# 

# 

**GOVERNORS STATE UNIVERSITY**

**CPSC-8845-01\_22SU: Advanced Database Concepts**

**PROJECT**

# **MOVIE TICKET RESERVATION SYSTEM**

# 

# 

# 

# 

Student Name: Sanjay Boga

Student ID: 1241032

Instructor: Yili Tseng

**Introduction**

During last era, motion pictures became very widely known. It demonstrates our cultures, and innumerable movies are launched every year by large international companies such as Netflix, Amazon, HBO, and others. Earlier, the one and only way to watch a film was in a "cinema," and humans became obsessed to even bluescreen. Theses days, thanks to the world of technological advancements, it becomes much simpler to watch a film.

Having followed the tremendous feedback from moviegoers over the last few decades, a new industry, movie ticket booking, has emerged as a subset of the same industry. With the expansion of cinema across the globe, distributing and purchasing tickets has become one of the most difficult tasks. To address this issue, a movie ticket reservation system has been implemented. This allows moviegoers to purchase movie tickets in a matter of minutes from anywhere in the world. The objective of movie ticket reservation system is provide moviegoers with a service that allows them to reserve seats in movie theaters anywhere and at any moment. Movie information is publicly available. The customer can quickly learn about the new cinema and then decide.

It is a mechanism that is accessible through the internet. Moviegoers  can purchase tickets online and cancel their seats at a later date. To improve the reimbursement feature, all consumers must sign up to become a member before purchasing a ticket.

**Creating DataBase**

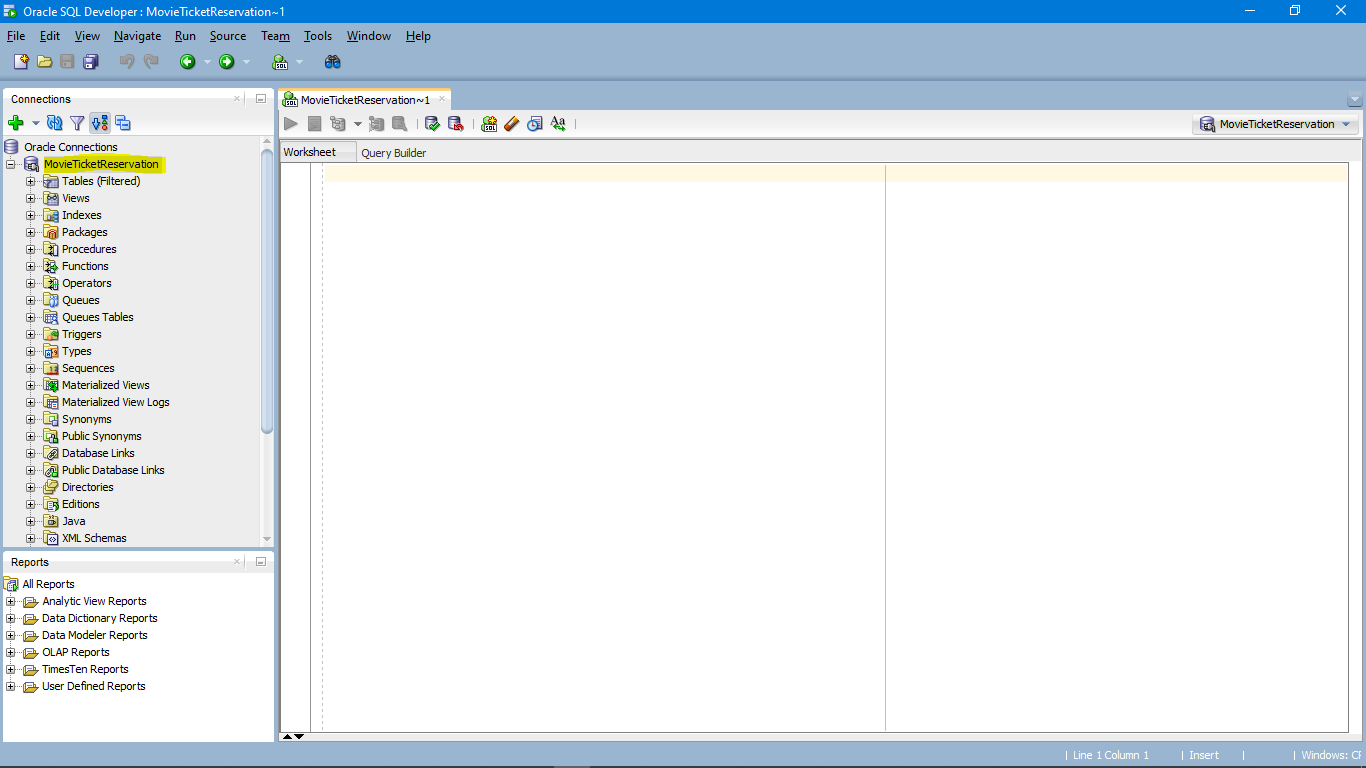


Figure 1: Database Creating & Design

**UML Diagram**

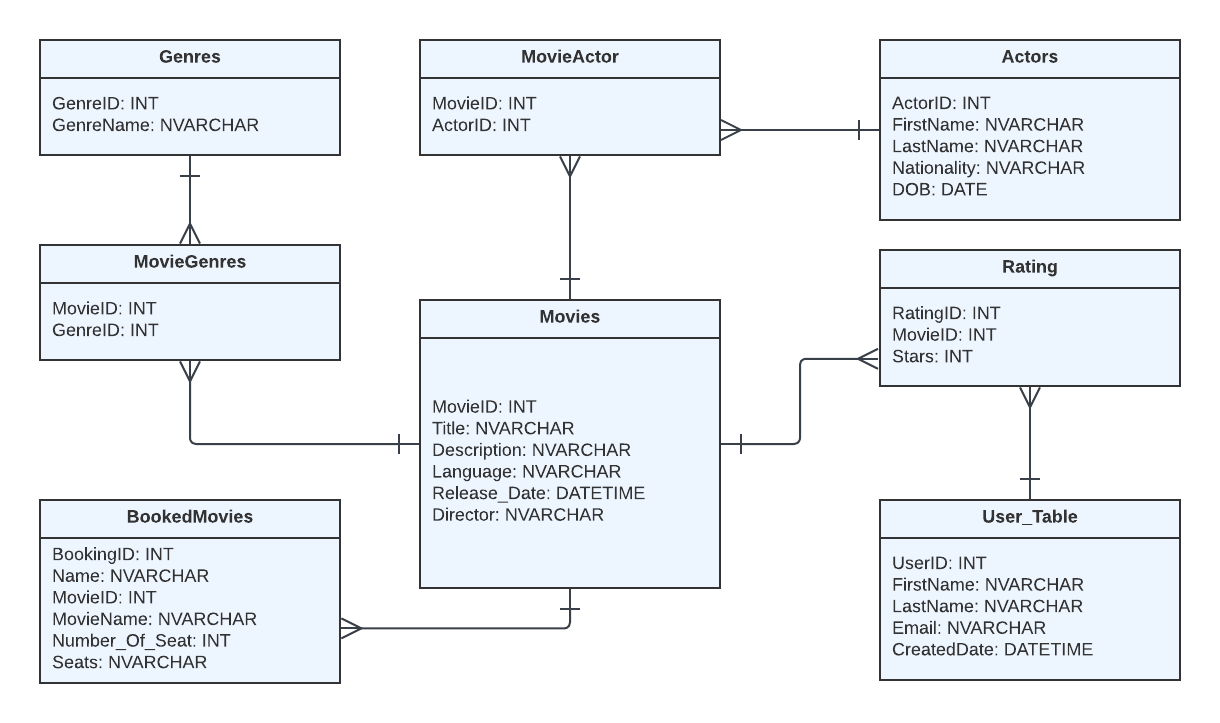


Figure 2: UML Diagram of Movie Ticket Reservation

**Above UML Diagram Consist of following Relationship and attributes**

|  |  |
| --- | --- |
| **Table** | **Relationship** |
| Genres 🡪 MovieGenres | One-to-Many |
| Movies 🡪 MovieGenres | One-to-Many |
| Movies 🡪 Ratings | One-to-Many |
| Movies 🡪 MovieActor | One-to-Many |
| Actors 🡪 MovieActor | One-to-Many |
| User 🡪 Ratings | One-to-Many |
| Movies 🡪 BookedMovies | One-to-Many |

**Entities & Attributes**

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Columns** | **Description** |
| **Genres** | GenreID , GenreName | This table uses to store information related to category of movies such as action, horror or crime |
| **Movies** | MovieID, Title, Description,  Language, ReleaseDate, Director | Movies table stores the basic but essential information related to specific movie such as movie name, who is the director, when did it got released and many more. |
| **Actor** | ActorID, FirstName, LastName,  Nationality, DOB | This table stores the actors details who has worked as a primary lead actor in the respective movie which is linked by another table called MovieActor. |
| **Ratings** | RatingID, MovieID, Stars | This table keep holds of number of stars/rating received by IMDB or users. |
| **MovieGenres** | MovieID, GenreID | This schema shows the link between two table i.e. movie and genres and link both of them using foreign key to fetch the category of gerne movie belongs to. |
| **MovieActor** | MovieID, ActorID | MovieActor is a relational table between actor and movie which link up using foreign key to fetch the lead actor working in respective films. |
| **User** | UserID, FirstName, LastName, Email, Created\_Date | This is the user details table which is specific to our application. Talkies allows the user to first register in our portal and then allow to book the movie tickets. |
| **BookedMovies** | BookingID, Name, MovieID, MovieName, Number\_of\_Seat, Seats | This table is used to stored the booking details when user book any movies. |

# **Normalization**

Normalization is the prcess of reducing redundancy and impove the efficiency of retriving the information from database.

Below is the un-normalized data, to reduce duplication of data in “MovieTicketReservation” Database we must have atlest 3NF to be implemented.

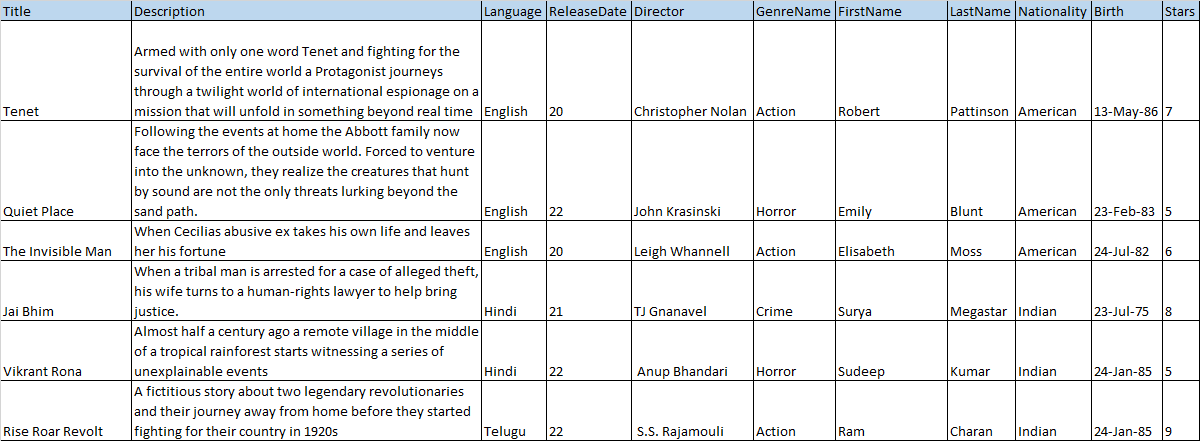


Figure 3: Un-normalized Data

## **Third Normal Form (3NF) – Transitively Functional Dependency**

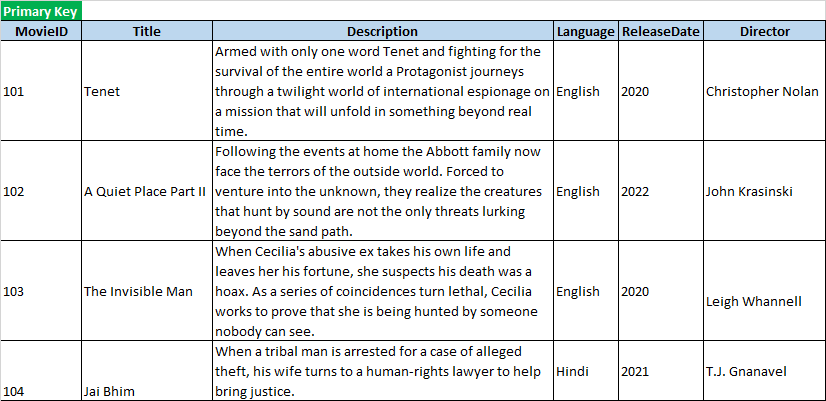


Table 1: Movies

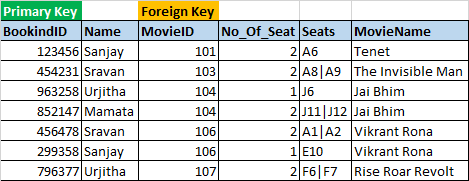


Table 2: BookedMovies

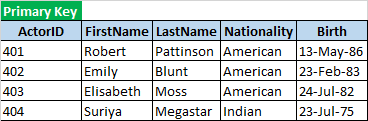


Table 3: Actors

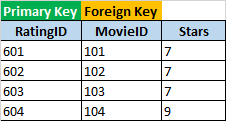
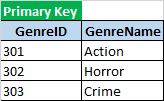
 

Table 4: Rating Table 5: Genres

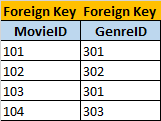
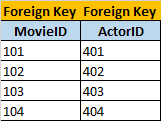
 

Table 6: MovieGenres Table 7: MovieActor

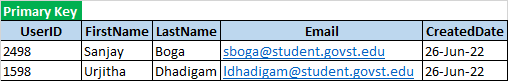
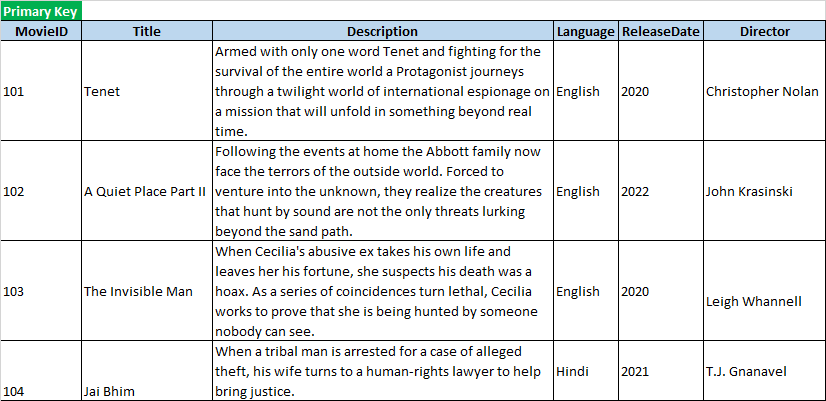


Table 8: User\_Table

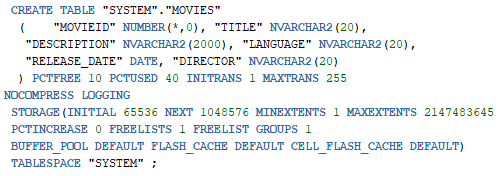
# **Implementing and Applying Constraints**

Constraints are used to define the regulations that guide the data in a table. Constraints are used to restrict the types of data that can be entered into a table.

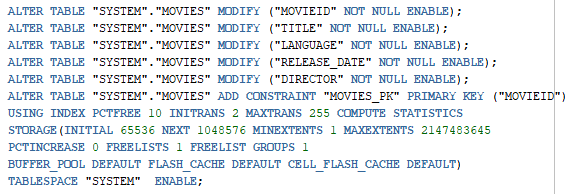
**Table 1: Movies**



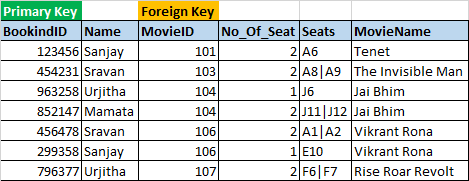
**DDL Command**



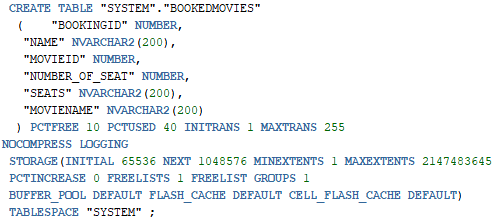
**Constraints**



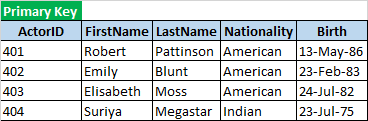
**Table 2: BOOKEDMOVIES**



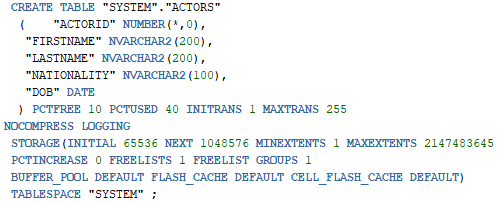
**DDL Command**



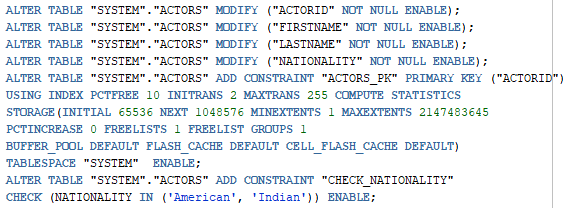
**Table 3: ACTORS**



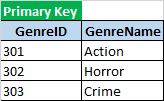
**DDL Command**



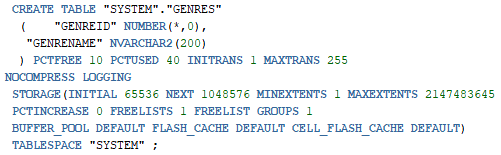
**Constraints**



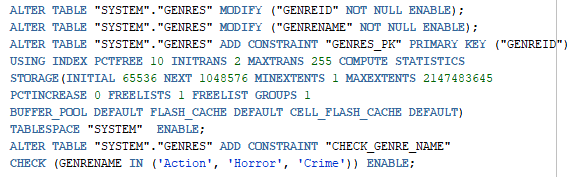
**Table 4: GENRES**



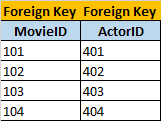
**DDL Command**



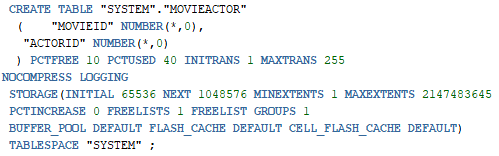
**Constraints**



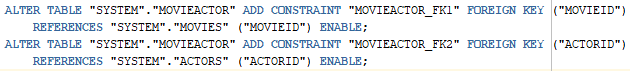
**Table 5: MOVIEACTOR**



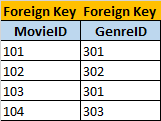
**DDL Command**



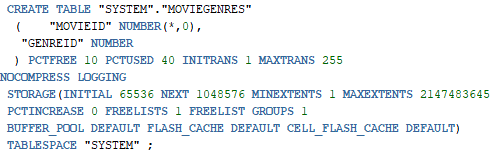
**Constraints**



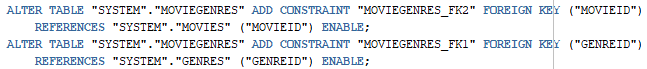
**Table 6: MOVIEGENRES**



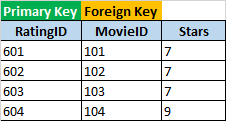
**DDL Command**



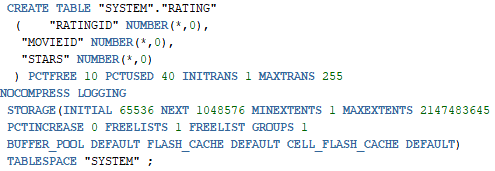
**Constraints**



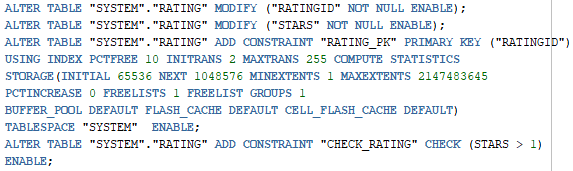
**Table 7: RATING**



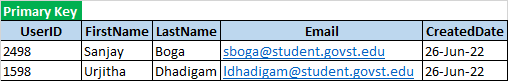
**DDL Command**



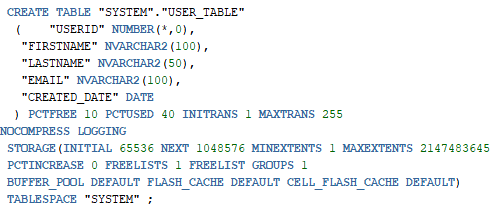
**Constraints**



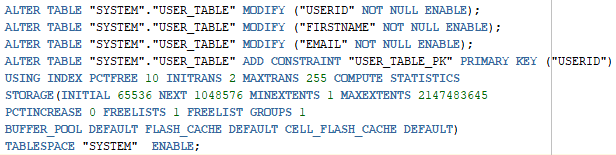
**Table 8: USER\_TABLE**



**DDL Command**



**Constraints**



**Implementation of key constraint, entity integrity constraint, and referential integrity:**

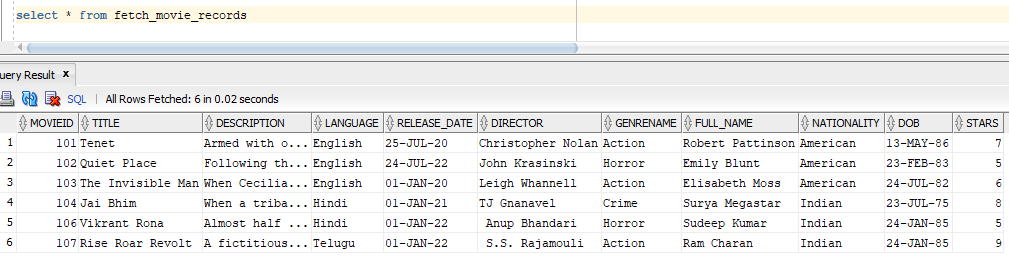
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Primary Key** | **Foreign Key** | **Check Constrains** | **Not Null** |
| ACTORS | ACTORID |  | NATIONALITY IN ('American', 'Indian') | ACTORID  FIRSTNAME  LASTNAME  NATIONALITY |
| MOVIEGENRES |  | MOVIEID  GENREID |  |  |
| USER\_TABLE | USERID |  |  | USERID  FIRSTNAME  EMAIL |
| BOOKEDMOVIES | BOOKINGID | MOVIEID |  |  |
| GENRES | GENREID |  | GENRENAME IN ('Action', 'Horror', 'Crime') | GENREID  GENRENAME |
| MOVIEACTOR |  | MOVIEID  ACTORID |  |  |
| MOVIES | MOVIEID |  |  | MOVIEID  TITLE  LANGUAGE  RELEASE\_DATE  DIRECTOR |
| RATING | RATINGID | MOVIEID | STARS > 1 | RATINGID  STARS |

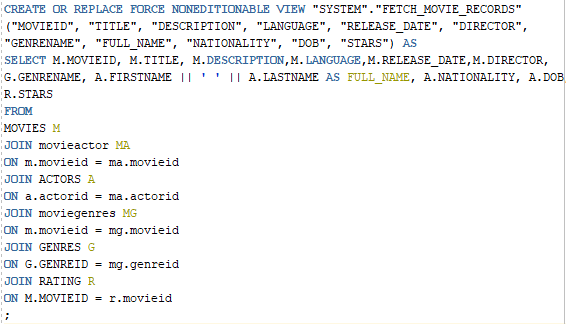
**PL/SQL, and Objects**

1. **Views**

View are nothing but a virtual or temperory table formed by actual table which also consist of rows and column. Main advantage of views are it is simple to execute where actually may consist of complex queries which can be stored as views and execute normally.

In our application we are using the below view to fetch and load the table in front-end which shows the list of movies available to book.





1. **Stored Procedure**

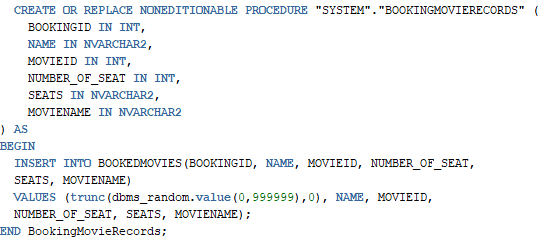
Stored Procedure are simple set of queries or set of SQL statement which are executed together. The primary goal is to hide direct queries from the code and improve the performance of database operation such as DDL or DML.

Based on the project requirement we have implemented 2 stored procedure

1. **BOOKINGMOVIERECORDS**

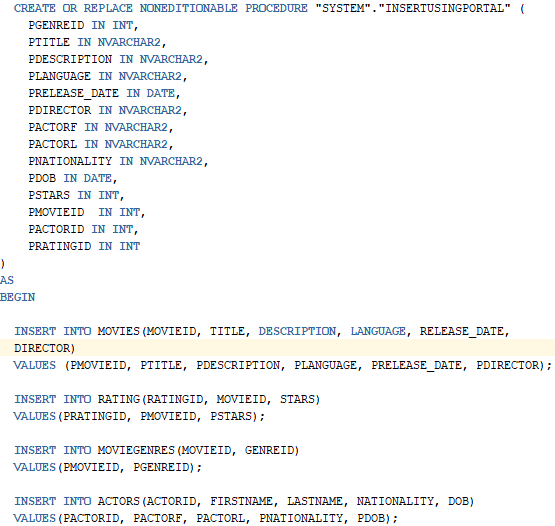
This procedure is to simply insert booking information from front-end using insert query.

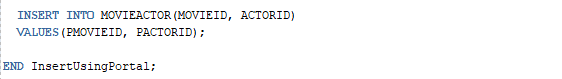
The parameters are below which are entered by used during booking are stored in bookedmovies table.



1. **INSERTUSINGPORTAL**

This procedure is used to insert the new record in all the respective table. In the GUI we have given an admin option where only registered user are allowed to add new movies if any one got released which can be also added into our database using front-end option which will call this procedure to store new record in all the respective table.





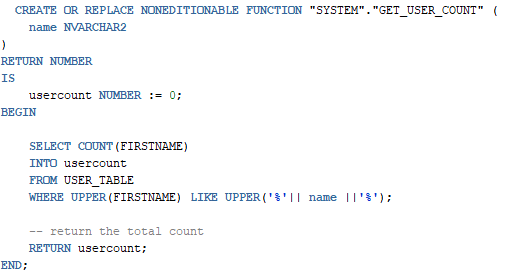
1. **Functions**

Oracle functions are database objects that contain a set of SQL statements to perform a specific task.

Based on the project requirement we have implemented 1 function

1. **GET\_USER\_COUNT**

This function is used to return user count from user\_table, which will result in evaluating this user is allowed to add movie into our database or not. Example User “Sanjay” got register using our GUI then he will also be eligible to add new movie into our database. If “Sanjay” has not registered then he will be getting an alert to first register and then add movie.



1. **Trigger**

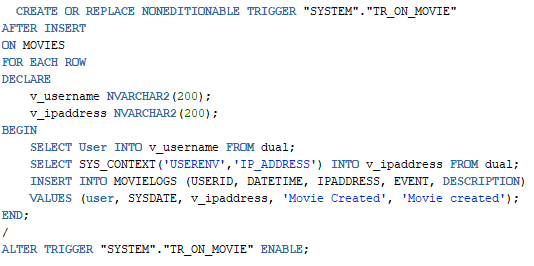
A trigger is a type of stored procedure that executes automatically when an event occurs in a database table.

Based on the project requirement we have implemented 1 trigger name “TR\_ON\_MOVIE”

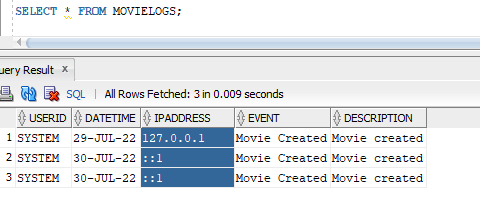
1. **TR\_ON\_MOVIE**

This trigger is used to automatically insert the log record into log table which is created when even a new movie is added into our database.

This trigger will store information in MOVIELOGS which will hold userid, datetime, Ipaddress, event and description.



**Execution**



**Web Application Using ASP .Net C#:**

We have Designed and Implemented “Movies Ticket Reservation ” which is a .Net based application used to provide the service to book and reserve the movie to watch in threater.

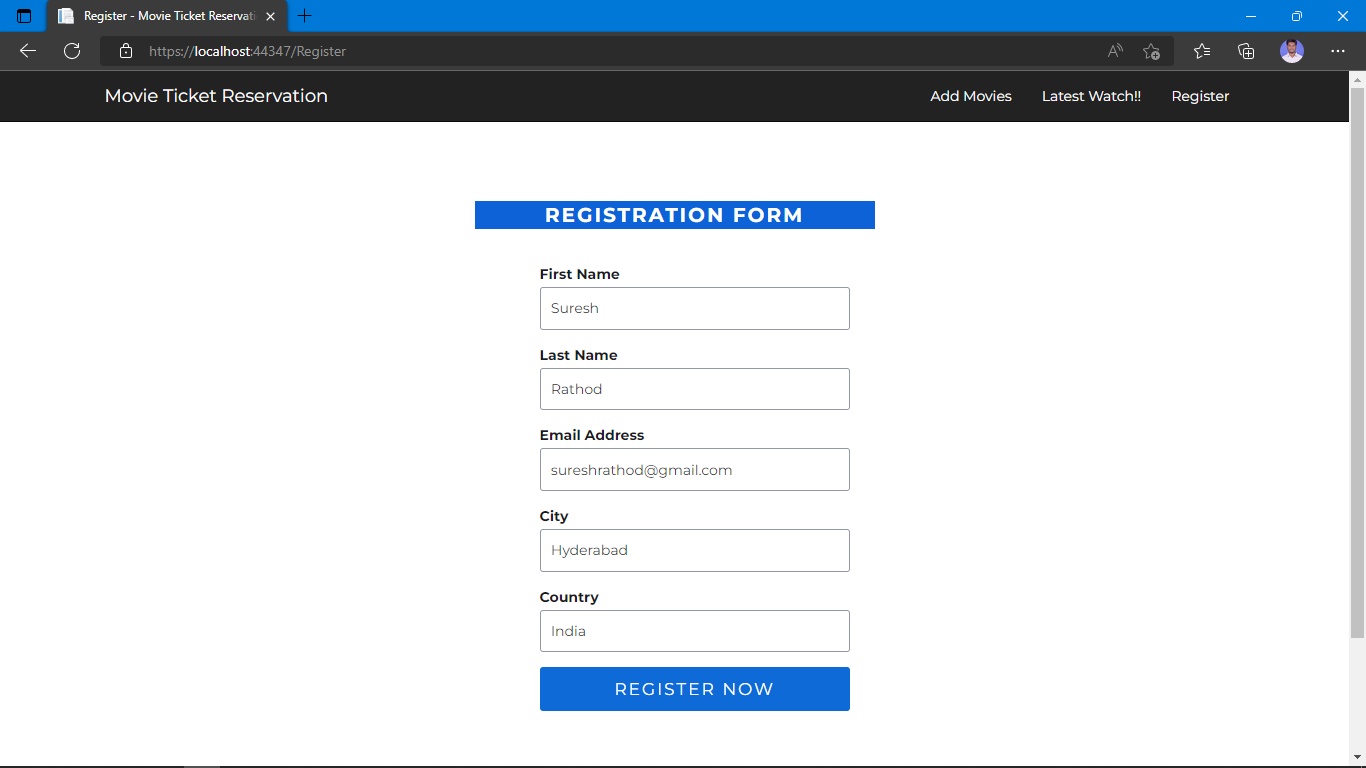
This application is running on 4.7 NET framework and Oracle 19c database application, Using the ASP.Net as framework other techstack used in this application is HTML, CSS, Bootstrap, Javascript, Jquery and C#

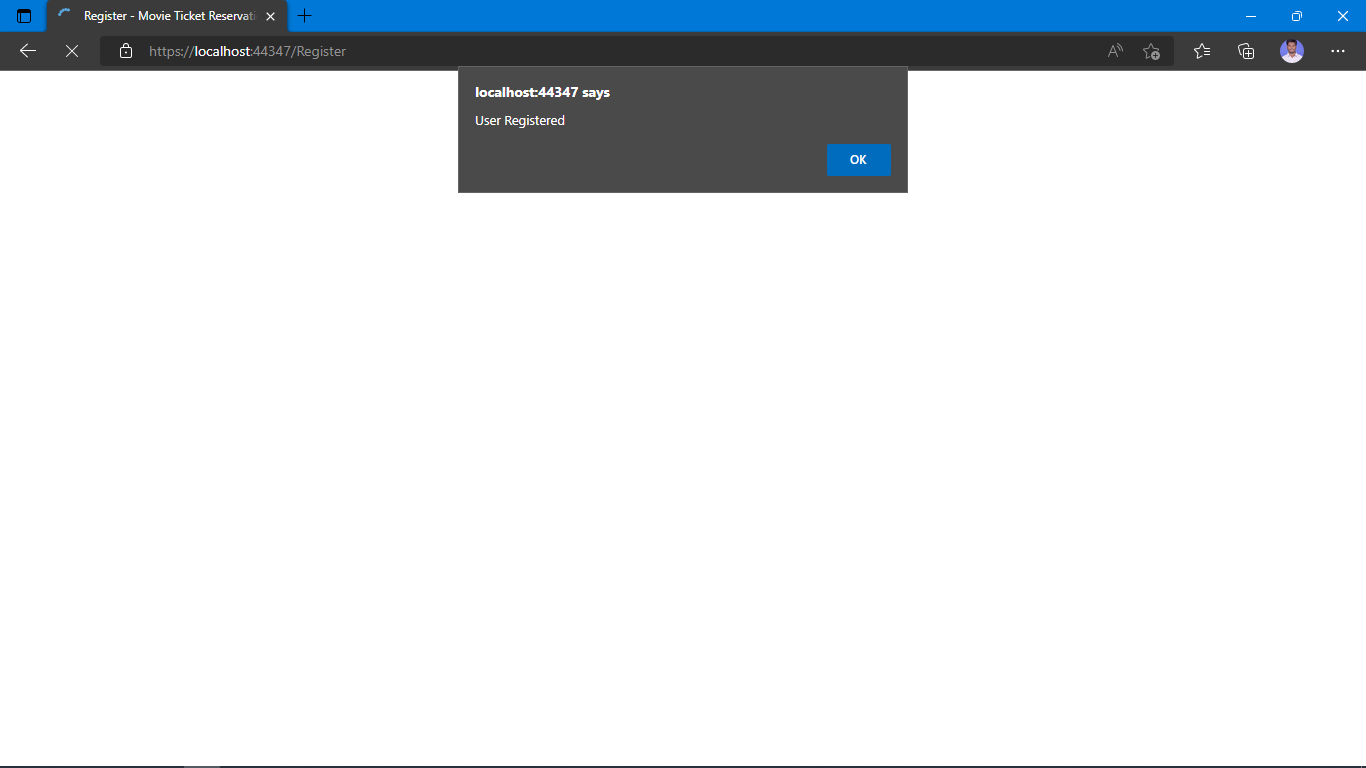
**This application consist of 4 Modules**

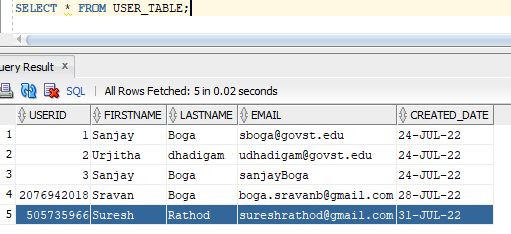
1. Registration
2. Latest Movies
3. Book Movies
4. Add New Movies

All these modules are publicly visible to user and only contraints to uses has to first register to our portal inorder to add new movies, which can only be done when user get registered with our otherwise they will be getting a warning alert to first register and then add movie to our database. This application is compatible with all the browser such as google chrome, opera and Microsoft egde.

1. **Registration**

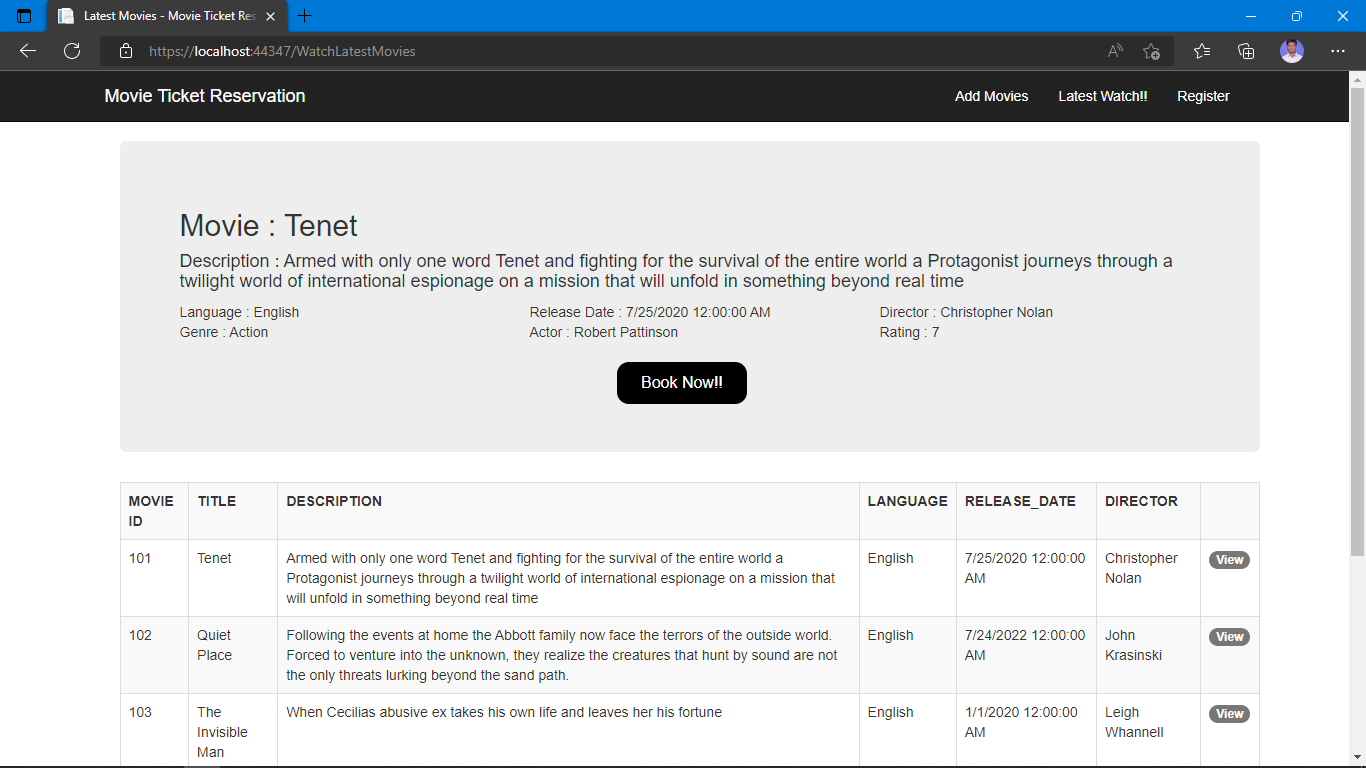




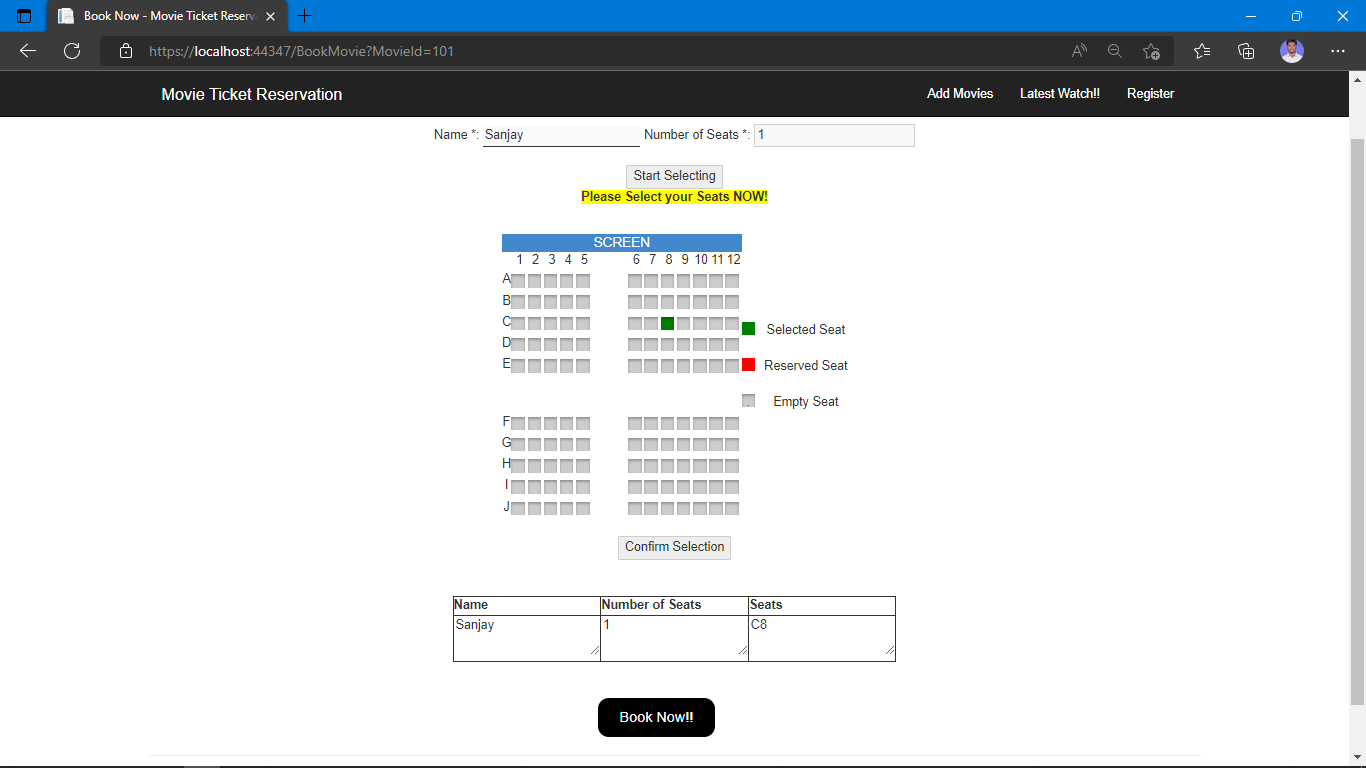


1. **Latest Watch**

Highlighted box show the important information related to specific movie when user click on view button. Information such as movie name, description, language, rating, director and so on.

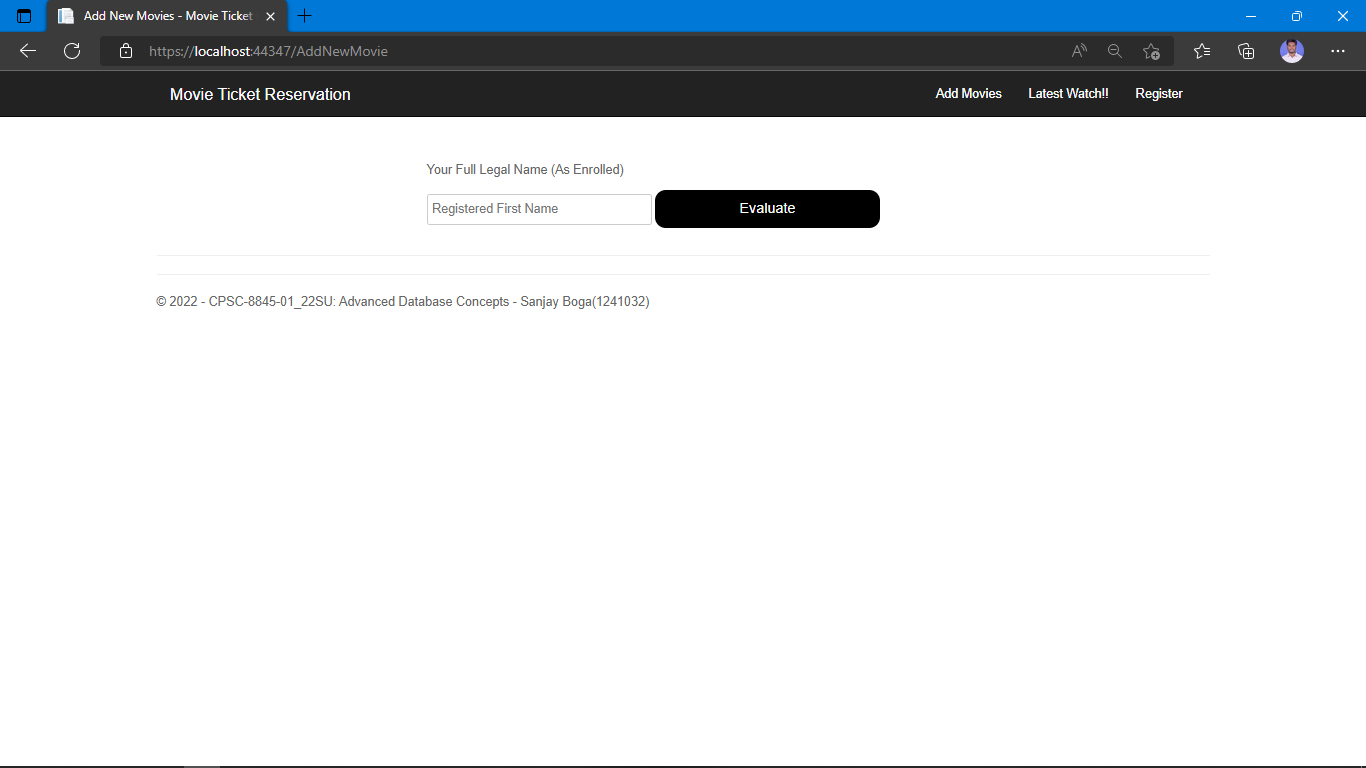


When user click on book now!! It will be navigating to booking package which allows the user to book the movie, where user given important information such as name, no of seats to book and below seat and confirms by clicking on book now which will call backend procedure to store into bookedmovies table.

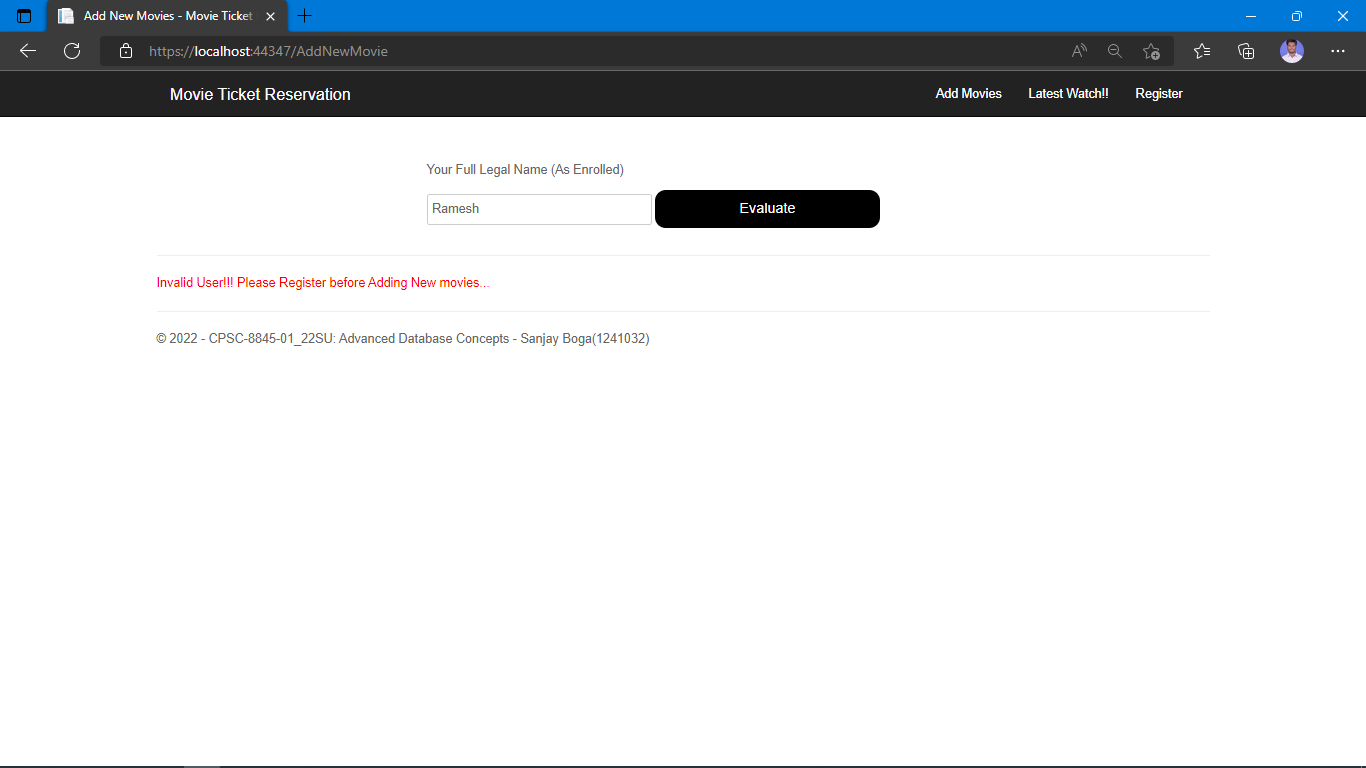


1. **Add Movies**

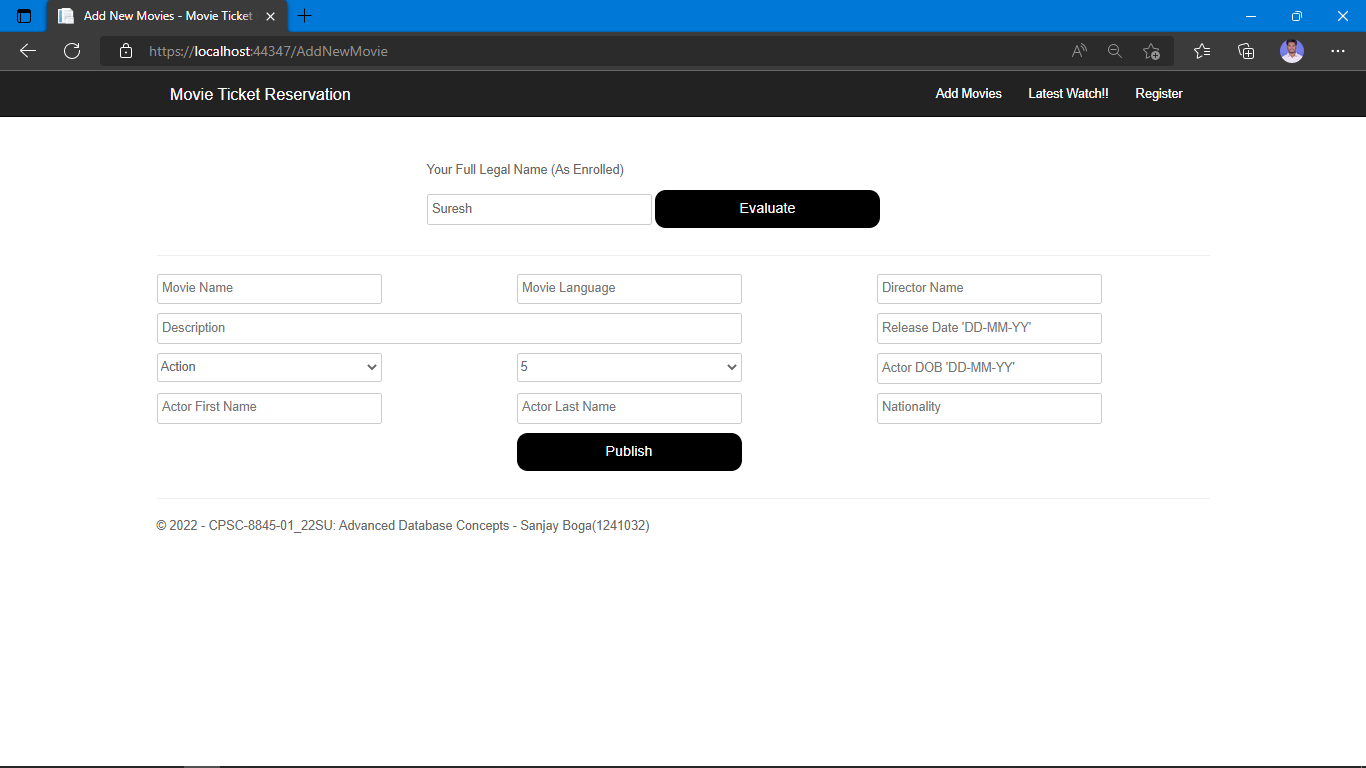
First user gives the register name ex: Sanjay or Suresh if the user is registered with us then our portal will allow them to add movie into our database else return alter to register first and then add movie.



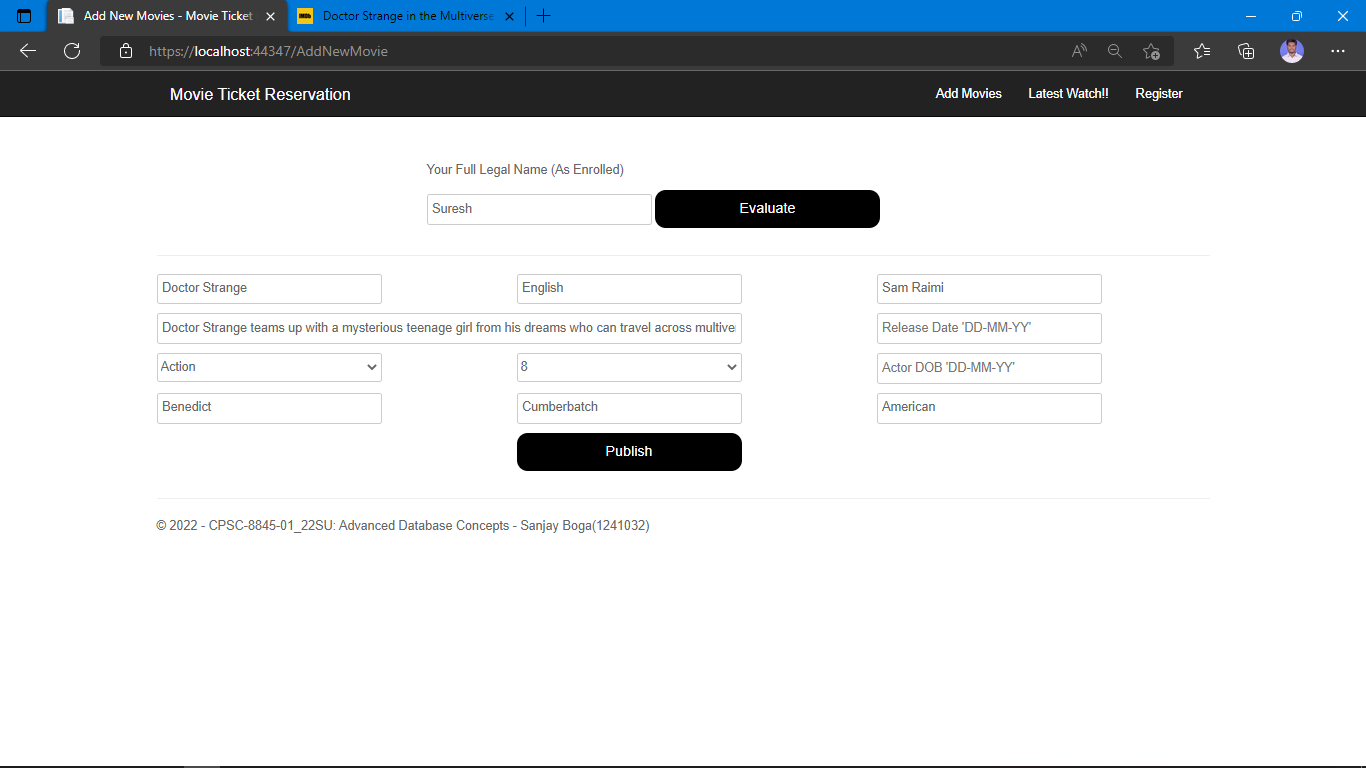
As Ramesh is not register with us our tool is asking him to first register and then add movie

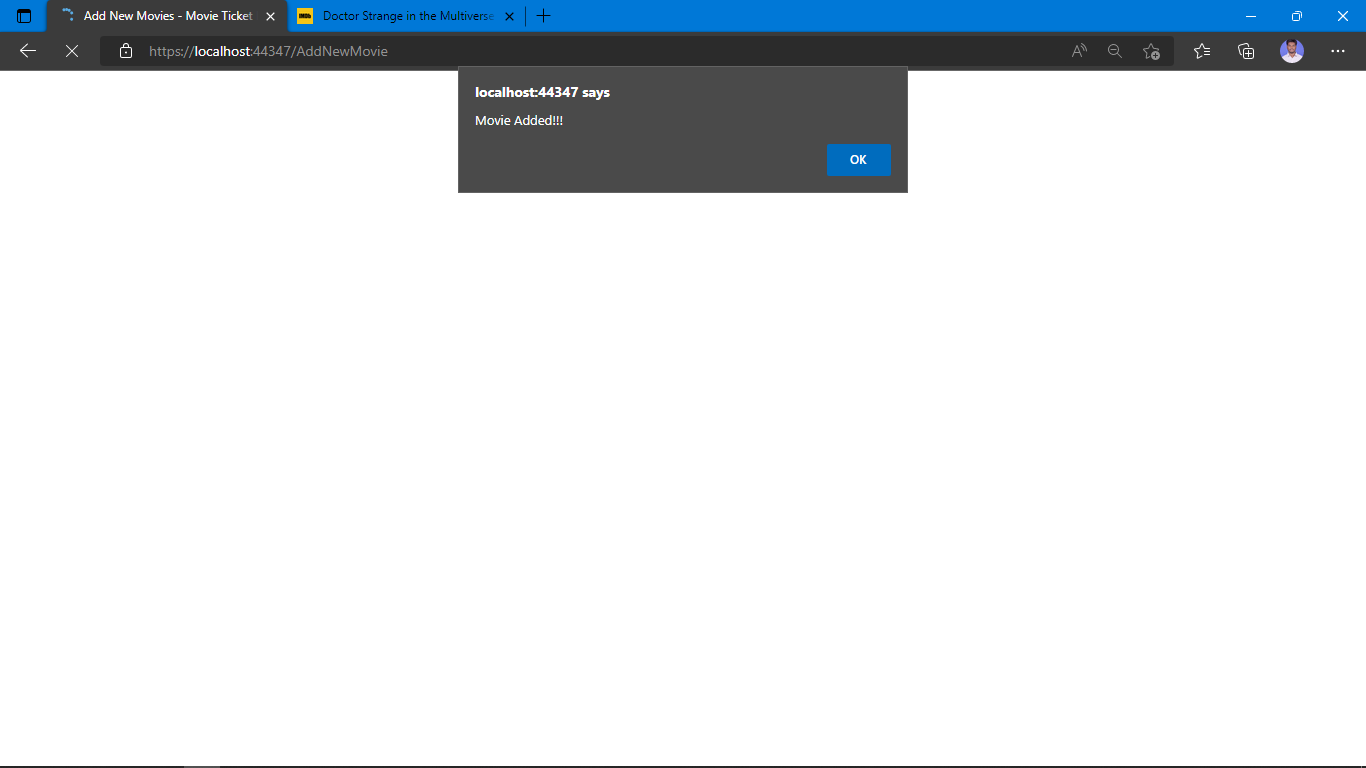


As name provided is Suresh who recently got register he is now allowed to add movie to our database



Adding Sample movie to our database using tool.





As we can see here once the user has added the movie through GUI now we can see one new record in our database name 108 MovieID, name Doctor Strange which can also been see now in latest movie section

