MOVIE TICKET MANAGEMENT

A MINI-PROJECT REPORT

Submitted by

HARIKRISHNAN S 241901032

BHARATH KRISHNAN H 241901017

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI - 602105

An Autonomous Institute

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"MOVIE TICKET MANAGEMENT"** is the bonafide work of "HARIKRISHNAN S (241901032), BHARATH KRISHNAN H (241901017)" Who carried out the project work under my supervision.

SIGNATURE

Ms. R. Rupmala

ASSISTANT PROFESSOR

Department of Computer Science

and Engineering (Cyber Security)

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the mini project report **Movie Ticket Management**, submitted as part of the curriculum requirements for the Bachelor of Engineering (B.E) degree affiliated to Anna University, is a bonafide work carried out by us under the supervision of Ms. R. Rupmala, Assistant Professor, Department of Computer Science Engineering and Cyber Security, Rajalakshmi Engineering College, Chennai.

This submission represents our ideas in our own words, and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea, fact, or source in our submission. We understand that any violation of the above will be grounds for disciplinary action by the institute and/or the University and may also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Rajalakshmi Engineering College, **HARIKRISHNAN S**

Chennai **BHARATH H**

November 2025

ABSTRACT

Managing ticket bookings manually can be time-consuming and prone to errors. This project is a desktop-based **Movie Ticket Management System** designed to simplify the process of browsing movies, selecting showtimes, and booking tickets in a fast and user-friendly way.

The application is built using **JavaFX** for an interactive user interface and **MySQL** as the backend database. It focuses on efficiency, accuracy, and a smooth customer experience. The system allows users to view currently running movies, check seat availability, book tickets, and cancel bookings easily.

A well-structured database is used to store movie details, show timings, theatre information, seat layouts, and ticket records. All interactions with the database are done using **JDBC**, ensuring reliable and secure communication between the application and MySQL.

Key features include adding, updating, and deleting movie details (CRUD operations), selecting seats through an intuitive UI, generating digital tickets, and viewing booking history. The system also includes admin controls to manage movies, schedules, and ticket records.

This project provides a practical and scalable solution for theatres or users who want to digitize the movie-ticket booking process with accuracy and ease.

Keywords

JavaFX, MySQL, JDBC, CRUD Operations, Ticket Booking System, Show Timings, Seat Selection, Movie Management, Desktop Application

ACKNOWLEDGEMENT

We like to convey our sincere appreciation to all who have supported and mentored us during the successful completion of this project work.

We express our profound gratitude to **Mr. Benedict J.N**., Associate Professor (SG) and Head of the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for furnishing us with essential resources, support, and an enabling environment to execute this project.

We express our profound gratitude to **Ms. R. Rupmala**, Assistant Professor in the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for her unwavering support, invaluable insights, and collaboration throughout our endeavor.

We would like to convey our gratitude to our faculty members and colleagues for their valuable feedback and encouragement.

We express our gratitude to our families and friends for their steadfast support, patience, and encouragement, which were instrumental in the effective execution of this seminar.

HARIKRISHNAN S

BHARATH H

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	i
	ACKNOWLEDGEMENT	ii
	LIST OF ABBREVIATION	v
	LIST OF FIGURES	vi
1	INTRODUCTION	1
1.1	Project Overview	1
1.2	Scope of the Work	2
1.3	Problem Statement	3
1.4	Aim and Objectives of the Project	4
2	SYSTEM SPECIFICATIONS	5
2.1	Hardware Specifications	5
2.2	Software Specifications	5
3	MODULE DESCRIPTION	6
3.1	User Management Module	6
3.2	Movie Ticket Management Module	6
3.3	Seat Management & Validation Module	7
3.4	Database Module	7

3.5	User Interface Module	7
3.6	ER Diagram	8
3.7	Database Schema of Bookings	9
4	CODING	10
4.1	Booking Token Util	10
4.2	Database Connection	12
4.3	Add Booking Scene	12
5	SCREENSHOTS	14
6	CONCLUSION AND FUTURE ENHANCEMENT	19
6.1	Conclusion	19
6.2	Future Enhancements	19
	REFERENCES	20

LIST OF ABBREVIATION

Abbreviation	Full Term
MTMS	Movie Ticket Management System
UI	User Interface
GUI	Graphical User Interface
DB	Database
SQL	Structured Query Language
JDBC	Java Database Connectivity
CRUD	Create, Read, Update, Delete
UX	User Experience
ID	Identifier
JSON	JavaScript Object Notation
RDBMS	Relational Database Management System
API	Application Programming Interface

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO
3.1	ER Diagram	8
3.2	Database Schema of Credentials	9
3.3	Database Schema of Users	9
5.1	Login Interface Screenshot	14
5.2	Home Page Screenshot	15
5.3	Adding Credential Screenshot	16
5.4	Booking Details Screenshot	17
5.5	Conformation Screenshot	18

CHAPTER 1
INTRODUCTION

1.1 Project Overview

The Movie Ticket Management System is a desktop application developed using JavaFX to simplify and digitalize the movie ticket booking process.
It allows users to browse movies, view show timings, check seat availability, and book tickets through an intuitive interface.
A MySQL database is used to securely store movie details, schedules, seat layouts, and booking records.
Database operations are handled through JDBC, ensuring fast and reliable communication.
The system provides both user and admin functionalities for managing movies, shows, and bookings.
Its modern UI design enhances user experience while maintaining accuracy and efficiency.
By eliminating manual errors and long queues, the system delivers a smooth and organized ticket-booking workflow.
Overall, the project offers a scalable and practical solution for theatre ticket management.

1.2 Scope of the Work

The scope of this project is to develop a complete and efficient movie ticket booking and management system suitable for both users and theatre administrators. Key features include:

• Providing users with options to browse movies, view show timings, and check seat availability.

• Allowing secure and accurate ticket booking, cancellation, and seat selection.

• Ensuring data integrity and preventing issues like double booking or invalid seat allocation.

• Offering an intuitive, modern, and user-friendly graphical interface built with JavaFX.

• Enabling administrators to manage movies, show schedules, theatres, and booking records.

• Integrating with a persistent MySQL database for storing movies, seats, users, and booking details.

1.3 Problem Statement

Managing movie ticket bookings through traditional manual methods presents several challenges for both users and theatre management. Customers often experience long queues, delayed bookings, and a lack of real-time visibility into seat availability, show timings, and movie schedules. These issues frequently result in inconvenience, last-minute booking failures, and uncertainty about seat confirmation. Moreover, manual processes are prone to human errors such as duplicate entries, incorrect seat allocation, and inconsistent record keeping.

From the perspective of theatre administrators, the absence of a centralized digital system makes it difficult to maintain updated movie listings, manage multiple show schedules, monitor booking trends, and track seat occupancy accurately. Data handling becomes inefficient, and generating reports or analyzing demand becomes time consuming.

The core problem this project addresses is the lack of a reliable, efficient, and user-friendly application that provides a seamless ticket-booking experience while ensuring accuracy in seat management and scheduling. The system aims to eliminate manual errors, automate booking operations, and offer a centralized platform that benefits both end users and theatre management. By integrating real-time seat updates, organized movie listings, and a structured database, this project creates a smooth, transparent, and error-free movie ticket booking environment.

1.4 Aim and Objectives of the Project

The main aim of this project is to develop an efficient, accurate, and user-friendly movie ticket management system that simplifies the process of browsing movies, selecting showtimes, and booking tickets. The system is designed to enhance user convenience while helping theatres manage schedules, bookings, and seat availability effectively.

The specific objectives are:

• To provide users with real-time access to movie listings, show timings, and seat availability.

• To implement a reliable ticket booking workflow that prevents double-booking and ensures accurate seat allocation.

• To design a modular and scalable system architecture that supports easy maintenance and future enhancements.

• To build an intuitive JavaFX-based user interface that enables seamless movie selection, seat selection, and booking confirmation.

• To integrate a MySQL database for secure and efficient handling of movies, schedules, theatres, and booking records.

• To include an admin module for managing movies, show schedules, theatre screens, and booking data.

CHAPTER 2
SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Minimum Specification
Processor	Dual-core 2.0 GHz or higher
Memory (RAM)	4GB (Minimum), 8GB (Recommended)
Storage	100MB free space
Display	1280x720 resolution

2.2 SOFTWARE SPECIFICATIONS

Component	Specification
Operating System	Windows 10/11, macOS 10.15+, Linux
Front-End	JavaFX 21.0.6
Back-End	MySQL 8.0.33
Core Language	Java 25
Dependencies	Maven (Build Tool), JUnit (Testing), HikariCP (Connection Pooling)

CHAPTER 3

MODULE DESCRIPTION

The application is built on a modular architecture, with a clear separation of concerns across five primary modules:

3.1 User Management Module

This module handles all user-related operations within the Movie Ticket Management System. It is responsible for:

• **User Registration:** Allowing new users to create an account by providing basic information such as name, email, and contact details.

• **User Login:** Verifying registered users through secure authentication before granting access to booking features.

• **Role Management:** Differentiating between regular users (customers) and administrators (theatre staff) with appropriate access levels.

• **Profile Handling:** Enabling users to update personal information and view their booking history.

• **Session Management:** Maintaining active user sessions during the booking process to ensure smooth navigation and preventing unauthorized access.

• **Security Measures:** Preventing unauthorized activities by validating user inputs and ensuring only authenticated users can make or modify bookings.

3.2 Movie & Show Management Module

This module represents the core business logic of the Movie Ticket Management System. It manages the complete lifecycle of movies, shows, and theatre schedules.

• **CRUD Operations:** Creating, Reading, Updating, and Deleting movie details, show timings, theatre screens, and seat layouts.

• **Show Scheduling:** Allowing administrators to add or modify showtimes, assign screens, and manage daily or weekly schedules.

• **Real-Time Availability:** Updating seat availability instantly when bookings or cancellations occur to ensure accurate, conflict-free data.

• **Movie Information Handling:** Managing movie metadata such as title, genre, duration, language, and description for display in the user interface.

3.3 Seat Management & Validation Module

This module handles all seat-related operations to ensure accurate, real-time booking and prevention of booking conflicts.

• **Seat Allocation Logic:** Managing the real-time assignment of seats during booking to prevent duplicate or conflicting reservations.

• **Availability Tracking:** Updating seat availability immediately after each booking or cancellation to ensure consistent, conflict-free data.

• **Validation Rules:** Ensuring users can only select valid seats based on screen layout, availability, and show timing constraints.

• **Seat Layout Handling:** Loading and displaying dynamic seating charts for each theatre screen, including categories like Regular, Premium, or VIP.

• **Consistency Checks:** Verifying that selected seats remain available at the moment of payment/confirmation to avoid race conditions during peak hours.

3.4 Database Module

This module is responsible for managing all interactions with the backend **MySQL** database, ensuring reliable and efficient data storage for movies, shows, users, and bookings.

• **Connection Management:** Establishing and maintaining stable database connections using optimized JDBC configurations for smooth interaction with the application.

• **Data Access Layer (Repository):** Implementing Data Access Objects (DAOs) such as *MovieRepository*, *ShowRepository*, *SeatRepository*, and *BookingRepository* to abstract database operations from the business logic.

• **CRUD Operations:** Handling all Create, Read, Update, and Delete operations for movies, show schedules, seat availability, and booking records.

• **Prepared Statements:** Using prepared statements for all SQL queries to ensure security and protect the system from SQL injection attacks.

• **Data Integrity:** Enforcing constraints and relational integrity to prevent duplicate bookings, invalid schedules, or inconsistent seat states.

3.5 User Interface Module

The presentation layer of the Movie Ticket Management System, developed using **JavaFX**, focuses on delivering a smooth and intuitive user experience.

• **Scene Management:** Managing navigation between views such as Movie Listing, Show Selection, Seat Layout, Booking Summary, Payment, and Admin Dashboard.

• **User Experience:** Designing a visually appealing, responsive interface with clear layouts, easy navigation, and interactive elements to enhance the booking flow.

• **Dynamic Displays:** Rendering real-time seat availability, show timings, and theatre details using visually structured components.

• **Input Validation:** Ensuring all user inputs—such as seat selection, user details, and show selection—are validated for accuracy and usability.

• **Accessibility:** Providing clean fonts, readable layout, and interactive elements to support a smooth user experience across different screen sizes.

3.6 ER Diagram

The above diagram represents the E-R model of the **Movie Ticket Management System**.
It consists of several key entities:
Users — stores details of registered users such as user_id, name, email, and role (customer/admin).
Movies — contains movie-specific information such as movie_id, title, genre, duration, and language.
Shows — represents scheduled showtimes, including show_id, movie_id, screen_id, date, and time.
Screens — defines the theatre screens with screen_id, screen_name, and seating capacity.
Seats — stores individual seat records for each screen, including seat_id, seat_number, category, and screen_id.
Bookings — maintains booking information such as booking_id, user_id, show_id, seat_id, and booking_timestamp.
A set of relationships exist among these entities:

- A **Movie** can have multiple **Shows**, forming a one-to-many (1-M) relationship.
- Each **Show** occurs in one **Screen**, but a screen can host multiple shows (1-M).
- Each **Screen** contains multiple **Seats**, also forming a one-to-many (1-M) relationship.
- A **User** can make multiple **Bookings** (1-M).
- Each **Booking** is associated with one **Show** and one **Seat**, linking the user's reservation to a specific time and place.

The ER model ensures structured data storage, accurate relationships, and efficient retrieval of booking and movie information.

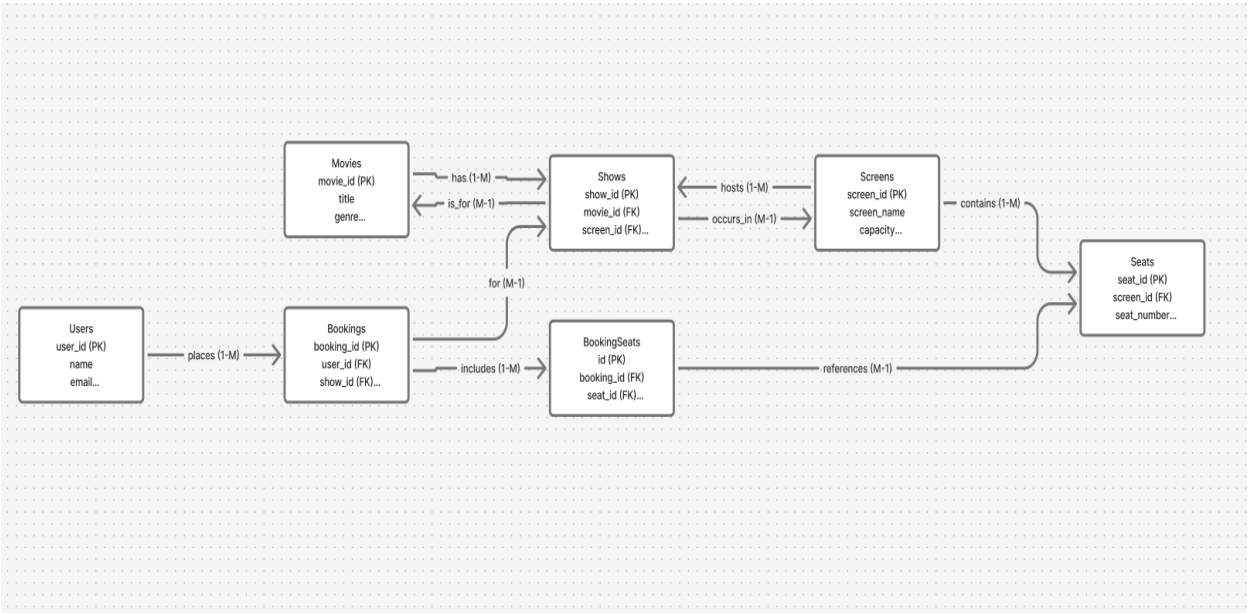

Fig.3.1 ER Diagram

3.7 Database Schema of Bookings

Explanation: Database Schema of Bookings

The **bookings** table stores all movie ticket reservations made by users.

Each booking is linked to a specific user through the user_id foreign key.

The table records details such as show_id, seat information, booking time, and total amount.

This structure ensures accurate tracking of all user ticket bookings.

Column Name	Data Type	Description
booking_id	INT (PK)	Unique ID for each booking
user_id	INT (FK)	Identifies the user who made the booking
show_id	INT (FK)	Identifies the selected movie show
seat_number	VARCHAR(10)	Seat reserved for the user

Fig.3.2 Database Schema of Credentials

Explanation: Database Schema of Users

The **users** table stores the account information of each registered customer or administrator.

Each user has a unique ID along with essential details such as name, email, and role.

If password authentication is used, passwords are securely stored as hashed values rather than plain text.

This table forms the core of the system's user management and maintains secure access to booking features.

Column Name	Data Type	Description
user_id	INT (PK)	Unique ID for each registered user
name	VARCHAR(100)	Full name of the user
email	VARCHAR(150)	User's unique email / login ID
password_hash	VARCHAR(255)	Securely stored hashed password

Fig.3.3 Database Schema of Users

Chapter 4

CODING

This chapter includes key code components from the Movie Ticket Management project, focusing on Booking Token, Database Connection, and Add Booking Scene implementations.

4.1 BookingTokenUtil

The BookingTokenUtil class generates a secure booking reference/token for each confirmed reservation. Tokens are short, URL-safe strings created from a UUID and HMAC-SHA256 signature to prevent tampering and to provide a verifiable reference that can be used in tickets and email confirmations.

```
package com.example.movie_ticket_system.utils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import java.util.Base64;
import java.util.UUID;

public class BookingTokenUtil {

    private static final String HMAC_ALGO = "HmacSHA256";
    private static final byte[] SECRET_KEY = new byte[32];

    static {
        new SecureRandom().nextBytes(SECRET_KEY);
    }

    public static String generateToken() throws Exception {
        String uuid = UUID.randomUUID().toString();
        byte[] signature = hmac(uuid.getBytes(StandardCharsets.UTF_8));
        byte[] combined = new byte[uuid.getBytes().length + signature.length];
        System.arraycopy(uuid.getBytes(StandardCharsets.UTF_8), 0, combined, 0,
uuid.getBytes().length);
        System.arraycopy(signature, 0, combined, uuid.getBytes().length, signature.length);
        return Base64.getUrlEncoder().withoutPadding().encodeToString(combined);
    }

    public static boolean verifyToken(String token) throws Exception {
        byte[] decoded = Base64.getUrlDecoder().decode(token);
        int uuidLen = 36;
        if (decoded.length <= uuidLen) return false;

        byte[] uuidBytes = new byte[uuidLen];
        System.arraycopy(decoded, 0, uuidBytes, 0, uuidLen);
```

```
        byte[] sig = new byte[decoded.length - uuidLen];
        System.arraycopy(decoded, uuidLen, sig, 0, sig.length);

        byte[] expected = hmac(uuidBytes);
        if (expected.length != sig.length) return false;

        for (int i = 0; i < sig.length; i++) {
            if (expected[i] != sig[i]) return false;
        }
        return true;
    }

    private static byte[] hmac(byte[] data) throws Exception {
        Mac mac = Mac.getInstance(HMAC_ALGO);
        mac.init(new SecretKeySpec(SECRET_KEY, HMAC_ALGO));
        return mac.doFinal(data);
    }
}
```

4.2 Database Connection

The DatabaseConnection class establishes a secure connection to the MySQL database using JDBC, enabling the app to store and retrieve credentials.

```
package com.example.movie_ticket_system.database;


import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;


public class DatabaseConnection {


    private static final String URL = "jdbc:mysql://localhost:3306/movie_ticket_db";

    private static final String USER = "root";

    private static final String PASSWORD = "root";


    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(URL, USER, PASSWORD);

    }

}
```

4.3 Add Booking Scene

The AddBookingScene class provides a JavaFX UI to create a new booking. It collects show, seat and user details, generates a secure booking token, and saves the booking using a prepared statement. The UI notifies the user on success/failure.

```
package com.example.movie_ticket_system.scenes;

import com.example.movie_ticket_system.database.DatabaseConnection;
import com.example.movie_ticket_system.utils.BookingTokenUtil;
import javafx.geometry.Insets;
```

```
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import java.sql.Connection;
import java.sql.PreparedStatement;

public class AddBookingScene {

    public static Scene create(Stage stage) {

        Label heading = new Label("Add Booking");
        heading.setStyle("-fx-font-size: 20px; -fx-font-weight: bold;");

        Label userIdLabel = new Label("User ID");
        TextField userIdField = new TextField();

        Label showIdLabel = new Label("Show ID");
        TextField showIdField = new TextField();

        Label seatLabel = new Label("Seat Number");
        TextField seatField = new TextField();

        Label amountLabel = new Label("Total Amount");
        TextField amountField = new TextField();

        Button save = new Button("Book");
        Button back = new Button("Back");

        save.setOnAction(e -> {
            try {
                int userId = Integer.parseInt(userIdField.getText().trim());
                int showId = Integer.parseInt(showIdField.getText().trim());
                String seat = seatField.getText().trim();
                double amount = Double.parseDouble(amountField.getText().trim());

                String token = BookingTokenUtil.generateToken();

                String insert = "INSERT INTO bookings (user_id, show_id, seat_number, total_amount,
booking_time, status, booking_token) VALUES (?, ?, ?, ?, CURRENT_TIMESTAMP, ?, ?)";

                Connection conn = DatabaseConnection.getConnection();
                PreparedStatement stmt = conn.prepareStatement(insert);

                stmt.setInt(1, userId);
                stmt.setInt(2, showId);
                stmt.setString(3, seat);
                stmt.setDouble(4, amount);
                stmt.setString(5, "CONFIRMED");
                stmt.setString(6, token);
```

```
            int rows = stmt.executeUpdate();

            if (rows > 0) {
                new Alert(Alert.AlertType.INFORMATION, "Booking created. Reference: " +
token).showAndWait();
            } else {
                new Alert(Alert.AlertType.ERROR, "Could not create booking.").showAndWait();
            }

        } catch (Exception ex) {
            new Alert(Alert.AlertType.ERROR, "Error: " + ex.getMessage()).showAndWait();
        }
    });

    GridPane grid = new GridPane();
    grid.setVgap(10);
    grid.setHgap(10);

    grid.add(userIdLabel, 0, 0);
    grid.add(userIdField, 1, 0);

    grid.add(showIdLabel, 0, 1);
    grid.add(showIdField, 1, 1);

    grid.add(seatLabel, 0, 2);
    grid.add(seatField, 1, 2);

    grid.add(amountLabel, 0, 3);
    grid.add(amountField, 1, 3);

    HBox buttons = new HBox(10, save, back);
    buttons.setAlignment(Pos.CENTER_RIGHT);

    VBox root = new VBox(15, heading, grid, buttons);
    root.setPadding(new Insets(20));
    root.setAlignment(Pos.TOP_CENTER);

    return new Scene(root, 450, 320);
  }
}
```

CHAPTER 5
SCREENSHOTS

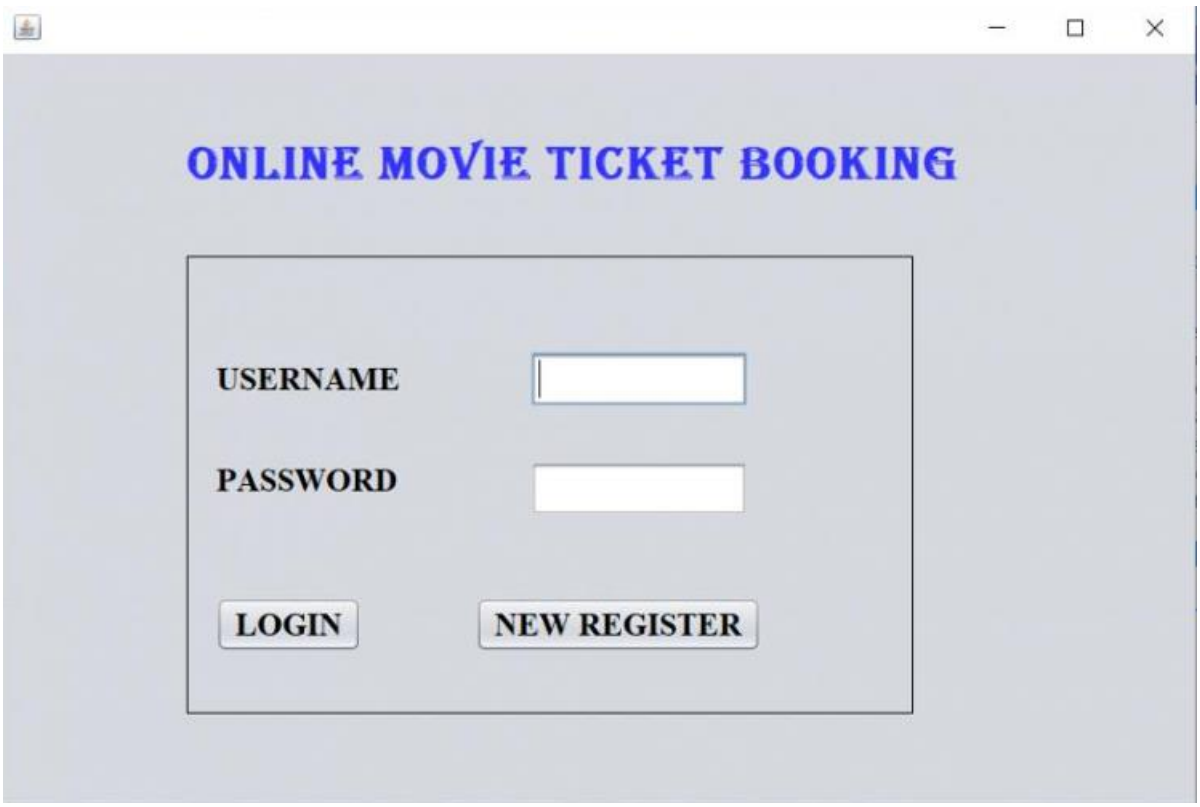

Fig.5.1 Login Interface Screenshot

Fig.5.2 Home Page Screenshot

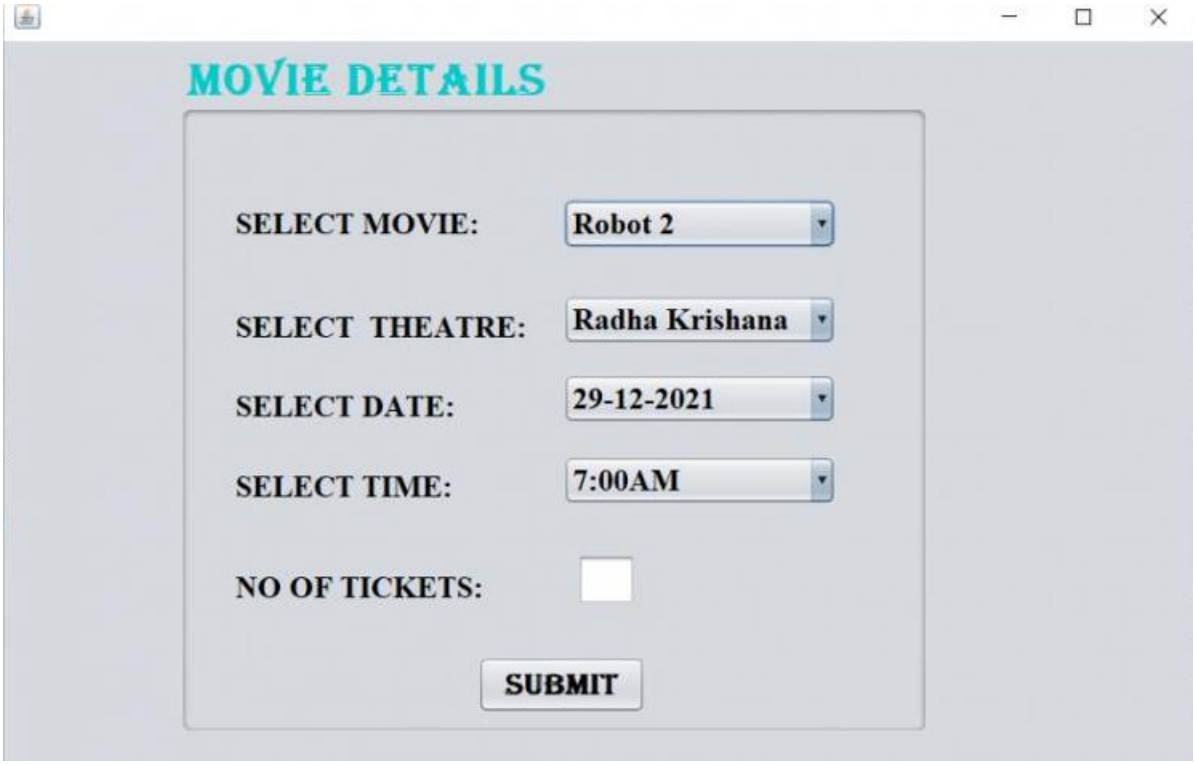

Fig.5.3 Adding Credential Screenshot

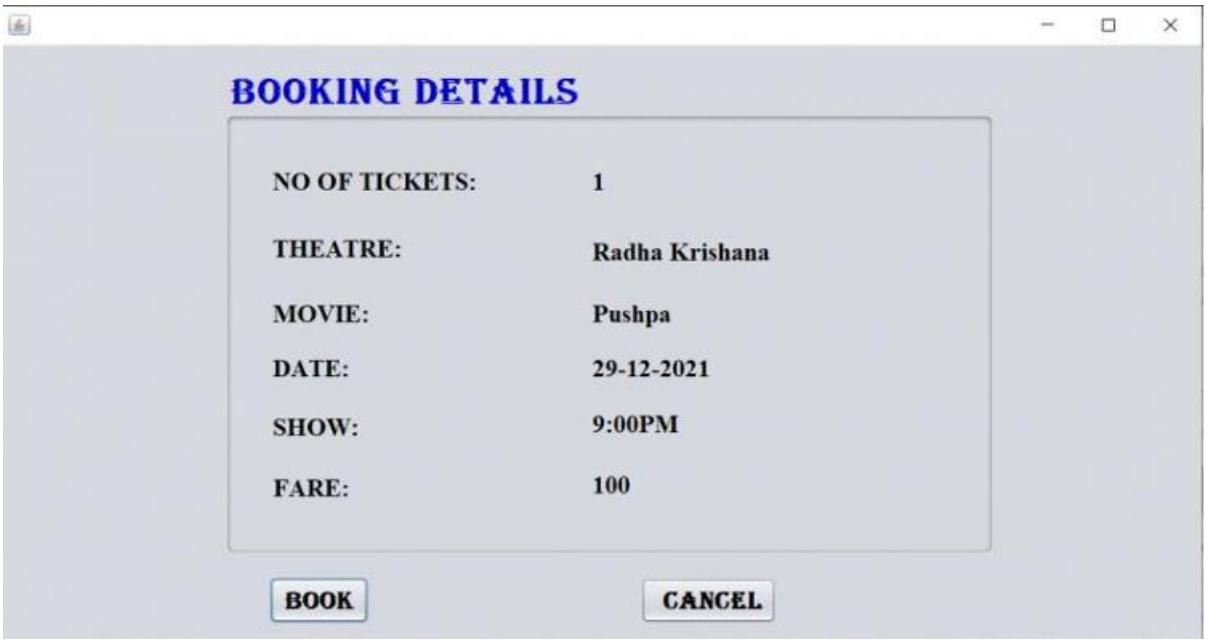

Fig.5.4 Booking Details Screenshot

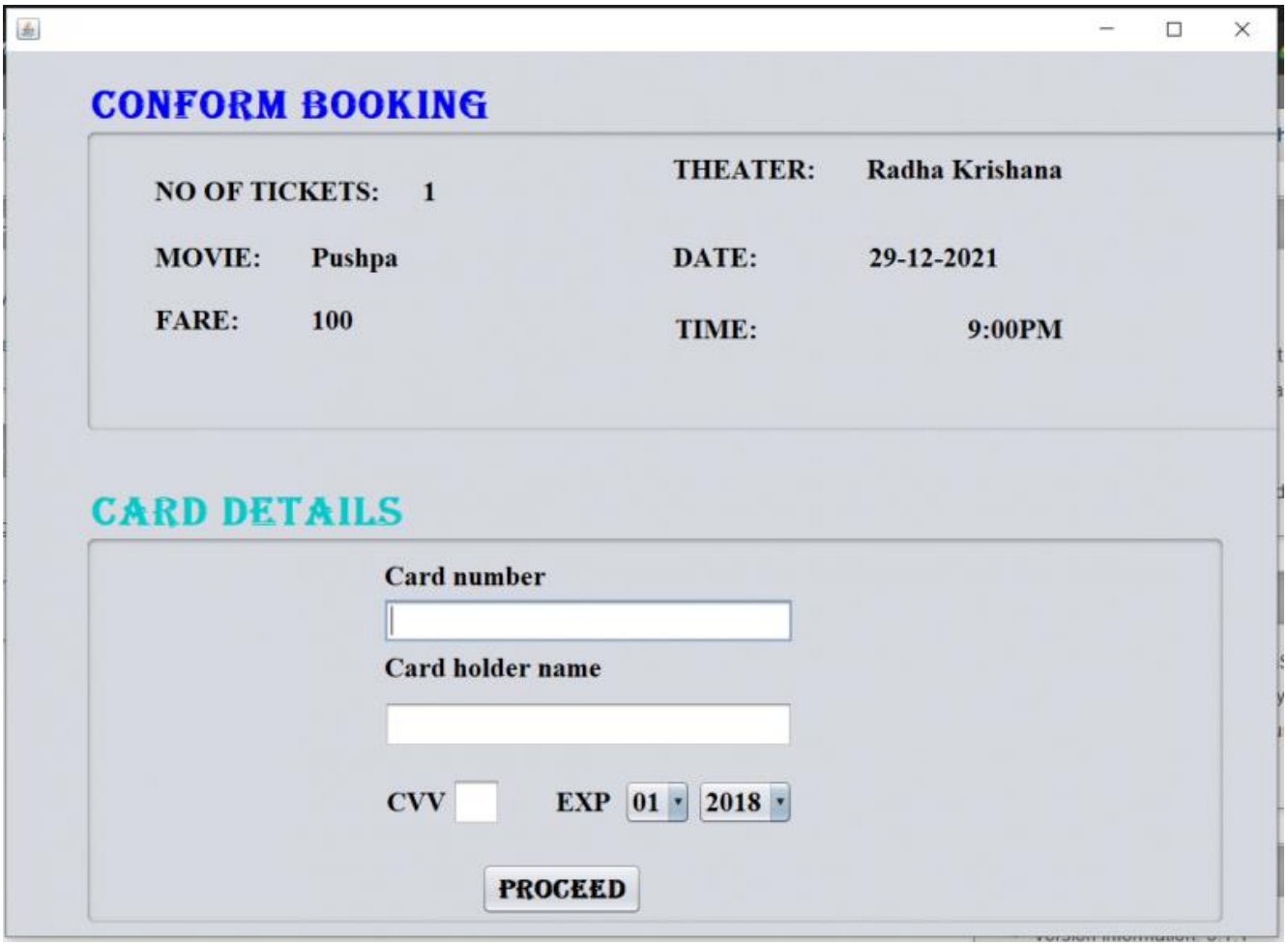

Fig.5.5 Conformation Screenshot

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The Movie Ticket Management System successfully delivers an efficient, user-friendly, and well-organized platform for managing movie bookings. By integrating a reliable MySQL database, modular JavaFX interface, and streamlined booking workflow, the application effectively addresses the challenges of manual ticket handling. The system enables users to browse movies, check availability, select seats, and book tickets with ease, while administrators can manage shows and theatre screens conveniently. Overall, the project provides a practical and scalable solution that enhances both user experience and theatre operations.

6.2 Future Enhancements

The system is designed for scalability and can be extended with several valuable features in future versions:

• **Online Payment Integration:** Adding support for secure payment gateways to allow users to complete bookings digitally.

• **Mobile Application Support:** Developing Android and iOS apps for easier, on-the-go movie browsing and ticket booking.

• **QR-Based E-Tickets:** Generating digital tickets with QR codes for quick and contactless entry at theatres.

• **Dynamic Pricing:** Implementing intelligent price adjustments based on seat category, demand, and showtime.

• **Notification Alerts:** Providing SMS or email reminders for upcoming shows, seat confirmations, and cancellations.

• **Analytics Dashboard:** Offering theatre administrators insights into occupancy trends, popular movies, and peak booking times.

REFERENCES

[1] Oracle, Java Cryptography Architecture (JCA) Reference Guide, Oracle Corporation, 2024. [Online]. Available: https://docs.oracle.com/en/java/

[2] Oracle, MySQL 8.0 Reference Manual, Oracle Corporation, 2024. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/

[3] OpenJFX Community, JavaFX Documentation and Tutorials, 2024. [Online]. Available: https://openjfx.io

[4] S. P. Chacon and M. Straub, Maven: The Definitive Guide, O'Reilly Media, 2023.

[5] B. Schneier , Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., Wiley, 1996.

[6] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2003.

[7] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008.

[8] Oracle, JDBC API Specification, Oracle Corporation, 2024. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/