# PROJECT : ONLINE MOVIE TICKET BOOKING SYSTEM

## FUNCTIONAL SPECIFICATION:

| | |
|---|---|
| Project Code: | |
| Project Name: | Online Movie Ticket Booking System |

## FUNCTIONAL SPECIFICATION:

**Table of Contents**

# 1. INTRODUCTION:

In this project movie ticket is booked using movie Ticket booking system. We enter into Web page by logging with User Name and Password. Then we select the Movie and later in which Theatre movie is running. Later choose Show Timings and enter no of tickets you want . Finally it displays the details of the procedure and print the form to show at respective ticket counter to get ticket.

The main purpose of our online ticket booking system is to provide an alternate and convenient way for a customer to buy cinema tickets. It is an automatic system. After the data has been fed into the database, the staff does not need to do anything with the order once it is received through the system.

Movie Ticket Booking System has been developed to override the problems of existing manual system.

Every organization, whether is it big or small has challenges to overcome and maintaining the information of Customer, Movie, Payment, Shows, Seats etc.

This website is supported to eliminate and in some cases reduce the hardships faced by existing offline system. Moreover this system is designed to carry out the particular operation in smooth and effective manner.

No former or prior knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly.

## Scope and Overview:

The scope of the "Online Movie Ticket Booking System" project is to provide hassle free movie ticket booking experience to the end users. The system will be developed on a Windows operating system using Spring Boot, JPA, Hibernate and Angular framework.

In this project we book ticket using online movie Ticket reservation system. We enter into Web page by logging with User Name and Password.

Then we select the Movie and later in which Theatre movie is running. Later choose Show Timings and enter no of tickets you want .Finally it displays the details of the procedure and print the form to show at respective ticket counter to get ticket.

## 2. SYSTEM OVERVIEW

The “Online movie ticket booking” should support basic for all below listed users.

- Administrator (A)
- Customer (C)

### 2.1 Authentication & Authorization

#### 2.1.1 Authentication:

Any end-user should be authenticated using a unique userid and password. We enter into Web page by logging with User Name and Password.

#### 2.1.2 Authorization :

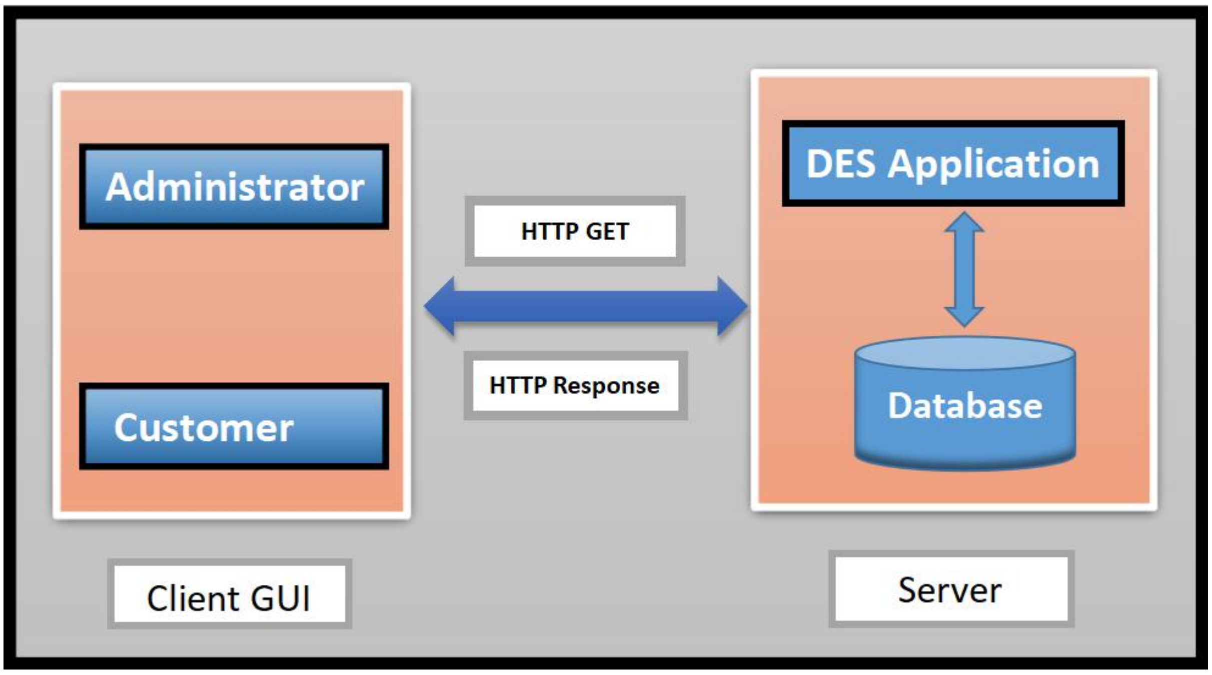
The operations supported and allowed would be based on the user type. For example, Administrator has the rights to add or remove different genres to or from the application to build rich product line. Edit movie details like name, ticket price language, description, and show timings, Enable or disable movie shows, delete the movies. He can also view movie ticket booking details.

Whereas Customer has a right to change password, Search movie ticket based on keyword. Add all the selected movie tickets to a cart and customize the purchase at the end, Experience payment process, Remove and Clear all the products from cart, Receive a booking summary page once a payment is complete.

## 2.2 FUNCTIONAL FLOW

The functional flow of the messages across different application components is shown below.

Ex. - Web Application



## 2.3 ENVIRONMENT:

The system will be developed on any Windows OS machine using Hibernate, Spring Boot and Angular Framework.

· Intel hardware machine (PC P4-2.26 GHz, 512 MB RAM, 40 GB HDD)

· Server – Apache Tomcat 8 or higher

· Database –MySQL

· JRE 8

· Eclipse IDE or Spring Tool Suite

## FUNCTIONAL SPECIFICATION

## 3 SUB-SYSTEM DETAILS:

In this project we book ticket using online movie Ticket reservation system. We enter into Web page by logging with User Name and Password. Then we select the Movie and later in which Theatre movie is running. Later choose Show Timings and enter no of tickets you want .Finally it displays the details of the procedure and print the form to show at respective ticket counter to get ticket.

Mainly there are two Roles of application **1) User/Customer 2) Admin.** Everyone can perform different operations according to their role and responsibilities.

### Admin:

- **Admin** is the root user of the application. Admin can handle all the customers and listed movies like booking, payment details.
- Admin Can Add/Delete/Update movie records.
- Admin Can Add/Delete/Update User/Customers.
- Admin can check the booking status and history of the user.
- Admin can check the payment details.

### User/Customers

- Customers are the front end-user of the application. So customers can view all the listed movies in a grid view and book tickets for those movies.
- Users can register and login to the application and check the booking history after login.
- Users can make online payments and book any movie according to dates.

## 3.1 Administrator

The administrator as a user is defined to perform below listed operations after successful login.

| ID | Objects | Operations | Data to include | Remarks |
|---|---|---|---|---|
| | Customer | View | Username, email, phone number, password | |
| | Theatres | Add<br>View<br>Delete<br>Modify | Manager contact, manager name, theatre city, theatre name. | |
| | Movie | Add<br>View<br>Delete<br>Modify | Movie id, movie date, movie description, movie genre, language, name, ratings. | |
| | Shows | Add<br>View<br>Delete<br>Modify | Show date, show end time, show name, show start time. | |
| | Seat | Add<br>View<br>Delete<br>Modify | Seat id, price, seat number, status, type. | |
| | Booking Details | View | Ticket id, no of seats, ticket status, booking transaction id. | |

## 3.2 Customer

The customer as a user is defined to perform below listed operations after successful login.

| ID | Object | Operations | Data to include | |
|---|---|---|---|---|
| | Sign up | Add<br>View<br>Delete<br>Modify | Username, email, phone number, password. | |
| | Login | Add<br>View<br>Delete<br>Modify | Username, password | |
| | Theatres | View<br>Delete<br>Modify | Manager contact, manager name, theatre city, theatre name. | |
| | Movie | View<br>Delete<br>Modify | Movie id, movie date, movie description, movie genre, language, name, ratings. | |
| | Shows | Add<br>View<br>Delete<br>Modify | Show date, show end time, show name, show start time. | |

**FUNCTIONAL SPECIFICATION**

## 3.3 Login | Logout

[Web Application – Spring , Hibernate, Angular framework]

- Go to Registration screen when you click on Register link.

- Go to Success screen when you login successfully after entering valid username & password fetched from the database.

- Redirect back to same login screen if username & password are not matching.

- Implement Session tracking for all logged in users before allowing access to application features. Anonymous users should be checked, unless explicitly mentioned.

# 4 Data Organization

This section explains the data storage requirements of the Product Order Entry System and indicative data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

## 4.1 Table: User

The user specific details such as username, email, role etc. Authentication, and authorization / privileges should be kept in one or more tables, as necessary and applicable.

| Field Name | Descriptions |
|---|---|
| UserID | UserID is auto generated after registration and it is used as loginID. Here userID will be primary key |
| password | User Password |
| role | Role of the user customer or admin |
| username | Username of the customer |
| Customer_ID | CustomerID used as foreign key from user register as customer. |

## Table: Customer

This table contains information related to customer details.

| Field Name | Description |
|---|---|
| Customer_ID | Customer ID is auto generated and here customer_ID is a primary key. |
| Address | User address |
| Customer_name | Username of the Customer |
| email | Customer Email Id |
| mobile_number | 10-digit contact number of users |
| Password | User Password |

## Table: booking

This table contains information related to booking details.

| Field Name | Descriptions |
|---|---|
| transaction_ID | Transaction id is auto generated and here used as a primary key. |
| booking_date | Date of movie ticket boking |
| total_cost | Cost of ticket |
| transaction_mode | Mode of payment |
| transaction_status | Status of payment |
| customer_customer_ID | CustomerID used as foreign key from user register as a customer. |

## Table: Ticket

This table contains information related to ticket details.

| Field Name | Descriptions |
|---|---|
| ticket_ID | ticketid is auto generated after booking ticket and here ticketid is a primary key. |
| no of seats | Number of seats to be booking. |
| Ticket_status | Status of ticket. |
| booking_transaction_ID | TransactionID used as foreign key from the booking. |

## Table: seat

This table contains information related to seat.

| Field Name | Descriptions |
|---|---|
| seat_id | Seatid is primary key |
| price | Price of seat |
| seat_number | Enter the seat number |
| status | Check status of seat empty or full |
| type | Type of the seat we needed |
| ticket_ticket_id | Ticket id used as foreign key it is MUL its repeatedly used |

**Table: show**

This table contains information related to show details.

| Field Name | Descriptions |
|---|---|
| show_id | Seattid is auto generated and here ticketid is a primary key. |
| show_date | Enter show date |
| show_end_time | Show ending time |
| show_name | Show name like morning show evening show like that |
| show_start_time | Show starting time |
| booking_transaction_id | It is a MUL . |
| screen_screen_id | It is a MUL . |
| theatre_theatre_id | It is a MUL . |

**Table: movie**

This table contains information related to movie details.

| Field Name | Descriptions |
|---|---|
| movie_id | Movieid is auto generated and here movieid is a primary key |
| movie_date | Movie date |
| movie_description | About movie |
| movie_genere | Genre of movies |
| movie_hours | Number of hours |
| movie_language | Movie language |
| movie_name | Name of the movie |
| movie_rating | Ratings of the movie |
| show_show_id | It is a MUL . |

## Table: theatre

This table contains information related to theatre details.

| Field Name | Descriptions |
|---|---|
| theatre_id | Theatreid is auto generated and here movieid is a primary key |
| manager_contact | Manager address |
| manager_name | Manager name |
| theatre_city | City  of the theatre |
| theatre_name | Name of the theatre |

## Table: screen

| Field Name | Descriptions |
|---|---|
| screen_id | screeid is auto generated and here movieid is a primary key |
| columnss | Number of columns in a screen |
| rowss | Number of rows in a screen |
| screen_name | Name of the screen first, second like that. |
| theatre_theatre_id | Theatreid is a foreign key |

## 5 REST APIs to be Built

Create following REST resources which are required in the application,

**1. Creating User Entity:** Create Spring Boot with Micro services Application with Spring Data JPA Technology stack:

- Spring Boot
- Spring REST
- Spring Data JPA

Here will have multiple layers into the application:

1. Create an Entity: User

2. Create a UserRepository interface and will make use of Spring Data JPA

a) Will have findByUserName method.

b) Add the User details

3. Create a UserService class and will expose all these services.

4. Finally, create a UserRestController will have the following Uri's:

| URI | METHODS | Description | Format |
|---|---|---|---|
| /user/adduser | POST | Add the user details | JSON |
| /user/removeuser | DELETE | Delete user | JSON |

### 2. Creating Customer Entity:

Build a RESTful resource for Customer manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: Customer

6. Create a CustomerRepository interface and will make use of Spring Data JPA

a. Add the Customer details.
b. Update Customer details.
c. Delete Customer details.

7. Create a Customer Service class and will expose all these services.

8. Finally, create a CustomerRestController will have the following Uri's:

| URI | METHODS | Description | Format |
|---|---|---|---|
| Customer/add | POST | Add the customer details | JSON |
| Customer/update | PUT | Update customer details | JSON |
| Customer/remove | DELETE | Delete customer | JSON |

## 3. Creating theatre Entity:

Build a RESTful resource for theatre manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: theatre

6. Create a CustomerRepository interface and will make use of Spring Data JPA

   a. Add the theatre details.
   b. Update theatre details.
   c. Delete theatre details.
   d. Get theatre details .

7. Create a theatre Service class and will expose all these services.

8. Finally, create a theatreRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| theatre/insert | POST | Add theatre details | JSON |
| theatre/update | PUT | Update theatre details | JSON |
| theatre/all | GET | View the theatre details | JSON |
| theatre/remove | DELETE | Delete theatre | JSON |

**4. Creating show Entity:**

Build a RESTful resource for show manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: show

6. Create a CustomerRepository interface and will make use of Spring Data JPA

   a. Add the show details.
   b. Update show details.
   c. Delete show by show id details.
   d. View  show details by using showid .

7. Create a showService class and will expose all these services.

8. Finally, create a showRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| show/add | POST | Add show details | JSON |
| show/update | PUT | Update show details | JSON |
| show/all | GET | View the show details | JSON |
| show/remove | DELETE | Delete shows. | JSON |
| showtheatre/theatre_id | GET | View list of shows with theatreid | JSON |

**5. Creating movie Entity:**

Build a RESTful resource for theatre manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: theatre

6. Create a adminRepository interface and will make use of Spring Data JPA

   a. Add the Movie details.
   b. Update Movie details.
   c. Delete Movie details.
   d. View Movie details by thatreid
   e. View movies with date.

7. Create a Movie Service class and will expose all these services.

8. Finally, create a movieRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| movie/add | POST | Add movies details | JSON |
| movie /map | PUT | Update movie details | JSON |
| movie /findall | GET | View Movie details | JSON |
| movie/remove | DELETE | Delete Movies | JSON |
| viewmovie/movieid | GET | Movie with movieid fetched | JSON |

**6. Creating admin Entity:**

Build a RESTful resource for admin manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: admin

6. Create a AdminRepository interface and will make use of Spring Data JPA

   a. Add admin details

7. Create a adminService class and will expose all these services.

8. Finally, create a adminRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| /admin | POST | Add admin details (admin created) | JSON |

**7. Creating booking Entity:**

Build a RESTful resource for booking manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: booking

6. Create a bookingRepository interface and will make use of Spring Data JPA

   a. Add Customer details and book a ticket.
   b. Book ticket by movie id.

c. View total cost of booking.
d. View booking details.
e. Update booking details.
f. Delete booking.

7. Create bookingService class and will expose all these services.

8. Finally, create bookingRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| insert/ | POST | Booking for customer with customer_id | JSON |
| findall/ | GET | Booking list is fetched | JSON |
| ticket/bookingid | DELETE | Booking deleted successfully | JSON |
| /cost/bookingid | GET | Total cost of the booking is displayed | JSON |
| /update | PUT | Update the booking details | JSON |
| /Bydate/date | POST | Booking with date | JSON |
| /Bymovie/movieid | POST | Booking with movieid | JSON |
| /viewbooking/ | GET | Screen is visible with booking details | JSON |

## 8. Creating seat Entity:

Build a RESTful resource for seat manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: seat

6. Create a s seat Repository interface and will make use of Spring Data JPA

a. Add seat details and book a ticket.
b. View list of seats.
c. Update seat details.
d. Delete the seat.
e. Block the seat.

7. Create screen Service class and will expose all these services.

8. Finally, create screenRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| add/ | POST | Add the seat details | JSON |
| findall/ | GET | List of seat is fetched | JSON |
| update/ | PUT | Seat is updated | JSON |
| book/ | POST | Book a seat | JSON |
| cancel/ | DELETE | Seat is cancelled | JSON |

**9. Creating screen Entity:**

Build a RESTful resource for screen manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: screen

6. Create a screenRepository interface and will make use of Spring Data JPA

   a. Add screen to the theatre.
   b. Update screen details.
   c. View list of screen details.

7. Create screenService class and will expose all these services.

8. Finally, create screenRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| add/ | POST | Screen added successfully to the theatre | JSON |
| findall/ | GET | List of screen is fetched | JSON |
| viwscreen/screenid | GET | Screen is found | JSON |
| update/ | PUT | Screen updated | JSON |

## 10. Creating ticket Entity:

Build a RESTful resource for ticket manipulations, where following operations to be carried out. Here will have multiple layers into the application:

5. Create an Entity: ticket

6. Create a ticketRepository interface and will make use of Spring Data JPA

   a. Add a ticket.
   b. View list of ticket
   c. View ticket with ticket id.

7. Create ticketService class and will expose all these services.

8. Finally, create ticketRestController will have the following Uri's:

| URI | METHOD | Description | Format |
|---|---|---|---|
| Add/ | POST | Ticket added successfully | JSON |
| /findall | GET | List of ticket fetched | JSON |
| /ticketid | GET | Ticket with ticket id fetched | JSON |

## 6 .ASSUMPTIONS

- ·User Interface: The type of client interface (front-end) to be supported - Angular based
- The administrator can add and remove Movie, theatre, shows into the database.
- You must not allow user to book same movie ticket twice.
- When you add product into cart the No. of Products selected will be incremented.
- If you cancel the booking ticket, the ticket booking list will be decremented.
- The total amount will be calculated based on the movie, accordingly, change the booking counter & total amount.

## 7 .General Expectations

- Participants must create the Class Diagram, Sequence Diagram and ER Diagram.
- Participants must do Unit testing and Functional Testing using POSTMAN tool.
- Integration of Angular and Spring Boot with Microservices should be done, referring Project 2 -Frond End Development Project.
- The server should be a concurrent server servicing multiple clients.
- Database can be implemented using Mysql or above.
- To begin with, the application should support at least 1 admin and 2 customers.
- Compilation and Build should be done using Eclipse IDE or STS
- Source-code and all documents must be maintained (checked-in) in configuration management system (subversion)
- Coding standards (for Java) should be followed.

## 8 Acceptance Criteria

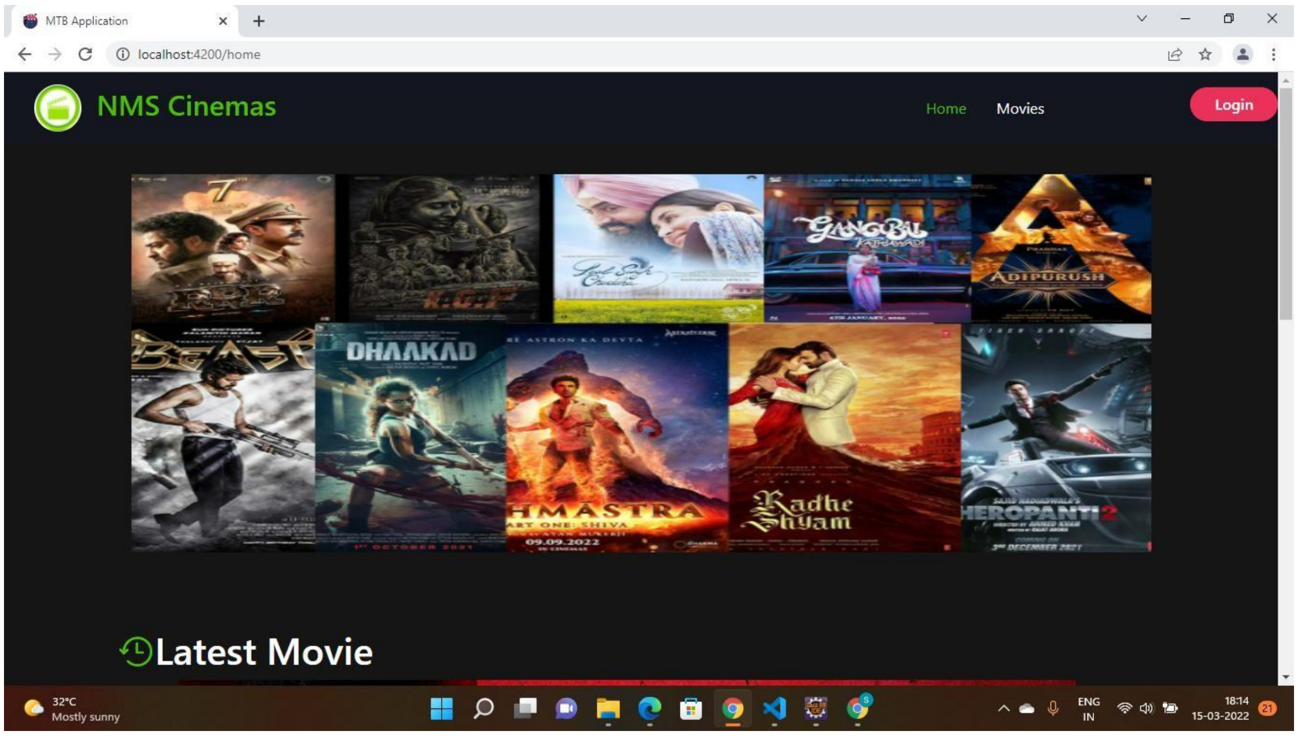
All P1 requirements must be mandatorily implemented.

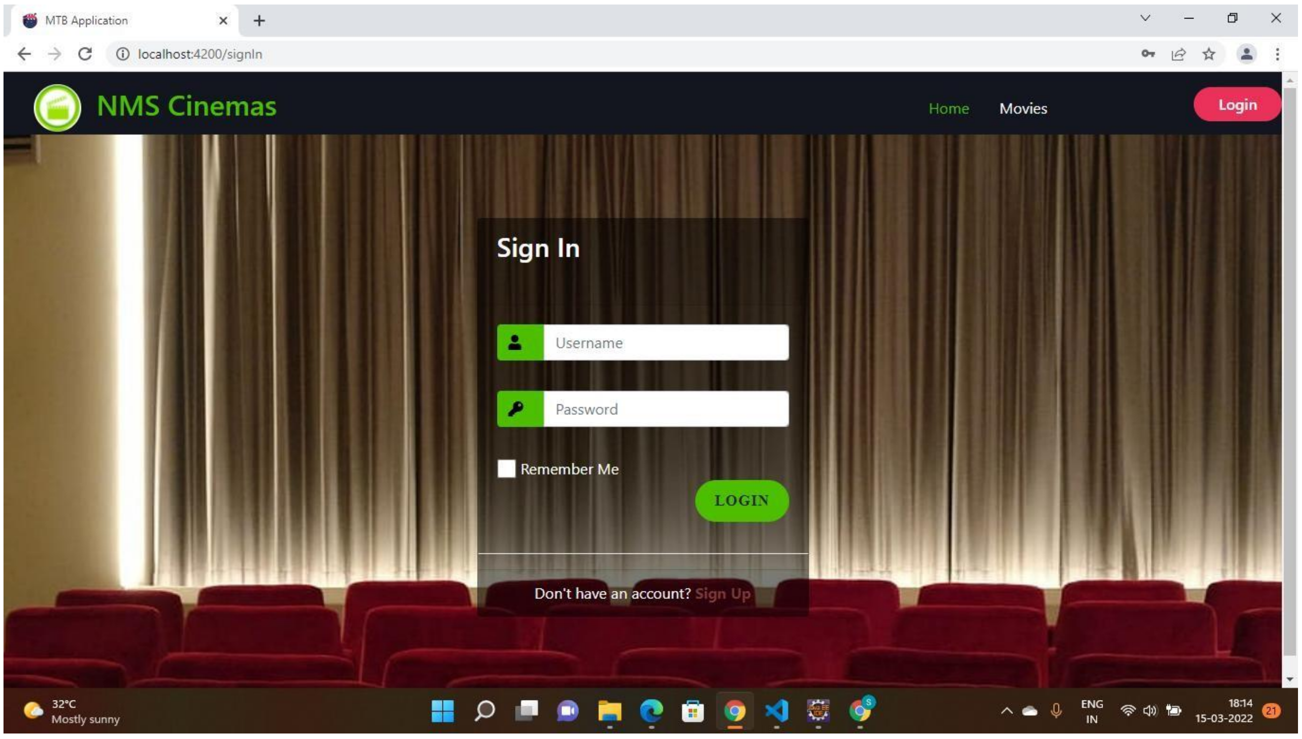
## 9 Traceability to Requirements

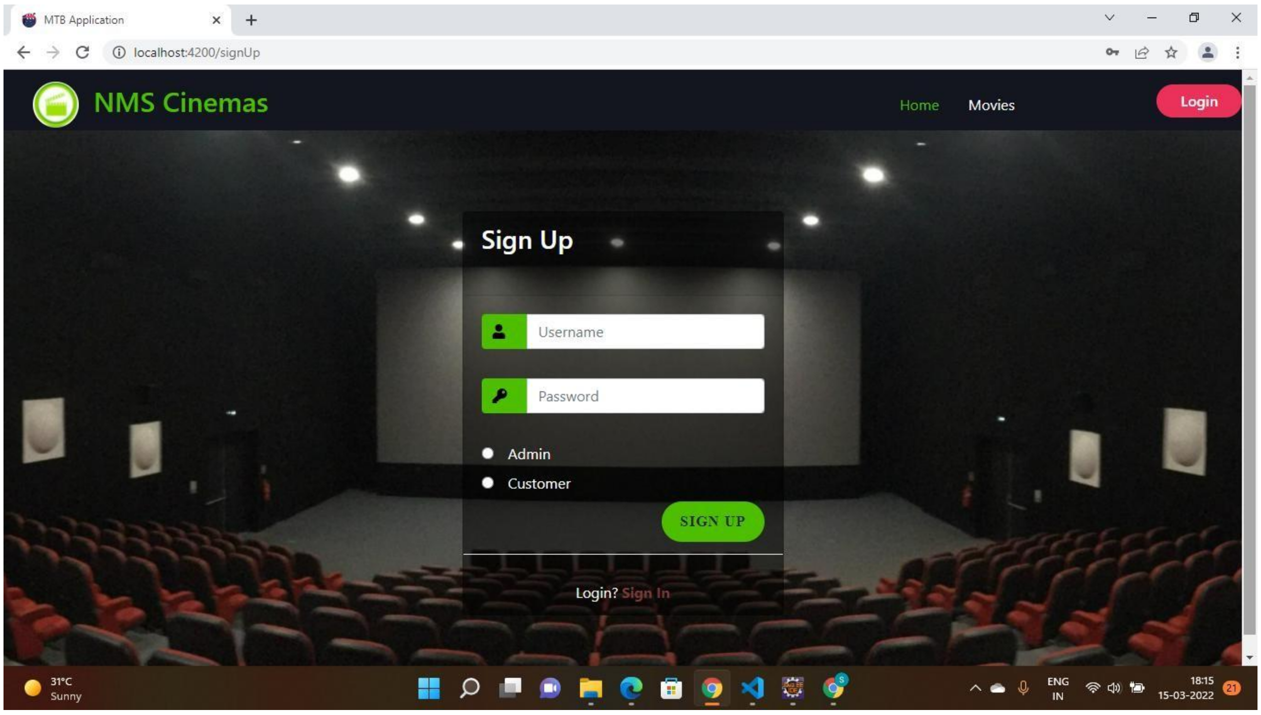
Appropriate requirements from RS and FS are mapped here.

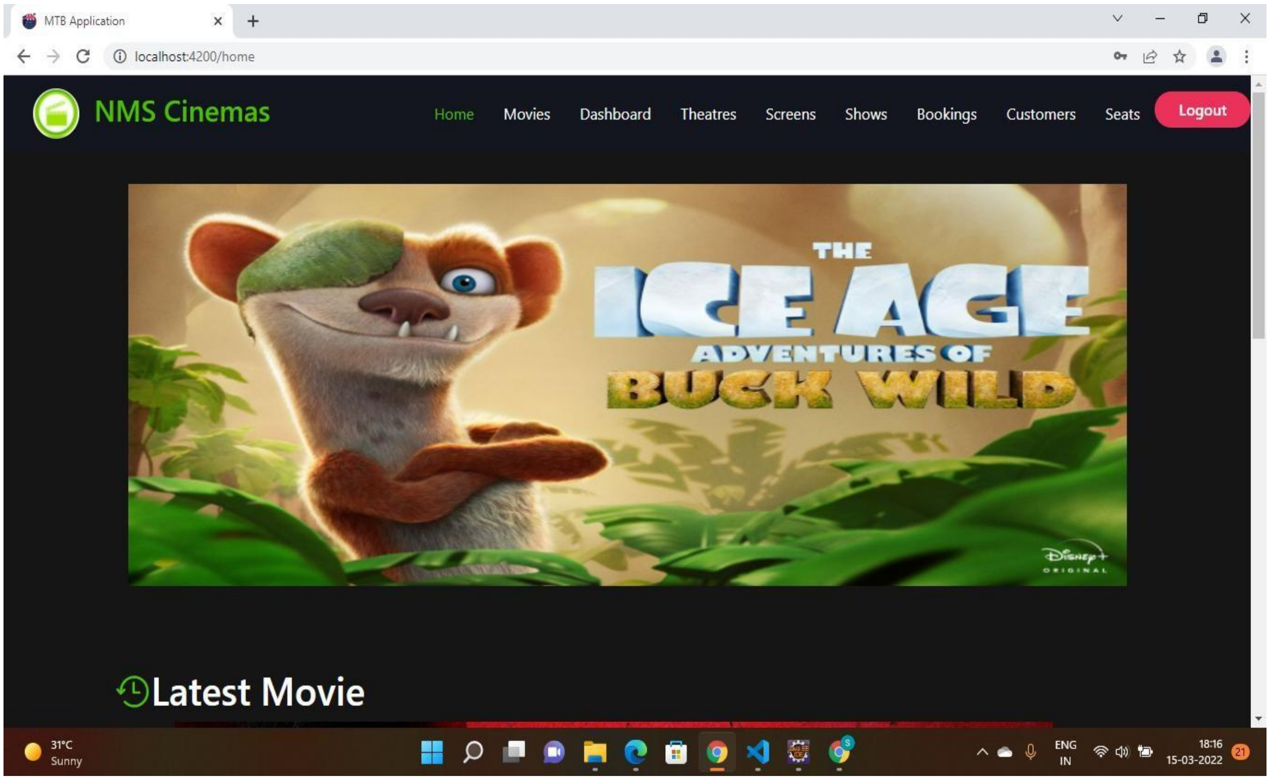
| Document Reference ID & Description: (Doc ID from which this document is derived) | | |
|---|---|---|
| Sl. No. | Reference document: RS Requirement/Feature (Section ID/Name) | Current document: FS Location (Section ID/Name) |
| 1 | | |
| 2 | | |

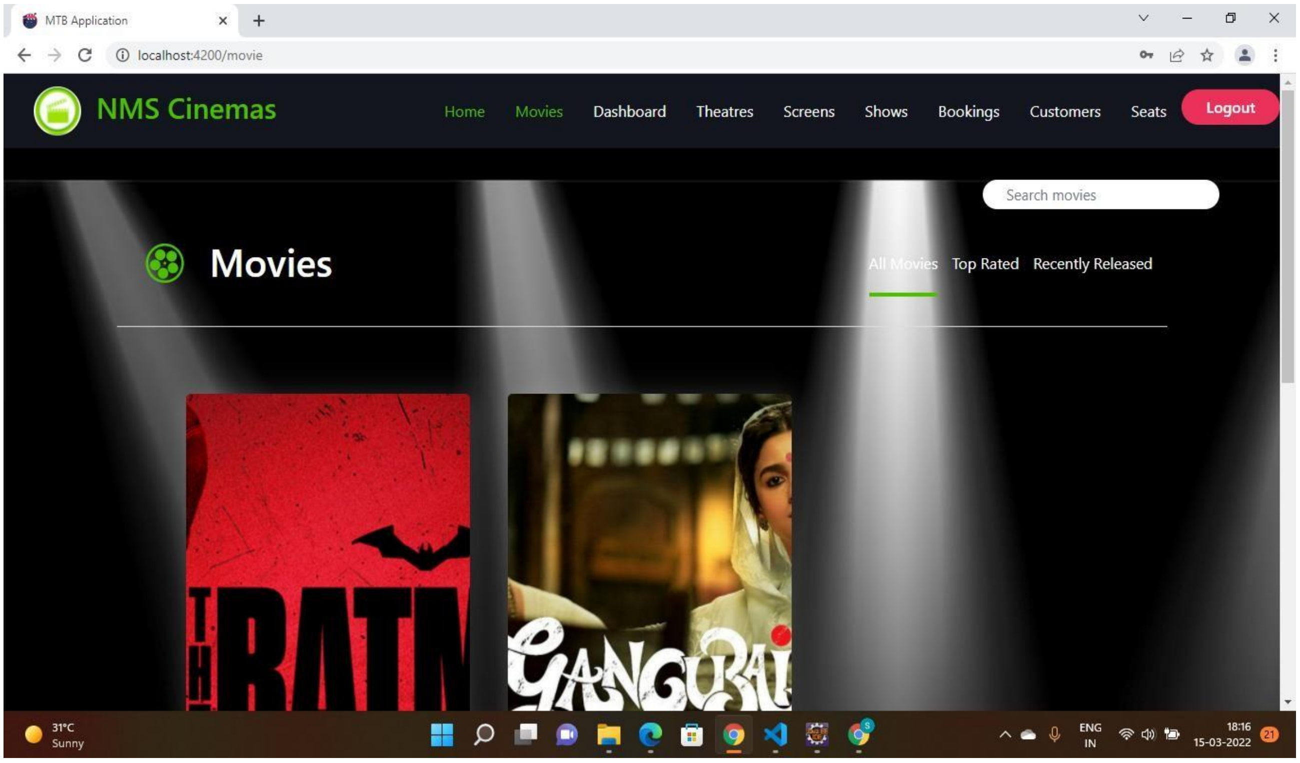
## 10. Output Screenshots for the Project

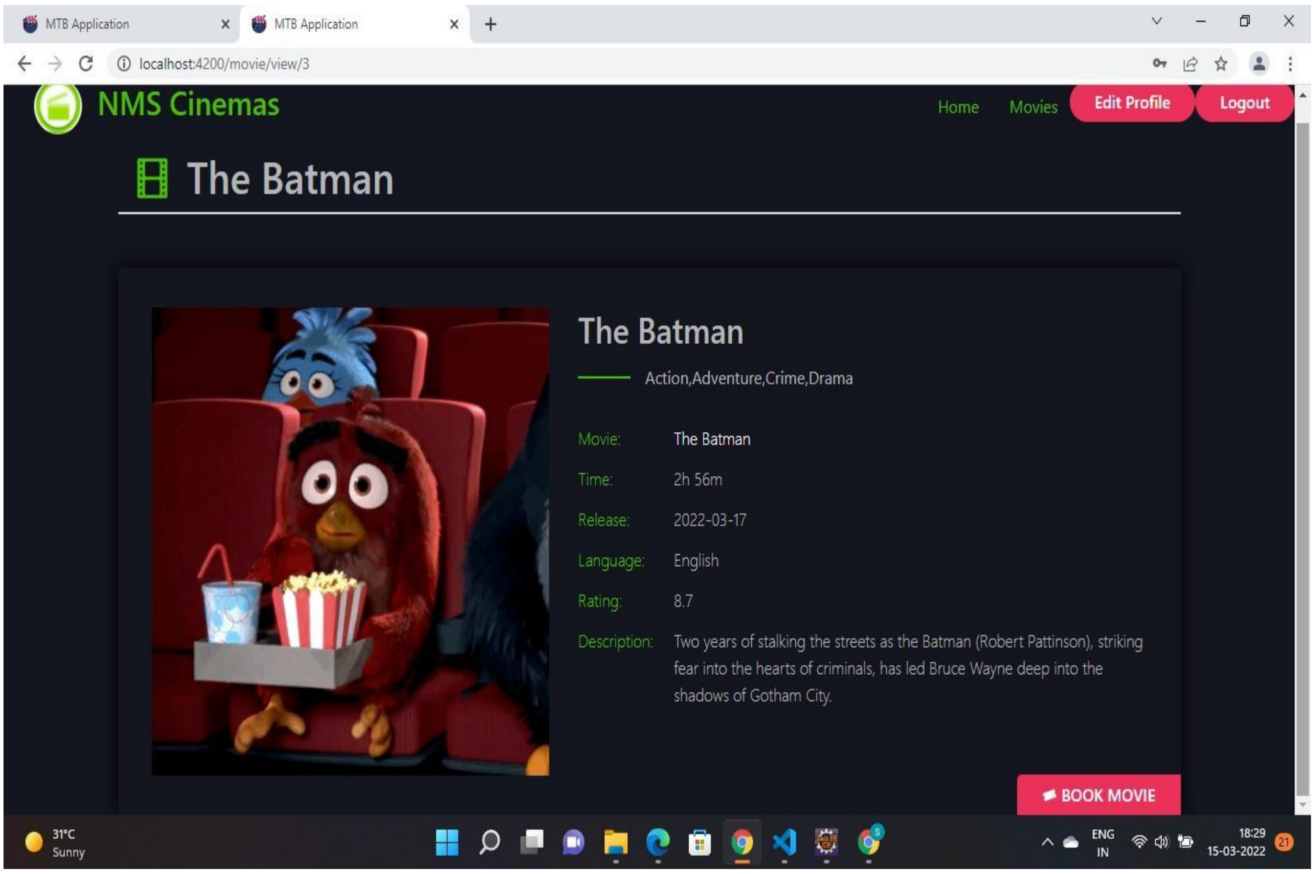





MTB Application
localhost:4200/signUp
NMS Cinemas
Home
Movies
Login
Sign Up
Username
Password
Admin
Customer
SIGN UP
Login? Sign In


MTB Application
localhost:4200/home
NMS Cinemas
Home
Movies
Dashboard
Theatres
Screens
Shows
Bookings
Customers
Seats
Logout
THE ICE AGE ADVENTURES OF BUCK WILD
Latest Movie



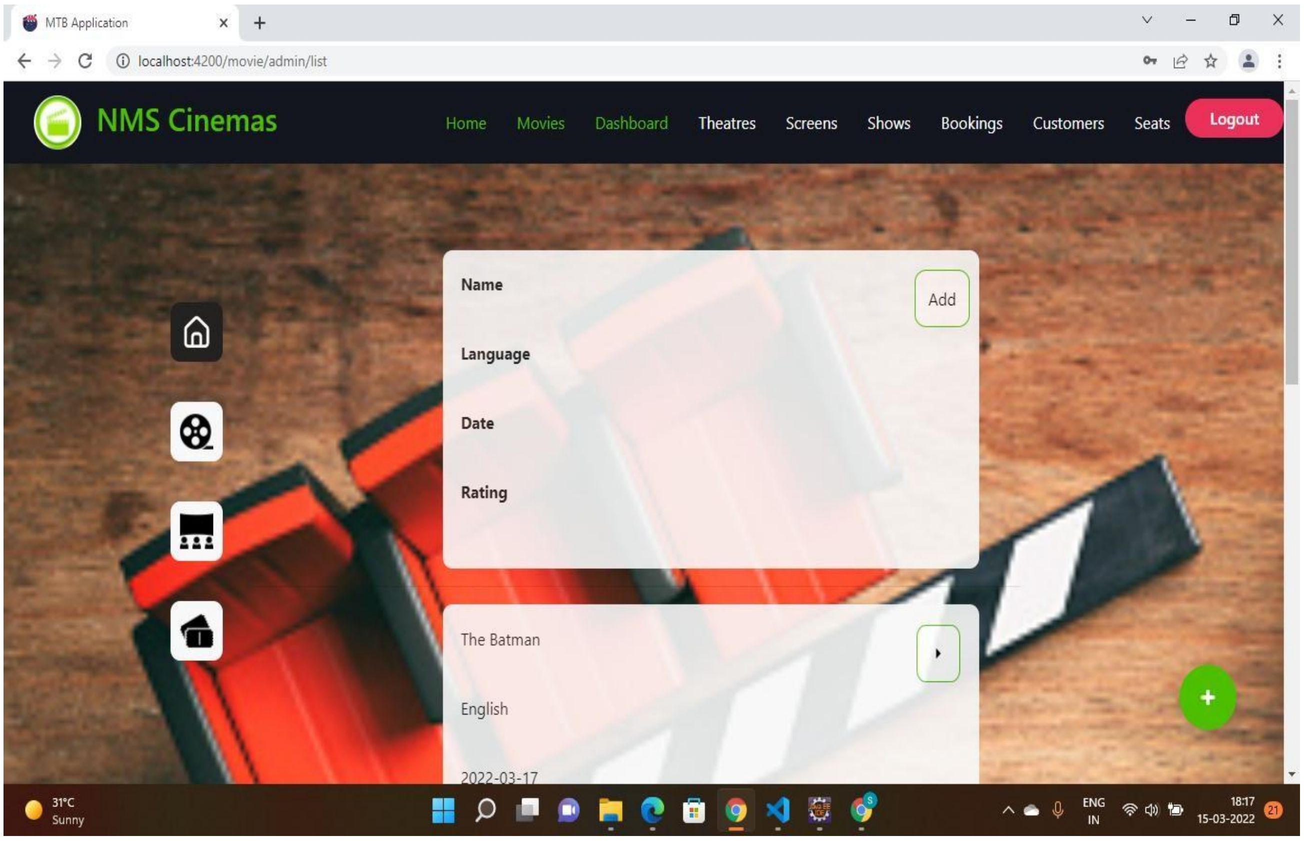


The Batman

The Batman

Action,Adventure,Crime,Drama

| | |
|---|---|
| Movie: | The Batman |
| Time: | 2h 56m |
| Release: | 2022-03-17 |
| Language: | English |
| Rating: | 8.7 |
| Description: | Two years of stalking the streets as the Batman (Robert Pattinson), striking fear into the hearts of criminals, has led Bruce Wayne deep into the shadows of Gotham City. |



Name

Language

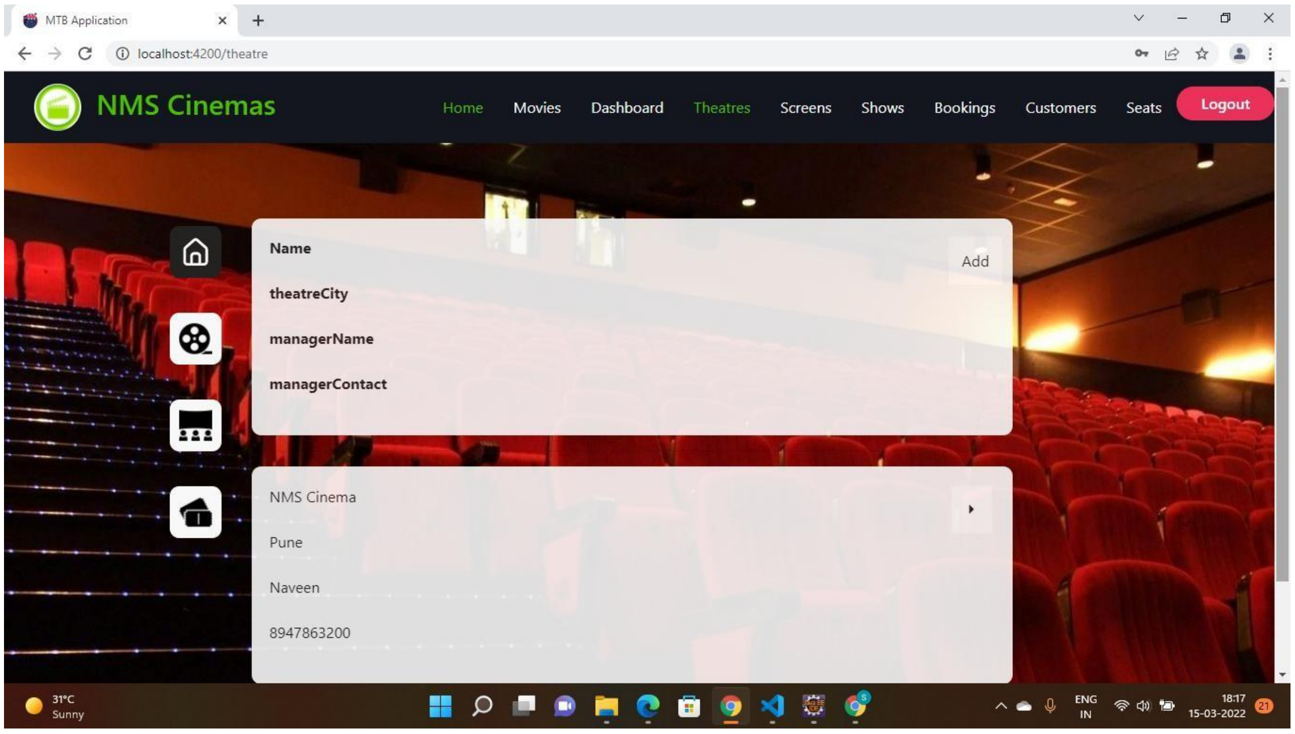
Date

Rating

The Batman

English

2022-03-17



Name

theatreCity

managerName

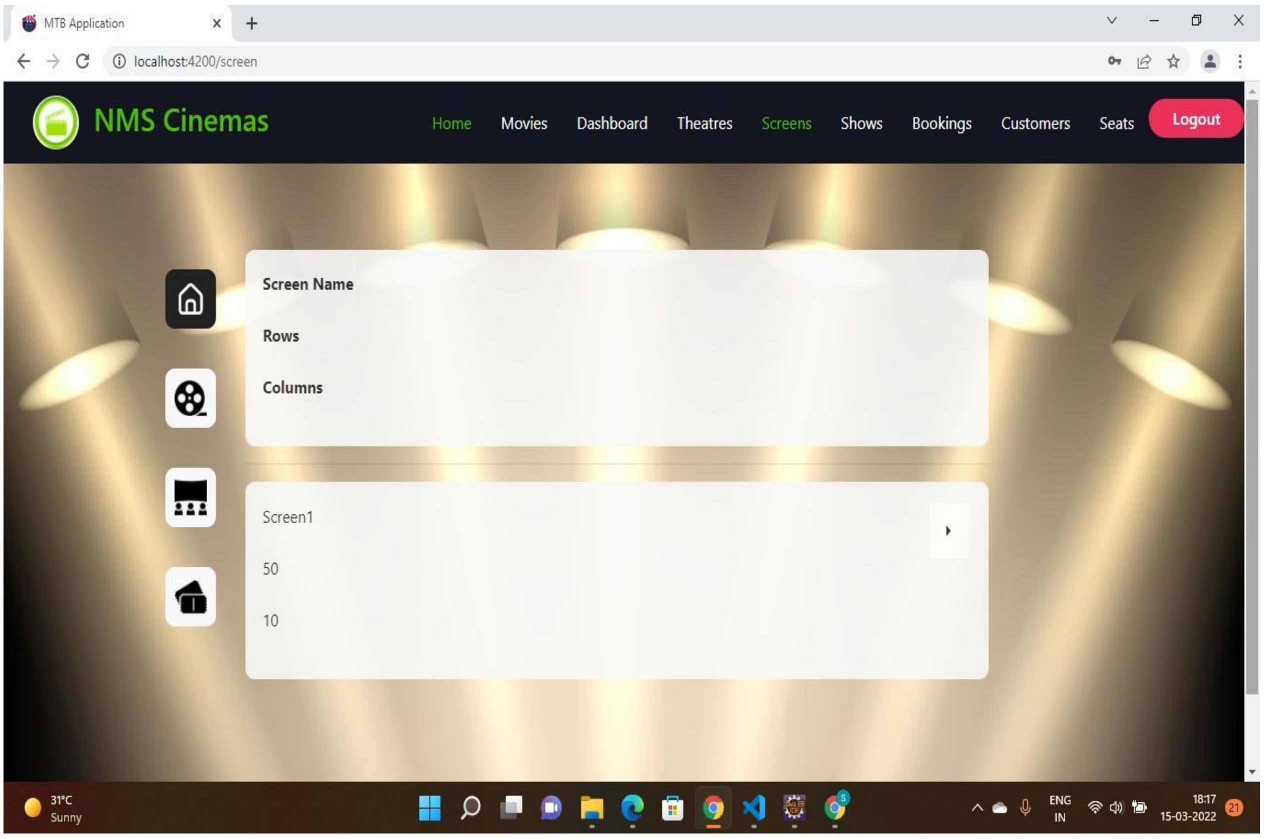
managerContact

NMS Cinema

Pune

Naveen

8947863200



Screen Name

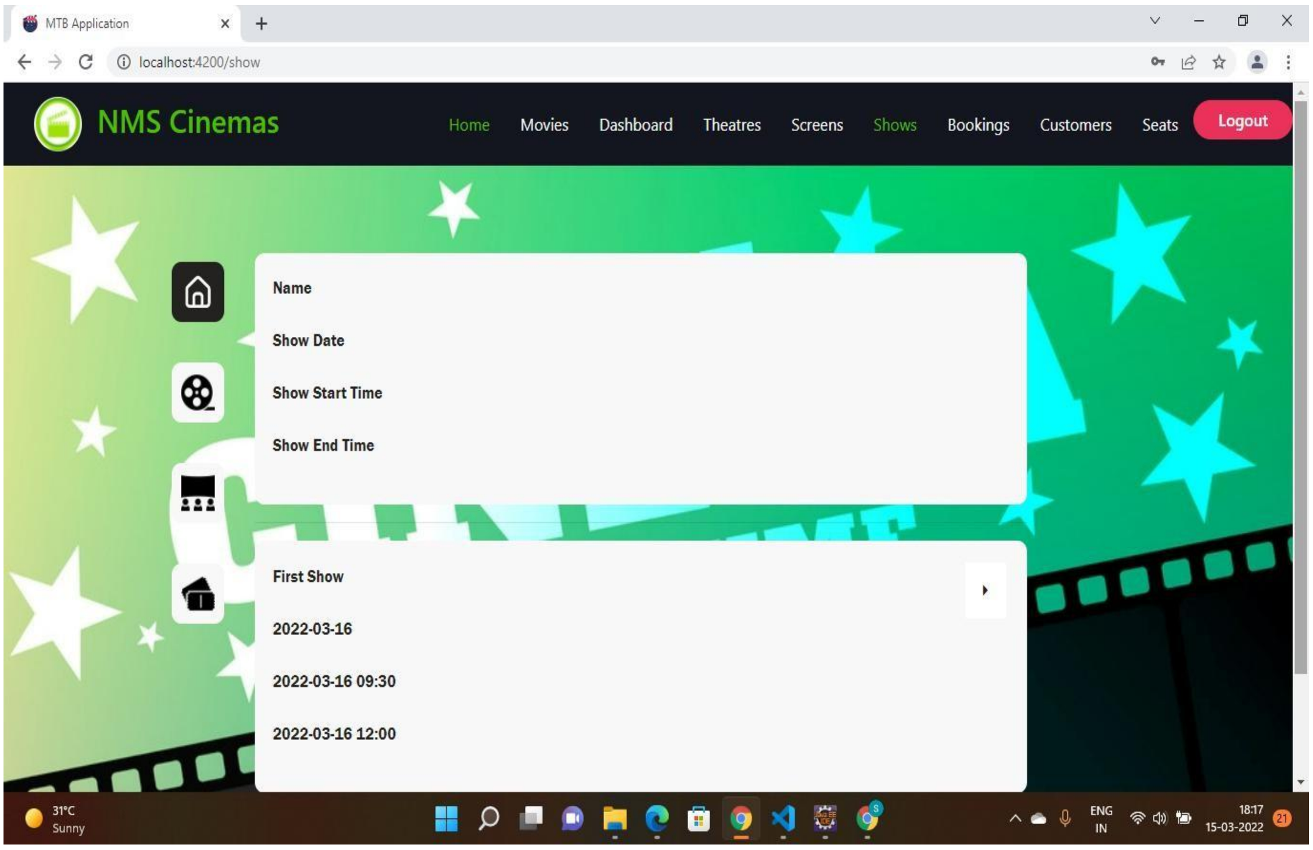
Rows

Columns

Screen1

50

10



Name

Show Date

Show Start Time

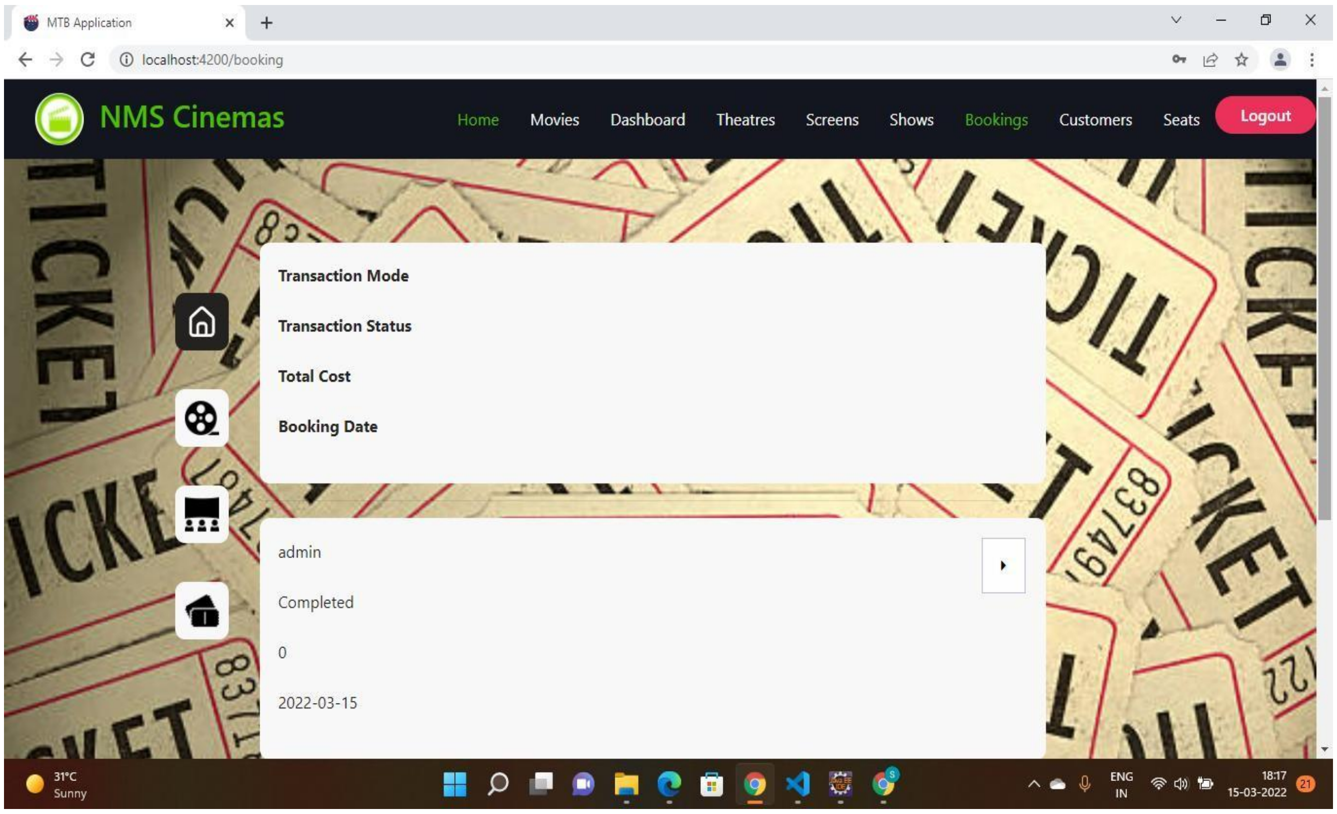
Show End Time

First Show

2022-03-16

2022-03-16 09:30

2022-03-16 12:00



**Transaction Mode**

**Transaction Status**

**Total Cost**

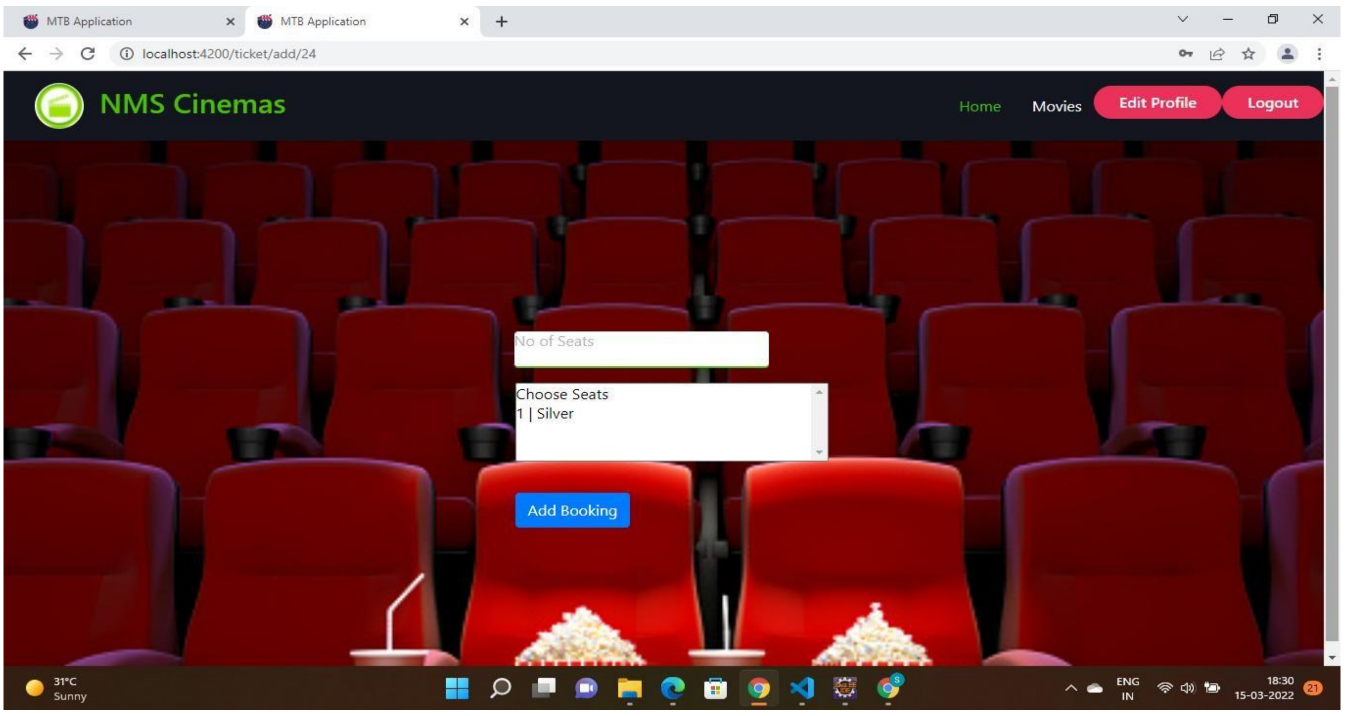
**Booking Date**

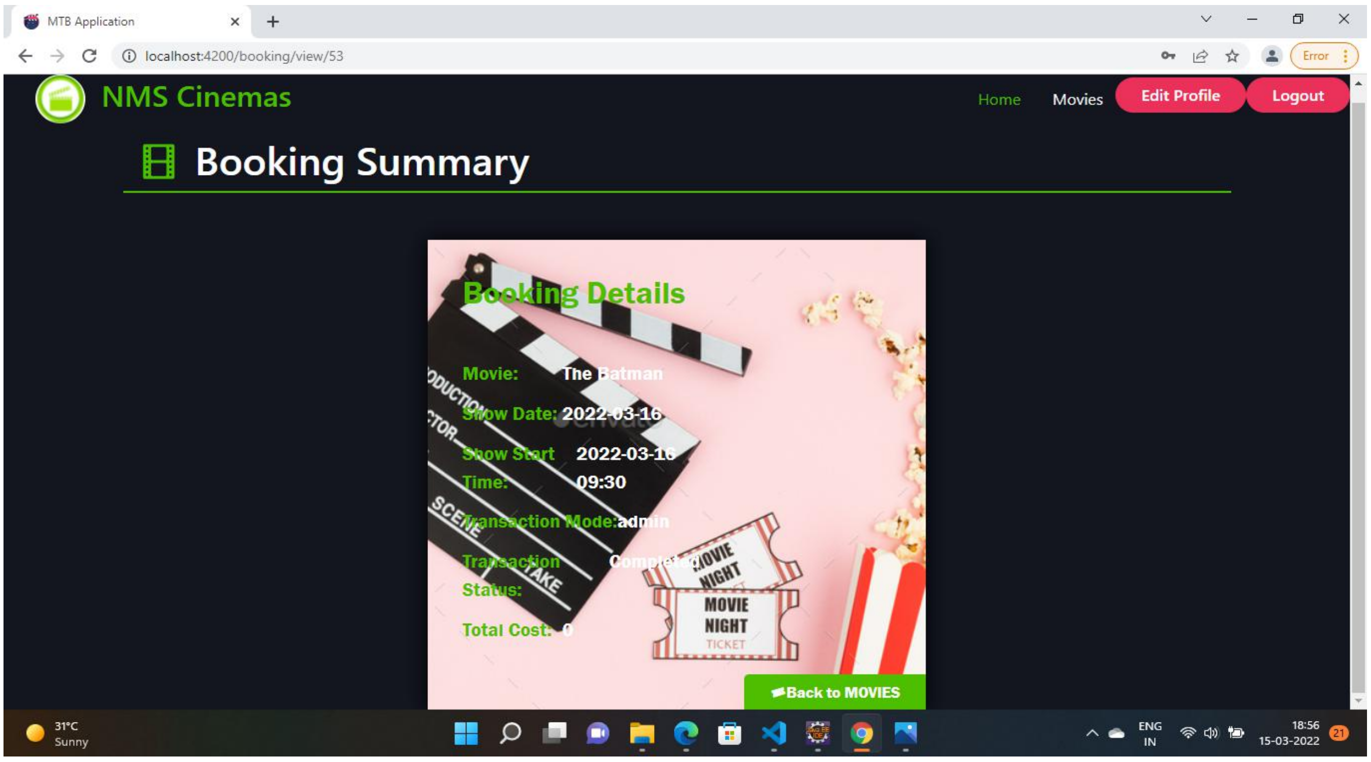
admin

Completed

0

2022-03-15





Movie: The Batman

Show Date: 2022-03-16

Show Start Time: 2022-03-16 09:30

Transaction Mode: admin

Transaction Status: Completed

Total Cost: 0