2/5/2025

DATABASE SYSTEM

COM5014M

Movie Theatre Management System

Ishara Dilshan De Silva Sobana Handi
230189784

Table of Contents

1.0 Introduction

1.1 System Overview

This Movie Theatre Management system is a database-driven solution designed to streamline the day-to-day operations of theatres. This database system allows administrators to manage theatres and empower movie theatre owners to manage their operations. Administrators can add theatre owners to theatres, while theatre owners can add movies, screening schedules, ticket prices, and seating configurations, and can check and manage the number of tickets purchased. Customers can browse available movies, select showtimes, and book tickets based on availability. The system aims to increase operational efficiency, improve overall customer experience and reduce manual effort.

1.2 Purpose of Database Design

The goal of this database design is to ensure seamless interaction between administrators, theater owners, and customers, facilitating efficient and organized management of theater operations. The system is built to: Provide secure access control for different user roles. Store and retrieve information about movies, theaters, shows, and reservations. Ensure data consistency and integrity through well-defined relationships and normalization. Support scalability for adding new theaters, movies, and features in the future.

1.3 Scope of the System

Administrator Role: Administrators can add theaters, assign permissions to theater owners, and remove unwanted movies from the entire system. Theater Owner Role: Theater owners can add movies, set ticket prices, schedule shows, and manage seat availability.

Customer Role: Customers can view movie details, select showtimes, and book tickets for specific dates and times. Database Management: The system will manage data related to theaters, movies, screens, schedules, ticket reservations, and user accounts. User Interaction: Provide a user-friendly interface for theater owners and customers to interact effectively with the system through a website.

Part 01: Database Specification and Design

2.0 Requirements Analysis

2.1 Functional Requirements

1. **Customer Registration & Authentication:**

   Customers must register and log in to book tickets and post-login, they can book seats and reserve tickets anytime. Customers can browse movies, view schedules, and book tickets for specific dates and times. Customers can book tickets and cancel them.

2. **Theatre Management:**

   Admin can add theatres and can manage theatres, including details such as Name, Place, Address, and State, and unwanted movies can be deleted. Theatre can add their Movies and movie shows, and shows times and they can start and stop movies

3. **Seat Reservation:**

   Customers can reserve seats and confirm availability, and if a seat is unavailable, can select alternative dates or seats.

4. **Tickets Reservation:**

   Customers can book movie tickets based on schedule and availability. If a ticket is unavailable, can you select alternative dates

5. **Booking History:**

   Customers can view booking history and cancel tickets. Theatres can manage all bookings, Check Day by day bookings.

6. **Shows Scheduling:**

   Theatre can create shows such as First, Second and Noon and can give them time. They specify start and end times

2.2 Non-functional Requirements

1. Performance

The system must handle multiple simultaneous user requests efficiently.

2. Scalability

The system must accommodate increased users, movies, and theatres over time

3. Usability

The interface should be intuitive and user-friendly for all user roles.

4. Maintainability

The database and application should be easy to update and maintain over time

5. Security

Sensitive data like passwords

User credentials and sensitive data must be encrypted to ensure privacy and security.

2.3 User Roles and Access Control

Administrator Role:

- Add theatres with their username passwords and states and their details.
- Unwanted movies can be deleted.
- Add Upcoming Movie news.

Customer Role:

- Access personal profiles and booking history.
- Access to view movies, view upcoming movie news, view movie trailer, and book tickets with number of seats.
- Can create their own account.
- Can view seat availability.

Theatre Role:

- Add their theatre movies and their Shows and show times and screens, number of seats and prices.
- Manage movie details such as start movie and end movies and stop it.
- Access to manage their theatres, including adding movies, shows, and ticket pricing.
- Can view booking and manage seat availability.
- They can check day-to-day bookings.

Access Control:

- Customers can only manage their own bookings.
- Only admins have permission to add theatres and add upcoming movie news.
- Data privacy ensures no unauthorized access to sensitive information

3.0 Conceptual Design

3.1 Entities

1. Login
2. Theatre
3. Movie
4. Screen
5. Show Times
6. Booking
7. User
8. Show

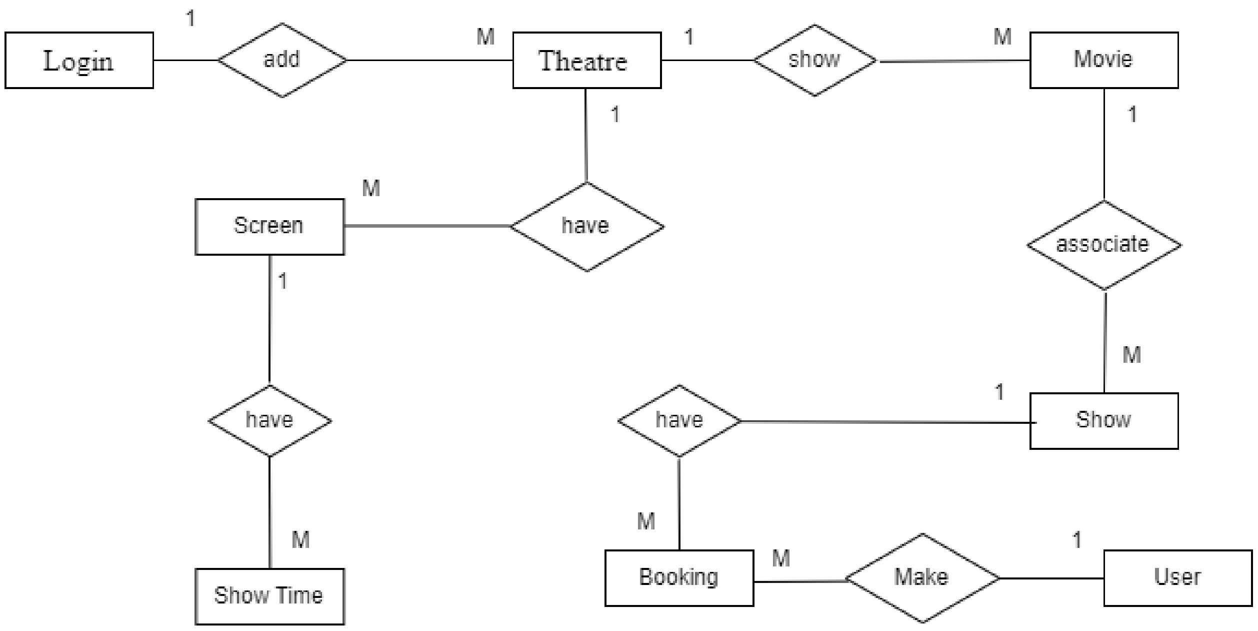


Figure 1 - Entities

3.2 Attributes

Here are the attributes for each entity.

1. **Theatre**
   - **Attributes**:
     - T_ID (Primary Key)
     - Name
     - Place
     - Address
     - Pin
     - State
2. **Login**
   - **Attributes**:
     - ID (Primary Key)
     - Username
     - Password
     - User_Type
     - UserID (Foreign Key referencing User.User_ID)
3. **Screen**
   - **Attributes**:
     - Screen_ID (Primary Key)
     - Screen_Name
     - Seats
     - Charge
     - T_ID (Foreign Key referencing Theatre.T_ID)

4. **Show**
   - **Attributes**:
     - S_ID (Primary Key)
     - Start_Date
     - Status (Availability)
     - R_Status
     - M_ID (Foreign Key referencing Movie.Movie_ID)
     - T_ID (Foreign Key referencing Theatre.T_ID)
5. **Movie**
   - **Attributes**:
     - Movie_ID (Primary Key)
     - Movie_Name
     - Release_Date
     - Status
     - Description
     - Image
     - Video_URL
     - Cast
6. **ShowTime**
   - **Attributes**:
     - ST_ID (Primary Key)
     - Start_Time
     - Name
     - Screen_ID (Foreign Key referencing Screen.Screen_ID)
     - S_ID (Foreign Key referencing Show.S_ID)

7. **Booking**
   - **Attributes**:
     - Book_ID (Primary Key)
     - Amount
     - No_Seats
     - Ticket_Date
     - Date
     - Status
     - User_ID (Foreign Key referencing User.User_ID)
     - S_ID (Foreign Key referencing Show.S_ID)

8. **User**
   - **Attributes**:
     - User_ID (Primary Key)
     - Name
     - Gender
     - Age
     - Email
     - Phone_No

3.3 Entity Relationships

1. Theatre and Screen (1:M)
    - A Theatre can have multiple Screens.
    - Key Mapping: tbl_theatre.t_id → tbl_screens.t_id.
2. Theatre and Movie (1:M)
    - A Theatre can show multiple Movies.
    - Key Mapping: tbl_theatre.t_id → tbl_movie.t_id.
3. Screen and Show Time (1:M)
    - A Screen can have multiple Show Times.
    - Key Mapping: tbl_screens.screen_id → tbl_show_time.screen_id.
4. Show Time and Show (1:M)
    - A Show Time can have multiple Shows.
    - Key Mapping: tbl_show_time.st_id → tbl_shows.st_id.
5. Movie and Show (1:M)
    - A Movie can be associated with multiple Shows.
    - Key Mapping: tbl_movie.movie_id → tbl_shows.movie_id.
6. Show and Booking (1:M)
    - A Show can have multiple Bookings.
    - Key Mapping: tbl_shows.s_id → tbl_bookings.show_id.
7. User and Booking (1:M)
    - A User can make multiple Bookings.
    - Key Mapping: tbl_registration.user_id → tbl_bookings.user_id.
8. Login and Theatre (1:M)
    - Admin can add multiple theatres.
    - Key Mapping: tbl_login.user_id → tbl_theatre.t_id
9. Booking and Screens (1:M)
    - Bookings are connected to Screens through Show and Show Time
    - Key Mapping: tbl_bookings.screen_id → tbl_screens.screen_id
10. Show and Screen (1:M)
    - Bookings are linked to Theatres via Shows, Show Time, and Screens.
    - Key Mapping: tbl_bookings. t_id → tbl_theatre.t_id

11. Show and Theatre (1:M)
   - Shows are linked to Theatres via Screens.
   - Key Mapping: tbl_shows. theatre_id → tbl_theatre.theatre_id.

3.4 ER Diagram

Entities & Relationships:

- Theatre ↔ Screen (1:M)
- Theatre ↔ Movie (1:M)
- Screen ↔ Show Time (1:M)
- Show Time ↔ Show (1:M)
- Movie ↔ Show (1:M)
- Show ↔ Booking (1:M)
- User ↔ Booking (1:M)
- Login ↔ Theatre (1:M)
- Booking ↔ Screens (1:M)
- Show ↔ Screen (1:M)
- Show ↔ Theatre (1:M)

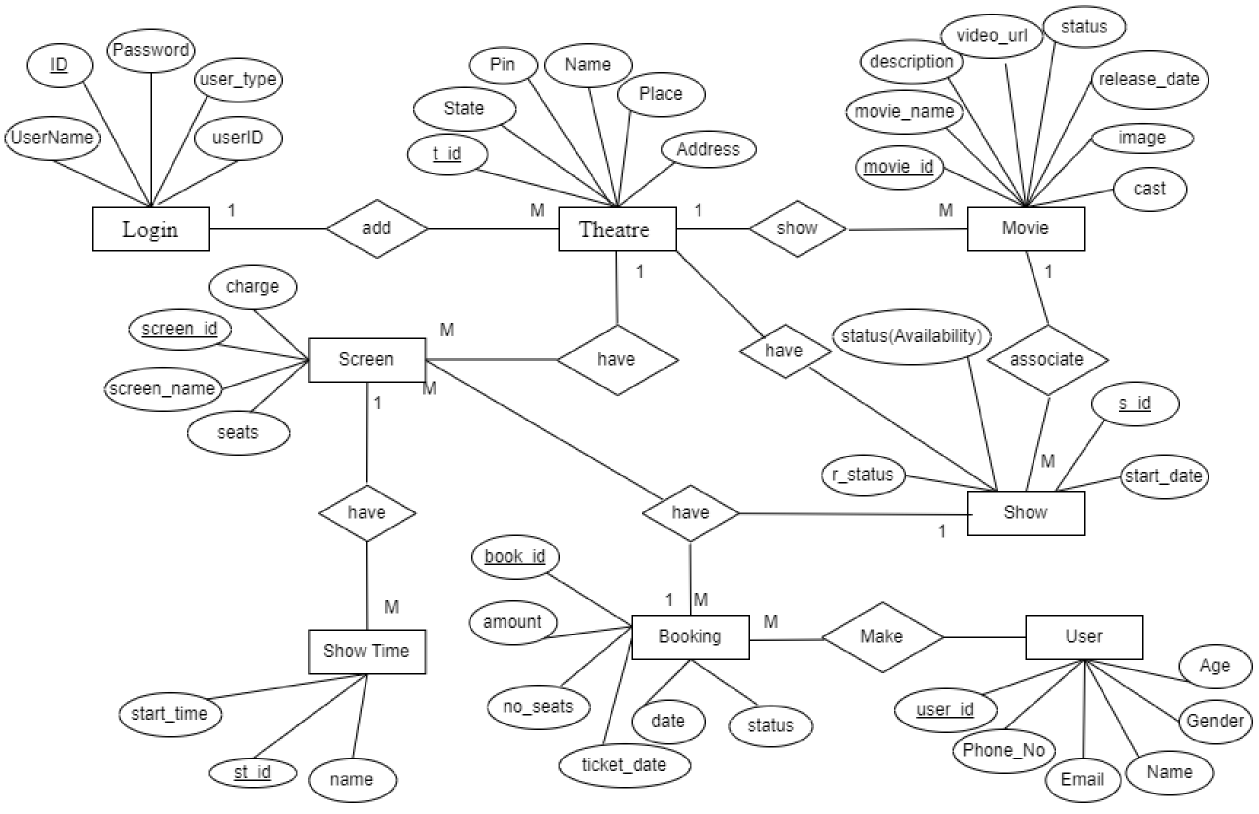
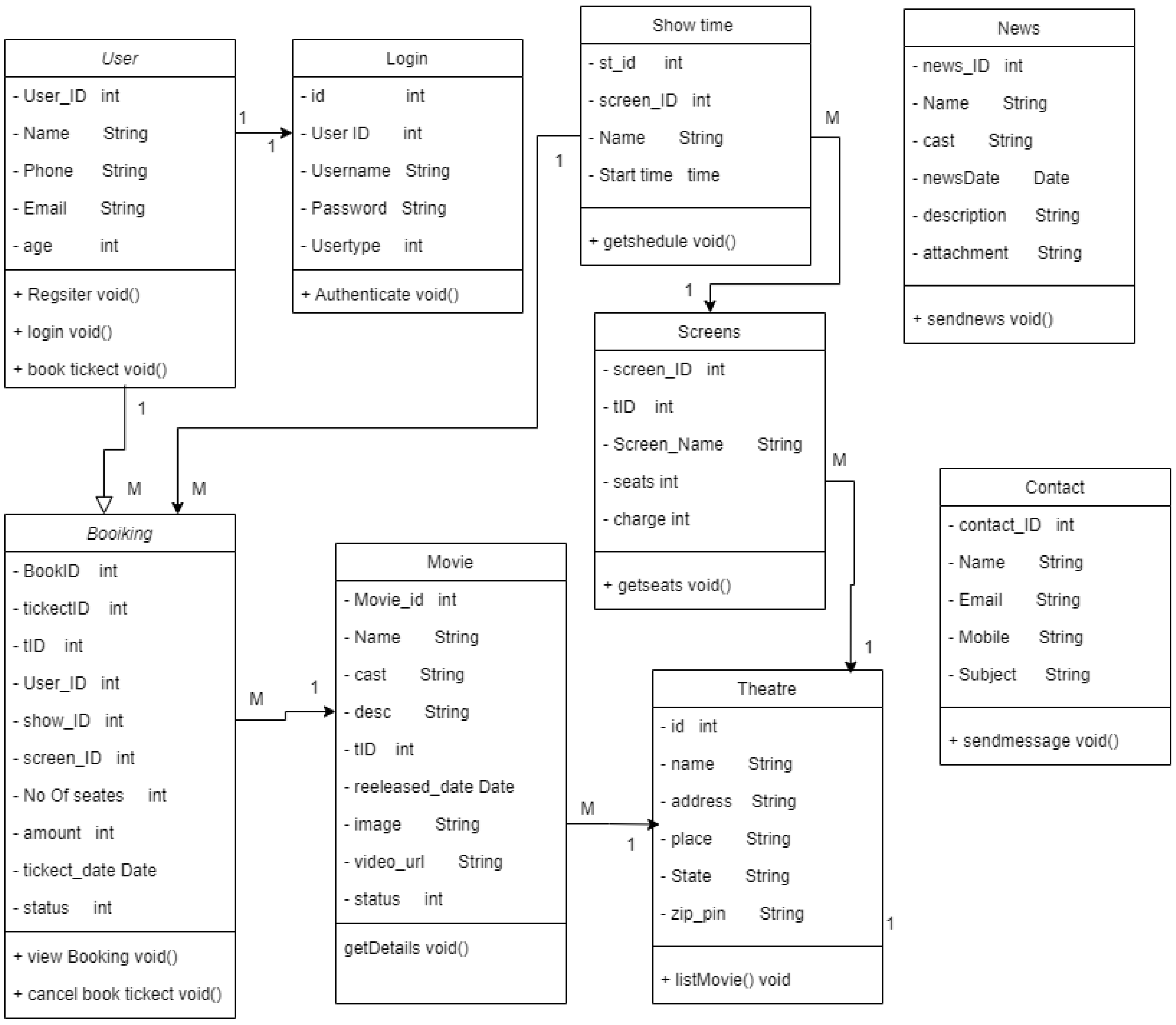


Figure 2 - ER Diagram

3.5 UML Diagram



Above class diagram have one to one relationships (1:1), one to many relationships (1:M) and many to one relationships (M:1)

1. User – Login (1:1)

   A user has one Login credential and Login Entry belongs to one User.

2. User – Booking (1:M)

   Connect User_ID in User to User_ID in Booking.

   User can book Multiple tickets and A booking is associated with one user

3. Theatre – Screens (1:M)

   Connect ID in Theatre to tID in Screens

   A Theatre has multiple Screens and A Screen belongs to one Theatre.

4. Theatre – Movie (1:M)
   Connect tID in Theatre to tID in Movie.
   A Theatre can show multiple movies, and A Movie is available in one Theatre.

5. Screens – Show Time(1:M)
   Connect screen_ID in Screens to screen_ID in Show Times.
   A Screen can have multiple showtimes and A Show Time is assigned to one screen.

6. Contact (Independent Table)
   The Contact table is separate, where users can send messages.
   The news_ID is unrelated to other entities.

7. News (Independent Table)
   The News table is separate, where admin can add New Upcoming News.
   The news_ID is unrelated to other entities.

8. Booking - Movie (M:1)
   Multiple Bookings can be made for the same Movie and A Movie is linked to many bookings.
   Connect tID in Booking to tID in Movie.

9. Show Time - Booking (1:M)
   A Show Time can have multiple Bookings and A Booking is linked to one Show Times.
   Connect show_ID in Show Time to show_ID in Booking.

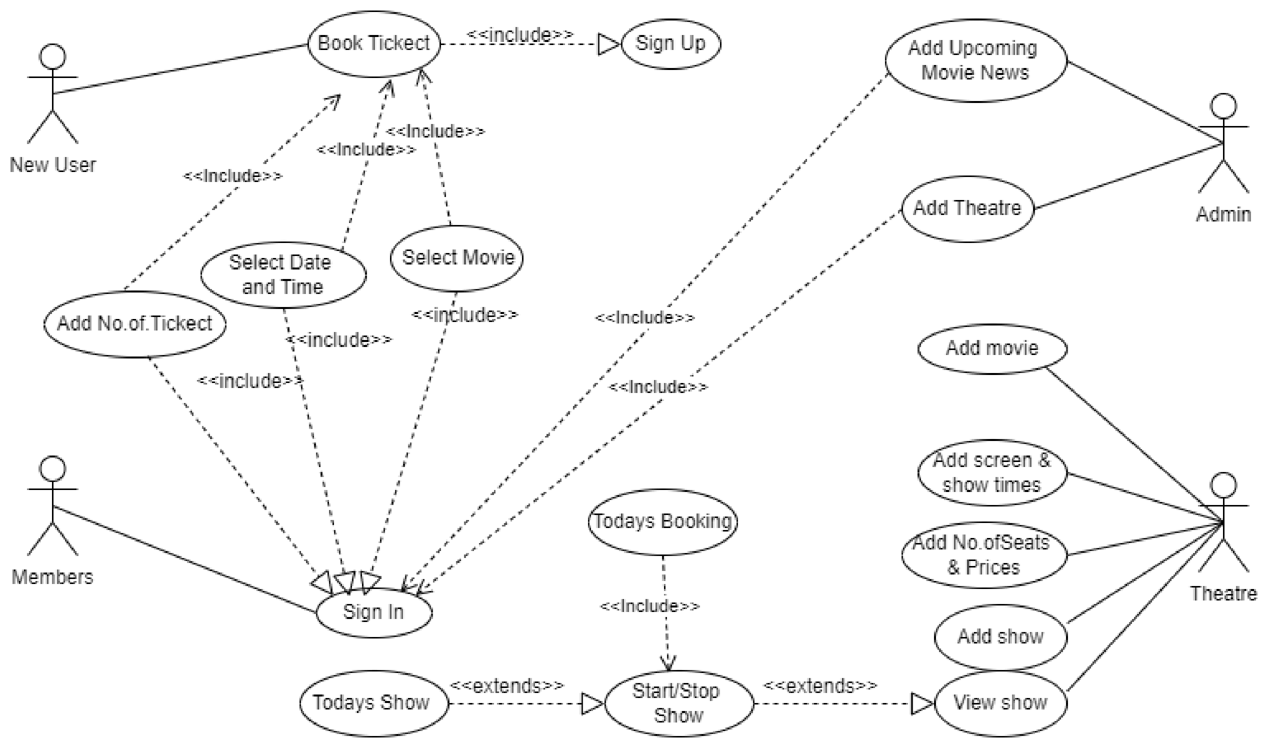


Figure 3 - Use Case Diagram

Actors: Newcomers, Members, Admin, Theatre

Here the above Use case diagram displays about system uses respective actions. The system users are Newcomers, Members Admin and Theatres. The relationships between use cases are shown with **<<include>>** and **<<extend>>.** New users must sign up before booking tickets. Admin can add only theatres and upcoming movie news. Members can view movies, Upcoming movie news and movie trailers. But members must log in to the account and select movie and select time then can select date and number of seats before booking tickets. Theatre can add any number of movies and can add screens in theatre have and number of seats and their prices. After that Theatre can create shows using movie screens and show time with start date. The View Show can start, stop and end movies using web applications, and then can check day-to-day shows and after start show can check day-to-day booking.

3.6 Business Rules

Here is a list of critical business rules that the database must enforce based on the provided model:

1. **User and Login**:
    - ➢ Users must have a unique login ID and username.
    - ➢ Three users are admin, theatre, and customer admin is already added
    - ➢ A login must be associated with exactly one user.
2. **Theatre**:
    - ➢ Each theatre must have a unique identifier (T_ID).
    - ➢ A theatre can have multiple screens, but each screen must belong to only one theatre.
3. **Screen**:
    - ➢ Each screen must have a unique identifier (Screen_ID).
    - ➢ A screen must be associated with a single theatre.
    - ➢ Each screen must define the number of available seats and charge per seat.
4. **Movie**:
    - ➢ Each movie must have a unique identifier (Movie_ID).
    - ➢ A movie can be shown in multiple theatres and at multiple times, but each show must link to one movie.
    - ➢ Movies must include key details like name, release date, cast, and video URL.
5. **Show**:
    - ➢ A show must have a unique identifier (S_ID).
    - ➢ A show must belong to one theatre and be associated with one movie.
    - ➢ A show must indicate its availability status.
6. **ShowTime**:
    - ➢ A showtime must have a unique identifier (ST_ID).
    - ➢ Each showtime must belong to a specific screen.
    - ➢ A screen can have multiple showtimes but only one showtime per time slot.

7. **Booking**:
    - ➢ Each booking must have a unique identifier (Book_ID).
    - ➢ A booking must be linked to one user and one show.
    - ➢ The number of seats booked must not exceed the available seats for the selected showtime.
    - ➢ Each booking must include details like the amount paid, number of seats booked, ticket date, and status.
8. **User**:
    - ➢ A user must have a unique identifier (User_ID) and valid contact details (email and phone number).
    - ➢ A user can make multiple bookings, but each booking must be linked to one user.
9. **General Rules**:
    - ➢ The database must enforce referential integrity by ensuring that all foreign key relationships are valid.
    - ➢ No duplicate entries are allowed for primary keys in any table.

4.0 Logical Design

4.1 Data Model

1. tbl_bookings

- book_id: (PK) Unique ID for each booking.
- ticket_id: Unique ticket identifier.
- t_id: (FK) Theatre ID, links to tbl_theatre.
- user_id: (FK) User ID, links to tbl_registration.
- show_id: (FK) Show ID, links to tbl_shows.
- screen_id: (FK) Screen ID, links to tbl_screens.
- no_seats: Number of seats booked.
- amount: Total amount for the booking.
- ticket_date: Date when the ticket is issued.
- date: Date of the show.
- status: Status of the booking (active or cancelled).

2. tbl_contact

- contact_id: (PK) Unique ID for each contact.
- name: Name of the person contacting.
- email: Contact email address.
- mobile: Contact phone number.
- subject: Subject of the inquiry or feedback.

3. tbl_news

- news_id : (PK) Unique ID for each contact
- name: Name of the movie
- cast: List of cast members.
- news_date: released date
- description: Description of the movie
- attachment: Upcoming movie image

4. tbl_login
    - id: (PK) Unique login ID.
    - user_id: User ID Unique
    - username: Unique Username, typically the email address.
    - password: User's password (hashed).
    - user_type: Type of user (Admin: 0, Theatre: 1, Customer: 2).
5. tbl_movie
    - movie_id: (PK) Unique ID for each movie.
    - t_id: (FK) Theatre ID, links to tbl_theatre.
    - movie_name: Name of the movie.
    - cast: List of cast members.
    - desc: Description of the movie.
    - release_date: Movie's release date.
    - image: Image URL for the movie poster.
    - video_url: URL for the movie's trailer or video.
    - status: Movie's status (active or inactive).
6. tbl_registration
    - user_id: (PK) Unique ID for the user.
    - name: User's name.
    - email: User's email address.
    - phone: User's phone number.
    - age: User's age.
7. tbl_screens
    - screen_id: (PK) Unique ID for the screen.
    - t_id: (FK) Theatre ID, links to tbl_theatre.
    - screen_name: Name of the screen.
    - seats: Total number of seats in the screen.
    - charge: Charge per seat.
8. tbl_shows
    - s_id: (PK) Unique ID for the show.

- st_id: (FK) Show time ID, links to tbl_show_time.
- t_id: (FK) Theatre ID, links to tbl_theatre.
- movie_id: (FK) Movie ID, links to tbl_movie.
- start_date: Show's start date.
- status: Status of the show (available or not).
- r_status: Reservation status (can indicate if the show is fully booked or open).

9. tbl_show_time
   - st_id: (PK) Unique ID for show time.
   - screen_id: (FK) Screen ID, links to tbl_screens.
   - name: Name of the show time (e.g., "Noon", "Second").
   - start_time: Time at which the show starts.

10. tbl_theatre
    - t_id: (PK) Unique ID for the theatre.
    - name: Name of the theatre.
    - address: Full address of the theatre.
    - place: Place (city) where the theatre is located.
    - state: State where the theatre is located.
    - pin: Postal code (PIN) of the theatre's location.

Relationships:

1. **tbl_bookings**:
   - Many-to-One relationship with tbl_theatre (theatre).
   - Many-to-One relationship with tbl_registration (user).
   - Many-to-One relationship with tbl_shows (show).
   - Many-to-One relationship with tbl_screens (screen).
2. **tbl_contact**:
   - No direct relationships with other tables.

3. **tbl_news**
    - No direct relationships with other tables.
4. **tbl_login**:
    - No direct relationships with other tables.
5. **tbl_movie**:
    - Many-to-One relationship with tbl_theatre (theatre).
6. **tbl_registration**:
    - One-to-Many relationship with tbl_bookings (user can have multiple bookings).
7. **tbl_screens**:
    - Many-to-One relationship with tbl_theatre (theatre).
8. **tbl_shows**:
    - Many-to-One relationship with tbl_theatre (theatre).
    - Many-to-One relationship with tbl_movie (movie).
    - Many-to-One relationship with tbl_show_time (show time).
9. **tbl_show_time**:
    - Many-to-One relationship with tbl_screens (screen).
10. **tbl_theatre**:
    - One-to-Many relationship with tbl_movie (theatre can have multiple movies).
    - One-to-Many relationship with tbl_screens (theatre can have multiple screens).
    - One-to-Many relationship with tbl_shows (theatre can host multiple shows).
    - One-to-Many relationship with tbl_bookings (theatre can have many bookings).

4.2 Tables and Columns

1. **User (tbl_registration)**

   Attributes: user_id, name, email, phone, age, gender

2. **Login (tbl_login)**

   Attributes: id, user_id, username, password, user_type

3. **Theatre (tbl_theatre)**

   Attributes: t_id, name, address, place, state, pin

4. **Movie (tbl_movie)**

   Attributes: movie_id, t_id, movie_name, cast, desc, release_date, image, video_url, status

5. **Screen (tbl_screens)**

   Attributes: screen_id, t_id, screen_name, seats, charge

6. **Show Time (tbl_show_time)**

   Attributes: st_id, screen_id, name, start_time

7. **Show (tbl_shows)**

   Attributes: s_id, st_id, t_id, movie_id, start_date, status, r_status

8. **Booking (tbl_bookings)**

   Attributes: book_id, ticket_id, t_id, user_id, show_id, screen_id, no_seats, amount, ticket_date, date, status

9. **Contact (tbl_contact)**

   Attributes: contact_id, name, email, mobile, subject

10. **Upcoming Movie News (tbl_news)**

    Attributs: news_id, name, cast, news_date, description, attachment

4.3 Normalization Process

1. First Normal Form (1NF)

   Criteria: Eliminate repeating groups; ensure each column contains atomic values, and each row is unique.
   Evaluation:
   - All tables have unique primary keys (AUTO_INCREMENT for each table ensures uniqueness).
   - Columns contain atomic values (e.g., tbl_contact.subject and tbl_movie.cast are strings, not lists).

   Conclusion: Satisfies 1NF

2. Second Normal Form (2NF)

   Criteria: Meet 1NF and eliminate partial dependencies (no non-prime attribute should depend on part of a composite primary key).
   Evaluation:
   - Tables like tbl_bookings, tbl_movies, and tbl_registration have simple primary keys.
   - There are no composite keys in the schema, so partial dependencies are not an issue.

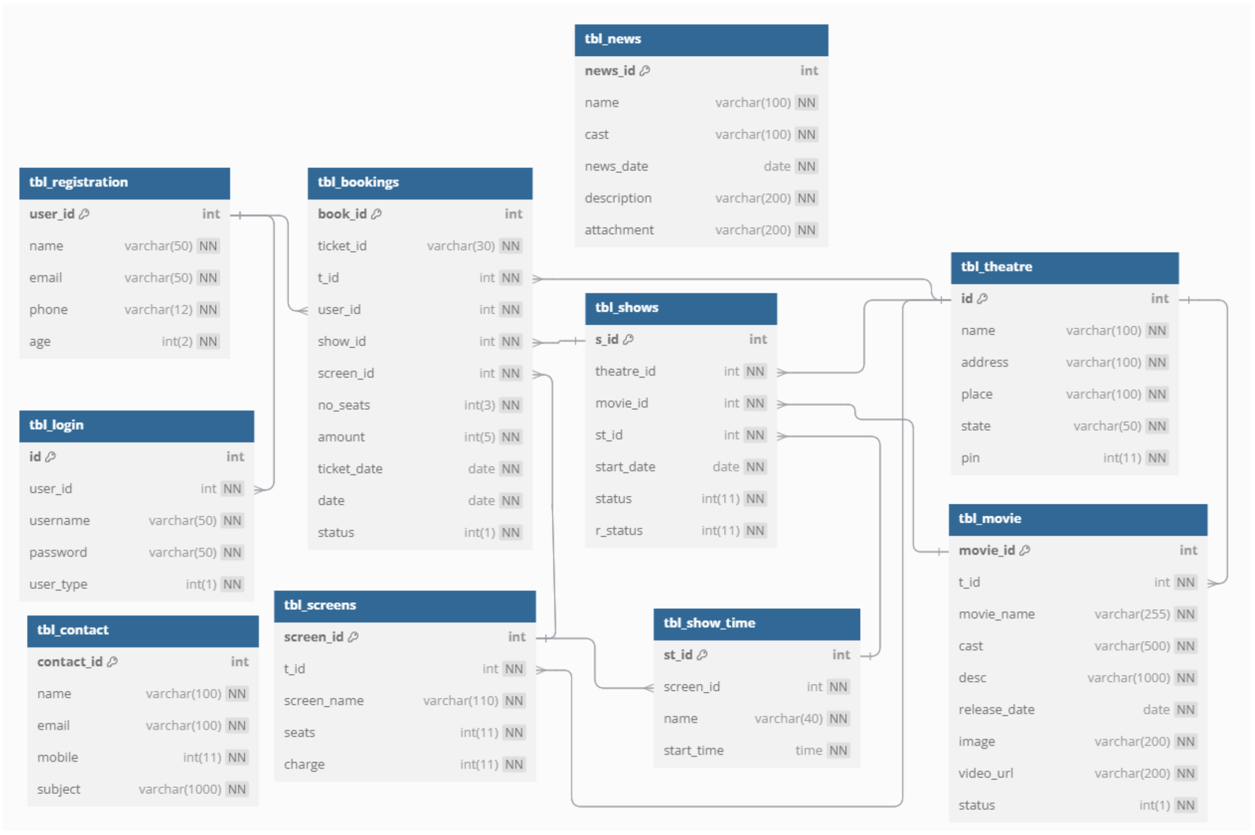
   Conclusion: Satisfies 2NF.

3. Third Normal Form (3NF)

   Criteria: Meet 2NF and eliminate transitive dependencies (no non-prime attribute should depend on another non-prime attribute).
   Evaluation:
   - Tables are well-designed with logical attributes grouped (e.g., tbl_registration holds user details, tbl_movie holds movie-specific details).
   - All non-key attributes depend directly on the primary key.
   - For example:
     - In tbl_bookings, attributes like no_seats and amount depend directly on book_id.
     - In tbl_movie, attributes like movie_name, cast, and release_date depend on movie_id.

   Conclusion: Satisfies 3NF.

4.4 Relational Schema



tbl_news		
news_id	int	
name	varchar(100)	NN
cast	varchar(100)	NN
news_date	date	NN
description	varchar(200)	NN
attachment	varchar(200)	NN

tbl_registration		
user_id	int	
name	varchar(50)	NN
email	varchar(50)	NN
phone	varchar(12)	NN
age	int(2)	NN

tbl_bookings		
book_id	int	
ticket_id	varchar(30)	NN
t_id	int	NN
user_id	int	NN
show_id	int	NN
screen_id	int	NN
no_seats	int(3)	NN
amount	int(5)	NN
ticket_date	date	NN
date	date	NN
status	int(1)	NN

tbl_theatre		
id	int	
name	varchar(100)	NN
address	varchar(100)	NN
place	varchar(100)	NN
state	varchar(50)	NN
pin	int(11)	NN

tbl_shows		
s_id	int	
theatre_id	int	NN
movie_id	int	NN
st_id	int	NN
start_date	date	NN
status	int(11)	NN
r_status	int(11)	NN

tbl_login		
id	int	
user_id	int	NN
username	varchar(50)	NN
password	varchar(50)	NN
user_type	int(1)	NN

tbl_movie		
movie_id	int	
t_id	int	NN
movie_name	varchar(255)	NN
cast	varchar(500)	NN
desc	varchar(1000)	NN
release_date	date	NN
image	varchar(200)	NN
video_url	varchar(200)	NN
status	int(1)	NN

tbl_screens		
screen_id	int	
t_id	int	NN
screen_name	varchar(110)	NN
seats	int(11)	NN
charge	int(11)	NN

tbl_contact		
contact_id	int	
name	varchar(100)	NN
email	varchar(100)	NN
mobile	int(11)	NN
subject	varchar(1000)	NN

tbl_show_time		
st_id	int	
screen_id	int	NN
name	varchar(40)	NN
start_time	time	NN

Figure 4 - Class Diagram

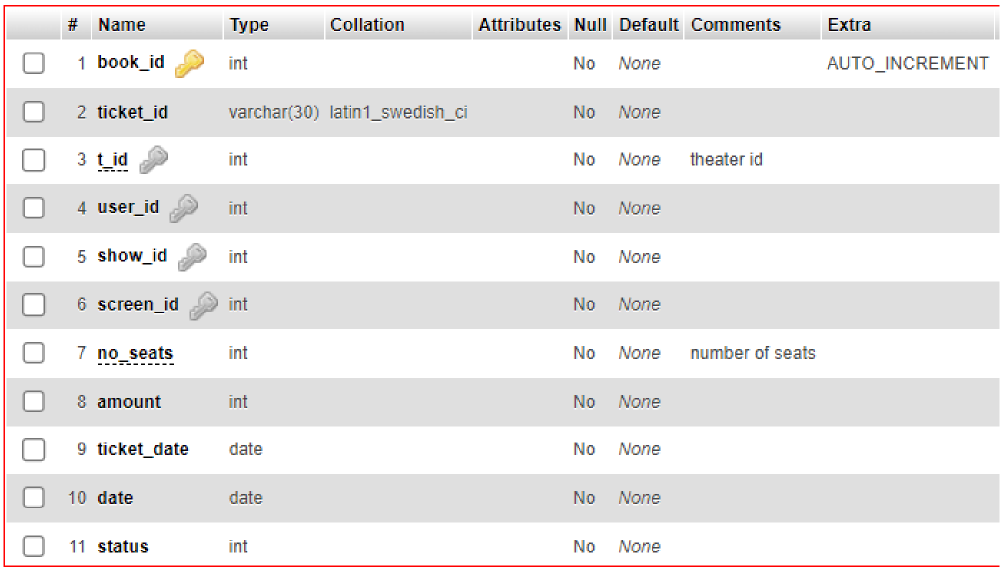
https://dbdocs.io/MovieDB?view=ClassDiagram

The provided class diagram represents a movie theater ticket booking system with multiple interconnected tables. The **tbl_registration** table stores user information such as name, email, phone, age, and **tbl_login** handle user authentication with fields for email or username, password, and user type. **tbl_bookings** records ticket bookings, linking users, and maintaining shows, screens, and theater seat numbers, ticket dates, and payment amounts. The **tbl_movie** table manages movie details including cast, description, release date, and multimedia links. The **tbl_theatre** table stores theater-related information such as name, address, and location. The **tbl_screens** table associates screens with theaters, providing details about seating capacity and ticket prices. Showtimes are handled through **tbl_shows** and **tbl_show_time**, defining movie screenings and their times. Additionally, the **tbl_news** table maintains updates on movies and related information, while **tbl_contact** manages user queries.

5.0 Physical Design

5.1 Table Definitions

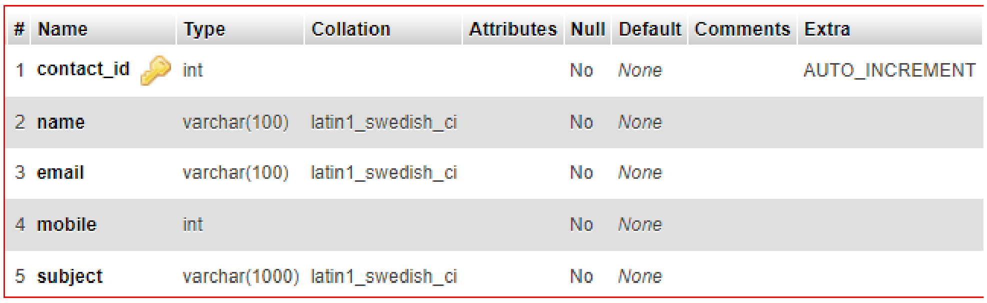
1. Table Name – tbl_booking



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	book_id	int			No	None		AUTO_INCREMENT
☐	2	ticket_id	varchar(30)	latin1_swedish_ci		No	None		
☐	3	t_id	int			No	None	theater id	
☐	4	user_id	int			No	None		
☐	5	show_id	int			No	None		
☐	6	screen_id	int			No	None		
☐	7	no_seats	int			No	None	number of seats	
☐	8	amount	int			No	None		
☐	9	ticket_date	date			No	None		
☐	10	date	date			No	None		
☐	11	status	int			No	None		

Figure 5 - Booking Table

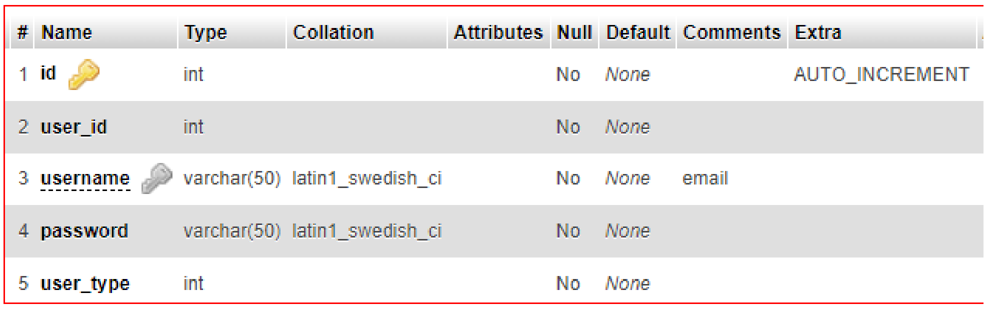
2. Table Name – tbl_contact



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	contact_id	int			No	None		AUTO_INCREMENT
2	name	varchar(100)	latin1_swedish_ci		No	None		
3	email	varchar(100)	latin1_swedish_ci		No	None		
4	mobile	int			No	None		
5	subject	varchar(1000)	latin1_swedish_ci		No	None		

Figure 6 - Contact Table

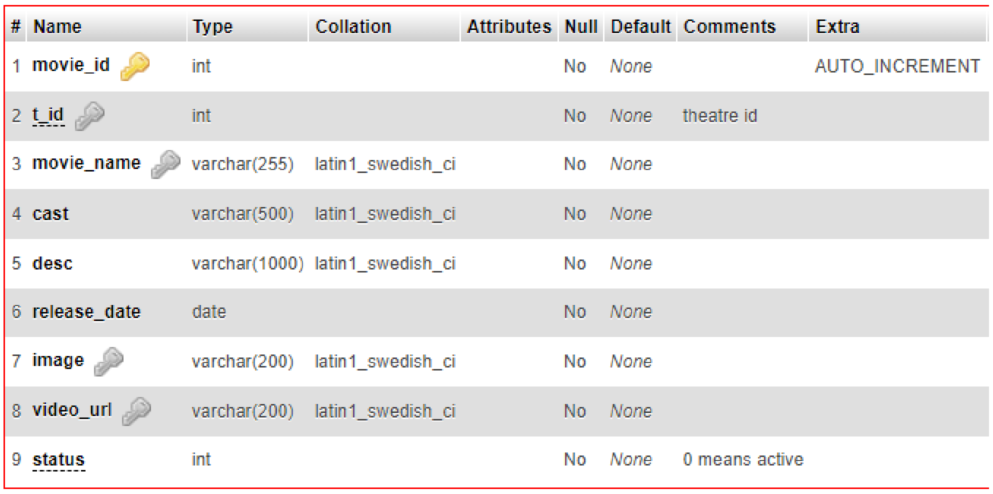
3. Table Name – tbl_login



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	None		AUTO_INCREMENT
2	user_id	int			No	None		
3	username	varchar(50)	latin1_swedish_ci		No	None	email	
4	password	varchar(50)	latin1_swedish_ci		No	None		
5	user_type	int			No	None		

Figure 7 - Login Table

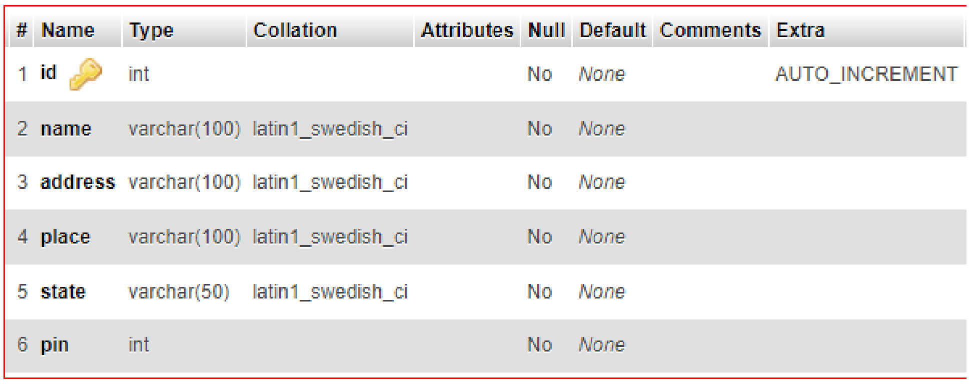
4. Table Name – tbl_movie



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	movie_id	int			No	*None*		AUTO_INCREMENT
2	t_id	int			No	*None*	theatre id	
3	movie_name	varchar(255)	latin1_swedish_ci		No	*None*		
4	cast	varchar(500)	latin1_swedish_ci		No	*None*		
5	desc	varchar(1000)	latin1_swedish_ci		No	*None*		
6	release_date	date			No	*None*		
7	image	varchar(200)	latin1_swedish_ci		No	*None*		
8	video_url	varchar(200)	latin1_swedish_ci		No	*None*		
9	status	int			No	*None*	0 means active	

Figure 8 – Movie Table

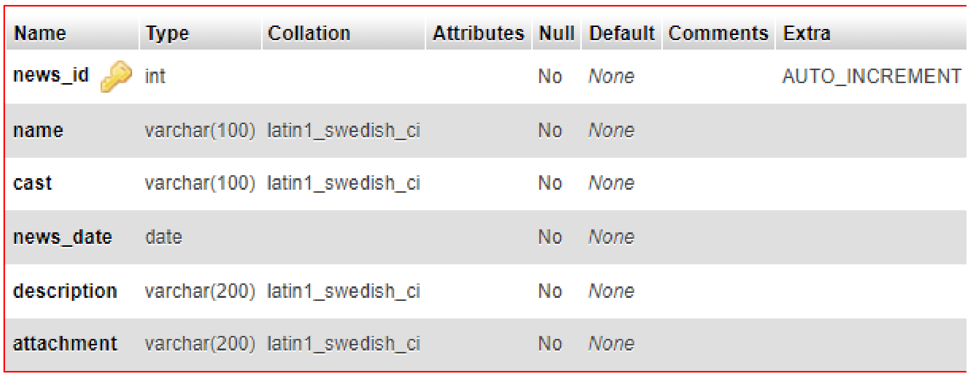
5. Table Name – tbl_theatre



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	*None*		AUTO_INCREMENT
2	name	varchar(100)	latin1_swedish_ci		No	*None*		
3	address	varchar(100)	latin1_swedish_ci		No	*None*		
4	place	varchar(100)	latin1_swedish_ci		No	*None*		
5	state	varchar(50)	latin1_swedish_ci		No	*None*		
6	pin	int			No	*None*		

Figure 9 – Theatre Table

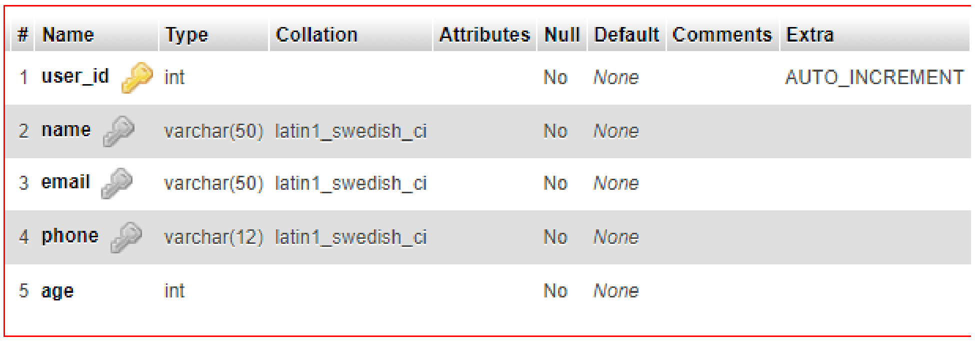
6. Table Name – tbl_news



Name	Type	Collation	Attributes	Null	Default	Comments	Extra
news_id	int			No	*None*		AUTO_INCREMENT
name	varchar(100)	latin1_swedish_ci		No	*None*		
cast	varchar(100)	latin1_swedish_ci		No	*None*		
news_date	date			No	*None*		
description	varchar(200)	latin1_swedish_ci		No	*None*		
attachment	varchar(200)	latin1_swedish_ci		No	*None*		

Figure 10 – News Table

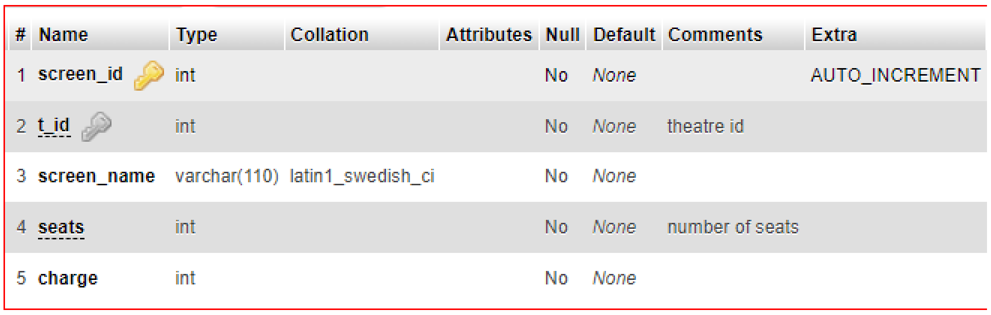
7. Table Name – tbl_registration



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	user_id 🔑	int			No	*None*		AUTO_INCREMENT
2	name 🔑	varchar(50)	latin1_swedish_ci		No	*None*		
3	email 🔑	varchar(50)	latin1_swedish_ci		No	*None*		
4	phone 🔑	varchar(12)	latin1_swedish_ci		No	*None*		
5	age	int			No	*None*		

Figure 11 - Register Table

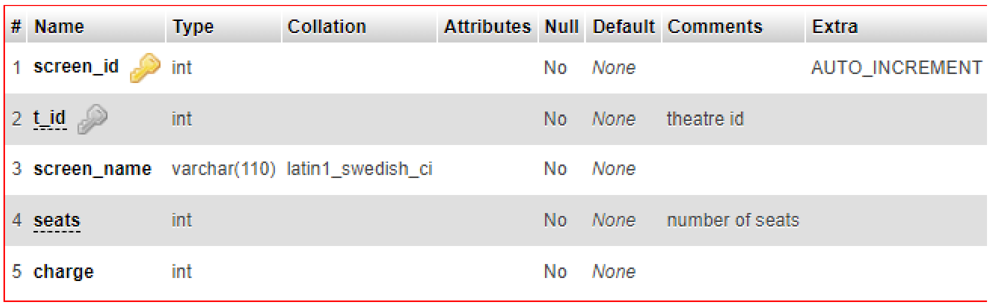
8. Table Name – tbl_screens



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	screen_id 🔑	int			No	*None*		AUTO_INCREMENT
2	t_id 🔑	int			No	*None*	theatre id	
3	screen_name	varchar(110)	latin1_swedish_ci		No	*None*		
4	seats	int			No	*None*	number of seats	
5	charge	int			No	*None*		

Figure 12 - Screen Table

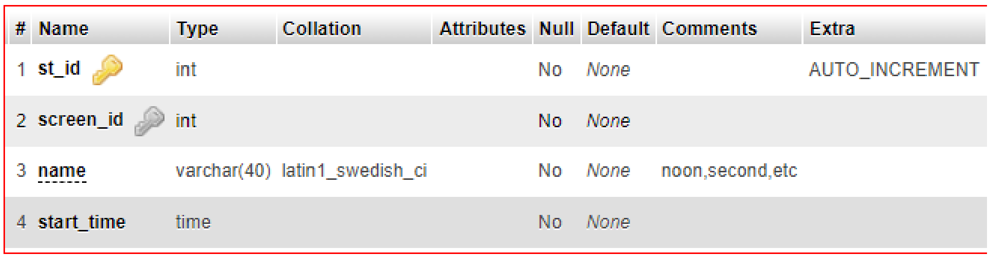
9. Table Name – tbl_shows



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	screen_id 🔑	int			No	*None*		AUTO_INCREMENT
2	t_id 🔑	int			No	*None*	theatre id	
3	screen_name	varchar(110)	latin1_swedish_ci		No	*None*		
4	seats	int			No	*None*	number of seats	
5	charge	int			No	*None*		

Figure 13 - Show Table

10. Table Name – tbl_show_time

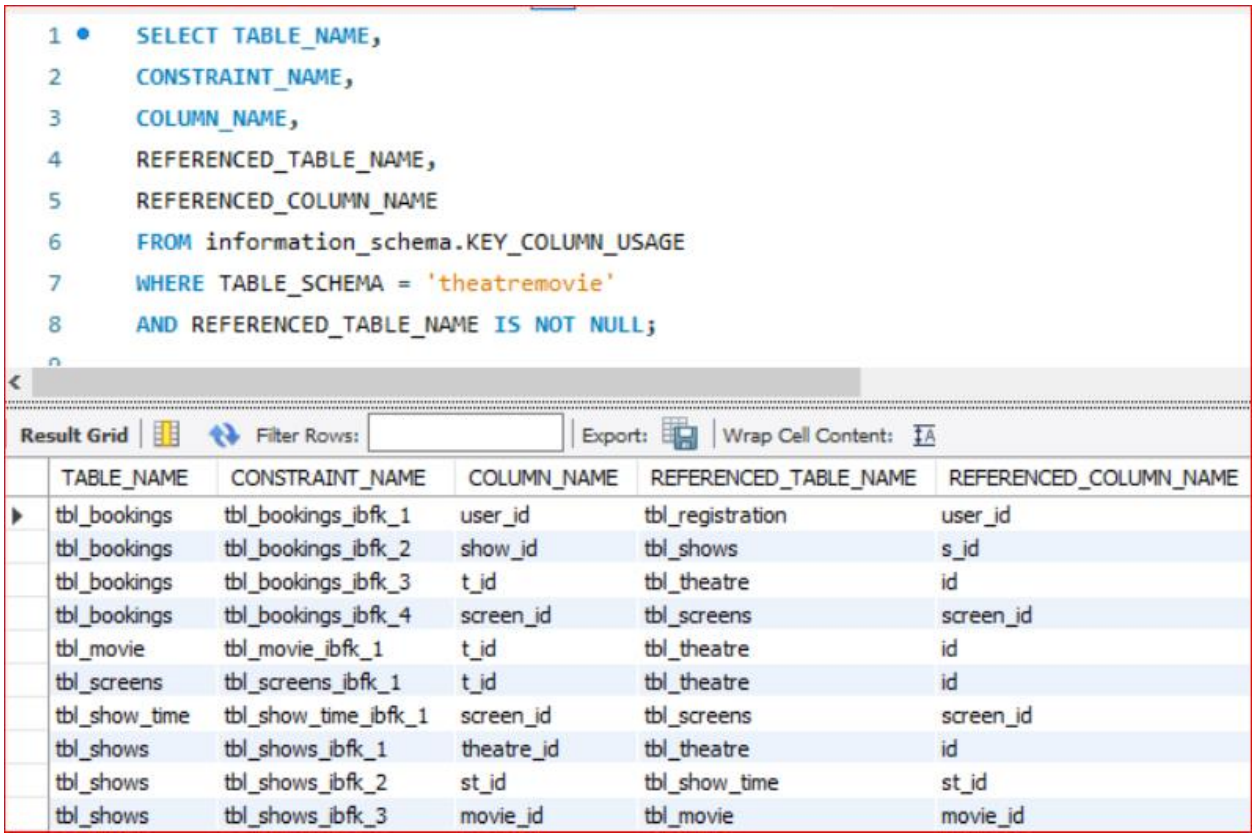


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	st_id 🔑	int			No	*None*		AUTO_INCREMENT
2	screen_id 🔑	int			No	*None*		
3	name	varchar(40)	latin1_swedish_ci		No	*None*	noon,second,etc	
4	start_time	time			No	*None*		

Figure 14 - Show Time Table

5.2 Indexing and Optimization

Below table discuss about the selection of primary and foreign keys for indexing.



```sql
SELECT TABLE_NAME,
CONSTRAINT_NAME,
COLUMN_NAME,
REFERENCED_TABLE_NAME,
REFERENCED_COLUMN_NAME
FROM information_schema.KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = 'theatremovie'
AND REFERENCED_TABLE_NAME IS NOT NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TABLE_NAME	CONSTRAINT_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
tbl_bookings	tbl_bookings_ibfk_1	user_id	tbl_registration	user_id
tbl_bookings	tbl_bookings_ibfk_2	show_id	tbl_shows	s_id
tbl_bookings	tbl_bookings_ibfk_3	t_id	tbl_theatre	id
tbl_bookings	tbl_bookings_ibfk_4	screen_id	tbl_screens	screen_id
tbl_movie	tbl_movie_ibfk_1	t_id	tbl_theatre	id
tbl_screens	tbl_screens_ibfk_1	t_id	tbl_theatre	id
tbl_show_time	tbl_show_time_ibfk_1	screen_id	tbl_screens	screen_id
tbl_shows	tbl_shows_ibfk_1	theatre_id	tbl_theatre	id
tbl_shows	tbl_shows_ibfk_2	st_id	tbl_show_time	st_id
tbl_shows	tbl_shows_ibfk_3	movie_id	tbl_movie	movie_id

5.3 Storage Considerations

Consider storage-related decisions, such as partitioning large tables, selecting appropriate storage engines (InnoDB for ACID compliance), and optimizing space utilization.

6.0 Transaction Analysis

6.1 Transaction Requirements

Define the transactions that must be supported by the database, such as placing orders, making payments, updating inventory, and generating invoices.

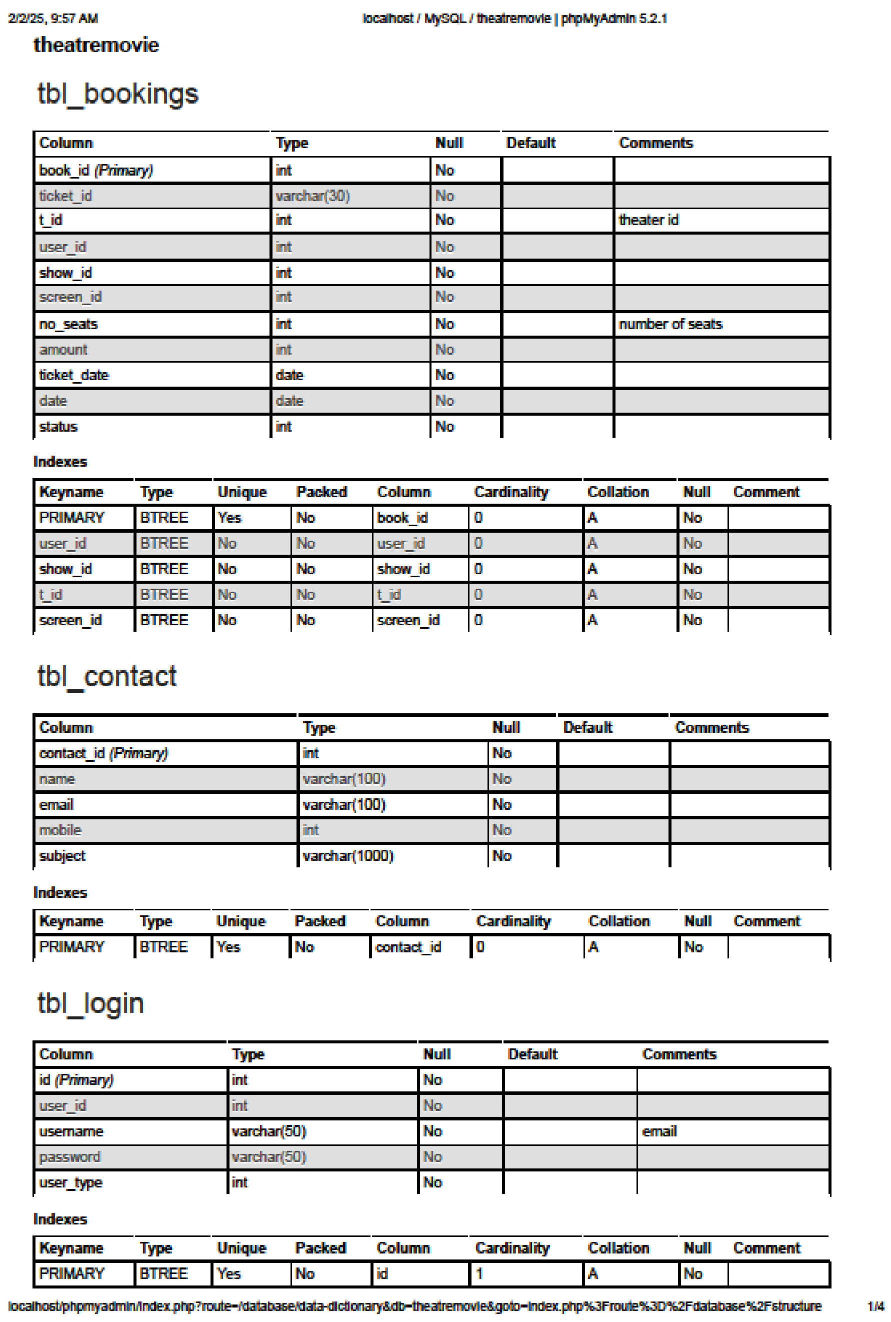
6.2 ACID Properties

Discuss how the database will maintain **ACID** properties (Atomicity, Consistency, Isolation, Durability) during transactions to ensure data integrity.

6.3 Sample Transaction Flow

Walk through a sample transaction, detailing the steps involved (e.g., customer order placement, inventory check, payment processing, and ticket generation).

7.0 Data Dictionary



2/2/25, 9:57 AM localhost / MySQL / theatremovie | phpMyAdmin 5.2.1

theatremovie

tbl_bookings

Column	Type	Null	Default	Comments
book_id *(Primary)*	int	No		
ticket_id	varchar(30)	No		
t_id	int	No		theater id
user_id	int	No		
show_id	int	No		
screen_id	int	No		
no_seats	int	No		number of seats
amount	int	No		
ticket_date	date	No		
date	date	No		
status	int	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	book_id	0	A	No	
user_id	BTREE	No	No	user_id	0	A	No	
show_id	BTREE	No	No	show_id	0	A	No	
t_id	BTREE	No	No	t_id	0	A	No	
screen_id	BTREE	No	No	screen_id	0	A	No	

tbl_contact

Column	Type	Null	Default	Comments
contact_id *(Primary)*	int	No		
name	varchar(100)	No		
email	varchar(100)	No		
mobile	int	No		
subject	varchar(1000)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	contact_id	0	A	No	

tbl_login

Column	Type	Null	Default	Comments
id *(Primary)*	int	No		
user_id	int	No		
username	varchar(50)	No		email
password	varchar(50)	No		
user_type	int	No		

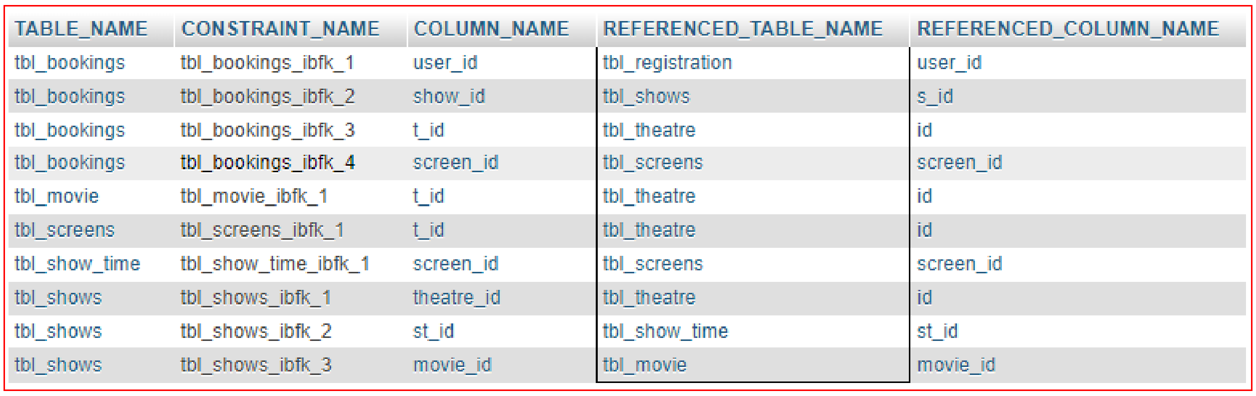
Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	

localhost/phpmyadmin/index.php?route=/database/data-dictionary&db=theatremovie&goto=index.php%3Froute%3D%2Fdatabase%2Fstructure 1/4

Figure 15 - Data Dictionary

Click above the data dictionary. Then we can go to data dictionary pdf.



TABLE_NAME	CONSTRAINT_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
tbl_bookings	tbl_bookings_ibfk_1	user_id	tbl_registration	user_id
tbl_bookings	tbl_bookings_ibfk_2	show_id	tbl_shows	s_id
tbl_bookings	tbl_bookings_ibfk_3	t_id	tbl_theatre	id
tbl_bookings	tbl_bookings_ibfk_4	screen_id	tbl_screens	screen_id
tbl_movie	tbl_movie_ibfk_1	t_id	tbl_theatre	id
tbl_screens	tbl_screens_ibfk_1	t_id	tbl_theatre	id
tbl_show_time	tbl_show_time_ibfk_1	screen_id	tbl_screens	screen_id
tbl_shows	tbl_shows_ibfk_1	theatre_id	tbl_theatre	id
tbl_shows	tbl_shows_ibfk_2	st_id	tbl_show_time	st_id
tbl_shows	tbl_shows_ibfk_3	movie_id	tbl_movie	movie_id

Part 02: Database System Development

8.0 Database Implementation and Testing

8.1 Implementation Strategy

Discuss how the database will be implemented in MySQL, including database creation scripts, table creation, and initial data loading.

8.2 Testing and Validation

9.0 Conclusion

10.1 Summary of Database Design

Summarize the key points of your database design, highlighting the essential tables, relationships, and transaction processes.

10.2 Future Considerations

Discuss potential future enhancements, such as database scaling, additional features, or handling increased data volume.

10.0 References

1. **Behra, H.** (2022). *A Final Year Report Project on Online Movie Ticket Booking System.* Submitted to the Department of Computer Science & IT, Arka Jain University, Jharkhand. Guided by Prof. Akash Kumar Bhagat. Retrieved from https://arkajainuniversity.ac.in/.

11.0 Appendix