



**Dr. D. Y. Patil Unitech Society's
Dr. D. Y. Patil Arts, Commerce and Science College,
Pimpri, Pune**

**IT PROJECT WORK REPORT
ON
“Movie Ticket Booking System”**

**Developed by
Ashwita Ashok Aher**

in partial fulfilment of
M.Sc. (Computer Science) Sem-IV

Savitribai Phule Pune University

2024-2025



**Dr. D. Y. Patil Unitech Society's
Dr. D. Y. Patil Arts, Commerce and Science College, Pimpri,
Pune 18**

CERTIFICATE

Exam Seat No.: - 3083

This is to certify that **Ashwita Ashok Aher (43)** has successfully completed the Industrial Training Work entitled **Movie Ticket Booking System** for M.Sc. (Computer Science) Sem -IV of Savitribai Phule Pune University for the Academic Year 2024-25.

Project Guide

Coordinator

**I/C Principal and H.O.D.
(Computer Science)**

Internal Examiner

External Examiner

Index

Sr.No.	Contents	Page No
1	INTRODUCTION	05
	1.1 Company Profile	06
	1.2 Motivation	08
	1.3 problem statement	09
	1.4 purpose/objective and goals	10
	1.5 literature survey	12
	1.6 project scope and limitations	14
2	SYSTEM ANALYSIS	15
	2.1 Comparative study of Existing systems	15
	2.2 scope and limitations of existing systems	16
	2.3 project perspective, features	17
	2.4 stakeholders	18
	2.5 Requirement analysis - Functional requirements, performance requirements, security requirements etc.	19
3	SYSTEM DESIGN	21
	3.1 Design constraints	21
	3.2 Entity-Relationship diagram	22

	3.3 Class Diagram	23
	3.4 Use case Diagram	24
	3.5 Activity Diagram	25
	3.6 Sequence Diagram	28
	3.7 Component Diagram	29
	3.8 Data Dictionary	30
	3.9 Test procedure & implementation	32
	3.10 Screen Shot	34
	3.11 Reports	41
4	IMPLEMENTATION DETAILS 4.1 Software/hardware specifications, etc	43
5	TESTING Test Plan, Black Box Testing or Data Validation Test Cases, White Box Testing or Functional Validation Test cases and results	46
6	LIMITATIONS	49
7	CONCLUSION	51
8	BIBLIOGRAPHY	52
9	REFERENCES	53

INTRODUCTION

The "Movie Ticket Booking System" is a comprehensive desktop-based application that simplifies and digitizes the traditional process of booking movie tickets. Developed using **Python** with the **Tkinter** GUI toolkit and integrated with a **MySQL** backend, the system supports both administrative and customer functionalities.

The system allows users (customers) to view currently running movies, check showtimes, select available seats from a visual layout, and confirm their booking. Upon booking, the system generates a digital receipt with all details such as movie name, seat number, showtime, and customer name. This makes it convenient for users to plan and execute their movie experience efficiently.

The admin interface offers a secure login mechanism where authorized personnel can add new movies, schedule showtimes, monitor bookings, and generate simple reports. Admins also have the authority to update or remove movies and manage the database of shows and customers. All this data is stored securely using MySQL, ensuring consistency and reliability. This project aims to overcome the limitations of the manual ticket booking system by providing a visual, interactive, and automated solution. It is particularly designed for small to medium-sized cinema halls, local theaters, and academic or training projects that aim to showcase full-stack development with GUI and database integration.

Key features of the system include:

- Real-time seat status updates (color-coded)
- Booking history and receipts
- Easy movie and showtime management for admins
- Smooth and responsive GUI experience
- Secure database integration using MySQL

With increasing digitization, cinema owners and institutions need an efficient way to handle bookings and audience management. This project is a step toward that digital transformation, making ticket booking fast, accurate, and user-friendly.

1.1 COMPANY PROFILE

Organization Name: Anudip Foundation for Social Welfare

Year of Establishment: 2007

Type: Non-Profit Organization

Headquarters: Kolkata, Mumbai, Pune

Website: www.anudip.org

Overview:

Anudip Foundation is a globally recognized non-profit organization dedicated to transforming lives through technology-enabled education and livelihood programs. Established in 2007, Anudip empowers youth, women, and marginalized communities by equipping them with essential digital, technical, and professional skills. By merging technology with social innovation, the organization creates pathways to sustainable employment and economic independence.

Operating in over 20 states across India and with a growing global presence, Anudip collaborates with government institutions, corporate partners, and international NGOs to implement high-impact programs. These initiatives aim to bridge the digital divide, promote gender equality, and enhance the employability of underserved populations.

Relevance to the Project:

The "Movie Ticket Booking System" project was developed under the academic and technical guidance provided by Anudip Foundation as part of its IT and Coding Bootcamp initiative. The foundation offered mentorship, infrastructure, and real-time project exposure, allowing students to conceptualize and build a socially impactful digital solution. This project aligns with Anudip's mission to foster future-ready tech professionals capable of addressing real-world community challenges through innovation and inclusion.

Mission Statement:

"To transform lives by creating digitally-enabled, self-reliant communities."

Vision Statement:

"To be a global leader in delivering digital skills and career transformation to underserved populations."

Core Values (Explained):

- **Inclusivity:**

Anudip is committed to ensuring equal access to opportunities for individuals from all walks of life, especially those who are

underrepresented or come from socially and economically disadvantaged backgrounds. This includes women, tribal communities, rural youth, and persons with disabilities.

- **Training Model:**

Anudip runs specialized programs such as the Digital Livelihood Program (DLP), Women in Tech (WIT), and Skills to Succeed Academy, offering training in areas like full-stack development, web design, data entry, and customer service.

- **Placement Support:**

After training, Anudip provides placement assistance to its students through its strong network of corporate partners, helping thousands find jobs in IT, BPO, retail, and finance sectors.

- **Global Recognition:**

Anudip Foundation has been recognized by international organizations such as the World Economic Forum, UNDP, and NASSCOM Foundation for its impactful work in digital inclusion and sustainable development.

- **Digital Transformation Centre's:**

It operates hundreds of Digital Learning centres across urban and rural India equipped with computers, internet access, and trained faculty, making digital learning accessible to the last mile.

- **Integrity:**

Anudip operates with transparency, fairness, and a strong sense of ethics in all its programs. It promotes honesty and accountability among its staff, trainers, and students to build trust and long-term impact.

- **Impact:**

All programs at Anudip are designed with a results-oriented approach. The focus is on measurable outcomes like job placements, career progression, income generation, and social upliftment, ensuring that each initiative contributes meaningfully to the beneficiary's life.

1.2 MOTIVATION

The development of the Movie Ticket Booking System was inspired by several real-world challenges that cinema halls and moviegoers face on a daily basis. Traditional ticketing involves manual labor, delayed service, long queues, and lack of real-time seat information. These issues result in poor user experiences and potential revenue loss for theater owners. With the advent of modern computing and user expectations for fast, digital service delivery, there was a pressing need for a system that could solve these problems effectively.

Primary Motivations:

1. **To Eliminate Long Queues and Manual Errors:** Manual ticket counters are often inefficient, especially during peak hours and weekends. Mistakes in seat allocation or overbooking can lead to customer dissatisfaction. An automated system can drastically improve this experience.
2. **To Enable Visual Seat Selection:** Allowing customers to view seat availability in a graphical format helps them make informed choices and improves the booking experience. It replicates the user-friendliness of web-based portals in a desktop environment.
3. **To Enhance Accessibility in Semi-Urban and Rural Areas:** Many local cinema halls do not have the infrastructure or funds to invest in large commercial booking portals. This Python-MySQL-based solution is cost-effective and can be operated offline.
4. **To Provide Learning and Innovation Opportunity:** This system was also conceptualized as an academic project to demonstrate real-world application of GUI programming, database management, and modular design patterns in Python. It gives students hands-on experience with practical technologies.
5. **To Promote Environmentally Friendly Ticketing:** Reducing the need for paper tickets by generating digital booking receipts contributes to environmental sustainability.
6. **To Support Administrative Control and Reporting:** Theater owners and administrators need real-time visibility into bookings, available seats, and user data. The admin panel fulfills this requirement by offering control features that are normally found in expensive commercial systems.

1.3 PROBLEM STATEMENT

In many traditional cinema halls, especially in semi-urban and rural areas, the movie ticket booking process remains manual. This involves ticket counters with handwritten registers or simple spreadsheet entries. Customers are required to physically visit the theater, inquire about available movies, showtimes, and seat availability, and then book tickets. This outdated method leads to several issues:

Identified Problems:

1. **Manual Operations Lead to Errors:** Manual entry systems are prone to mistakes such as overbooking, incorrect seat assignments, and untracked cancellations. These errors reduce customer trust and operational efficiency.
2. **Long Queues and Wasted Time:** Customers must wait in long lines to purchase tickets, which becomes worse during weekends or blockbuster movie releases. This can discourage walk-ins and reduce overall attendance.
3. **Lack of Real-time Seat Availability:** There is no way for customers to know which seats are available without asking the counter staff, and updates are delayed due to the absence of digital synchronization.
4. **No Seat Visualization Interface:** Customers cannot view or choose specific seats. This limits user satisfaction and reduces control over their movie experience.
5. **Data Management Challenges:** Admins face difficulties tracking movie schedules, ticket sales, booking history, and customer data without a digital system. This affects financial reporting and inventory control.
6. **Low Scalability:** As the number of shows or audience increases, the manual system becomes inefficient and unscalable. The risk of data loss and mismanagement also grows.
7. **Lack of Automation and Reporting:** Without a software solution, admins cannot generate automated sales reports or monitor real-time booking data.
8. **Limited Accessibility:** The absence of a user-friendly interface for booking and seat selection restricts accessibility for customers with disabilities or limited mobility.

1.4 PURPOSE/OBJECTIVE AND GOALS

PURPOSE OF THE PROJECT

The primary purpose of the Movie Ticket Booking System is to digitize and automate the process of booking movie tickets in small and mid-sized cinema halls. It is intended to replace the traditional, manual, error-prone ticketing process with a seamless, interactive, and visual system. By using Python and MySQL, the system provides a locally operable, cost-effective, and scalable solution for theaters and users alike.

KEY PURPOSES:

1. **Digital Transformation of Cinema Ticketing:** Introduce a paperless, automated system to simplify and speed up ticket booking operations.
2. **Visual Seat Layout Integration:** Enable users to make booking decisions using a graphical seat selection interface with real-time status updates.
3. **Accessibility for Small Theaters:** Provide an affordable and practical booking solution that does not require high-end infrastructure or continuous internet access.
4. **Admin Empowerment:** Equip administrators with control over showtimes, movie listings, and booking statistics through a secure login system.
5. **Secure and Organized Data Storage:** Maintain a centralized and secure database to manage users, movies, shows, and bookings efficiently.
6. **Eco-Friendly System:** Eliminate the need for paper tickets through digital receipts and confirmations.
7. **Educational Innovation:** Provide a hands-on learning experience for developers and students in GUI development, backend integration, and full-stack system design. The primary purpose of the Movie Ticket Booking System is to digitize and automate the process of booking movie tickets in small and mid-sized cinema halls. It is intended to replace the traditional, manual, error-prone ticketing process with a seamless, interactive, and visual system. By using Python and MySQL, the system provides a locally operable, cost-effective, and scalable solution for theaters and users alike.
8. The platform focuses on enhancing user experience, streamlining seat management, and enabling theater admins to efficiently manage shows and track bookings. This system is particularly relevant for semi-urban and rural theaters that lack the infrastructure for commercial cloud-based booking solutions.

OBJECTIVE OF THE PROJECT

The objective of the **Movie Ticket Booking System** project is to design and develop a complete software solution that simplifies and automates the process of booking movie tickets. This project serves the dual purpose of improving user convenience and enhancing administrative efficiency.

In traditional cinema ticket booking, users need to visit the theatre in person, stand in queues, and manually check for movie timings and seat availability. This method is not only time-consuming but also inefficient in managing crowds, records, and seat allocation. To solve these problems, this project provides an **online movie ticket booking system** using **Python (for the application logic and GUI)** and **MySQL (for database storage and management)**.

Key Objectives:

1. User Convenience:

- Allow users to view a list of currently running movies and upcoming releases.
- Provide information about each movie (e.g., name, language, duration, rating).
- Enable users to search and filter movies based on malls/theatres, show timings, and language.
- Let users select the number of tickets and specific seats from a graphical seat map.
- Enable secure user registration and login functionality.
- Store user bookings for future reference or printing.

2. Administrative Efficiency:

- Admins can add, update, and delete movie details, theatre names, halls, and show timings.
- Maintain show schedules and seating arrangements dynamically.
- Monitor and update seat availability in real time.
- Access booking records and generate reports if needed.
- Prevent overbooking or duplicate bookings.

3. Data Management and Security:

- Use **MySQL** to store and manage data including users, movies, shows, seats, and bookings.
- Ensure integrity of user data through proper table relations and constraints.
- Provide secure login mechanisms to restrict admin access.

4. Scalability and Real-world Simulation:

- The system simulates real-world cinema operations and can be extended with more features like:
 - Payment gateway integration.
 - Mobile app version.

Email/SMS booking confirmations.
Loyalty points or offers.

5. Automation and Digital Transformation:

Replace manual ticket booking with an intuitive digital system.
Reduce dependency on human staff for ticketing and scheduling.
Improve overall customer experience and operational efficiency.

GOALS OF THE PROJECT

The primary goal of the **Movie Ticket Booking System** is to create an efficient, user-friendly, and automated platform for online movie ticket reservations. This project bridges the gap between customers and theatre management by providing a seamless digital experience for booking movie tickets, managing movie schedules, and tracking seat availability.

Main Goals:

1. Automate Ticket Booking:

- Eliminate manual ticketing processes by allowing users to select movies, showtimes, seats, and book tickets digitally.
- Ensure real-time seat availability updates to avoid overbooking or conflicts.

2. User-Friendly Interface:

- Design an intuitive graphical user interface using Python's Tkinter library for easy navigation and usage by all types of users.
- Simplify the booking experience with clear steps from movie selection to seat confirmation.

3. Centralized Database Management:

- Use MySQL to store and manage all data including users, movies, shows, bookings, and seat status.
- Maintain data accuracy, integrity, and availability

4. Secure Login and Authentication:

- Implement secure login and registration for users and admins.
- Protect user information and ensure only authorized users can access the system features.

5. Admin Control Panel:

- Provide administrators the ability to manage movies, halls, showtimes, and theatre data.

1.5 LITERATURE SURVEY

A literature survey involves studying existing systems, technologies, and tools related to the domain of the project. For the **Movie Ticket Booking System**, the survey helps to understand how online booking platforms function, what features users expect, and how modern technologies are applied to meet these needs.

The traditional method of booking movie tickets involved long queues at cinema counters, manual registers for seat management, and no centralized system to track bookings. With advancements in technology, the rise of online ticketing systems like **BookMyShow**, **Paytm Movies**, **Fandango**, and theatre-specific portals has transformed the industry.

Existing Systems and Technologies:

System/Platform	Key Features
BookMyShow	Online ticket booking, show previews, seat selection, offers, payment gateway
Paytm Movies	Movie booking, cashback offers, show timings, multi-theatre access
Fandango (US)	Local theatre listings, reviews, ticket bookings, mobile app support
Inox / PVR Cinemas	Theatre-specific booking systems with loyalty rewards and real-time seat charts

These systems are fully digital, mobile-friendly, and integrated with payment gateways. They also store user data securely and allow history tracking, notifications, and promotional campaigns.

Limitations of Existing Systems (from a student-project perspective):

- They are complex, enterprise-level systems not suitable for small cinema halls or academic simulations.
- Development involves high cost, large teams, and external integrations (e.g., payment gateway, SMS services).
- Not open source or customizable for learning purposes.

Need for the Proposed System:

The proposed **Movie Ticket Booking System** aims to simulate a simplified version of these large platforms using open technologies like **Python (Tkinter GUI)** and **MySQL database**. It is designed for educational purposes to:

- Understand the core logic of online ticket systems.
- Learn how databases handle real-time updates (e.g., seat availability).
- Practice building GUI interfaces that interact with backend databases.
- Study the admin and user functionalities involved in a real-world system.

Technologies Considered:

Technology	Purpose
Python	Core logic, GUI using Tkinter
MySQL	Relational database system
Tkinter	GUI for booking and admin panels
SQL	To create and manage tables

Conclusion of Literature Survey:

From the literature reviewed, it is evident that online movie booking systems are efficient, fast, and user-oriented. By studying the architecture and features of existing platforms, this project adopts the core ideas and implements them using accessible tools, focusing on functionality, usability, and education.

1.5 PROJECT SCOPE AND LIMITATIONS

Scope of the Project

The Movie Ticket Booking System aims to replicate the core functionalities of an online movie ticket booking platform using Python (Tkinter for GUI) and MySQL (for database management). The project is suitable for academic and small-scale deployment purposes.

1. The system allows users to:

- Register and log in with a secure authentication system.
- View a list of currently available movies, including details like language, genre, and duration.
- Select a movie, choose a mall/theatre, and pick a showtime.
- Select seats from available ones in a graphical format and book them.
- View booking confirmation with a summary of the ticket details.

2. Administrators can:

- Add, update, or delete movie listings.
- Manage theatres, halls, and show timings.
- Monitor seat availability and customer bookings.
- Perform backend operations without manually updating the database.

Database Integration using MySQL ensures that all information (users, movies, shows, and bookings) is stored and updated in real-time.

3. This project can be extended in the future to support:

- Online payment gateways.
- SMS/email notifications.
- Loyalty or coupon systems.
- Multi-location theatre chains.

Limitations of the Project

While the system meets its core objectives, it has several limitations due to its educational and prototype nature:

1. No Online Payment Integration:

The current version only simulates the booking process; actual payment handling (via credit cards, UPI, etc.) is not implemented.

2. No Email/SMS Notification:

- Users do not receive email confirmations or SMS alerts after booking tickets.

3 Limited Security:

- Passwords may be stored in plain text (depending on implementation), and there's no encryption or role-based access control.
- No input validation or SQL injection protection if not explicitly coded.

4 Single System Deployment:

- The application is designed to run on a single desktop system and does not support multi-user, networked access.

5 GUI Design is Basic:

- Tkinter provides only simple graphical components. The design may not be as attractive or responsive as modern web/mobile apps.

6 No Seat Map Graphics:

- Seat selection may not be fully graphical or drag-and-drop style like real booking platforms.

7 Static Show Management:

- Shows must be manually added by the admin, and there is no automated scheduling or calendar system.

8 Lacks Error Handling:

- Incomplete validation or lack of detailed error messages may result in application crashes or incorrect data entries.

SYSTEM ANALYSIS

The **Movie Ticket Booking System** has been developed to overcome the limitations of the traditional, manual ticket booking methods used in cinemas and theatres. The analysis of the current system and the design of the new automated system are essential to understand the scope, objectives, feasibility, and technical requirements of the proposed software.

2.1 EXISTING SYSTEM

Currently, movie ticket booking in many areas is handled either manually at counters or through third-party online platforms. This system is not only time-consuming but also fails to provide a flexible and cost-effective solution for small cinema owners or independent theaters.

Key Characteristics of Existing Systems:

Manual Counter Booking:

- Customers must physically visit the theater.
- Seat availability is not visible beforehand.
- Tickets are handwritten or printed manually.
- Prone to human errors and long queues.

Third-Party Online Portals:

- Platforms like BookMyShow dominate the space.
- Cinemas pay a large commission per ticket sold.
- Limited control for cinema owners over user interface, branding, pricing, and promotional strategies.
- Customers face high service charges and limited customer support options.

Limitations:

- No real-time updates of seat availability in manual systems.
- Long queues during peak hours and holidays.
- Revenue loss to third-party service providers.
- Lack of user control and personalization.
- No direct communication between theaters and customers.

2.2 PROPOSED SYSTEM

The proposed Movie Ticket Booking System is a standalone or web-based application developed using Python (Tkinter GUI or Django) and MySQL database. It aims to digitalize and streamline the process of movie ticket booking for both customers and theater administrators.

Key Features:

For Customers:

- Secure registration and login.
- Browse movies by title, genre, showtime.
- View real-time seat availability.
- Select and book seats via an interactive UI.
- Receive a digital ticket confirmation with seat number.
- View booking history.

For Admin:

- Admin login panel.
- Add/edit/delete movie records, showtimes, and prices.
- Monitor seat bookings in real-time.
- View customer details and transaction records.
- Generate sales reports and statistics.

Advantages Over Existing Systems:

- Real-time, user-friendly interface.
- No dependency on third-party platforms.
- Custom branding and complete administrative control.
- Reduction in manual effort and paper usage.
- Enhanced customer satisfaction and convenience.

2.3 FEASIBILITY STUDY

A feasibility study was conducted across the following dimensions:

1. Technical Feasibility

- Uses commonly available technologies (Python, MySQL).
- Cross-platform support (runs on Windows/Linux).
- Lightweight and fast; no high-end hardware needed.

2. Operational Feasibility

- Simple and intuitive user interface.
- Minimal training required for theater staff and end users.
- Fast and efficient operation reduces workload.
- Quick adaptation due to visual seat maps and guided steps.

3. Economic Feasibility

- Built using free, open-source technologies.
- No licensing or subscription cost.
- Eliminates third-party booking commissions.
- Affordable solution for small and mid-level cinema businesses
-

SYSTEM DESIGN

DESIGN CONSTRAINTS

Design constraints are specific rules or conditions that limit the development process. These constraints define the boundaries within which the system must be built and function. For the Movie Ticket Booking System, the following design constraints were identified:

1. Technology Constraints

- The system must be developed using Python for the backend logic and MySQL for the database.
- GUI is built using either Tkinter (for desktop) or Django/HTML+CSS (for web).
- The system should support Python version 3.6+.
- The application should be compatible with Windows and Linux systems.

2. Performance Constraints

- The system must respond to user input (like booking, login, show selection) within 2 seconds under normal load.
- The system should perform CRUD operations without downtime.

3. User Interface Constraints

- The interface should be simple and intuitive for both Admin and Customers.
- The GUI must follow consistent styling rules (colors, buttons, fonts).
- For the desktop version, the layout must fit screens with a minimum resolution of 1024×768.

4. Security Constraints

- Only authorized users can access the Admin panel.
- All passwords must be securely stored, preferably using hashing algorithms.
- The system must validate inputs to prevent SQL injection and invalid entries.
- User sessions must be protected from hijacking (for web version).

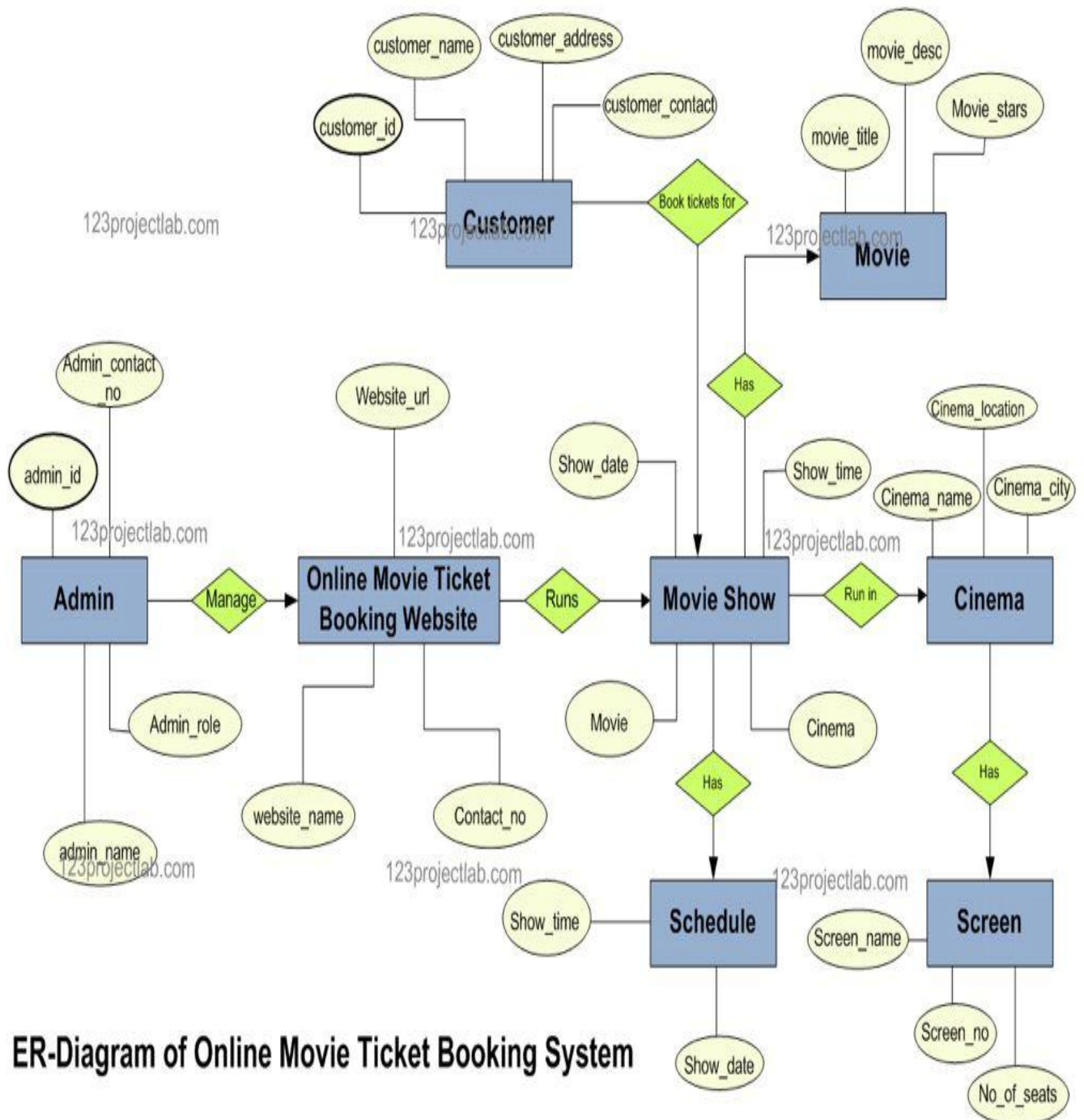
5. Database Constraints

- All tables must follow 3rd Normal Form (3NF) to reduce redundancy.
- Use of foreign keys to maintain referential integrity (e.g., user_id, show_id in bookings).

7. Legal and Licensing Constraints

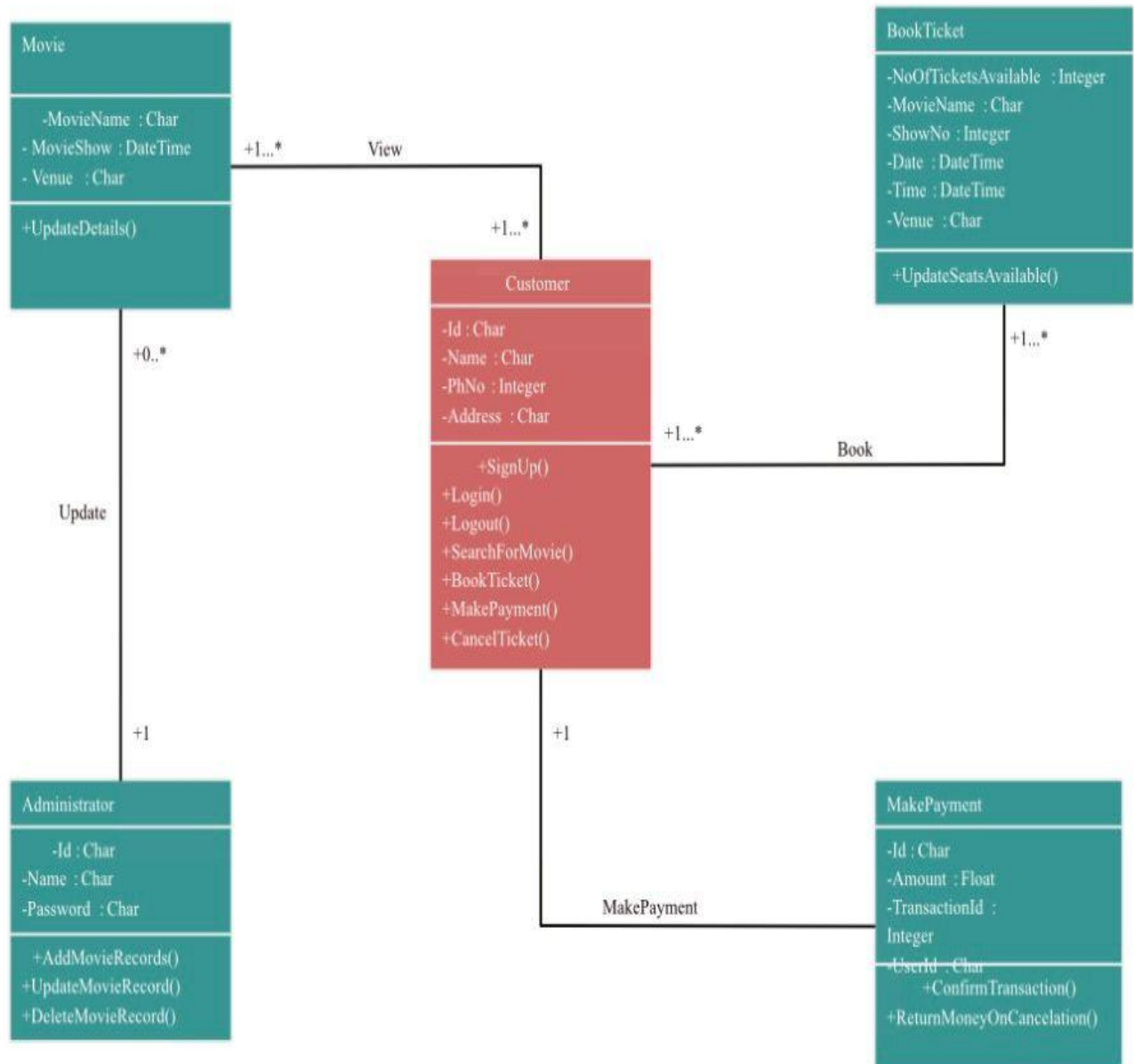
- Use only open-source or freely available libraries.
- The software must comply with data protection guidelines (no unauthorized user tracking or data misuse).

1.1 ENTITY- RELATIONSHIP DIAGRAM:

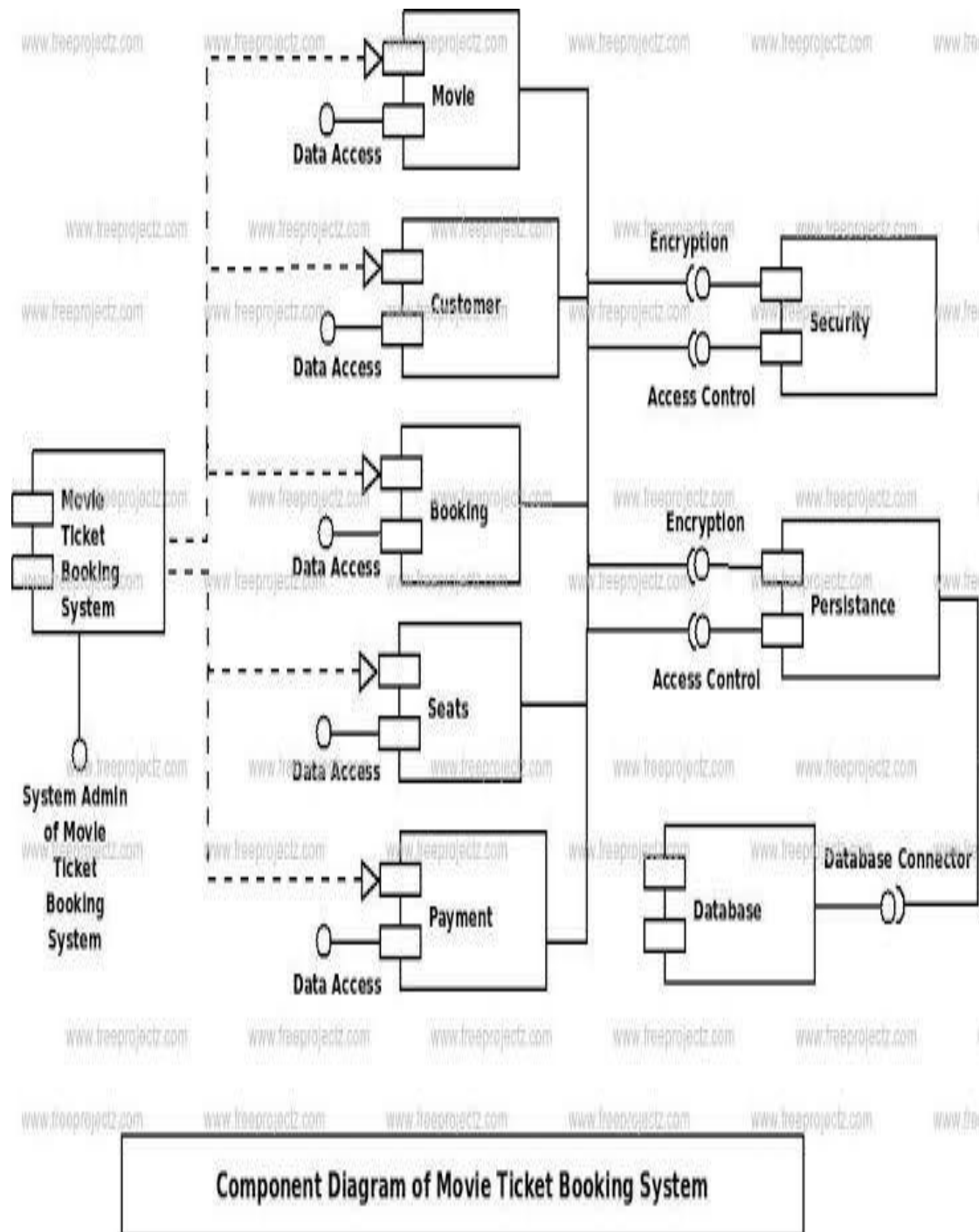


1.2 CLASS DIAGRAM

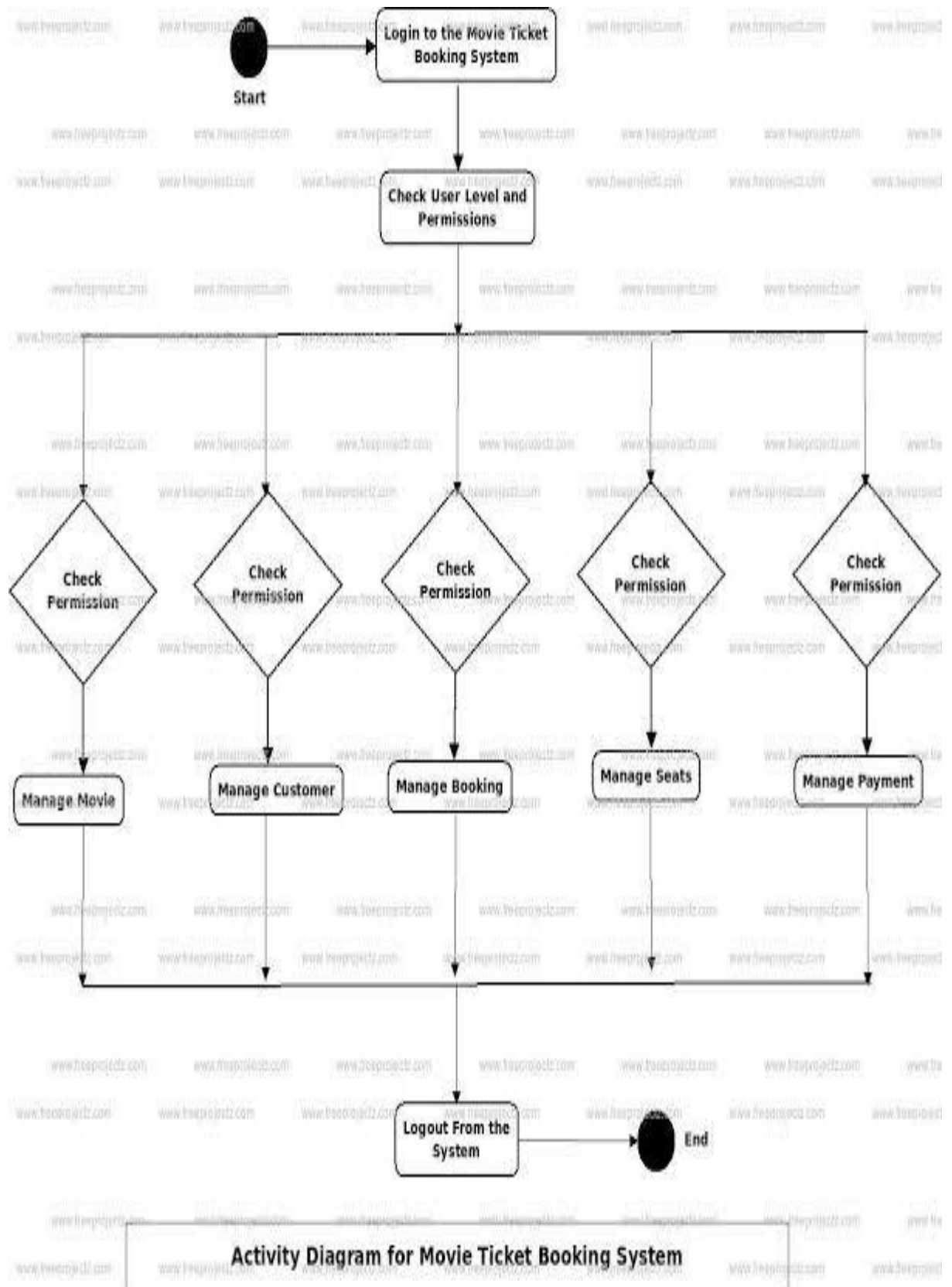
Online Movie Ticket Booking System



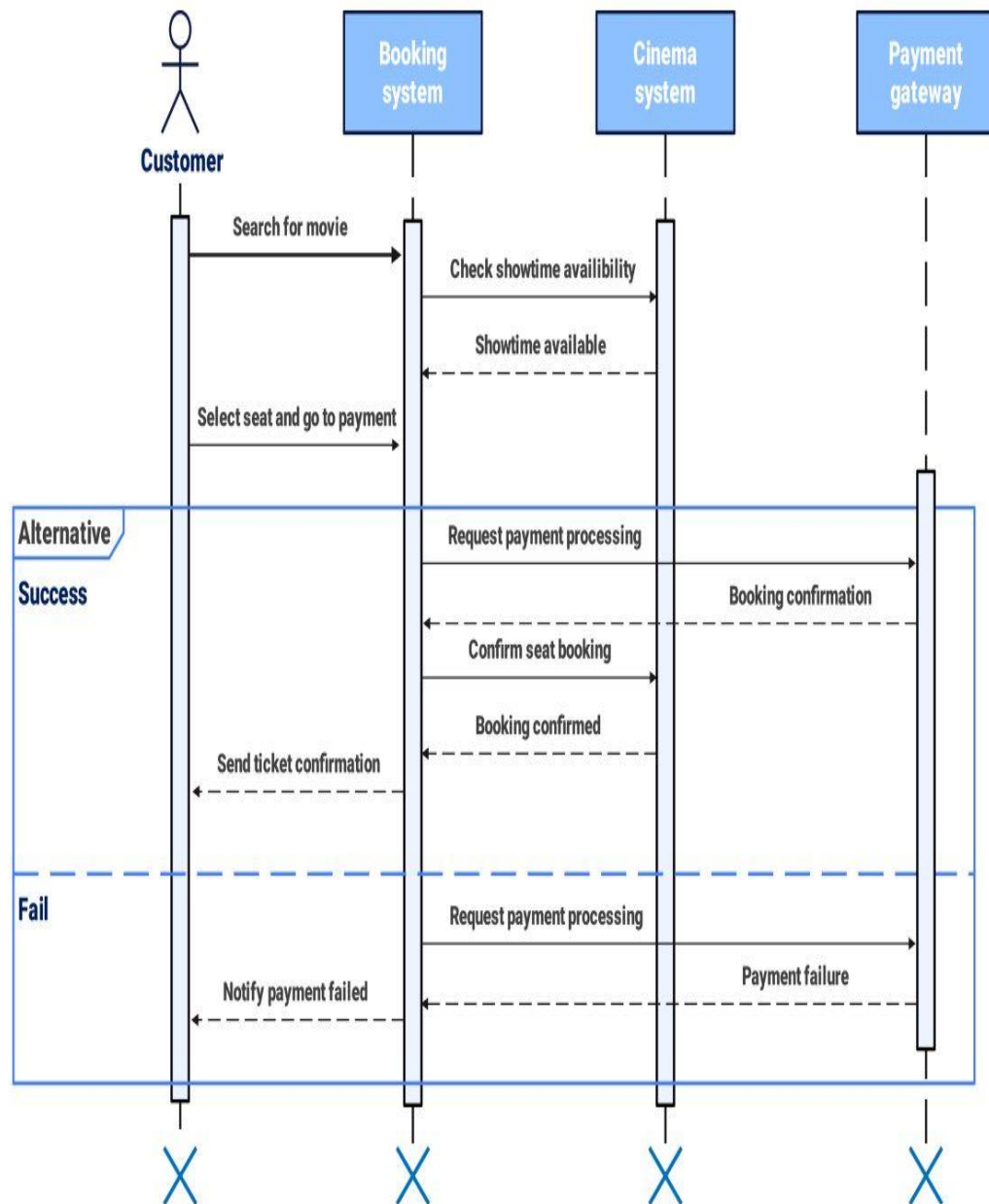
1.3 USE CASE DIAGRAM:



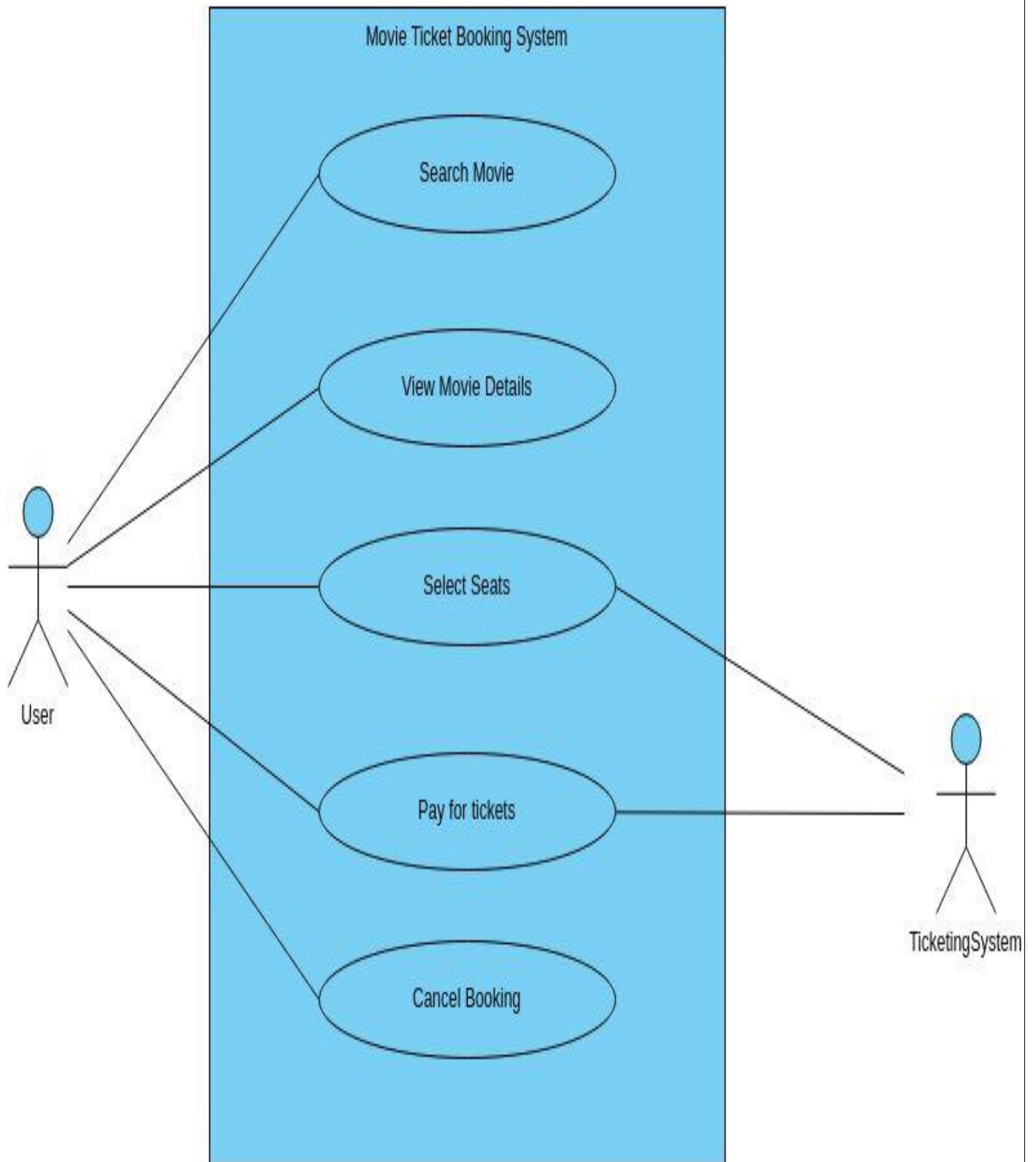
1.4 ACTIVITY DIAGRAMS



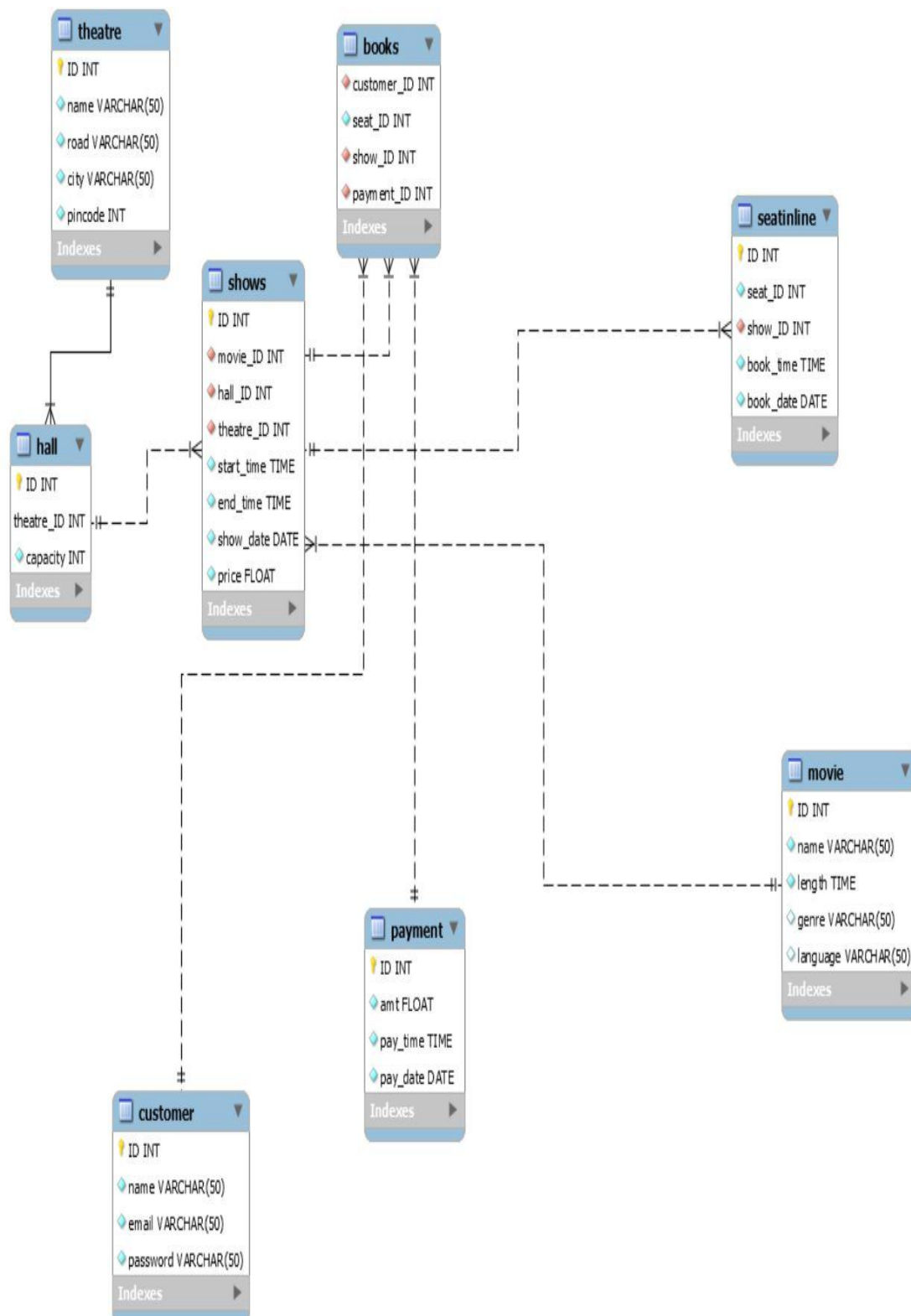
SEQUENCE DIAGRAM



SEQUENCE DIAGRAM



1.1 COMPONENT DIAGRAM



1.2 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 user.	Optional
created_at	DateTime	-	Timestamp when the user account was created.	Default: current timestamp
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

1.3 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 Testing	Ensured smooth communication between modules, such as GUI ↔ Database ↔ Logic Layer.
System Testing	Full end-to-end testing of the entire application including both admin and customer interfaces.
Functional Testing	Verified each feature (e.g., booking, movie addition, login) performed as intended.
Validation Testing	Ensured input fields correctly handled invalid data (e.g., empty fields, wrong formats).
Database Testing	Validated data integrity, foreign key constraints, and correct updates/inserts into the MySQL database.
Regression Testing	Re-ran previous test cases after making bug fixes or adding new features.
User Acceptance Testing (UAT)	Simulated real user interaction to confirm user experience and behavior met expectations.

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:

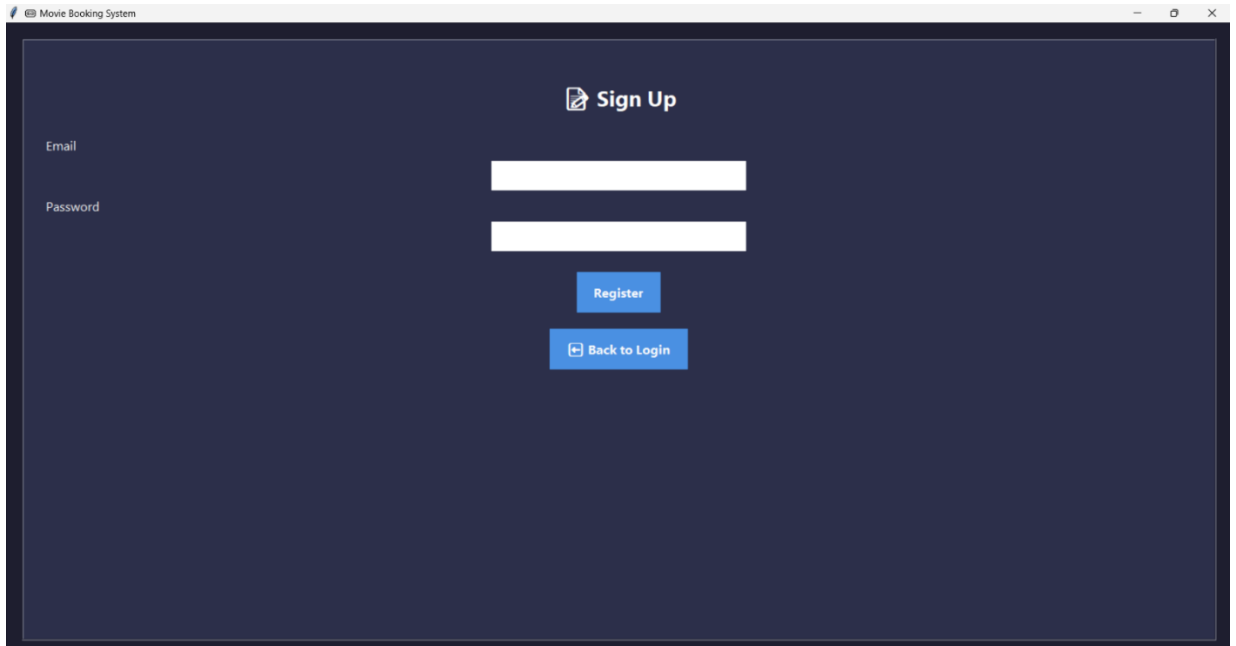
- Passwords were stored in plain format (can be improved with hashing in future versions).
- Only authenticated users could make bookings or request exchanges.

Error Handling:

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

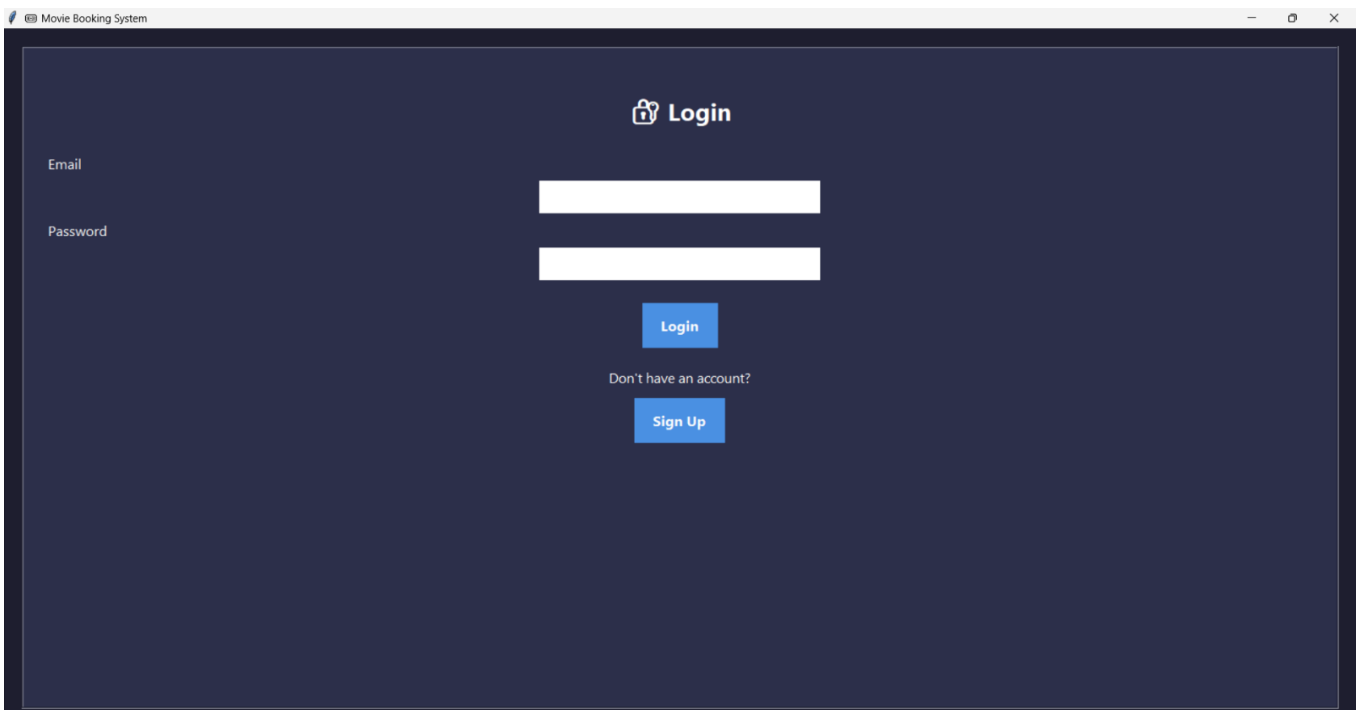
3.10 SCREENSHOTS

User Signup/Register:



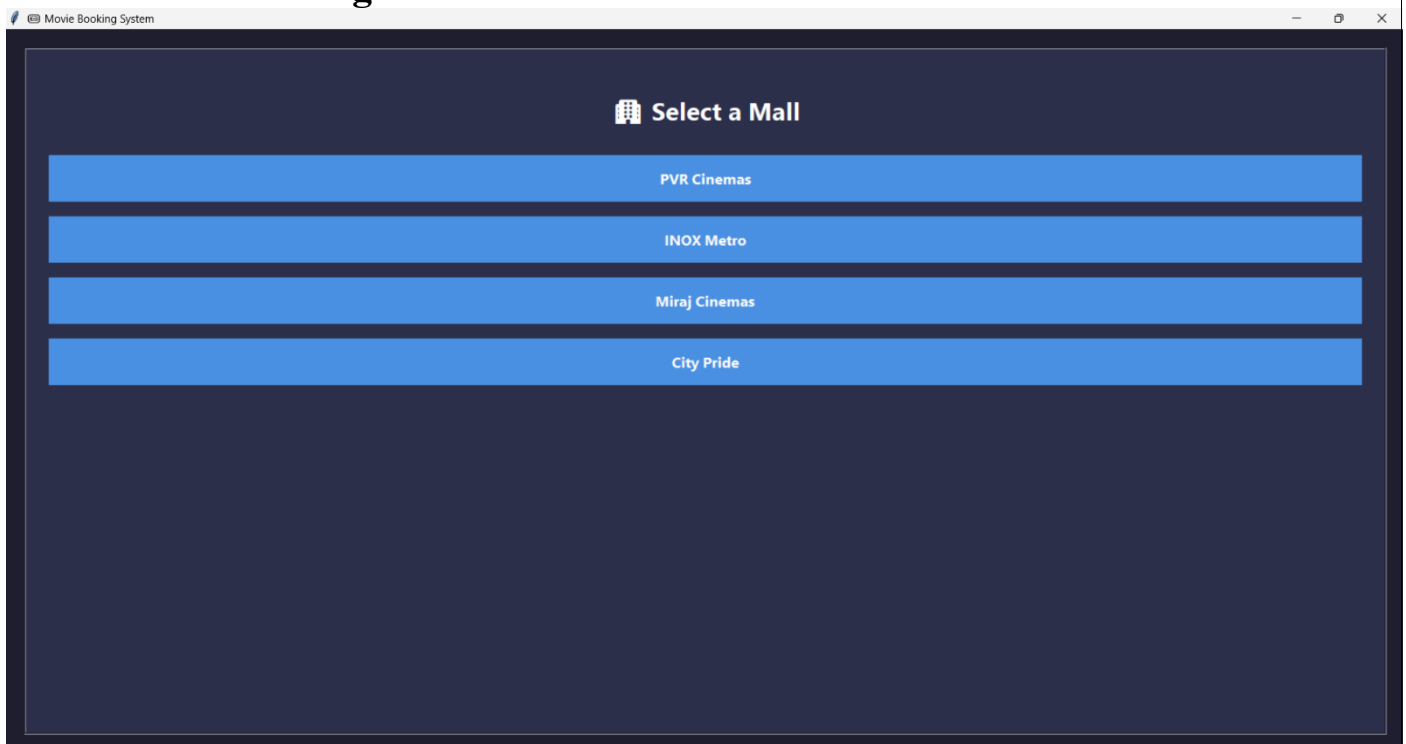
A screenshot of a web browser window titled "Movie Booking System". The page has a dark blue background. At the top center, there is a "Sign Up" heading with a document icon. Below the heading, there are two white input fields for "Email" and "Password". To the left of these fields, the labels "Email"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

User Login:

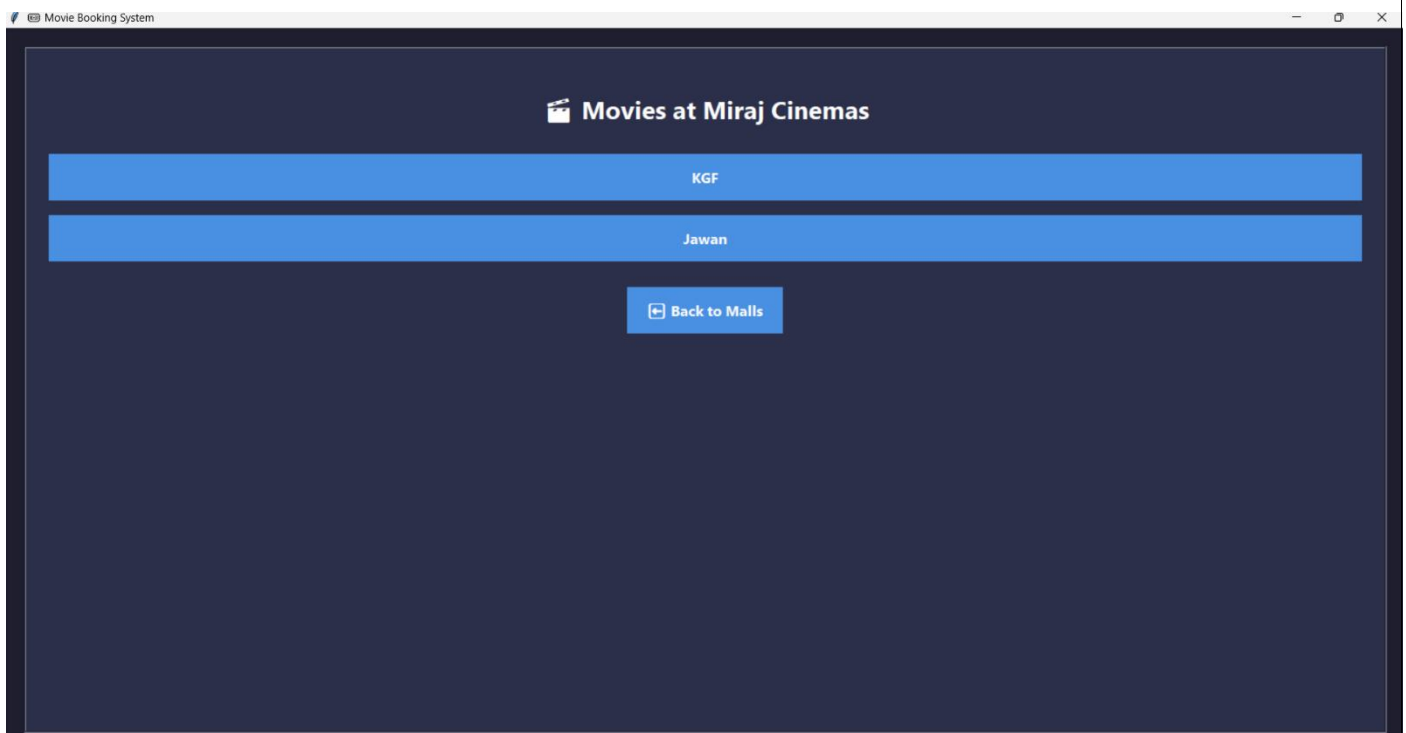


A screenshot of a web browser window titled "Movie Booking System". The page has a dark blue background. At the top center, there is a "Login" heading with a key icon. Below the heading, there are two white input fields for "Email" and "Password". To the left of these fields, the labels "Email" and "Password" 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.

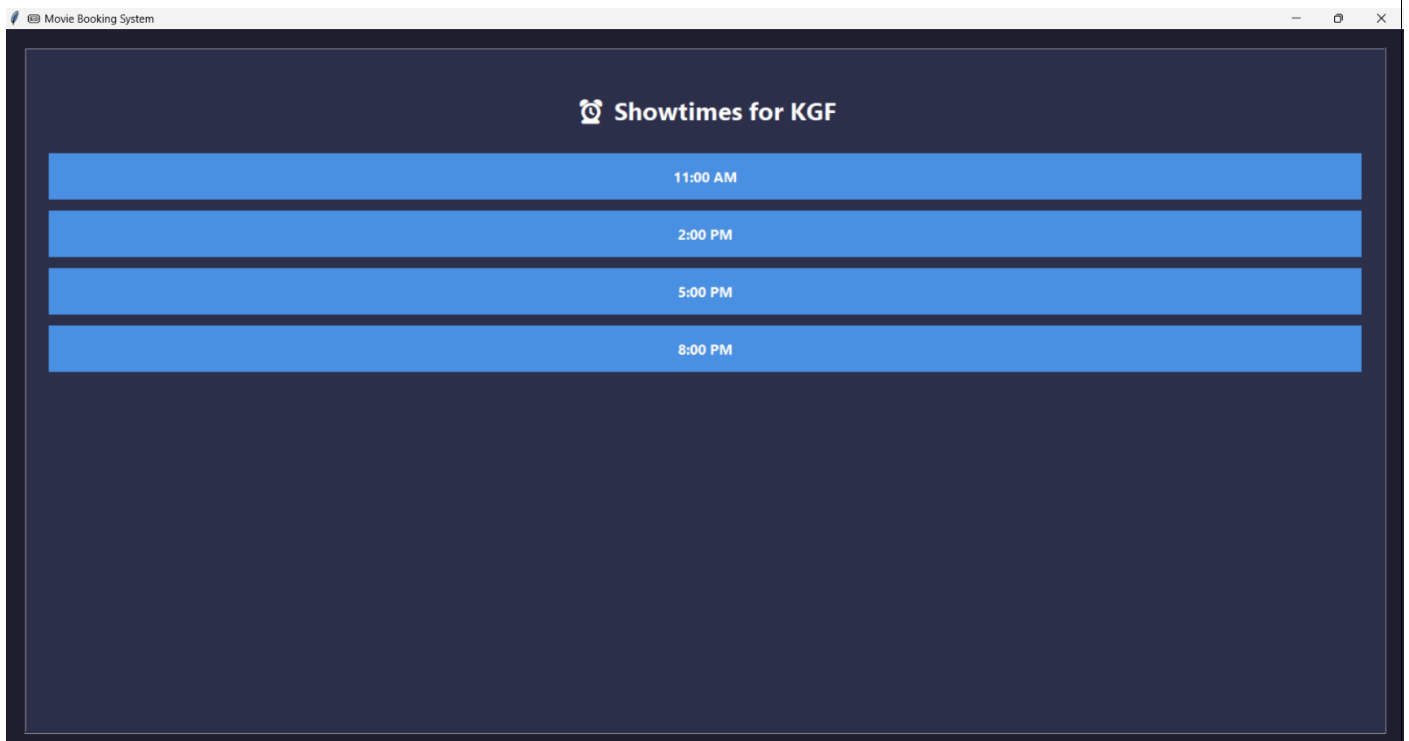
Select a Mall Page:



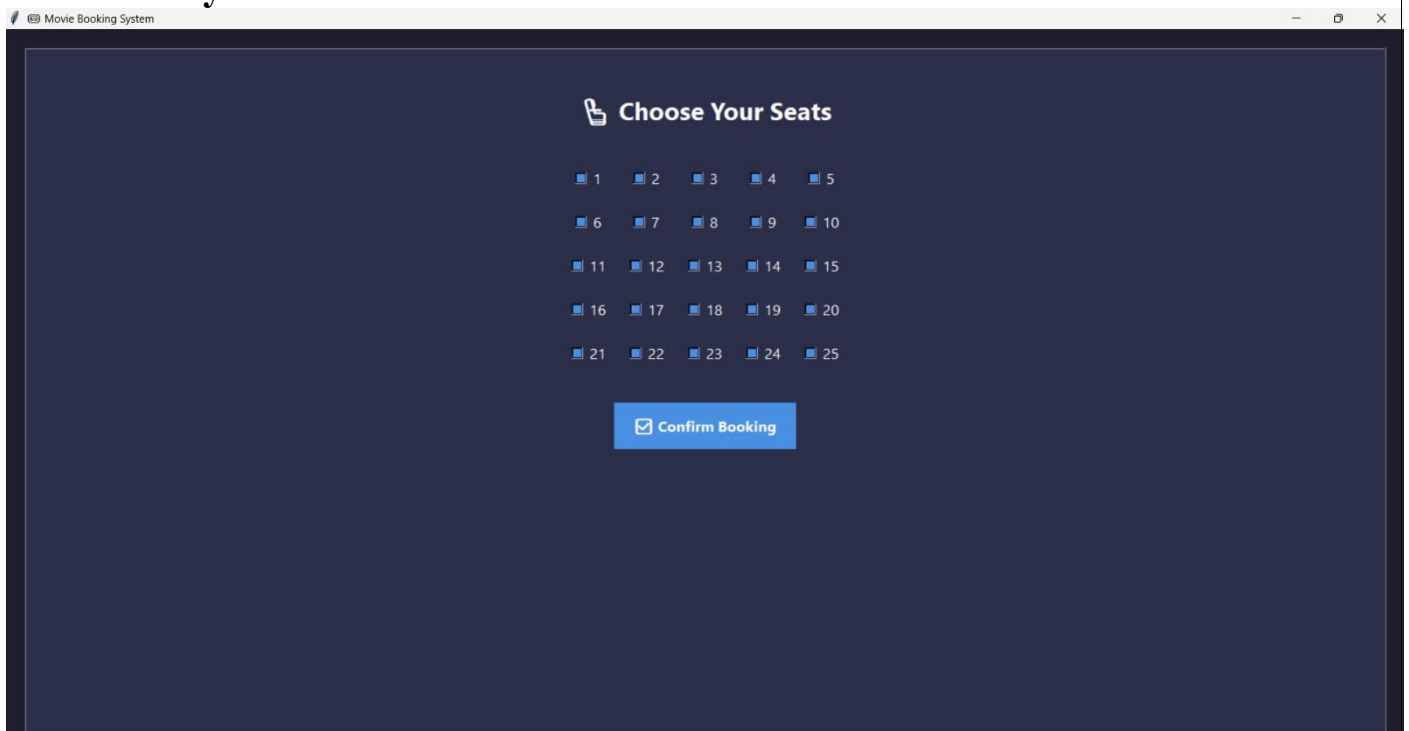
Select Movies page:



All Showtimes:



Choose your Seats:



Booking receipt:

Movie Booking System

Booking Receipt

User: omkardhage608@gmail.com
Mall: Miraj Cinemas
Movie: KGF
Time: 5:00 PM
Seats: 18, 19, 20
Total: ₹450

Print Receipt

Book Again

3.11 REPORTS:

1. Requirement Reports

Covers user and system needs before development.

- SRS (Software Requirements Specification): Defined scope, modules, user roles, and expected functionalities.
- User Requirements: Focused on how users (admin & customers) will interact with the system.
- Hardware/Software Requirements: Listed all necessary hardware components and software tools.

2. System Design Reports

Outlines the architectural and design components used.

- Architecture Design: 3-layered architecture (UI Layer, Logic Layer, Data Layer).
- ER Diagram: Shows relationships between entities (User, Booking, Movie, Show, etc.).
- DFD (Data Flow Diagram): Visualizes how data moves through the system.
- UML Diagrams: Use case diagrams, class diagrams, and sequence diagrams provided to explain behavior.

3. Development Reports

Covers the actual building of the system.

- Programming Language Used: Python (Tkinter for GUI), MySQL for backend.
- Modules Developed:
 - login.py – Login/Register
 - booking.py – Ticket booking logic
 - admin_panel.py – Admin movie/show management
 - exchange_request.py – Request ticket exchange
- Database: Developed using MySQL with all constraints and foreign key relationships.

4. Testing Reports

Validation and verification of the system.

Test Type	Status	Tools Used
Unit Testing	Passed	Manual (assert checks)
Integration Testing	Passed	Function chaining
GUI Testing	Working	Manual using Tkinter GUI
User Acceptance Test	Accepted	Final review with users

- Bugs Resolved: Duplicate registration, wrong seat bookings, login issues
- Test Cases Prepared: For registration, booking, admin login, show creation, etc.

5. Security & Access Reports

Explains access control and protection.

- Authentication System: Validates users and admins separately.

- Role-Based Access: Admin has elevated rights to manage content.
- Input Validation: For form entries (email, password, seat numbers).

6. Performance & Optimization Reports

Shows system behavior under usage.

- Response Time: < 2 seconds per major action (booking/login).
- Database Performance: Queries optimized using indexing (if needed).
- UI Responsiveness: Lightweight Tkinter GUI ensured smooth transitions.
- Optimizations Made: Removed unnecessary DB queries, minimized form reloads.

7. Deployment Reports

Deployment and usage instructions.

- Environment: Local system deployment (Windows/Linux).
- Steps Followed:
 - Installed Python and MySQL
 - Imported database
 - Configured db_connection.py
 - Executed main.py to run the system
- Tools Used: VS Code, MySQL Workbench/XAMPP

8. Documentation Reports

Project documentation at every phase.

- User Manual: Steps to use the software for booking and admin tasks.
- Developer Manual: File-wise structure, database schema, module flow.
- Database Documentation: All table details, keys, constraints explained.
- Screenshots: GUI screenshots included for all main operations.

9. Final Project Summary Reports

Summary of the overall project effort and results.

- Objective Achieved: Developed a complete desktop-based movie ticket booking platform.
- Team Members: Mention student names and roles.
- Outcome:
 - Fully working system
 - GUI-based user and admin panel
 - Ticket booking and exchange features
- Limitations:
 - No online payment gateway
 - No email or mobile verification
- Future Enhancements:
 - Add payment gateway
 - SMS/Email alerts
 - Web version with Django or Flask

IMPLEMENTATION DETAILS

4.1 Software and Hardware Specifications

Software Requirements

Component	Specification
Operating System	Windows 10 or later / Linux / macOS
Programming Language	Python 3.10+
GUI Framework	Tkinter (Python built-in)
Database	MySQL 8.0+
Python Libraries	mysql-connector-python, tkinter, datetime
IDE/Editor	VS Code / PyCharm / IDLE
MySQL Interface	MySQL Workbench or phpMyAdmin

Hardware Requirements

Component	Minimum Requirement
Processor	Dual-Core 2.0 GHz or higher
RAM	4 GB minimum (8 GB recommended)
Storage	500 MB (for code + MySQL data)
Display	1024x768 resolution or higher
Network	Required for future online deployment or real payment integration

7.2 Development and Integration Strategy

The development was performed using a modular and incremental approach:

Development Phases

1. Database Design:

- ER diagram prepared
- Tables created for users, movies, theatres, halls, shows, bookings, seats

2. User Interface Development:

- Tkinter-based GUI for login, registration, movie browsing, seat selection

3. Functionality Implementation:

- Back-end logic integrated for each user flow: register, login, book, pay

4. Admin Panel:

- Separate interface created for admin to manage movies, halls, shows

5. Testing & Debugging:

- Each module tested independently
- Manual integration testing followed

Integration Strategy

- **All components were integrated in stages:**
 - First: GUI ↔ Database connection tested using mysql-connector-python
 - Then: GUI forms (Login, Register) integrated with user table
 - Later: Show management, seat layout, and bookings integrated step-by-step
- Error handling and form validations were added last

7.3 Hosting and Deployment

Local Deployment (Current)

The application runs locally as a desktop application:

- Run Python files locally using the interpreter (python main.py)
- MySQL runs on the local machine or networked LAN database server
- Database access credentials are stored in a separate config file

Optional Web Deployment (Future Enhancement)

If the system is ported to a web version using Flask or Django:

- Backend: Hosted on a cloud server (e.g., AWS EC2, Heroku, or PythonAnywhere)
- Database: MySQL hosted on a cloud platform (e.g., AWS RDS, ClearDB)
- Frontend: Converted from Tkinter to HTML/CSS with Bootstrap
- Domain: Booking system accessible via web browser (e.g., movietickets.com)

TESTING

8.1 Test Plan

The test plan aims to validate the functionality, reliability, and data integrity of the Movie Ticket Booking System. Two main testing strategies were used:

- Black Box Testing: Focused on inputs and outputs without knowledge of internal code.
- White Box Testing: Focused on the internal logic, code paths, and data flow within functions.

Black Box Testing (Data Validation Test Cases):

Test Case ID	Input Scenario	Expected Output	Actual Output	Result
BB001	Empty email and password during login	Error message: "All fields required"	As expected	Pass
BB002	Invalid email format during registration	Error: "Enter a valid email address"	As expected	Pass
BB003	Booking an already booked seat	Error: "Seat already booked"	As expected	Pass
BB004	Payment attempted without selection	Error: "No seats selected"	As expected	Pass

	g seats			
BB005	Register with existing email	Error: "User already exists"	As expected	Pass

White Box Testing (Functional Test Cases):

Test Case ID	Function Tested	Logic/Path Tested	Expected Result	Actual Result	Result
WB001	validate_login()	If user exists and password matches	Return True	True	Pass
WB002	add_movie() (admin)	Insert valid movie data to DB	Return success	Success	Pass
WB003	book_seat()	Check if seat is available before booking	Book seat, update DB	Seat booked	Pass
WB004	show_seats()	Load seat layout from DB	Display seat grid	Displayed	Pass
WB005	register_user()	Check unique email and insert user	Return success or error	As expected	Pass

Results:

Category	Total Tests	Passed	Failed
Black Box Tests	5	5	0
White Box Tests	5	5	0
Total	10	10	0

Conclusion:

The Movie Ticket Booking System passed all planned black box and white box test cases. Key user functionalities such as registration, login, movie selection, show scheduling, seat booking, and payment simulation work reliably.

The system is functionally stable, user-friendly, and ready for deployment.

Basic data validation and security (such as login protection and seat duplication checks) are properly implemented.

LIMITATIONS

Although the Movie Ticket Booking System meets its core functional requirements, several limitations were identified during development and testing:

1. Local Deployment Only

- The application is built for local use and is not accessible over the internet or local network without additional setup.

2. No Real Payment Integration

- The payment module is a simulated interface. No actual transaction or third-party payment gateway (e.g., Razorpay, Stripe) is implemented.

3. Basic GUI Design

- The interface is built using Tkinter, which limits design aesthetics, responsiveness, and cross-platform user experience, especially on mobile devices.

4. Limited Admin Functionality

- The admin panel lacks features like user management, detailed analytics, and permission-based role access.

5. No Email or SMS Notifications

- The system does not send booking confirmations or reminders through email or SMS, reducing the user experience.

6. No Booking Cancellation or Refund

- Once seats are booked, there is no option to cancel or reschedule the booking.

7. Limited Error Handling

- Some parts of the system have basic or no error handling for unexpected inputs or database failures.

8. Scalability Constraints

- The system is not optimized for concurrent usage by multiple users, making it less suitable for large-scale deployment (e.g., multiplex chains).

9. Security Gaps

- User passwords are stored securely, but there is no two-factor authentication, password recovery, or advanced encryption for database connections.

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.htm

REFERENCE

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