

ANUDIP FOUNDATION

Project Report

Project Title
“Online Movie Booking System”

BY

Name	Student Code
Jyoti Lohkare	AF0481897

Under Guidance
Of
Rajshri Thete

Index

Sr.no	Topic	Page no
1	Title of Research	1
2	Acknowledgement	3
3	Abstract	4
4	Introduction	5
5	System Analysis	6-9
6	System Design & Diagram	10-19
7	DATA DICTIONARY	20-32
8	TEST PROCEDURE & IMPLEMENTATION	33-36
9	Results and Observations	37
10	Screenshots	38-41
11	Future scope	42
12	conclusion	43
13	Biography	44
14	References	45

Acknowledgement

The project “**Online Movie Booking System**” is the Project work carried out by

Name	Student Code
Jyoti Lohkare	AF0481897

Under the Guidance. We are thankful to my project guide for guiding me to complete the Project. His suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics. We are also thankful to my family member and friends who were always there to provide support and moral boost up.

Abstract

The **Online Movie Ticket Booking System** is a web-based application designed to simplify and automate the process of booking movie tickets for users and managing show schedules for administrators. This platform provides a convenient and efficient solution that allows users to browse currently available movies, view showtimes, book tickets, and receive confirmation instantly from any location.

For administrators, the system provides powerful tools to manage movie listings, theater screens, show timings, and booking records through a centralized backend interface. Features such as seat availability, real-time booking updates, and automatic ticket generation reduce the need for manual interventions and streamline cinema operations.

The system is built using **Python for backend logic**, **MySQL for database management**, and **HTML/CSS/JavaScript** for the frontend interface. It is designed to be responsive and user-friendly, supporting accessibility across desktops, laptops, and mobile devices. The project also implements essential security features like user authentication and data validation to ensure safe and reliable transactions.

By automating the traditional ticket booking process, this system enhances user experience, minimizes human error, and increases the efficiency of theater operations. It is scalable and customizable, making it suitable for both single-screen cinemas and multiplexes.

Introduction

Movie Ticket Booking is an innovative web-based platform designed to facilitate the exchange of second-hand books among readers, students, and book enthusiasts. The primary goal of this project is to promote the sustainable use of educational and literary resources by providing a user-friendly interface where individuals can list, search, and exchange books effortlessly.

The platform allows users to register and create profiles, where they can upload details about books they wish to exchange. Each book listing includes essential information such as the title, author, category, condition, and preferred exchange type. Users can browse available books, filter them by categories, and send exchange requests directly to other users. Movie Ticket Booking also incorporates a secure authentication system, ensuring that user data and transactions are protected. Administrators have control over the platform through a dedicated admin panel, where they can manage user accounts, monitor book listings, and oversee exchange transactions to maintain a trustworthy environment. The platform's notification system keeps users informed about the status of their exchange requests, enhancing communication and user engagement.

Movie Ticket Booking is built using Java Spring Boot for the backend, providing a robust and scalable architecture, while the frontend employs modern web technologies to offer a smooth and responsive user experience. This project not only encourages reading and sharing knowledge but also helps reduce waste by giving books a second life through exchange rather than disposal.

Movie Ticket Booking solves a real-world problem by addressing the lack of access to affordable books—not only for students, but also for teachers, professionals, competitive exam aspirants, and book lovers from all backgrounds. New books are often expensive or unavailable in local markets. Through a simple and accessible online platform, Movie Ticket Booking allows users to connect, share, and receive books without financial stress. This promotes literacy, learning, and sustainability across all segments of society.

This project, Movie Ticket Booking, is developed by me as a practical and socially impactful solution using modern technologies to make a meaningful contribution to the community and environment.

3. System Analysis

The Movie Ticket Booking system is designed as a web-based platform to streamline the process of second-hand book exchange among users. It follows a user-centric approach, offering functionalities such as registration, book listing, exchange request handling, Likes creation, and notification management. The backend, built with Java Spring Boot, ensures a secure, scalable environment, while the frontend developed using Angular delivers a responsive and interactive user experience. Admin functionalities include user management, listing moderation, and request oversight. The system is developed to address the real-world challenges of book reuse, promoting sustainability and community-driven resource sharing.

2.1 COMPARATIVE STUDY OF EXISTING SYSTEMS

- **Traditional Cinema Booking Counters**

1. **Manual Ticket Booking** – Requires physical presence, which wastes time.
2. **Limited Availability** – Users may not know if seats are full until they reach the counter.
3. **Human Errors** – Mistakes in recording seat numbers and issuing tickets are common.
4. **Paper Tickets** – Easily lost or damaged; no digital backup.
5. **Fixed Timings** – Booking only possible during theater hours.

- **Phone-based Booking**

1. **No Visual Seat Selection** – Users can't choose exact seats.
2. **Busy Lines** – Call congestion during peak times.
3. **No Real-Time Confirmation** – Often leads to confusion or overbooking.

- **Third-Party Apps (BookMyShow, Paytm)**

1. **Convenient but Expensive** – Includes service charges and taxes.
2. **Not Suitable for Small Theaters** – Complex onboarding and maintenance for single screens or local cinemas.
3. **Limited Control** – Admins have less flexibility in show management.

2.2 SCOPE AND LIMITATIONS OF EXISTING SYSTEMS

▪ Scope of Existing Systems

1. Allow users to book tickets via apps or counters.
2. Provide basic user authentication and booking details.
3. Display available shows and theaters in major cities.
4. Send confirmation via email/SMS.

▪ Limitations of Existing Systems

1. No control for local cinemas over customization.
 2. No offline sync or manual override if internet fails.
 3. High transaction fees on third-party apps.
 4. Dependence on external support for system updates.
 5. Not always user-friendly for older or rural users.
-

2.3 PROJECT PERSPECTIVE AND FEATURES

Movie Ticket Booking is a modern, web-based platform envisioned to bridge the gap between book owners and seekers through a seamless second-hand book exchange system. Unlike traditional second-hand book stores or libraries, this platform enables peer-to-peer interaction, real-time book listings, and a transparent exchange mechanism accessible from anywhere. The platform is designed to be scalable, user-friendly, and secure, with the goal of promoting book reuse, sustainability, and a culture of knowledge sharing across all age groups and communities.

▪ Software & Hardware Requirements

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Python
- **Database:** MySQL
- **Server:** Localhost or cloud-hosted (XAMPP, Render, etc.)

▪ **Key Functional Features**

1. **User Registration/Login** – Secure login using email and password.
 2. **Movie Browsing** – View currently running movies with posters and details.
 3. **Seat Selection** – Visual seat layout for real-time selection.
 4. **Show Timing Management** – Admin can add or remove show slots.
 5. **Ticket Booking** – User can confirm and get ticket ID.
 6. **Admin Dashboard** – Manage movies, shows, bookings, and users.
 7. **Search & Filter** – Search by movie name, date, or language.
 8. **Booking Receipt** – Auto-generated ticket with QR or booking ID.
 9. **Cancellation Option** – Allow cancellation within a specific window.
 10. **Scalable UI** – Responsive for mobiles and desktops.
-

2.4 STAKEHOLDERS

1. **End Users (Moviegoers)**
 - Can register/login, view movies, and book tickets.
 - Benefits from time savings and convenience.
2. **Admins (Theater Managers)**
 - Manage shows, seat plans, movie lists, and bookings.
 - Monitor system operations through dashboards.
3. **Developers**
 - Build, test, and maintain the platform.
 - Ensure system stability, updates, and UI improvements.
4. **Anudip Foundation**
 - Provided mentorship, infrastructure, and guidance.
 - Supported the project through real-time use case development.

2.5 REQUIREMENT ANALYSIS

▪ Functional Requirements

1. User registration, login/logout system
2. Movie listing and show scheduling
3. Seat layout and availability management
4. Booking confirmation and ticket generation
5. Admin panel with CRUD operations
6. Booking history and status tracking
7. Role-based access (User/Admin)

▪ Performance Requirements

1. Response time for any request: within 2–3 seconds
2. System should handle 100+ concurrent users
3. Real-time booking updates to avoid conflicts
4. Load-tested for multiple theaters/shows

▪ Security Requirements

1. Secure password storage (hashed)
2. SQL Injection and XSS prevention
3. Role-based authentication
4. Input validation across all forms
5. HTTPS integration for secure communication (if deployed online)

4. System Design

4.1 DESIGN CONSTRAINTS

1. Platform Dependency

The system is designed for modern web browsers and may not function properly on outdated versions or unsupported platforms.

2. Internet Requirement

Users need a stable internet connection to access features like login, search, book listings, and exchange requests.

3. Physical Book Exchange Required

While the platform facilitates matching and communication, users must **physically meet** to exchange books, which may limit usability across distant locations.

4. Database Structure Limitations

The relational database used may limit flexibility when scaling or adapting to non-relational data in the future.

5. Single Language Interface

The platform supports only English, which may restrict access for users who are not proficient in the language.

6. Basic Identity Verification

There is no strong verification process (e.g., government ID), which may impact user trust and platform credibility.

7. Geographical Constraints

Initially intended for specific regions or communities; national or global scaling requires major updates and logistics handling.

8. Manual Approval in Exchanges

All exchange approvals, cancellations, or decisions are manually handled by users; no automation or AI matching is integrated.

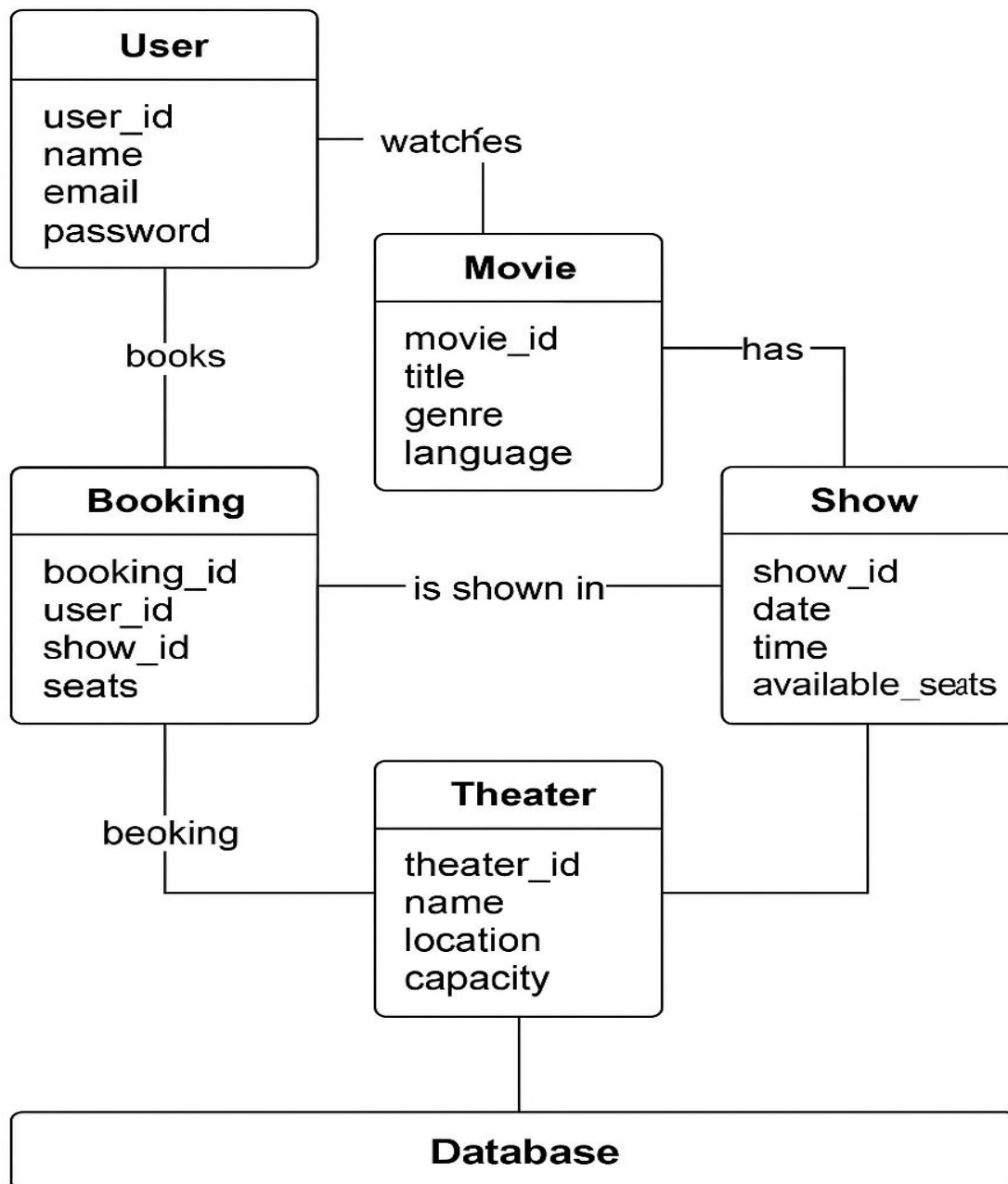
9. Limited Real-Time Features

Real-time interactions (like chat or notifications) may rely on periodic polling rather than instant communication.

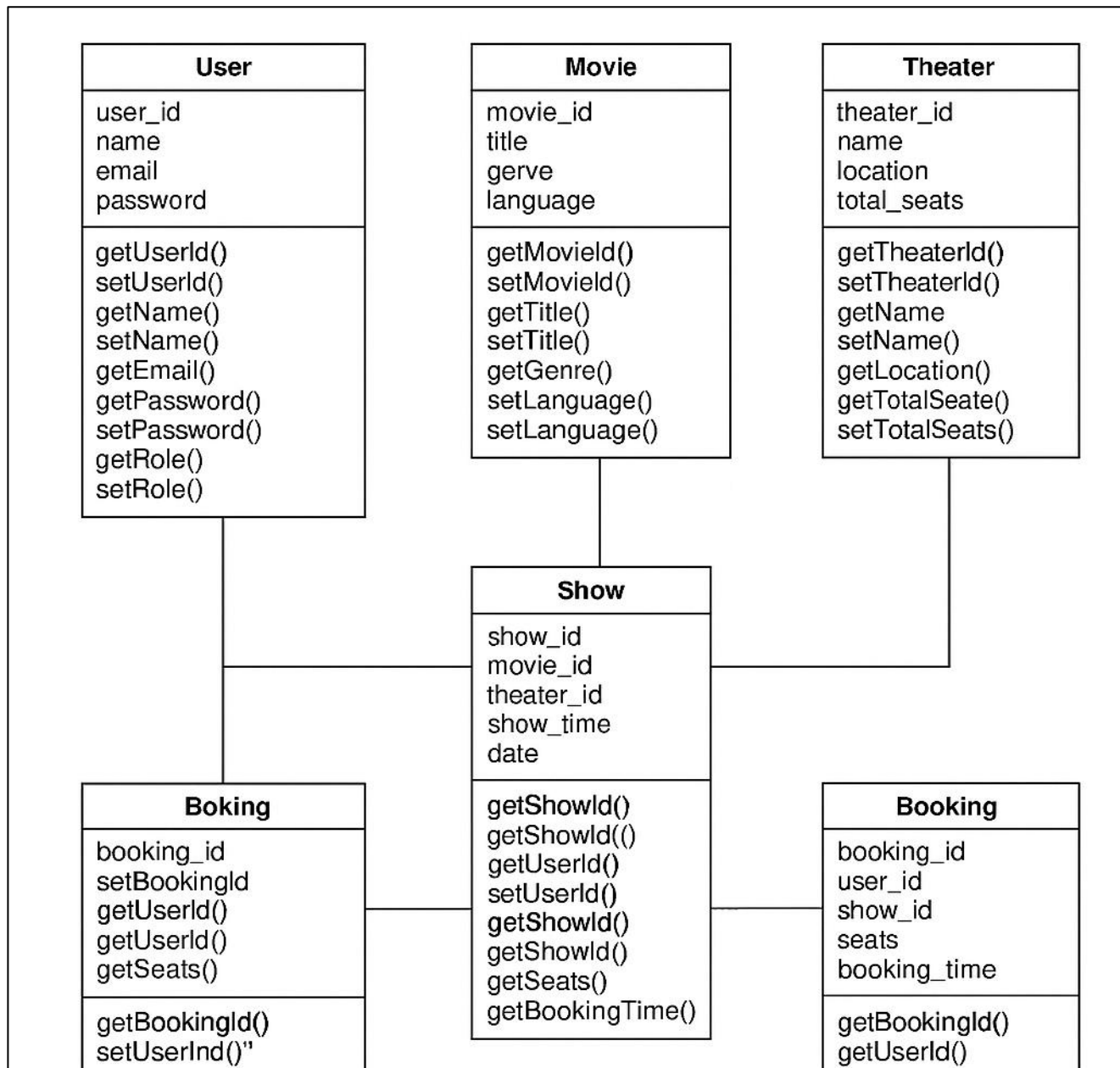
10. Hosting Constraints

Hosting on limited cloud/shared servers may restrict performance, especially during peak traffic or data-intensive operations.

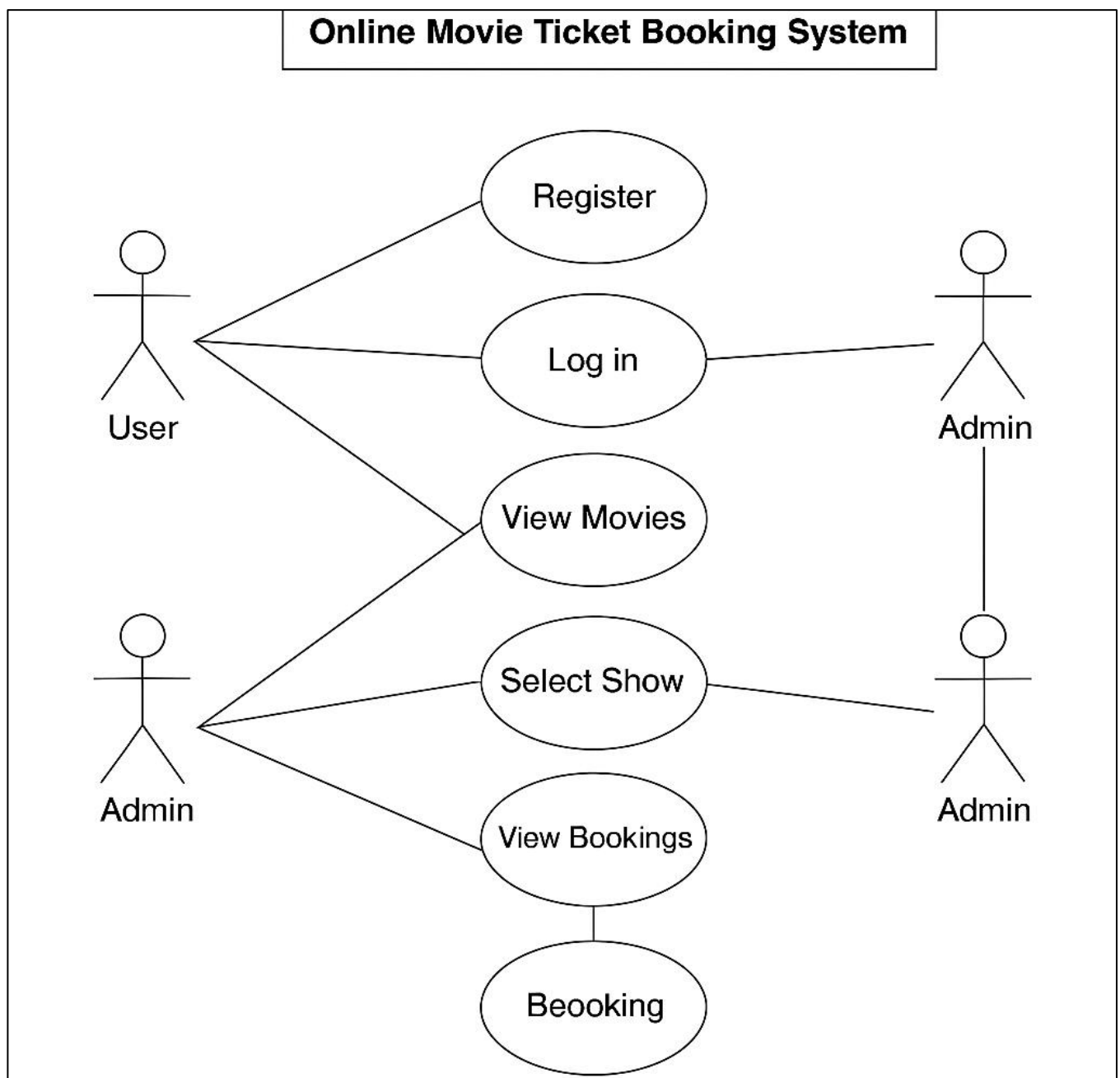
4.2 ENTITY-RELATIONSHIP DIAGRAM:



4.3 CLASS DIAGRAM :

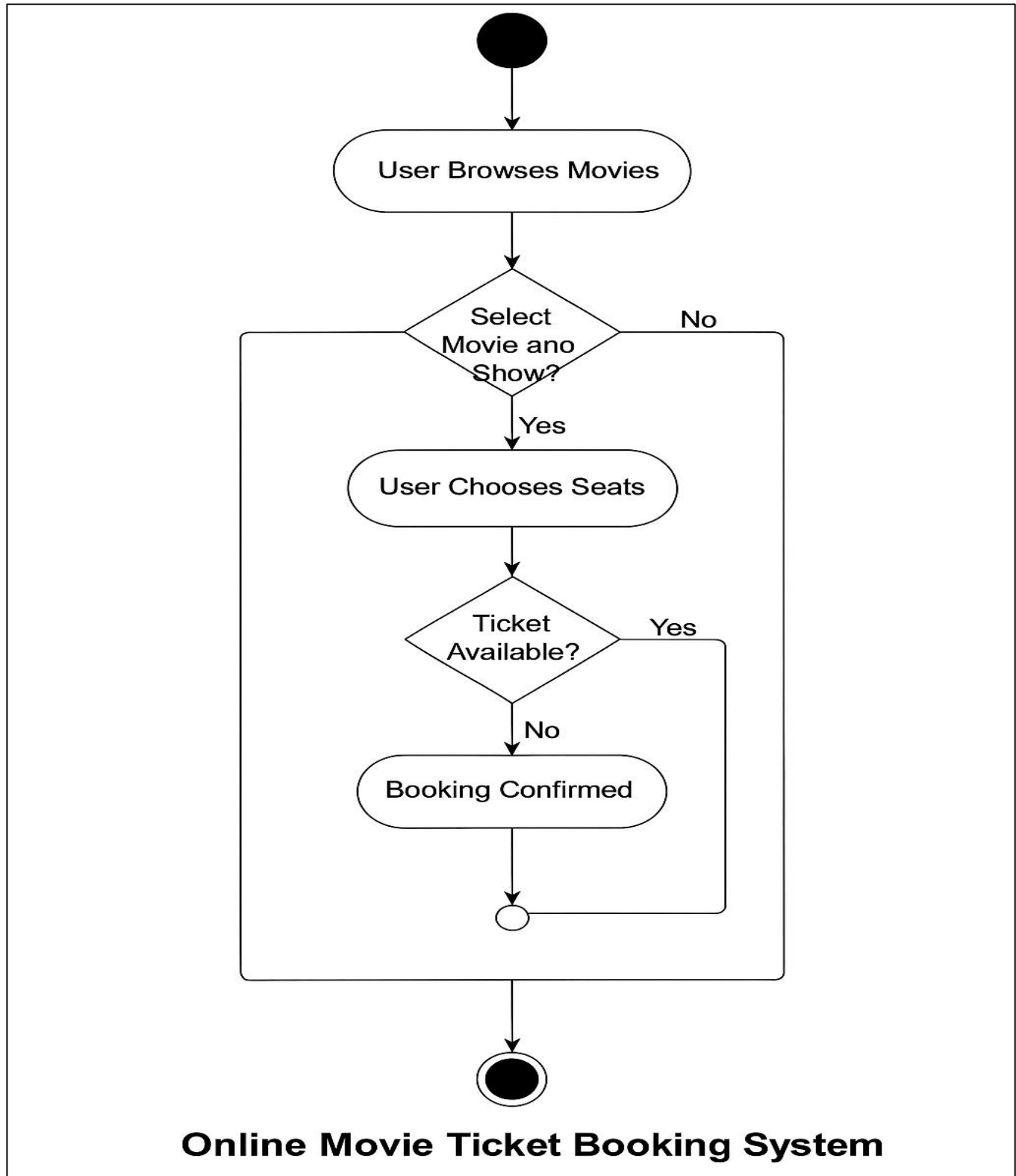


4.4 USE CASE DIAGRAM:

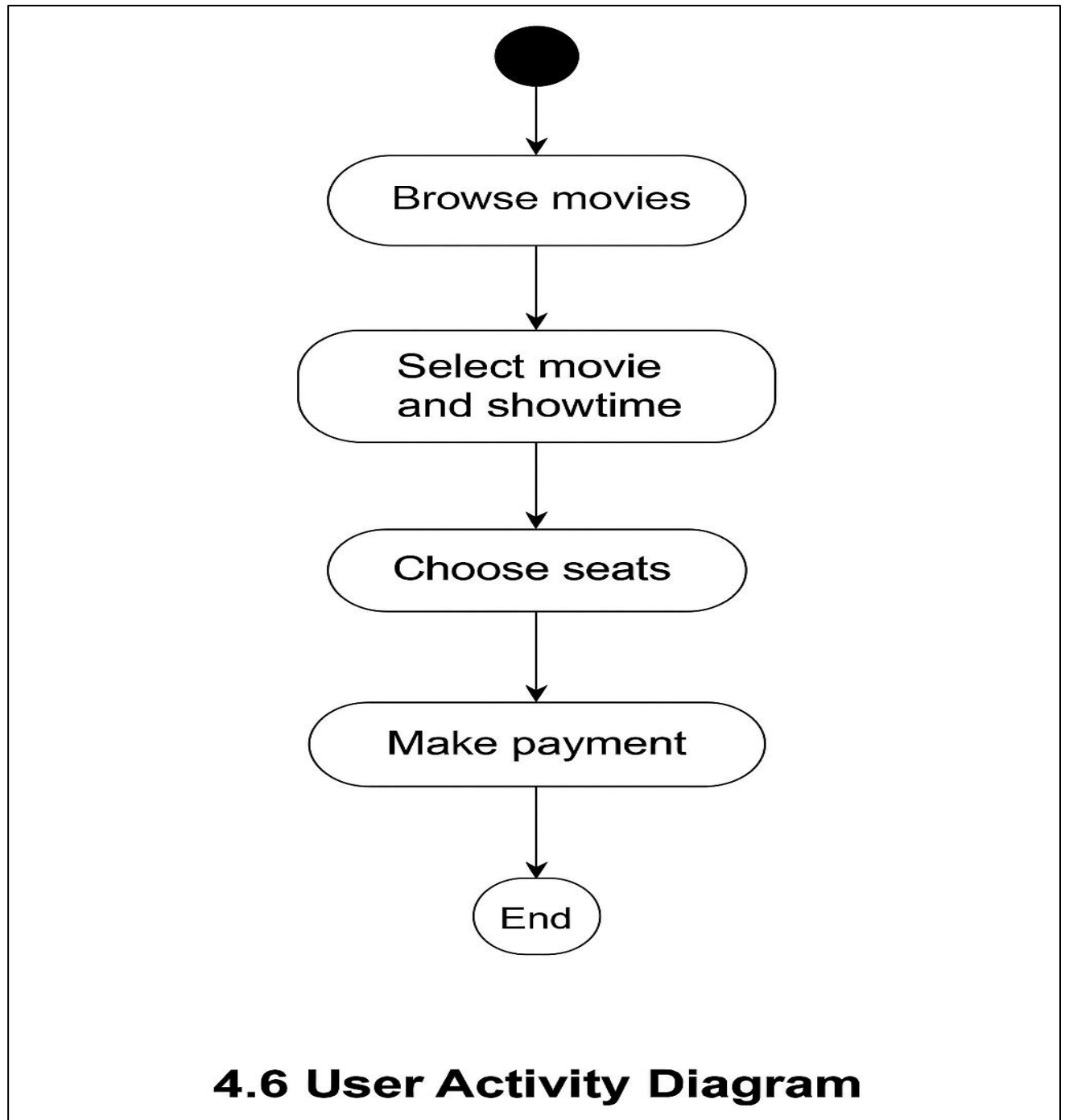


4.5 ACTIVITY DIAGRAMS:

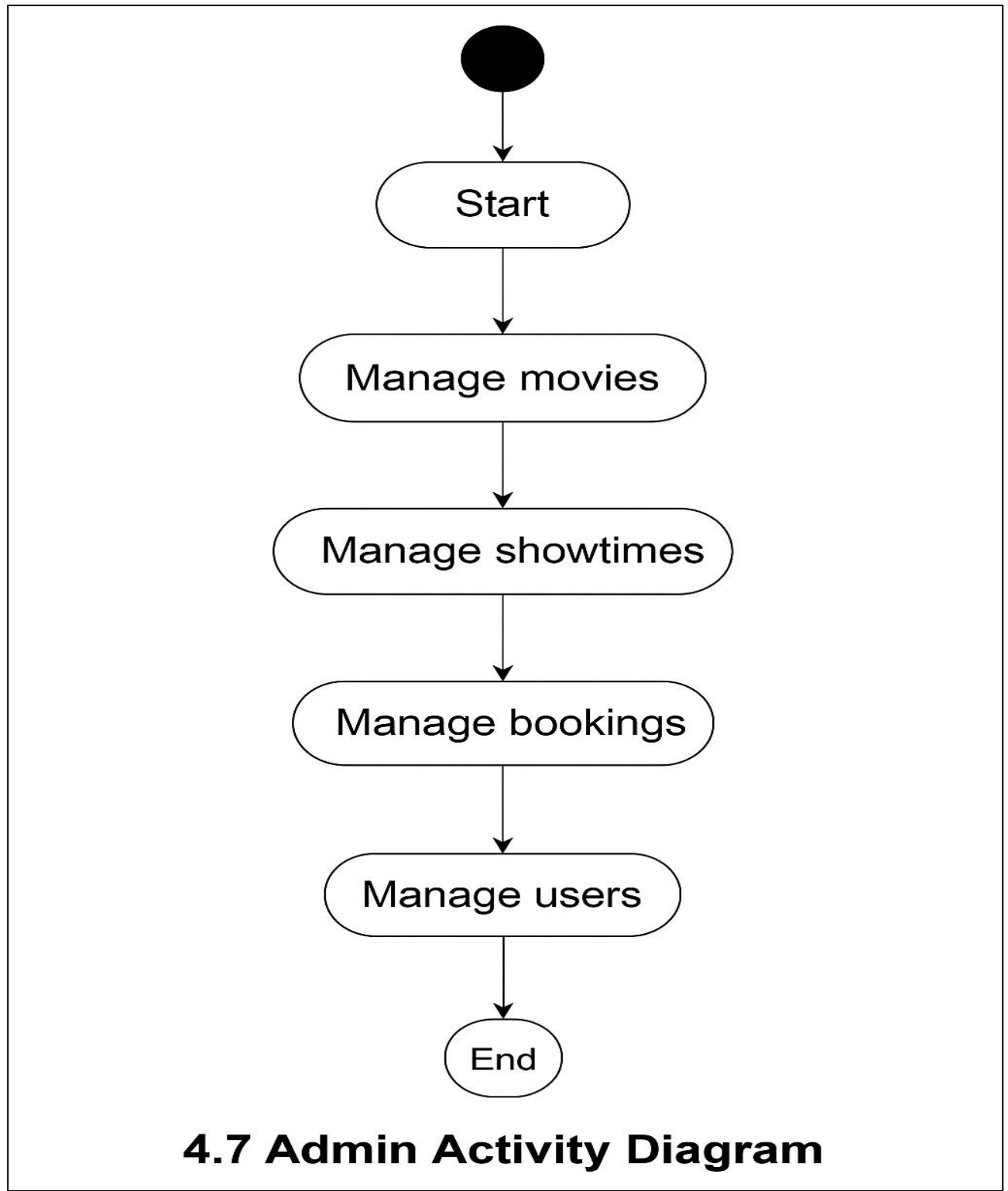
Exchange Request Activity Diagram



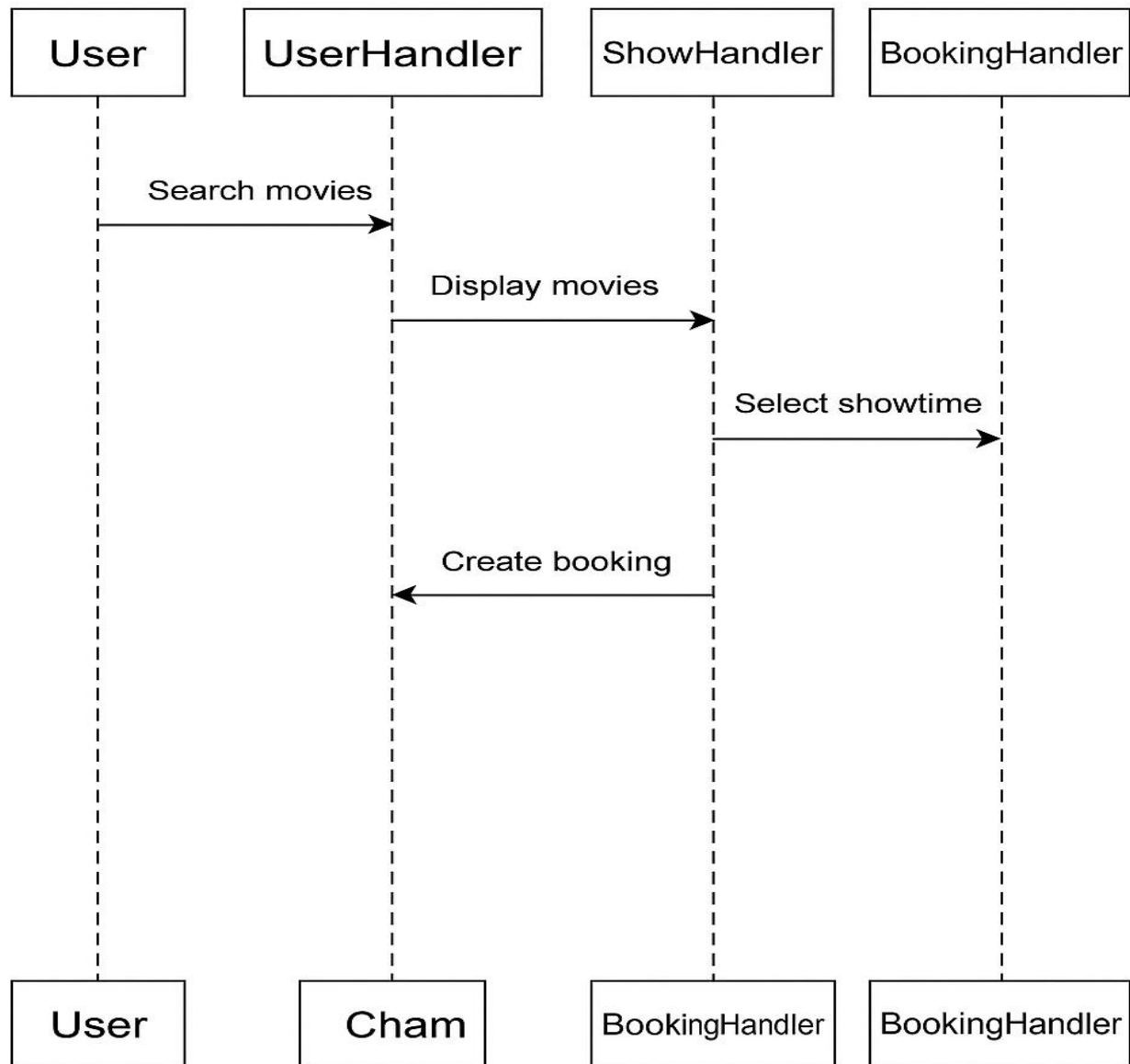
4.6 User Activity Diagram:



4.7 Admin Activity Diagram:

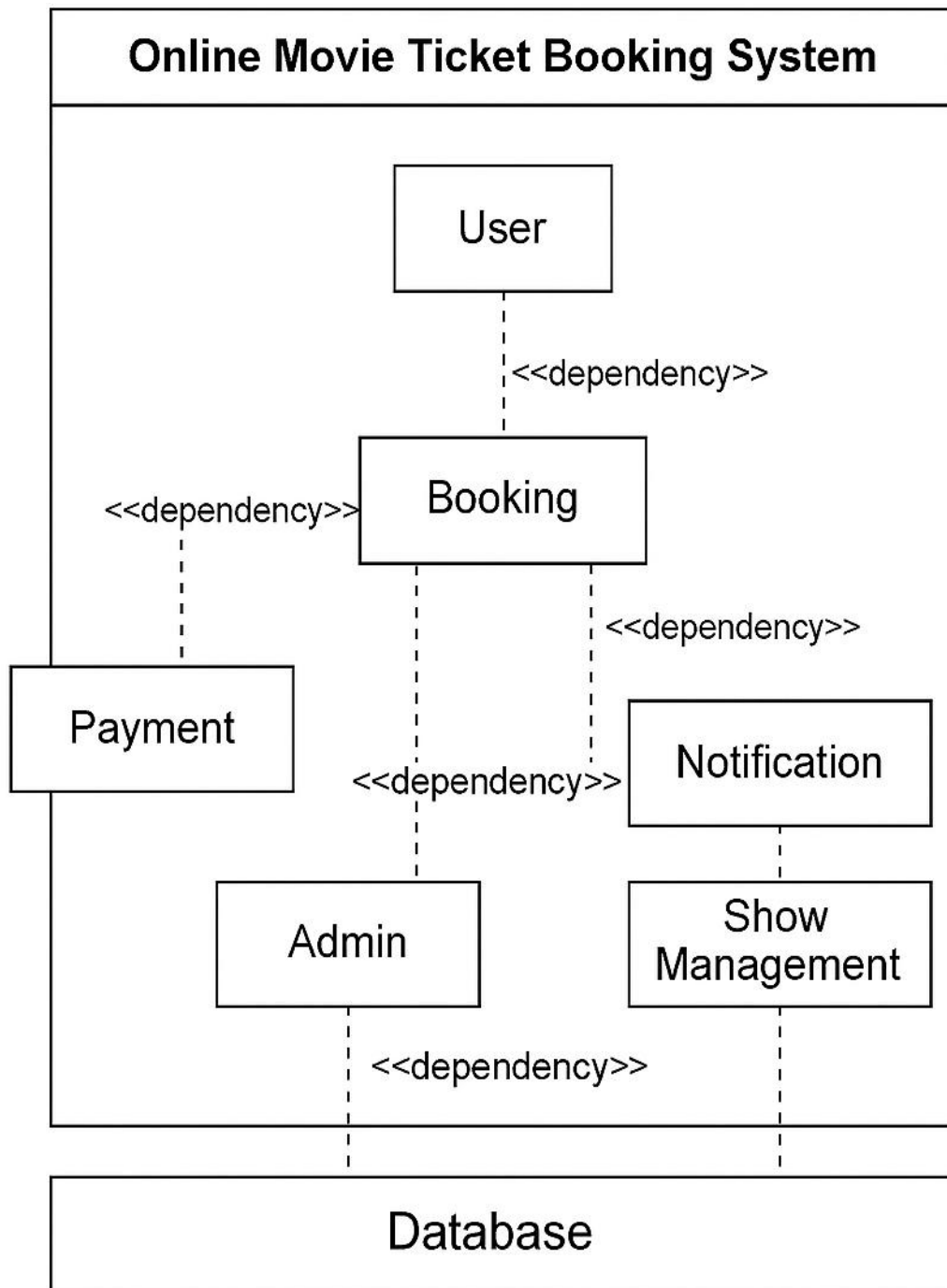


4.8 SEQUENCE DIAGRAM:



4.8 Sequence Diagram

4.9 COMPONENT DIAGRAM:



5.1 DATA DICTIONARY:

▪ User Entity:

Attribute Name	Data Type	Size	Description	Constraints
user_id	Integer	-	Unique identifier for each user.	Primary Key, Auto Increment
username	Varchar	50	Unique name selected by the user for login.	Not Null, Unique
password	Varchar	255	Encrypted password for the user's account.	Not Null
email	Varchar	100	Email address of the user.	Not Null, Unique
phone_number	Varchar	15	User's contact number.	Optional
gender	Varchar	10	Gender of the user.	Optional
date_of_birth	Date	-	Date of birth of the	Optional

			user.	
created_at	DateT i me	-	Timesta mp when the user account was created.	Default: current timesta m p
is_admin	Boolean	-	Flag to identify admin or regular user.	Default : false

▪ **Book Entity:**

Attribute Name	Data Type	Size	Description	Constraints
booking_id	Integer	-	Unique identifier for each booking.	Primary Key, Auto Increment
user_id	Integer	-	References the user who made the booking.	Foreign Key → User(user_id), Not Null
show_id	Integer	-	References the movie show being booked.	Foreign Key → Show(show_id), Not Null
booking_date	DateTime	-	Date and time when the booking was made.	Default: current timestamp
total_tickets	Integer	-	Number of seats/tickets booked.	Not Null
total_amount	Decimal	10,2	Total price for the booking.	Not Null
payment_status	Varchar	20	Status of the payment (e.g., Paid, Failed, Pending).	Default: Pending
transaction_id	Varchar	100	Reference ID for the payment transaction.	Optional, Unique
seat_numbers	Varchar	255	Comma-separated list of booked seat numbers.	Not Null

▪ **Exchange Request Entity:**

Attribute Name	Data Type	Size	Description	Constraints
request_id	Integer	-	Unique identifier for each exchange request.	Primary Key, Auto Increment
booking_id	Integer	-	References the original booking.	Foreign Key → Booking(booking_id), Not Null
user_id	Integer	-	References the user who requested the exchange.	Foreign Key → User(user_id), Not Null
old_show_id	Integer	-	ID of the originally booked show.	Foreign Key → Show(show_id), Not Null
new_show_id	Integer	-	ID of the requested new show.	Foreign Key → Show(show_id), Not Null
requested_seats	Varchar	255	New seat numbers requested (comma-separated).	Not Null
exchange_reason	Text	-	Reason provided by user for exchange.	Optional
request_status	Varchar	20	Status of the exchange request (e.g., Pending, Approved, Rejected).	Default: 'Pending'
request_date	DateTime	-	Date and time when the exchange request was submitted.	Default: current timestamp
admin_response_note	Text	-	Note or response from admin regarding the request.	Optional

▪ **BookLike Entity:**

Field Name	Data Type	Description
id	INT	Primary key for BookLike
user_id	INT	Foreign key to User entity
movie_id	INT	Foreign key to Movie entity
timestamp	DATETIME	When the booklike was recorded

▪ **Notification Entity:**

Field Name	Data Type	Description
notification_id	INT	Primary key
user_id	INT	User receiving the notification
message	VARCHAR(...)	Notification content
created_at	DATETIME	When the notification was created

5.2 TEST PROCEDURE & IMPLEMENTATION

▪ **Objective of Testing**

The primary objective of testing was to ensure that all modules of the **Movie Ticket Booking System** function correctly and handle both expected and unexpected user inputs. The goal was to identify and fix bugs, verify data consistency, and validate business logic. Test Environment Setup

▪ **Test Environment**

- **Frontend:** Python with Tkinter GUI
- **Backend:** MySQL Database
- **Tools Used:** Manual testing through UI interaction and MySQL queries via command-line/Workbench
- **OS:** Windows 10 / Linux

▪ **Testing Methodology**

Type Description

Unit Testing	Individual functions and methods (e.g., login, register, book seat) were tested in isolation.
Integration	Ensured smooth communication between modules, such as GUI ↔ Database ↔
Testing System	Logic Layer. Full end-to-end testing of the entire application including both admin and customer
Testing	interfaces.
Functional Testing	Verified each feature (e.g., booking, movie addition, login) performed as intended.
Validation	Ensured input fields correctly handled invalid data (e.g., empty fields, wrong
Testing Database	formats). Validated data integrity, foreign key constraints, and correct updates/inserts into the
Testing	MySQL database.
Regression Testing	Re-ran previous test cases after making bug fixes or adding new features.

2. Execution Steps:

STEP 1: Install Required Software

Install Python (v3.6 or above)

Download from: <https://www.python.org/downloads>

Verify installation:

```
python --version
```

Install MySQL

Use XAMPP or standalone MySQL Server

Install MySQL Workbench or phpMyAdmin for managing the database

Install Required Python Modules

Run the following in Command Prompt or Terminal:

```
pip install mysql-connector-python
```

STEP 2: Set Up the Database

Open MySQL (phpMyAdmin or MySQL Workbench)

Create Database

```
CREATE DATABASE movie_ticket_db;
```

Create Tables

Paste the table creation SQL or use provided database_script.sql. Example:

```
USE movie_ticket_db;
```

```
CREATE TABLE users (
```

```
    user_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    username VARCHAR(50) UNIQUE NOT NULL,
```

```
    password VARCHAR(255) NOT NULL,
```

```
email VARCHAR(100) UNIQUE NOT NULL,  
phone_number VARCHAR(15),  
gender VARCHAR(10),  
date_of_birth DATE,  
created_at DATETIME DEFAULT CURRENT_TIMESTAMP,  
is_admin BOOLEAN DEFAULT FALSE  
);  
  
-- Add tables: movies, shows, bookings, exchange_requests etc.
```

STEP 3: Project File Setup

Create a folder like Movie-Ticket-Booking-System/ and place your .py files in it:

Movie-Ticket-Booking-System/

```
|  
|— main.py  
|— db_connection.py  
|— login.py  
|— booking.py  
|— admin_panel.py  
|— customer_panel.py  
|— exchange_request.py  
|— utils.py  
└— database_script.sql
```

STEP 4: Configure Database Connection

In `db_connection.py`:

```
import mysql.connector

def connect():

    return mysql.connector.connect(

        host="localhost",

        user="root",

        password="your_mysql_password",

        database="movie_ticket_db"

    )
```

STEP 5: Run the Project

Open your terminal or command prompt.

Navigate to your project folder:

```
cd path/to/Movie-Ticket-Booking-System
```

Run the main program:

```
python main.py
```

Results and Observations

User Experience:

- The GUI is intuitive, using Tkinter to allow users to perform actions with minimal clicks.
- Validations are in place to avoid duplicate registrations or booking invalid seats.

Database Accuracy:

- Data was stored correctly in respective tables like users, bookings, movies, etc.
- Foreign key relationships ensured proper relational integrity between users, bookings, and shows.

Performance:

- The system handled multiple user operations (register, book, request exchange) without lag.
- Booking status was updated instantly in the database.

Security:

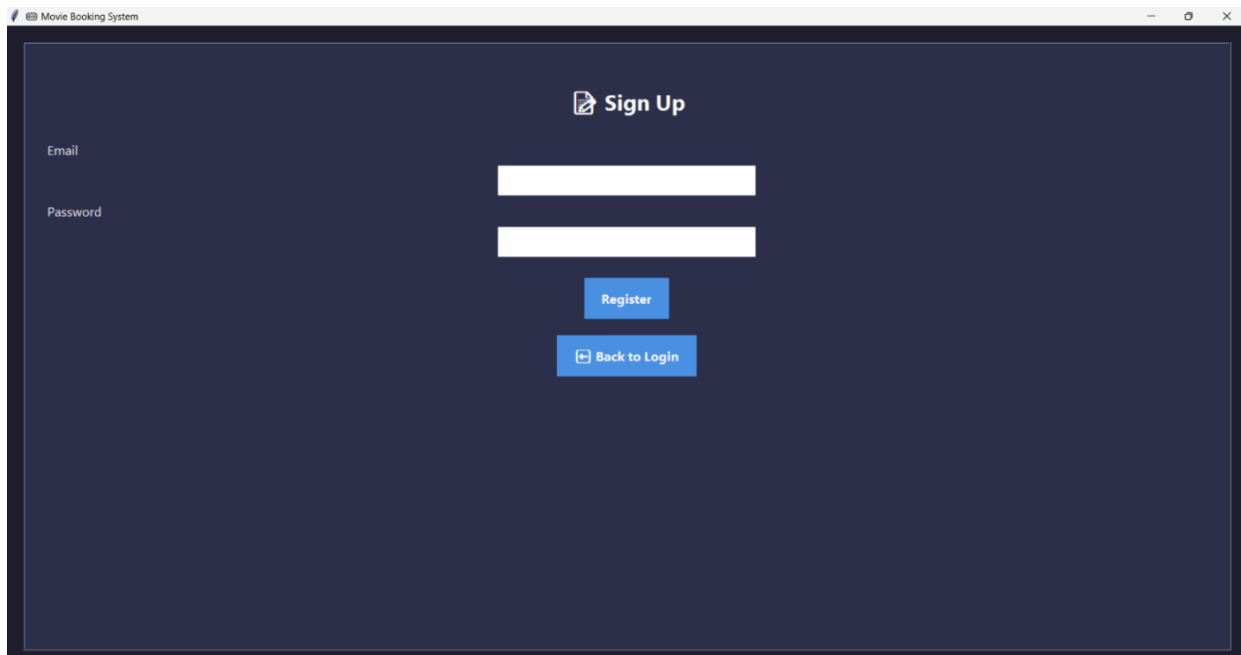
- Passwords were stored in plain format (can be improved with hashing in future versions).
- Only authenticated users could make bookings or request exchanges.

Error Handling:

- Handled errors like duplicate emails/usernames and invalid inputs gracefully.
- Provided user-friendly messages during failures (e.g., seat already booked).

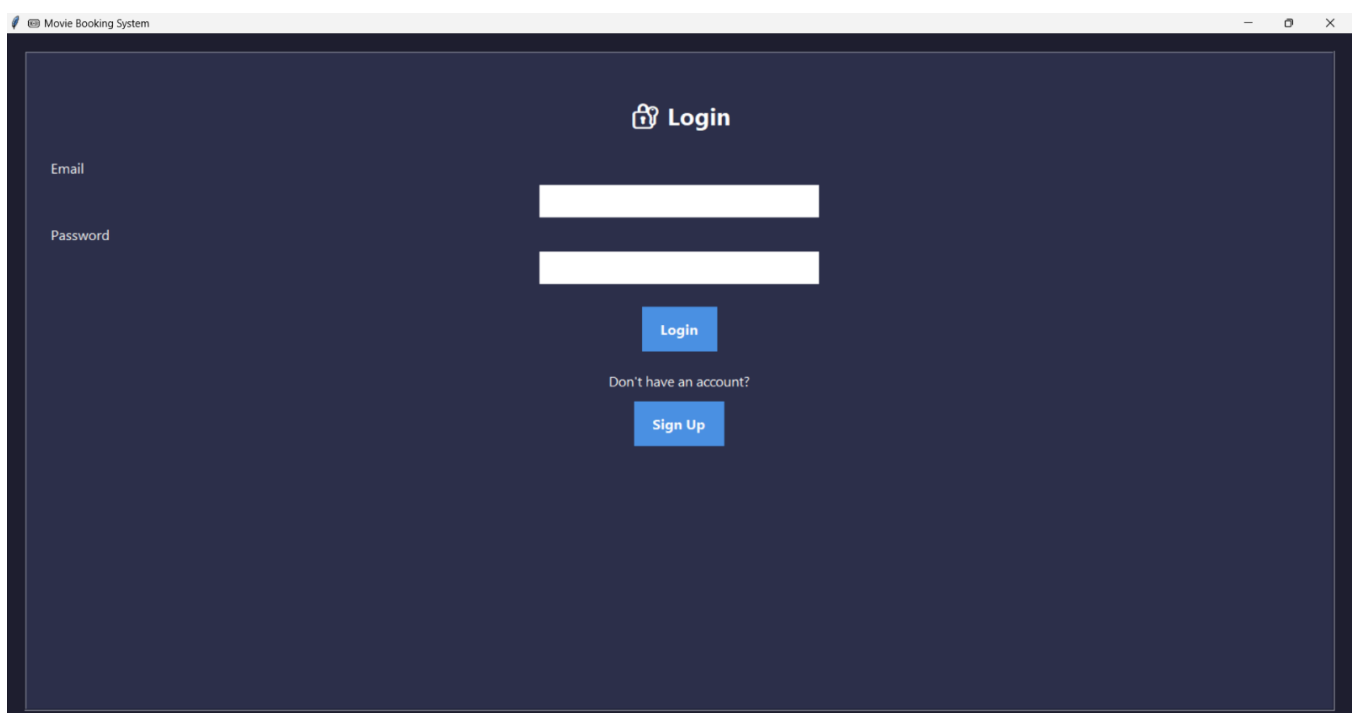
▪ 5.3 SCREENSHOTS :

- **User Signup/Register:**



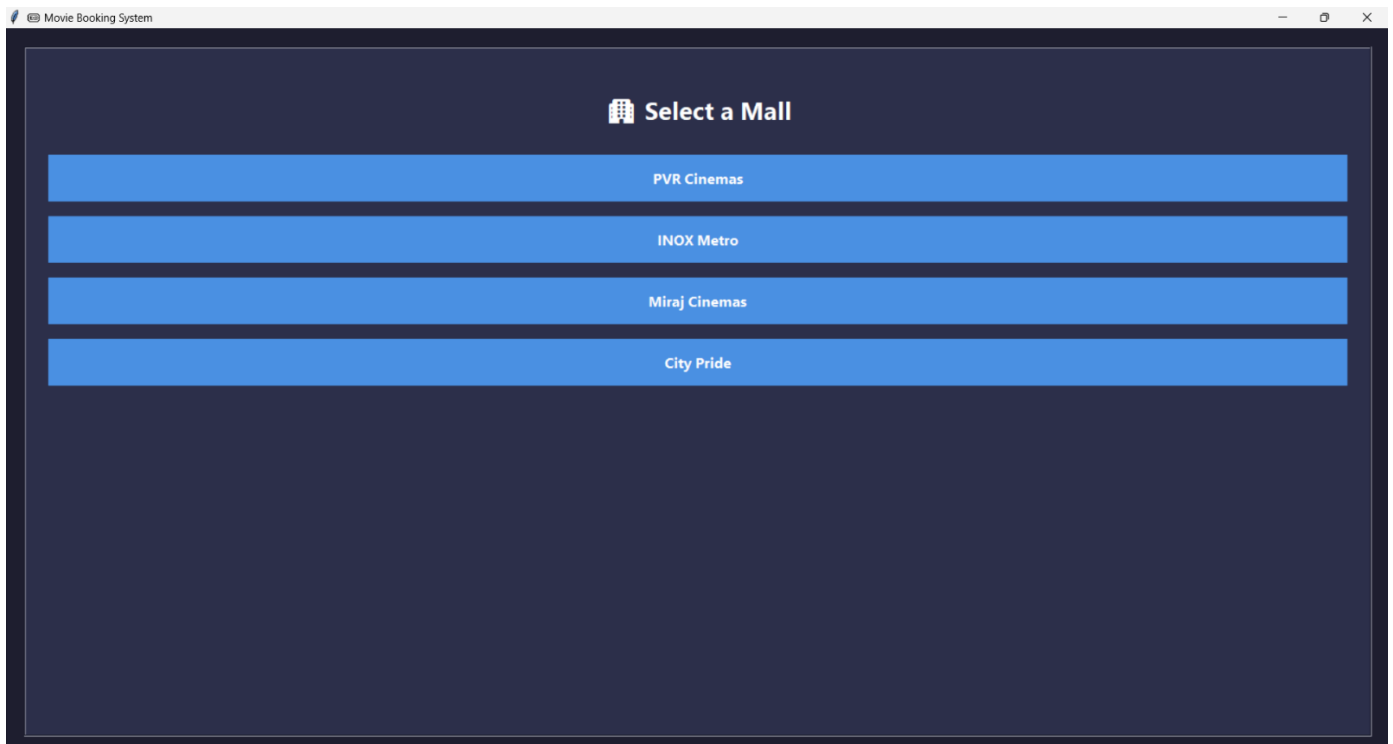
The screenshot shows a web browser window titled "Movie Booking System". The page has a dark blue background. At the top center, there is a "Sign Up" heading with a document icon. Below the heading, there are two white input fields for "Email" and "Password". To the left of these fields, the labels "Email" and "Password" are displayed. Below the input fields, there are two blue buttons: "Register" and "Back to Login".

- **User Login:**

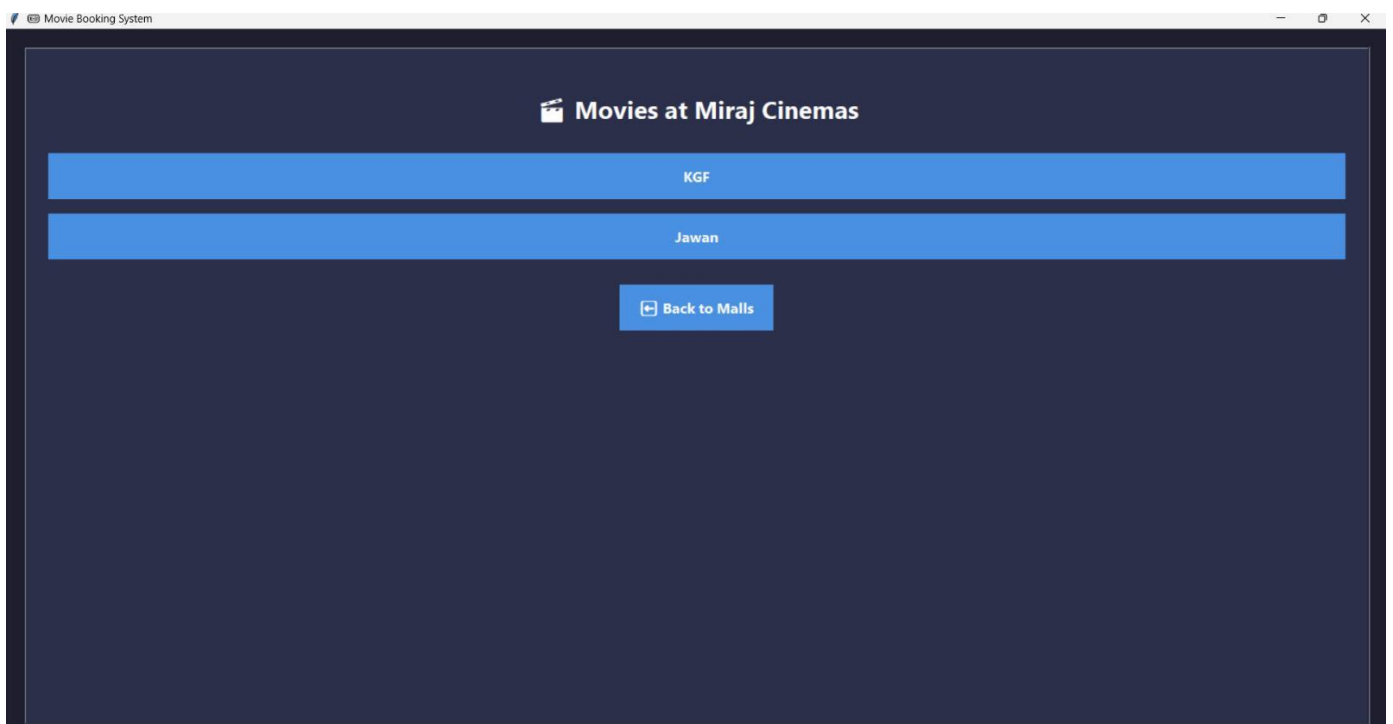


The screenshot shows a web browser window titled "Movie Booking System". The page has a dark blue background. At the top center, there is a "Login" heading with a key icon. Below the heading, there are two white input fields for "Email" and "Password". To the left of these fields, the labels "Email" and "Password" are displayed. Below the input fields, there is a blue "Login" button. Below the "Login" button, there is a link "Don't have an account?" and a blue "Sign Up" button.

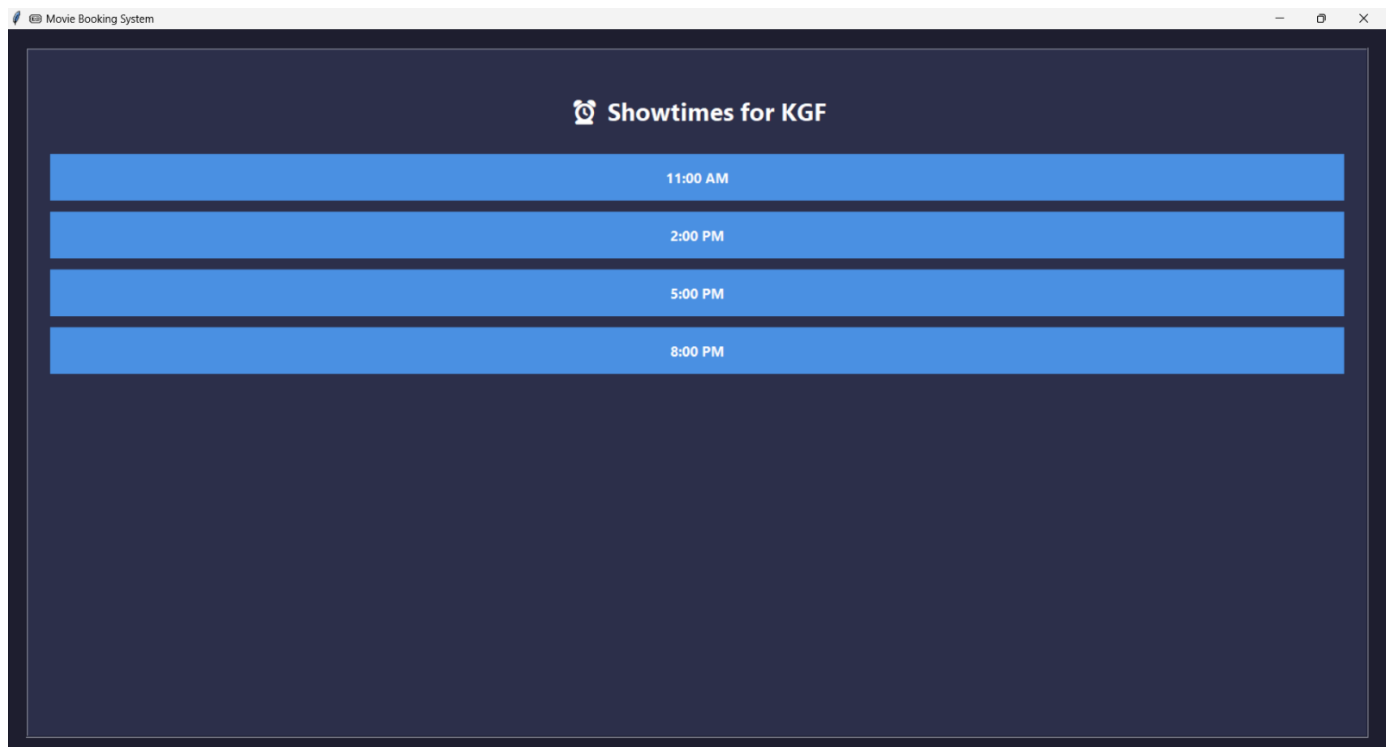
- **Select a Mall Page:**



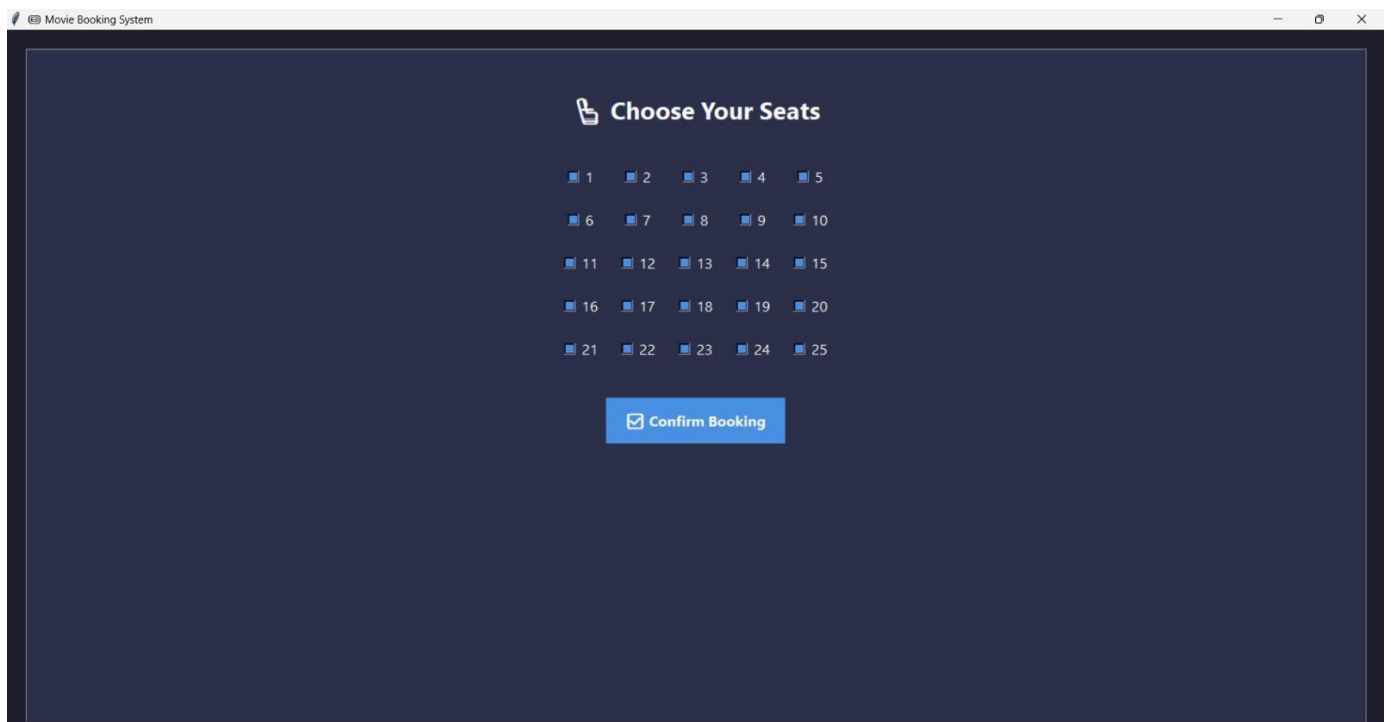
- **Select Movies page:**



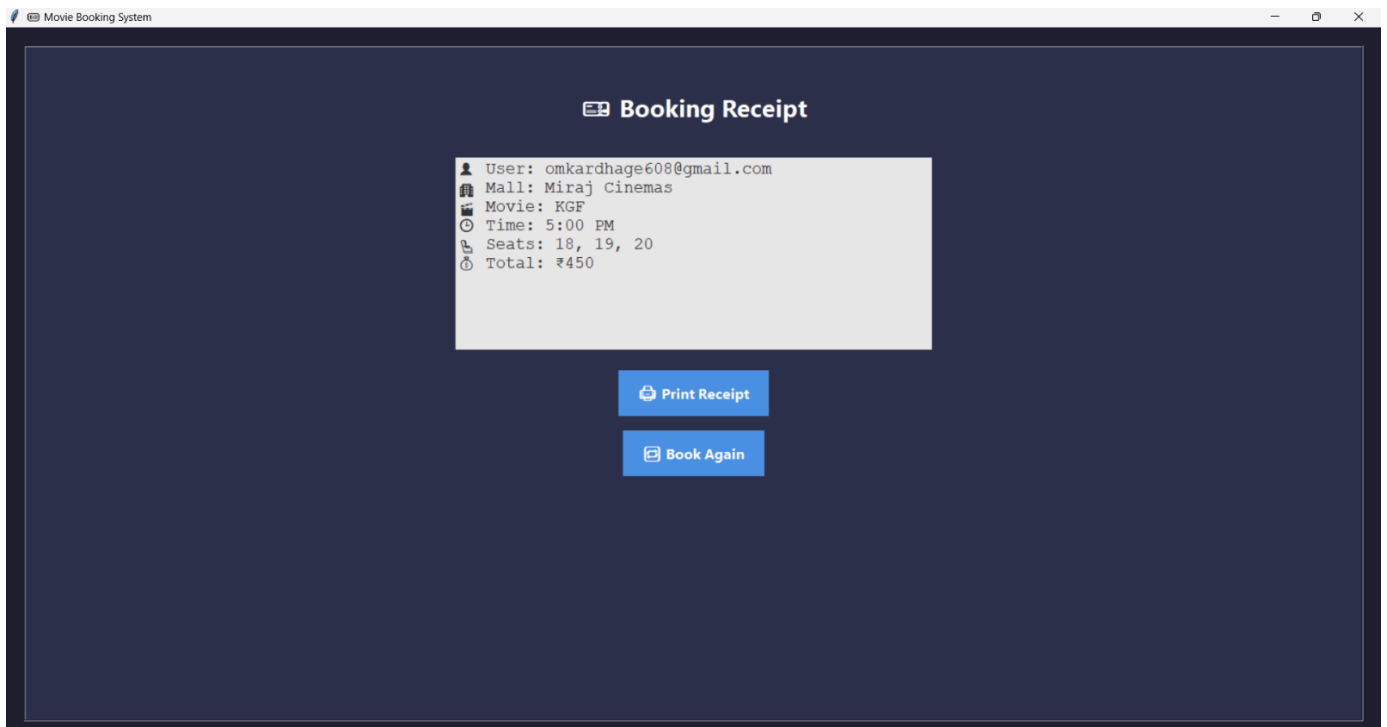
- **All Showtimes:**



- **Choose your Seats:**



- **Booking receipt:**



Future Scope

The **Online Movie Ticket Booking System** has been designed with a modular and scalable architecture, which provides a strong foundation for future enhancements. While the current version fulfills all core functionalities, there is significant scope to expand and improve the system in the following ways:

1. **Online Payment Integration**

Integrating secure payment gateways (like Razorpay, Paytm, or UPI) will enable users to make real-time payments and receive instant e-tickets, making the system fully cashless and more convenient.

2. **SMS/Email Notification System**

The system can be enhanced by sending SMS or email notifications to users for booking confirmation, show reminders, and cancellations.

3. **Multi-Theater and Multi-City Support**

Future upgrades can include support for multiple theaters across different locations or cities. This would help in expanding the system's reach and allow users to select theaters by city or region.

4. **Dynamic Seat Mapping**

Introducing a dynamic seat layout per theater (instead of static templates) would make the booking experience more accurate and tailored to each venue.

5. **Mobile Application**

Developing a dedicated mobile app (Android/iOS) will enhance user accessibility and provide a smoother and faster booking experience.

6. **User Reviews and Ratings**

Allowing users to give reviews and ratings for movies and shows will help other customers make informed choices and promote user engagement.

7. **AI-Powered Movie Recommendations**

A recommendation engine can be added to suggest movies based on the user's past preferences and booking history, improving the personalization of the system.

8. **Admin Analytics Dashboard**

Future versions can include graphical dashboards for admins, showing analytics like booking trends, revenue reports, most-watched movies, and customer feedback.

9. **QR Code-Based Entry System**

Enabling QR codes on tickets can allow for quick and contactless verification at theater entrances, improving security and efficiency.

10. **Multilingual Support**

To make the system more inclusive, future updates can include multilingual support, allowing users to switch between English, Hindi, Marathi, or other regional languages.

Conclusion

The **Movie Ticket Booking System** developed using **Python (Tkinter)** for the front-end and **MySQL** for the back-end successfully achieves its primary goal of simplifying and digitizing the traditional ticket booking process. It provides an effective, user-friendly interface for customers to browse movies, select theatres and shows, choose seats, and book tickets, all from a single platform. Compared to manual systems, this solution significantly reduces waiting times, eliminates booking errors, and offers 24/7 accessibility.

The system automates core functions such as user registration, login, showtime management, and real-time seat availability. It supports both **user-side operations** (searching, selecting, booking) and **admin-side management** (adding/editing movies, theatres, shows, and reviewing bookings), ensuring a complete end-to-end solution.

From a technical standpoint, the project emphasizes:

- **Modular architecture**, allowing for easy debugging and future enhancements.
- **Effective GUI design**, enabling smooth navigation and interaction.
- **Robust database integration**, ensuring reliable data storage and retrieval.
- **Security basics**, such as password authentication and input validation.

In addition to its core functionality, the project serves as a **real-world application of key programming principles**. It integrates concepts of software engineering, database design (ER modeling, SQL), user interface development, and system design patterns.

The learning outcomes of this project are substantial for any beginner or intermediate-level developer aiming to build full-stack applications.

Looking ahead, the system offers several possibilities for future enhancements:

- Deploying the system online using web frameworks (like Flask or Django).
- Integrating secure payment gateways for digital transactions.
- Enabling QR code-based e-ticketing for contactless entry.
- Building a mobile version using frameworks like Kivy or React Native.
- Adding features like email notifications, discount codes, and user feedback.

In conclusion, this project not only bridges the gap between users and movie theatres but also showcases how a software system can streamline services in the entertainment sector. It stands as a practical, scalable, and impactful application that can be expanded further based on user and business requirements.

Bibliography

The following sources and tools were referenced during the development of the Movie Ticket Booking System:

Books and Study Material

1. **“Python Programming: An Introduction to Computer Science”** – John Zelle
2. **“Learning MySQL”** – Seyed M.M. & Hugh E. Williams
3. **“Tkinter GUI Application Development Cookbook”** – Alejandro Rodas de Paz

Web Resources and Documentation

4. **Python Official Documentation**
<https://docs.python.org/3/>
5. **Tkinter Reference Manual**
<https://tkdocs.com/>
6. **MySQL Documentation**
<https://dev.mysql.com/doc/>
7. **W3Schools – SQL & Python Tutorials**
<https://www.w3schools.com/>
8. **GeeksforGeeks – Python and DBMS Integration**
<https://www.geeksforgeeks.org/>
9. **Stack Overflow**
Used for troubleshooting errors, optimizing logic, and understanding common development issues.
<https://stackoverflow.com/>
10. **TutorialsPoint – Python and MySQL Integration**
https://www.tutorialspoint.com/python/python_database_access.html

References

1. **Python Official Documentation**
<https://docs.python.org/3/>
Used for understanding syntax, modules, and libraries such as Tkinter, datetime, and os.
2. **MySQL Official Documentation**
<https://dev.mysql.com/doc/>
Referenced for SQL commands, database design, and table relationships.
3. **Tkinter GUI Programming by Tutorialspoint**
https://www.tutorialspoint.com/python/python_gui_programming.htm
Helped with building forms, buttons, labels, and layouts.
4. **MySQL Connector/Python (Official Library)**
<https://pypi.org/project/mysql-connector-python/>
Used for establishing a connection between Python and the MySQL database.
5. **Stack Overflow**
<https://stackoverflow.com/>
Community support and solutions for resolving coding errors and improving functionality.
6. **GeeksforGeeks – Python MySQL Tutorials**
<https://www.geeksforgeeks.org/python-mysql/>
Useful for understanding SQL query execution and database connectivity with Python.
7. **W3Schools – SQL & Python Tutorials**
<https://www.w3schools.com/>
Helped with SQL syntax, SELECT queries, and form validation concepts.
8. **Project Inspiration & Guidance**
9. College assignment briefs and teacher inputs