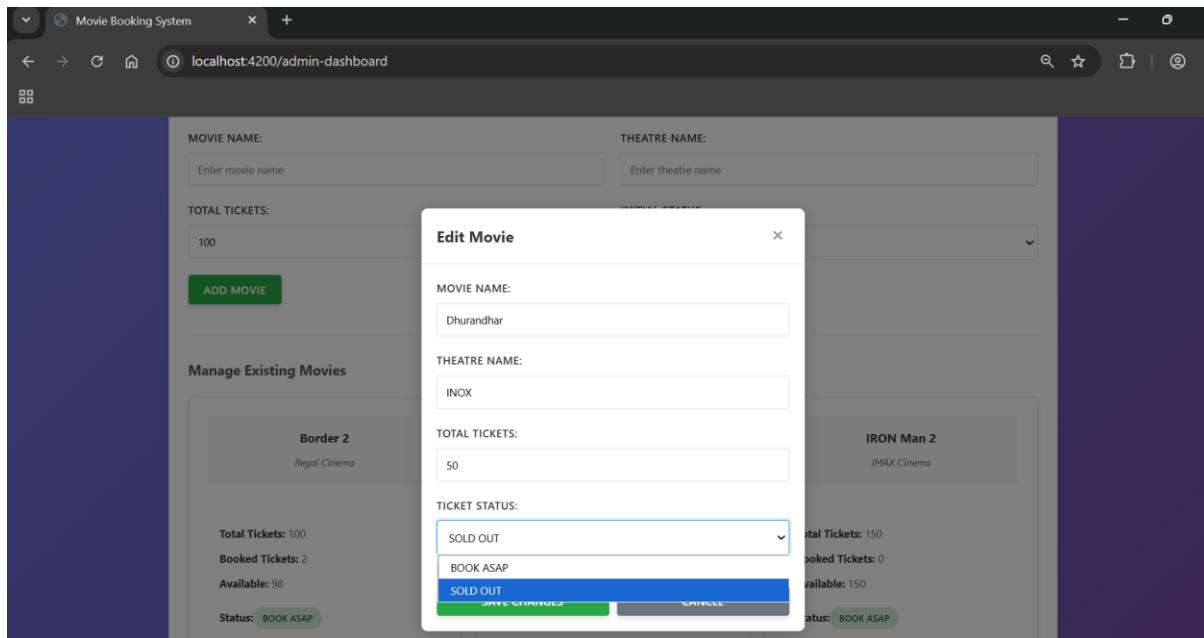
10.1.18 Added Movie Reflected in Dashboard



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)

A screenshot of the 'Manage Existing Movies' section of the admin dashboard. It displays three movie cards: 'Border 2' (Regal Cinema, 100 total, 2 booked, 98 available, Status: BOOK ASAP), 'Dhurandhar' (INOX, 50 total, 0 booked, 50 available, Status: SOLD OUT), and 'IRON Man 2' (IMAX Cinema, 150 total, 0 booked, 150 available, Status: BOOK ASAP). Each card has three buttons at the bottom: 'EDIT' (blue), 'TOGGLE STATUS' (yellow), and 'DELETE' (red).

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

The screenshot shows a web browser window for booking tickets at localhost:4200/book/Dhurandhar?theatre=INOX&totalTickets=50&availableTickets=50. The page title is "Book Tickets for Dhurandhar". It displays a seating chart with rows labeled A through J and columns labeled 1 through 10. Seats A1, A2, and B1 are highlighted in blue, while others are green. Below the chart, a message says "Selected Seats: A1, A2, B1". A red error message box says "Please select exactly 5 seat(s)". A "BOOK TICKETS" button is visible.

10.1.23 Successful Ticket Booked

The screenshot shows a confirmation message: "Tickets booked successfully! Redirecting...". Below it is a seating chart identical to the one in the previous screenshot, with the same highlighted seats (A1, A2, B1) and a "BOOK TICKETS" button.

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

The screenshot shows the Admin Dashboard of the Movie Booking System. At the top, there's a navigation bar with 'Movie Booking', 'Admin Dashboard', 'Profile', 'Change Password', and 'Logout'. Below the navigation, the main title is 'Admin Dashboard' with the subtitle 'Manage movies and ticket statuses'. A sub-navigation bar has 'Movie Management' selected and 'View All Tickets' highlighted. A modal window titled 'All Booked Tickets' is open, displaying a table with one row of data: Ticket ID 1, Movie Name Border 2, Theatre Regal Cinema, Number of Tickets 2, Seat Numbers A1,A2, and User ID 2. Above the table, it says 'Total Booked Tickets: 1' and has a 'REFRESH' button. The background shows other movie management cards for 'Border 2' and 'IRON Man 2'.

10.1.25 Admin Can Delete the Movie

The screenshot shows the Admin Dashboard with a modal window titled 'Delete Movie'. The modal contains a warning message: 'Warning! This will delete the movie "Dhurandhar" and all booked tickets associated with it. This action cannot be undone.' Below the message is a confirmation question: 'Are you sure you want to proceed?'. At the bottom of the modal are two buttons: 'YES, DELETE' (in red) and 'CANCEL'. The background shows movie management cards for 'Border 2' and 'IRON Man 2'. The 'Border 2' card shows Total Tickets: 100, Booked Tickets: 2, Available: 98, and Status: BOOK ASAP. The 'IRON Man 2' card shows Total Tickets: 150, Booked Tickets: 0, Available: 150, and Status: BOOK ASAP. Each card has 'EDIT', 'TOGGLE STATUS', and 'DELETE' buttons.

10.1.26 Admin Dashboard After Deleting Dhurandhar Movie

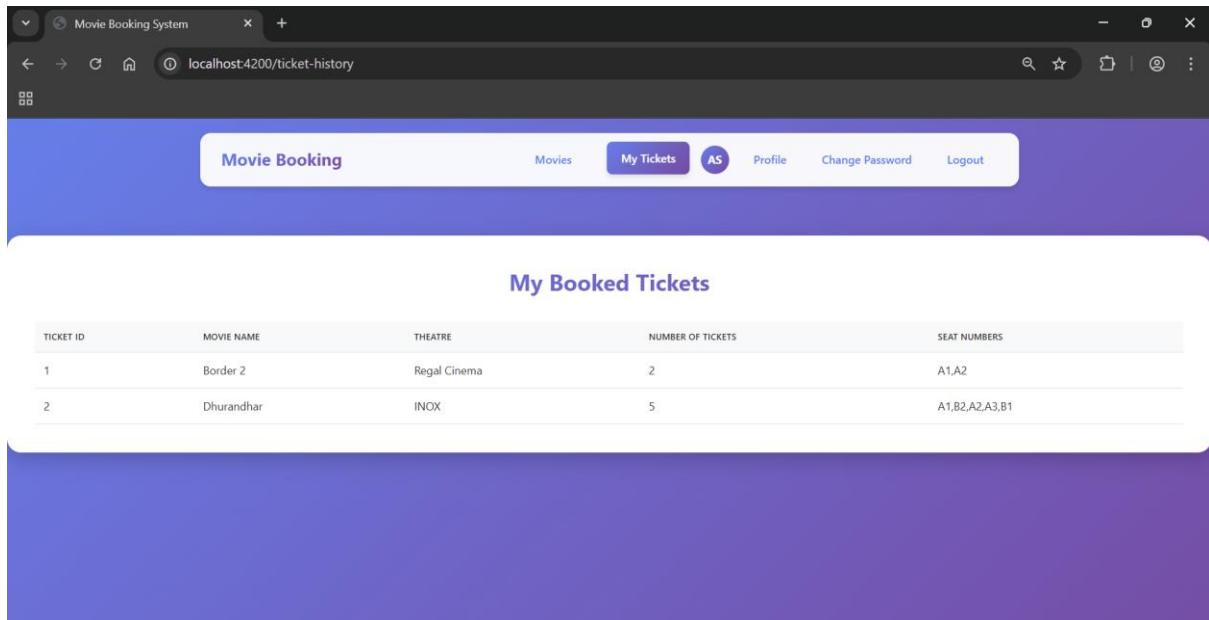
The screenshot shows the Admin Dashboard of the Movie Booking System. At the top, there are input fields for 'Enter movie name' and 'Enter theatre name'. Below these are sections for 'TOTAL TICKETS:' (set to 100) and 'INITIAL STATUS:' (set to 'BOOK ASAP'). A green 'ADD MOVIE' button is located below these fields. The main area is titled 'Manage Existing Movies' and displays two movie cards: 'Border 2' (Regal Cinema) and 'IRON Man 2' (IMAX Cinema). Each card provides movie details (Total Tickets, Booked Tickets, Available), current status ('BOOK ASAP'), and action buttons for 'EDIT', 'TOGGLE STATUS', and 'DELETE'. The 'Border 2' card has a blue border, while the 'IRON Man 2' card has a red border.

10.1.27 In User Home Page Movie Removed

The screenshot shows the User Home Page of the Movie Booking System. It features a purple header with the welcome message 'Welcome, Ankush singh!' and the sub-instruction 'Browse and book your favorite movies'. Below this is a section titled 'Available Movies' with a search bar labeled 'Search movies...'. Two movie cards are displayed: 'Border 2' (Theatre: Regal Cinema, Total Tickets: 100, Booked: 2, Available: 98, Status: BOOK ASAP) and 'IRON Man 2' (Theatre: IMAX Cinema, Total Tickets: 150, Booked: 0, Available: 150, Status: BOOK ASAP). Each card includes a 'BOOK TICKETS' button.

10.1.28 The ticket was also Removed for the Respective Deleted Movie

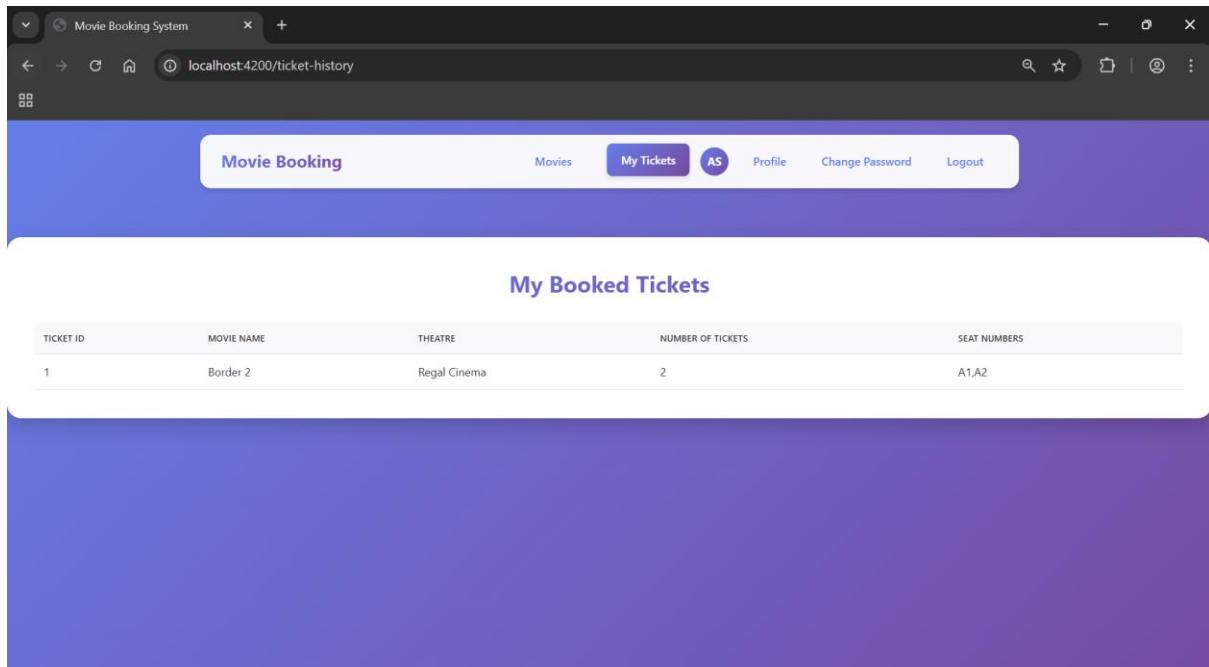
Before:



The screenshot shows a web browser window for the "Movie Booking System" at the URL "localhost:4200/ticket-history". The interface has a purple header bar with navigation links: "Movie Booking", "Movies", "My Tickets" (which is highlighted in blue), "Profile", "Change Password", and "Logout". Below the header is a section titled "My Booked Tickets" containing a table with the following data:

TICKET ID	MOVIE NAME	THEATRE	NUMBER OF TICKETS	SEAT NUMBERS
1	Border 2	Regal Cinema	2	A1,A2
2	Dhurandhar	INOX	5	A1,B2,A2,A3,B1

After:



The screenshot shows the same web browser window for the "Movie Booking System" at the URL "localhost:4200/ticket-history". The interface is identical to the "Before" screenshot, but the table now only displays one row of data:

TICKET ID	MOVIE NAME	THEATRE	NUMBER OF TICKETS	SEAT NUMBERS
1	Border 2	Regal Cinema	2	A1,A2