



Savitribai Phule Pune University Pune

MINI PROJECT REPORT

ON

Movie Ticket Booking System

SUBMITTED BY

1. Sanskruti Nikam

2. Rushikesh Patil

3. Vishnu Pawar

4. Yash Tekale

Under the Guidance of

Prof S.A. Bhalgonkar



DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION

P.E.S'S MODERN COLLEGE OF ENGINEERING

PUNE – 411005.

Academic Year : 2024-25

CERTIFICATE

This is to certify that

Sanskruti Nikam Exam No. (T1903103125)

Rushikesh Patil Exam No. (T1903103136)

Vishnu Pawar Exam No. (T1903103141)

Yash Tekale Exam No. (T1903103189)

Savitribai Phule Pune University Department of **Electronics and Telecommunication Engineering** This is to certify that the students of **Third Year (T.E.)** Electronics and Telecommunication Engineering have successfully completed a mini-project on the topic titled 'Movie Ticket Booking System' during the academic year **2024-25**. This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering.

Dr. K.R.Joshi
Principal
P.E.S's MCOE, Pune-05.

Dr.Mrs.R.S.Kamthe
H.O.D. E&TC

Prof. S.A. Bhalgaonkar
Project Guide

ACKNOWLEDGEMENT

With deep sense of gratitude, we would like to thank all the people who have lit our path with their kind guidance. We are very grateful to these intellectuals who did their best to help during our project work.

It is our proud privilege to express deep sense of gratitude to **Dr. Prof. (Mrs.) K. R. Joshi**, Principal of PES MCOE, Pune, for her comments and kind permission to complete this project. We remain indebted to **Dr. (Mrs.) R. S. Kamathe**, H.O.D. of Electronics and Telecommunication Department for her timely suggestion and valuable guidance.

The special gratitude goes to **Prof. Seema Bhalgaonkar** excellent and precious guidance in completion of this work. We are also thankful to our parents who providing their wishful support for our project completion successfully. And lastly, we thanks to our all friends and the people who are directly or indirectly related to our project work.

1. Sanskruti Nikam (T1903103125)
2. Rushikesh Patil (T1903103136)
3. Vishnu Pawar (T1903103141)
4. Yash Tekale (T1903103189)

Contents

Sr.no	Topic	Page no.
	Acknowledgement	i
	Contents	ii
	List of Tables/Collections	iii
	List of Forms	iv
	Acronyms	V
	Abstract	vi
Chapter No.1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement	2
1.3	Framework of the proposed work in project	3
Chapter No. 2	Literature Review	4
2.1	Introduction	4
2.2	Existing methodologies	5
2.3	Proposed methodologies	7
Chapter No. 3	Software Requirement Specification	9
Chapter No. 4	Assumptions	12
Chapter No. 5	Entity-Relationship Diagram	13
Chapter No. 6	Normalization	14
Chapter No. 7	Tables / Collections	15
Chapter No. 8	Forms/jFrames(GUI)	16
Chapter No. 9	Features	17
Chapter No. 10	Conclusion	19

List of Tables/Collections

Users

Columns:

- Id (INT, PRIMARY KEY, AUTO_INCREMENT)
- Username (VARCHAR(50), UNIQUE, NOT NULL)
- Password (VARCHAR(50), NOT NULL)
- Phone Number (VARCHAR(15)) - added later with ALTER TABLE

bookings

Columns:

- id (INT, PRIMARY KEY, AUTO_INCREMENT)
- user_id (INT, FOREIGN KEY referencing users(id), NOT NULL)
- movie_name (VARCHAR(100), NOT NULL)
- seats (INT, NOT NULL)
- booking_time (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- priceLabel (DECIMAL(10, 2))
- price (DECIMAL(10, 2)) - added later with ALTER TABLE

List of Forms

- First Normal Form (1NF): Each table (users, bookings) has atomic data (single values per column) and no repeating groups.
- Second Normal Form (2NF): There are no composite keys, and each column in both tables is fully dependent on its primary key (id in both users and bookings).
- Third Normal Form (3NF): No transitive dependencies are present. For example, in the bookings table, all non-key columns (like movie_name, seats, time_slot, and price) depend directly on the id of the booking, not on other non-key columns.

Acronyms

CRUD: Create, Read, Update, Delete (operations in the admin panel)

DB: Database

NF: Normal Form

1NF: First Normal Form

2NF: Second Normal Form

3NF: Third Normal Form

SQL: Structured Query Language

JDBC: Java Database Connectivity

Abstract

The movie booking system is designed to provide users with a seamless platform to browse, select, and reserve seats for movies in their favourite theatres. Users can create an account with a unique username, password, and phone number, which allows them to log in and securely manage their bookings. Once logged in, users can explore available movies, choose specific time slots, and select the number of seats they wish to book. The system calculates the total price based on the number of seats, and the booking is confirmed with a timestamp, ensuring users have a record of their reservations.

The backend structure of the system efficiently manages user data and bookings through relational tables. Each booking is linked to a user account, ensuring personalized experiences. The system is designed with scalability in mind, supporting multiple users and bookings while maintaining data integrity through foreign key relationships. The user-friendly interface simplifies the process of selecting movies and booking seats, making it a reliable and convenient platform for cinema-goers.

1. INTRODUCTION

1.1 Motivation

The motivation behind creating a movie booking system is to simplify and enhance the movie-going experience for both users and cinema operators. In today's fast-paced world, people value convenience and efficiency. A movie booking system allows users to easily browse available movies, choose showtimes, select seats, and purchase tickets all from the comfort of their homes or while on the go. This eliminates the need for waiting in long queues at the theatre and provides a seamless, hassle-free experience.

For cinema operators, the system streamlines ticket sales and seat management, improving operational efficiency. It reduces the chances of errors in bookings, ensures better crowd management, and offers insights through data analytics on movie popularity and user preferences. Additionally, it opens up opportunities for targeted marketing and promotions, helping theatres attract more customers. Overall, the movie booking system meets the demand for a user-friendly, accessible solution that enhances the movie-watching experience while optimizing cinema operation.

1.1 Problem Statement

The central problem that a movie booking system tries to solve is the hassle and inefficiency of the old ways to purchase movie tickets. People, in general, had to visit cinemas, stand in long lines in hope not to miss buying the tickets when they are all sold out already. It took such a long time, was frustrating and uncertain, especially during busy movie releases and at weekends.

From the cinema operator's viewpoint, managing high numbers, seat reservations, and dealing with last-minute cancellations or alterations were operational issues. Without a central system, monitoring how many seats are available and how many tickets have sold may result in incorrect info, overcrowding or underutilization of shows.

A movie booking system solves these problems by giving users a website where they can easily see showtimes, select their seats, and pay for tickets safely in advance. For cinemas, the system helps with organization, manages seats better, and gives real-time information for smarter choices. This solution makes the whole process easier, helping both movie fans and theater operator

1.3 Framework of the proposed work in project

The movie booking system framework includes the design and creation of a database to manage users and their movie bookings. Below is the step-by-step explanation of how the system is structured and works:

1. Database Setup

- The first step is to create a **database** named movie booking. This serves as the central repository for all data related to users and movie bookings. By dropping the database if it exists and creating it anew, we ensure a fresh start every time we run the system.

2. Users Table

A **users table** is created to store details about the users who will book movie tickets. The table contains:

- id: A unique identifier for each user, automatically generated.
- username: A unique username chosen by the user, which is mandatory.
- password: A password for user authentication, which is also mandatory.
- phone number : An optional field to store the user's contact number for communication purposes.

3. Bookings Table

A **bookings table** is created to manage the movie booking details. This table stores:

- id: A unique identifier for each booking.
- User id: A reference to the id of the user making the booking, establishing a relationship with the users table.
- Movie name: The name of the movie being booked.
- seats: The number of seats the user has booked.
- Booking time: The time when the booking was made, automatically set to the current timestamp.
- Time slot: The time when the movie is scheduled to be shown, which is provided during the booking

4. The **foreign key constraint** on user id ensures that each booking is linked to a valid user. The ON DELETE CASCADE option ensures that if a user is deleted, all their associated bookings will be

5. Querying the Data

- A query is provided to retrieve booking information along with user details. This query:
 - Joins the users and bookings tables using the user id to show information like the user's username, movie name, number of seats, booking time, movie time slot, and total price.

6.Booking Management

- After the system retrieves or processes the bookings, there is an option to **drop the bookings table** if needed, removing all booking data from the system. This is used for resetting or cleaning up the booking records.

7. System Queries

- The system can execute a query to **view all bookings** and their details at any time by selecting from the bookings table. This allows administrators or users to see all active bookings.

2.LITERATURE REVIEW

2.1 Introduction

The movie booking system introduction identifies the need for a modern and efficient means through which people can get their favorite films' tickets. In an economy with emphasis on convenience and time, older forms of movie ticket booking such as waiting in long lines at the cinema become frustratingly time-consuming. As entertainment evolves, movie viewers expect a more streamlined experience to plan outings with ease.

This project aims to develop an online movie booking system that not only meets these expectations but also enhances the overall movie-going experience. By leveraging technology, this system will enable users to browse current and upcoming movies, select their preferred showtimes, and choose seats—all from the comfort of their homes or on the go. Being able to book tickets early will remove the worry of sold-out shows and give users quick confirmation of their purchases.

This movie booking system will be a very significant tool for cinema operators to handle ticket sales and customer interactions. It assists theatres to do bookings adequately, check seat availability, and view sales data to improve their services overall. In this regard, this project is an important step in changing the way people enjoy movies and makes it easier, faster, and fun for everyone involve

2.2 Existing Methodologies

1. Database Management:

- The code starts by ensuring that the movie_booking database does not exist. If it does, it drops (deletes) it.
- It then creates a new movie_booking database and sets it as the current database for further operations.

2. Creating Tables:

users Table This table is created to store user information. It includes:

- id: A unique identifier for each user (auto-incremented).
- username: The user's name, which must be unique and cannot be null.
- password: The user's password, which cannot be null.
- phone_number: An optional field to store the user's phone number.

3. bookings Table:

This table is designed to record bookings made by users. It includes:

- id: A unique identifier for each booking (auto-incremented).
- user_id: A foreign key that links each booking to a user, ensuring that only existing users can make bookings.
- movie_name: The name of the movie being booked.
- seats: The number of seats booked.
- booking_time: The timestamp indicating when the booking was made, defaulting to the current time.
- time_slot: A new column to specify the time of the movie.
- price: A column to store the total cost of the booking.

4. The FOREIGN KEY constraint ensures referential integrity, meaning that a booking must be associated with a valid user. The ON DELETE CASCADE option means that if a user is deleted, their associated bookings will also be deleted automatically.

5. Fetching Data:

A query is then executed to retrieve booking details along with user information:

- It joins the users table with the bookings table, matching users with their bookings based on the user ID.
- The query fetches various details such as the user's ID, username, password, booking ID, movie name, number of seats, booking time, movie time slot, and price.

6. Dropping the Bookings Table:

The `DROP TABLE IF EXISTS bookings;` command is executed, which deletes the bookings table if it exists. This step will result in the loss of all booking data.

7. Selecting from the Bookings Table:

The last command attempts to select all data from the bookings table (`SELECT * FROM bookings;`), but this will fail if the previous drop command has executed, as the table will no longer exist

2.3 Proposed Methodologies

1. Database Creation:

It also drops the database called movie_booking if it already exists for a fresh start and then creates the same.

2. Switching to the New Database

The use of the statement `movie_booking;` modifies the existing database to movie_booking.

3. Creating the users Table:

A table named users is created to store information about users. It consists of these columns:

- 3.1.** ID: Unique number of each user which increases automatically when a new user is created.
- 3.2.** username: A unique username that should be provided.
- 3.3.** password: Each user had a password
- 3.4.** .phone_number: an optional phone number.

4. Making the book table:

The table of bookings contains information regarding movie bookings. Its columns include:

- 4.1.** id: unique identifier for each booking.
- 4.2.** user_id: This column refers to the ID of any user in the users table. It ensures each reservation is associated with a particular user. If a user is deleted, the database automatically deletes all the reservations related to that user (ON DELETE CASCADE).
- 4.3.** movie_name: name of the movie that will be rented.

5. Available Seats: The number of seats reserved.

- 5.1.** booking_time: This is the time when the booking takes place, which as usual is the current time.
- 5.2.** time slot: The specific time slot of the movie (5:00 PM, 8:00 PM, etc.).
- 5.3.** Price : The entire cost of booking is stored in two decimal points for the precision.

Requesting booking information with user details A question is posed to accept all possible detail booking information and related user data.

It receive User information like user identification, username, and password. Booking details shall comprise of the name of the movie, number of seats, time, slot, and total amount of money. The INNER JOIN ensures that only records of users who have made bookings will be fetched. Delete and Select from the bookings Table: If it exists, the bookings table will then be removed and a query will attempt to select from the now nonexistent bookings table, which would result in an error since the table no longer exists.

3. Software Requirement Specification

1. Software Requirements

Operating System:

Windows 10 and above, Linux, or macOS

2. Dev Environment:

IntelliJ IDEA for Java development

3. Programming Language:

Java (for backend logic and UI implementation)

4. Database:

MySQL (to store and retrieve information on movies, booking record, users, etc.)

5. Database connector

JDBC or Java Database Connectivity is used to connect a Java application with a MySQL database.

6. Framework/Libraries:

Java Swing: Used to build the UI-part consisting of sign-in, booking forms, etc.

JDBC Connector: Necessary for communication between Java and MySQL.

7. Other Software:

MySQL Workbench: It's for managing databases and running queries.

Functional Requirements

User Authentication:

Users should be able to log into and manage their own reservations.

8. **Movie Choices:** Users should be able to view available movies and select showtimes. Booking Management: Users can choose seats and confirm their orders. There is a requirement for real-time updating of the seat availability. Database Operations: The MySQL database must hold the information for user detail, movie information, and booking information.

ASSUMPTION

In a movie booking system, here are the common assumptions described in simple terms:

1. Users

- There are different types of users, such as:
 - Customers: They book movie tickets.
 - Admins: They manage movies, schedules, and bookings.

2. Movies

- A movie has basic details like its title, genre, duration, director, cast, and release date.
- Movies are shown in different theatre at specific times (called showtimes).

3. Theatre and Screens

- A theatre is a place with multiple screens or halls where movies are played.
- Each screen can show one movie at a time.
- A screen has a specific seating capacity, meaning a fixed number of seats.

4. Showtimes

- Movies are shown at specific times during the day (e.g., 3:00 PM, 6:00 PM).
- Users can choose a movie based on available showtimes.

5. Seats

- Each screen has rows and seats that can be booked.
- Users can see which seats are available and select the ones they want.

6. Booking Process

- Customers choose a movie, showtime, and seats.
- They must provide payment to confirm their booking.
- After payment, customers receive a booking confirmation, usually with a ticket number or QR code.

7. Cancellation and Refunds

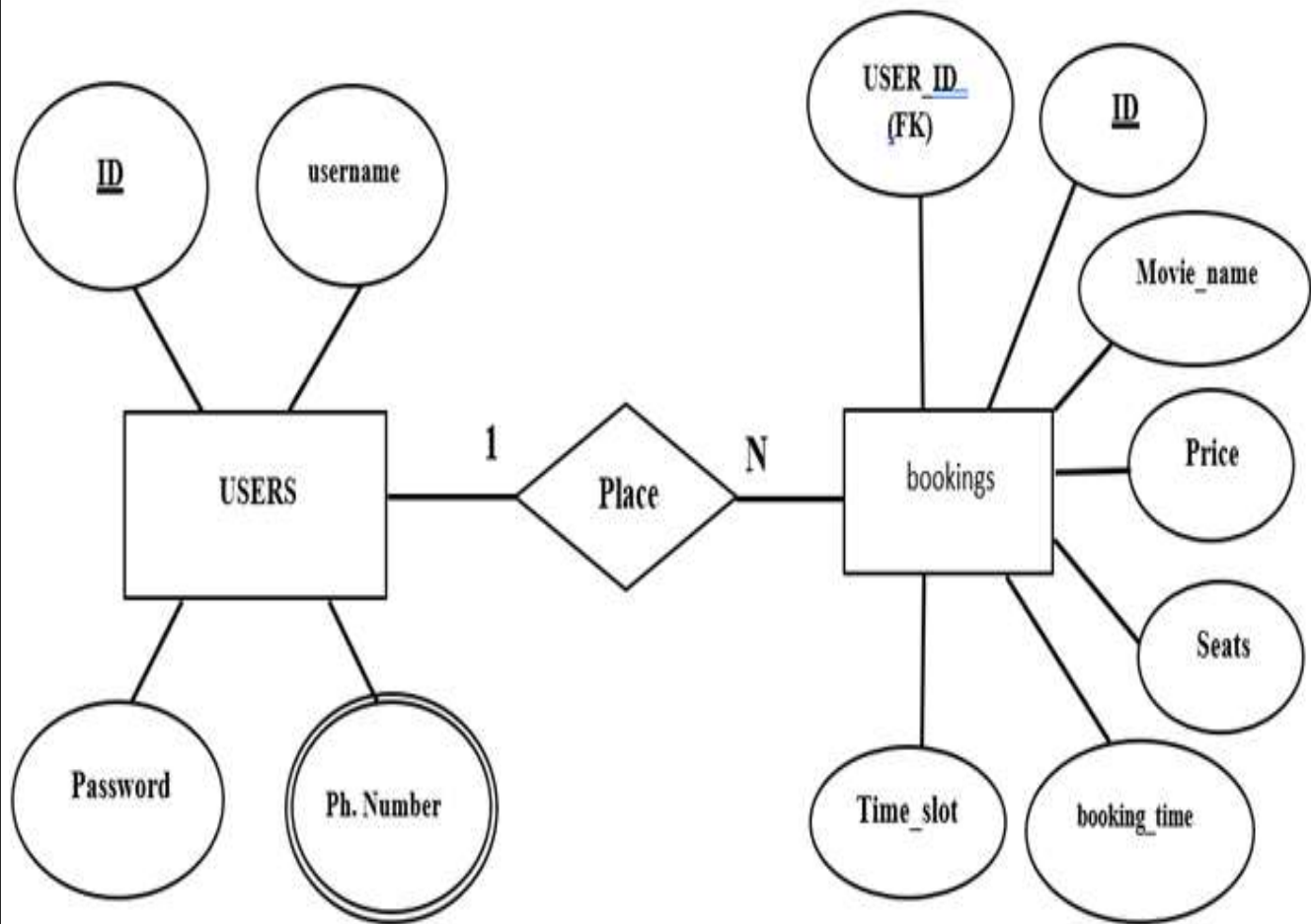
- Users can cancel bookings within a certain time frame.

8. Admin Management

- Admins can add new movies, update schedules, and manage user bookings.
- They can also generate reports like total bookings, revenue, and occupancy rates for each show.

5.Entity-Relationship Diagram

- Step 1 : Define Entities: user and booking
- Step 2: Adding Attributes For the User Entity, the Attributes are id, username, password, and phone number. For the Booking Entity, the Attributes are id, user id, movie name, seats, booking time, time slot, and price.
- Step 3: The user books a movie. The booking provides information like the name of the movie, seat number, time slot, and price.
- Step 4: Specify Cardinality. User place booking has one to many relationship
- Step 5: KEYS The primary key of the user is id. The primary key of booking is id. The column user_id within the bookings table is a foreign key referencing users(id).
- password, and phone number. For the Booking Entity, the Attributes are id, user id, movie name, seats, booking time, time slot, and price.
- Step 3: The user books a movie. The booking provides information like the name of the movie, seat number, time slot, and price.
- Step 4: Specify Cardinality. User place booking has one to many relationship
- Step 5: KEYS The primary key of the user is id. The primary key of booking is id. The column user_id within the bookings table is a foreign key referencing users(id).



6. Normalization

To achieve normalization in your database design, let's walk through the process of how we can apply the three common normal forms (1NF, 2NF, and 3NF) to your movie_booking schema.

- **First Normal Form (1NF)**

1NF requires that each column contains atomic values, and each row is unique. Your tables already satisfy 1NF, as each attribute contains a single value (e.g., one username, one movie_name, etc.), and each row has a unique identifier (id).

- **Second Normal Form (2NF)**

To meet 2NF, the table must already meet 1NF, and all non-key columns must depend on the whole primary key. Since your tables either have a simple primary key (id in both users and bookings), this is automatically satisfied. Each non-key column (e.g., username, movie_name) fully depends on the primary key.

- **Third Normal Form (3NF)**

3NF requires that no non-key column depends on another non-key column, i.e., there should be no transitive dependencies. In your case, none of the non-key columns (like movie_name, seats, priceLabel) depend on another non-key column, so your tables already comply with 3NF.

7.Tables /Collections

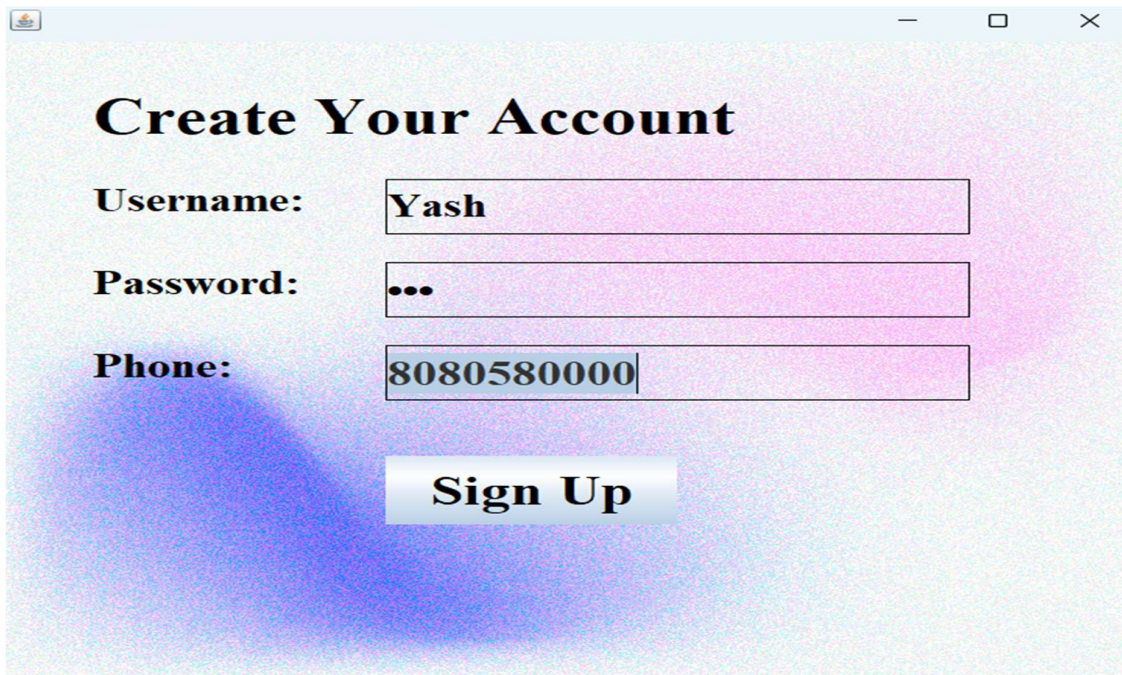
Show Structure of users Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(50)	NO	UNI	NULL	
password	varchar(50)	NO		NULL	

Show Structure of bookings Table:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO	MUL	NULL	
movie_name	varchar(100)	NO		NULL	
seats	int(11)	NO		NULL	
booking_time	timestamp	YES		CURRENT_TIMESTAMP	
priceLabel	decimal(10,2)	YES		NULL	

8.Forms/jFrames(GUI)



A screenshot of a Java Swing window titled "Create Your Account". The window has a light blue title bar with standard minimize, maximize, and close buttons. The background is a light blue gradient. The title "Create Your Account" is displayed in a large, bold, black serif font. Below the title, there are three labels: "Username:", "Password:", and "Phone:". Each label is followed by a text input field. The "Username" field contains the text "Yash". The "Password" field contains three black dots. The "Phone" field contains the text "8080580000". Below the input fields, there is a single button labeled "Sign Up" in a bold, black serif font. The button has a light blue gradient and a slight shadow.

Create Your Account

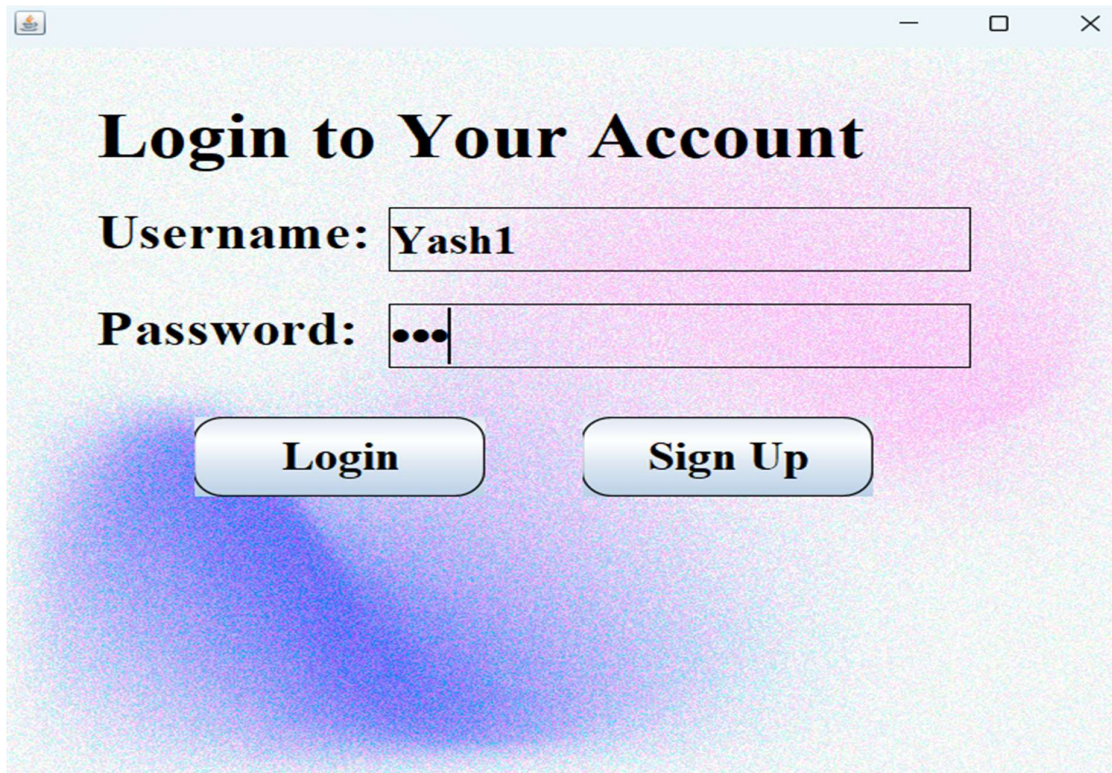
Username:

Password:

Phone:

Sign Up

Login Page:-



A screenshot of a Java Swing window titled "Login to Your Account". The window has a light blue title bar with standard minimize, maximize, and close buttons. The background is a light blue gradient. The title "Login to Your Account" is displayed in a large, bold, black serif font. Below the title, there are two labels: "Username:" and "Password:". Each label is followed by a text input field. The "Username" field contains the text "Yash1". The "Password" field contains three black dots. Below the input fields, there are two buttons: "Login" and "Sign Up". Both buttons are in a bold, black serif font and have a light blue gradient with a slight shadow. The "Sign Up" button is positioned to the right of the "Login" button.

Login to Your Account

Username:

Password:

Login **Sign Up**

Movie Ticket Booking Page :-

Movie Ticket Booking

Movie Ticket Booking

Select Movie:

Stree 2

Select Time Slot:

3:00 PM

Number of Seats:

2

Price: ₹120.00

Book Ticket

Display Info :-

Username	Phone Number	Movie Name	Seats	Booking Time	Time Slot	Price
Yash1	8080580000	Stree 2	2	2024-10-23 00:17:46.0	3:00 PM	₹400.00
Cancel Ticket						

Cancel Ticket :-

User Bookings

Username	Phone Number	Movie Name	Seats	Booking Time	Time Slot	Price
<div><div>Success</div><div><div>i</div><div>Ticket cancelled successfully</div><div>OK</div></div></div>						

9.Features

1.User Accounts:

- Sign Up: People can create accounts with a username and password.
- Log In: Once registered, users can log in with their credentials.

2. Movie Booking:

- Search for Movies: Users can look for movies by title or browse by category.
- Book a Movie: After finding a movie, users can select the number of seats they want and make a booking.
- See Showtimes: Showtimes for each movie will be displayed so users can choose when to watch.
- Seat Selection: Users can pick their preferred seats when booking tickets (if you choose to add this).

3. Security:

- Safe Passwords: User passwords are stored securely using encryption.
- Secure Login: Sessions and other security measures make sure only authorized people can log in and access data.
- Prevent Attacks: Measures like protecting against harmful attacks on the system, such as someone trying to guess passwords many times.

4. Movie Management (For Admins):

- Add New Movies: Admins can add new movies to the system with details like title, description, and showtimes.
- Update Movies: If movie information changes (like a release date or showtime), admins can update it.
- Delete Movies: Admins can also remove movies that are no longer available.

5. Booking Management:

- View Bookings: Users can see a list of their bookings, including upcoming and past reservations.
- Booking Confirmation: Users will receive an email or message confirming their booking after they reserve seats.
- Booking History: Users can see a history of all their bookings in their profile.
- Cancel Booking: If someone changes their mind, they can cancel a booking within certain limits.
- Discounts and Coupons: Users can enter discount codes during checkout to get a lower price on tickets.

8. Reports and Insights (Admin):

- Sales Reports: Admins can view reports on movie sales, such as how many tickets were sold and how much revenue was earned.
- User Analytics: Admins can see data about how often users book tickets and what movies are most popular.
- Movie Performance: Admins can track which movies are doing well and which aren't.

9. User-Friendly Design:

- Mobile-Friendly: The system will be designed to work smoothly on mobile phones and tablets.
- Social Logins: Users can sign up or log in using their social media accounts like Google or Facebook.

10. Conclusion

In conclusion, the movie booking system is designed to make the process of finding and reserving movie tickets simple and hassle-free for users. With a smooth user experience, people can easily sign up, log in, browse through movies, pick their preferred showtimes, and book tickets securely. Features like secure payments, seat selection, booking confirmations, and notifications make it a complete solution for anyone wanting a convenient way to plan their movie outings. The system also ensures user data protection and offers perks like discounts and loyalty points to enhance the user experience.

On the admin side, the system provides a range of tools to efficiently manage movies, showtimes, users, and bookings. Admins can add or update movie details, handle bookings, track sales, and generate reports on user activity and movie performance. This makes it easy to maintain the system and monitor its success. Overall, the system not only makes movie booking easier for users but also gives admins the power to manage the entire operation smoothly.

11.References

SQL References

1. MySQL Documentation

- Link: [MySQL Reference Manual](#)
- The official MySQL documentation provides in-depth information on all MySQL features, including data types, creating tables, and defining relationships between tables (like foreign keys). It's a comprehensive guide for anyone working with MySQL.

Java References

2. Oracle Java Documentation

- Link: [Java SE Documentation](#)
- This is the official documentation for Java SE, covering all classes and methods available in the Java Standard Edition. It's essential for any Java developer, providing explanations and examples for all Java functionalities.

3. Java Database Connectivity (JDBC) Tutorial

- Link: [JDBC Tutorial](#)
- This tutorial explains how to connect Java applications to a database using JDBC. It covers how to perform CRUD (Create, Read, Update, Delete) operations with databases, which would be relevant for your movie booking system.

4. Baeldung: Introduction to JDBC

- Link: [Introduction to JDBC](#)
- This article provides an overview of JDBC and how to use it effectively in Java applications. It includes code examples that can help you implement database interactions in your project.