

# **MOVIE TICKET BOOKING PLATFORM USING DJANGO**

*Submitted by,*

**Vignesh V - 20211CAI0140**

*Under the guidance of,*

**Ms. Deepthi S**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (Artificial Intelligence and  
Machine Learning)**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**MAY 2025**

# **PRESIDENCY UNIVERSITY**

## **PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

### **CERTIFICATE**

This is to certify that the Internship report **MOVIE TICKET BOOKING PLATFORM USING DJANGO** being submitted by Vignesh V bearing roll number 20211CAI0140 in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

**Ms. Deepthi S**  
Assistant Professor  
School of CSE  
Presidency University

**Dr. Zafar Ali Khan N**  
Associate Professor  
Selection Grade & HOD  
School of CSE & IS  
Presidency University

**Dr. MYDHILI NAIR**  
Associate Dean  
PSCS  
Presidency University

**Dr. SAMEERUDDIN KHAN**  
Pro-Vice Chancellor - Engineering  
Dean –PSCS / PSIS  
Presidency University

# **PRESIDENCY UNIVERSITY**

## **PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

### **DECLARATION**

I hereby declare that the work, which is being presented in the report entitled **MOVIE TICKET BOOKING PLATFORM USING DJANGO** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence and Machine Learning)**, is a record of my own investigations carried under the guidance of **Ms. Deepthi S, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

**Student Name**

**Roll No**

**Signature**

Vignesh V

20211CAI0140

# INTERNSHIP COMPLETION CERTIFICATE



CIN: U72200KA2007PTC044701

Date: 10/05/2025

## INTERNSHIP COMPLETION CERTIFICATE

This is to certify that **Vignesh V**, has successfully completed an internship at **Robowaves** from **01/02/2025 to 01/05/2025**.

During this period, he worked on **Robowaves**, gaining hands-on experience and industry exposure.

We appreciate **Vignesh V** for their dedication, hard work, and commitment throughout the internship. We believe this experience will help them excel in their professional career.

We wish them the best for their future endeavours.

Yours Sincerely,

For Robowaves

(A Unit of Test Yantra Software Solutions (India) Pvt. Ltd.)

"System Generated letter no need of Signature"



## ABSTRACT

The Online Movie Ticket Booking Platform is a sophisticated web application built with the Django framework, aimed at modernizing and simplifying the movie ticket purchasing experience. Featuring a user-friendly and responsive design, the platform allows users to explore a wide range of movies, view current and upcoming showtimes, choose seats, and securely finalize ticket purchases, all from a unified platform. The system prioritizes modularity and scalability, comprising a collection of interconnected Django applications, each dedicated to specific functionalities. At its foundation, the platform consists of several essential modules, beginning with the Accounts module, which manages user authentication and profiles, ensuring users can safely register, log in, and modify their accounts. The Movies module enables administrators to oversee film listings, including titles, descriptions, genres, and showtimes, while the Theatre module organizes details regarding cinema screens, such as seat layouts and show schedules. The Bookings module, one of the platform's most vital components, facilitates real-time seat selection and ticket purchases, allowing users to quickly and effortlessly secure seats for their preferred movie showtimes. The integrated Payments module ensures secure financial transactions, enabling users to pay for their tickets via various payment options, whereas the Dashboard module is tailored for administrators and theater operators to track and manage platform operations, from updating movie listings to supervising bookings and user information. Additional features include a Reviews system that permits users to express their feedback on films, enhancing the overall experience and aiding future viewers in making informed choices. To guarantee optimal performance and security, the platform includes a media and static file management system that securely processes multimedia content like images, movie posters, and promotional material. On the backend, the platform employs the Django ORM (Object-Relational Mapping) to enable secure, efficient, and scalable database interactions, ensuring that all user information, bookings, and transactions are quickly stored and retrieved. The frontend uses Django templates, which dynamically render content based on user interactions, ensuring a continuous experience. To enhance the platform's visual appeal and responsiveness, the frontend integrates Bootstrap, a well-known framework for developing mobile-friendly and visually attractive interfaces. Additionally, environment variables are utilized to securely manage sensitive information such as API keys, database credentials, and other confidential details, safeguarding both user and system data. The project is organized for straightforward deployment, making it ideal for academic presentations as well as practical commercial applications. The system is crafted to be highly extensible, permitting future upgrades like third-party service integration, enhanced payment gateways, or advanced recommendation systems based on user preferences.

## ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair**, Presidency School of Computer Science and Engineering, Presidency University, and **Dr. Zafar Ali Khan N**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Ms. Deepthi S, Assistant Professor**, and Reviewer **Mr. John Bennet, Assistant Professor**, Presidency School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the PIP4004 Internship/University Project Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K**, department Project Coordinators **Dr. Afroz Pasha** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**Vignesh V**

## LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 4.1	Workflow	8
2	Figure 4.2	Activity Diagram	9
3	Figure 4.3	Class Diagram	10
4	Figure A.1	Register page	34
5	Figure A.2	Login page	34
6	Figure A.3	Home page	35
7	Figure A.4	Movie-Info page	35
8	Figure A.5	Review Page	36
9	Figure A.6	Theatre Shows and Timings page	36
10	Figure A.7	Payment page	37
11	Figure A.8	Your Order page	37

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	<b>i</b>
	<b>ACKNOWLEDGMENT</b>	<b>ii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Overview of the Project	<b>1</b>
	1.2 Objective of the Project	<b>2</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>4</b>
<b>3</b>	<b>RESEARCH GAPS OF EXISTING METHODS</b>	<b>6</b>
<b>4</b>	<b>PROPOSED METHODOLOGY</b>	<b>7</b>
<b>5</b>	<b>OBJECTIVES</b>	<b>11</b>
<b>6</b>	<b>SYSTEM DESIGN AND IMPLEMENTATION</b>	<b>12</b>
	6.1 System Architecture	<b>12</b>
	6.2 Functional Modules Overview	<b>12</b>
	6.3 Database Design	<b>13</b>
	6.4 Implementation Strategy	<b>14</b>
	6.5 Testing Deployment	<b>14</b>
	6.6 Summary and Future work	<b>14</b>
<b>7</b>	<b>TIMELINE</b>	<b>16</b>
<b>8</b>	<b>OUTCOMES</b>	<b>17</b>
<b>9</b>	<b>RESULTS AND DISCUSSION</b>	<b>18</b>
<b>10</b>	<b>CONCLUSION</b>	<b>19</b>
<b>11</b>	<b>APPENDIX A</b>	<b>21</b>
<b>12</b>	<b>APPENDIX B</b>	<b>34</b>
<b>13</b>	<b>APPENDIX C</b>	<b>38</b>



# **Chapter 1**

## **INTRODUCTION**

In the current rapidly evolving digital environment, convenience and accessibility are crucial factors in consumers' buying choices. The conventional method of purchasing movie tickets—characterized by lengthy lines, limited access to showtime details, and manual seat selection—is rapidly becoming obsolete. To align with modern user expectations and utilize technological innovations, there is a distinct need for a dependable and user-focused online ticket reservation system.

The Online Movie Ticket Booking Platform addresses this requirement by offering a user-friendly, efficient, and secure web application designed to facilitate movie ticket bookings. Developed using the robust Django web framework, the platform delivers a comprehensive system for users to explore cinema listings, check showtimes, select seats in real time, complete payments, and receive booking confirmations directly on their personal devices.

This system benefits not just the end users but also cinema operators and platform administrators. Cinema operators can proficiently manage programming, seating layouts, and ticket pricing, while administrators have the capability to monitor user activity, update content, and uphold the platform's integrity. Its modular design—incorporating applications for user management, reservations, movies, theaters, payments, and reviews—ensures ongoing maintainability and adaptability, paving the way for future enhancements. By merging a robust backend with an appealing frontend, the project illustrates the effectiveness and usability of web development technologies, particularly the Django framework. It serves as both a valuable tool for users and a solid basis for academic exploration or potential business expansion.

### **1.1 Overview of the Project**

The Online Cinema Ticket Booking Platform is a cutting-edge web application created with the Django framework that enhances and streamlines the ticket booking experience for both users and cinema operators. It offers a user-friendly interface that enables customers to browse movies, check showtimes, choose seats in real time, and securely purchase tickets online from any device.

This platform transforms the traditional, cumbersome ticket purchasing process at physical locations into a fully digital experience. Built on the Django framework, known for its robustness and extensibility, the system enables quick development, exceptional performance, and secure data processing. The architecture consists of multiple Django applications, each dedicated to a specific function within the platform.

The Accounts application oversees user-related tasks such as registration, login, logout, profile management, and authentication. The Movies application keeps track of details related to movies, including title, genre, trailers, showtimes, and promotional imagery. The Bookings application handles the entire ticket booking procedure, allowing users to check seat availability and finalize reservations. The Theaters application enables administrators to manage theaters, screens, and schedules, while the Payments application facilitates transactions via both test and real payment gateways. The Dashboard application acts as a central interface for monitoring platform activities, managing content, and tracking ticket sales. The Ratings application allows users to provide feedback and ratings for films they have watched.

The frontend of the application utilizes Django templates and Bootstrap to create a responsive and aesthetically pleasing design that functions seamlessly on both desktop and mobile devices. For backend operations, the platform employs Django ORM for secure and effective database interactions with systems like SQLite. Additionally, static and media files such as images and video trailers are handled through Django's management system for static and media files.

Beyond its core functionality, the project is built with potential future enhancements in mind. These include the addition of SMS or email notifications, third-party APIs for payment processing or ticketing, a discount voucher system, user analytics, and more advanced dashboard features. The platform is user-friendly thanks to its clearly defined modular design and adaptable architecture.

## **1.2 Objective of The Project**

The primary objective of the Online Movie Ticket Booking Platform is to create a digital solution that provides convenience, efficiency, and simplicity, shifting traditional movie ticket

sales into a modern web-based format. By leveraging the Django framework, the platform allows users to seamlessly browse films, check showtimes, select seats in real-time, and securely process payments using any internet-connected device. This initiative aims to eliminate issues related to long queues, manual seat selection, and limited ticket availability by delivering an interactive system that enhances user control and satisfaction. Additionally, a key goal is to aid theater managers and staff in overseeing schedules, seating arrangements, and ticket pricing through a unified dashboard, improving operational efficiency and minimizing human mistakes. From a technical viewpoint, the project aspires to demonstrate best practices in full-stack development with Django, such as a modular application architecture, secure user authentication, and database management through Django ORM. It also aims to ensure a responsive design across various devices by utilizing Django Templates and Bootstrap. Furthermore, the platform seeks to provide a scalable and adaptable framework that can support future enhancements like payment gateway integrations, user notifications, discount systems, and connections to external APIs. Academically, the project functions as a practical illustration of web development principles, making it suitable for demonstrating a real-world application during internships, coursework assessments, or portfolio presentations. On a broader level, the platform is intended to symbolize the shift towards digital solutions in the entertainment industry and contribute a reliable, adaptable system that can be deployed or expanded into a market-ready product with little effort. Ultimately, the project's main aim is to integrate functionality, security, and user experience into a cohesive solution that benefits both end-users and business operators in the movie exhibition sector.

## Chapter 2

### LITERATURE SURVEY

In today's technology-driven world, digital platforms have significantly transformed the way people access entertainment, particularly in the domain of movie ticket booking. Prominent services such as **BookMyShow**, **Fandango**, and **AMC Theatres** have streamlined the movie-going experience by offering a range of features including movie listings, showtime selection, real-time seat reservations, and secure online payments. These platforms are typically built using modern web technologies such as **React**, **Angular**, **Node.js**, and **Java**, which allow them to deliver fast, responsive, and scalable services. Their infrastructure supports high traffic loads, multi-device accessibility, and seamless payment processing, setting a high standard for user convenience and system reliability.

Despite their efficiency, these commercial platforms come with inherent constraints. Their **closed-source nature** restricts external developers from accessing or customizing the underlying code, making them unsuitable for educational exploration, research, or adaptation by small theatres. Moreover, the licensing costs and integration complexities often make them inaccessible to smaller operators who might wish to build or customize their own systems. This has created a gap in the market—one where small business owners, local theatres, and academic institutions struggle to find practical, cost-effective, and customizable alternatives.

Open-source movie booking platforms, while freely accessible, often fall short in terms of modern design, features, and maintainability. Many are built with outdated frameworks and lack modularity, which makes extending their functionality or adapting them for current user expectations a challenging task. Features like **interactive seat selection**, **mobile responsiveness**, **payment gateway support**, and **user feedback systems** are often missing or poorly implemented. Developers looking to use these systems typically have to refactor significant portions of code, which increases the time and effort required to bring such a project to production quality.

Adding to this challenge, many **local or independently operated theatres** still depend on outdated manual systems such as spreadsheets or even handwritten records to manage bookings. These practices are not only prone to human error but also lack real-time updates, often resulting in double bookings, miscommunications, and customer dissatisfaction. Furthermore, these traditional methods lack integration with modern marketing tools such as

email notifications, customer loyalty programs, or online promotions, which are now standard in most digital services.

Given these challenges, the **Django web framework** emerges as a suitable and powerful tool for building scalable and secure web applications, including those for movie ticket booking. Django is a high-level framework written in **Python**, designed to support rapid development and clean, pragmatic design. It includes built-in components for managing user authentication, routing, templates, form processing, and database interaction via the **Django ORM**. Its **Model-View-Template (MVT)** architecture promotes separation of concerns, which enhances maintainability and supports scalable development. Complex relationships, manage user sessions, and integrate third-party services such as **email**, **SMS APIs**, and **payment gateways**. However, there are relatively few Django-based systems that focus on replicating the full scope of functionality provided by commercial movie ticket booking services. Most existing Django projects related to ticket booking are either overly simplified or lack critical features such as dynamic seat selection, theatre scheduling, or secure payment handling.

This project seeks to address that gap by designing and developing a Django-powered **Online Movie Ticket Booking System** that mirrors key features of industry-leading platforms while remaining open-source and educational. The proposed solution includes essential modules for:

- **Movie and Show Management:** Allowing admins to add and manage movies, showtimes, and theatre locations.
- **Interactive Seat Booking:** Offering real-time seat maps that update dynamically to prevent double bookings.
- **User Registration and Authentication:** Providing secure user accounts with profile management.
- **Payment Integration:** Enabling online transactions through services like **Razorpay**, **PayPal**, or similar gateways.
- **Theatre Scheduling:** Allowing theatre owners to schedule multiple shows and allocate screens accordingly.
- **User Reviews and Ratings:** Facilitating community interaction through comments and star ratings.
- **Notification System:** Sending booking confirmations and reminders via email or SMS.

In summary, while commercial platforms deliver a polished user experience, their proprietary frameworks make them unsuitable for open customization or academic purposes. On the other hand, available open-source options often lack critical features or require significant changes.

## Chapter 3

### RESEARCH GAPS OF EXISTING METHODS

Despite the advancements offered by current online movie ticketing platforms, several critical gaps remain unaddressed—particularly in the areas of customization, cost-effectiveness, and academic utility. Leading commercial services such as **BookMyShow**, **Fandango**, and others have revolutionized the user experience by delivering seamless booking processes and rich feature sets. However, these platforms are **proprietary in nature**, heavily tailored to large cinema chains, and offer **little to no flexibility for customization or study**. Their internal architectures and backend logic are not openly accessible, making them **unsuitable for developers, students, or researchers** who wish to explore or extend the system for learning or localized use cases.

Another major issue lies within the **open-source landscape**, where available ticket booking solutions often suffer from **technical debt and lack of modernity**. These platforms typically rely on **outdated technologies**, come with **sparse or poorly maintained documentation**, and attract minimal developer support. As a result, adapting or scaling such systems can be cumbersome. Moreover, they commonly miss out on **key functional components** like real-time seat selection, **secure and integrated payment systems**, or **multi-user role management**—features that are now considered standard in professional-grade applications. The situation is even more challenging at the grassroots level. **Local cinemas and small theatre operators** continue to use manual or legacy software systems for managing bookings. These methods are often prone to **human error**, lack coordination, and are not capable of handling multiple transactions or updates in real time. Additionally, they offer **limited reporting tools**, if any, and do not support essential user engagement features such as **feedback systems, event notifications, personalized promotions, or loyalty rewards**.

From the perspective of educators, learners, and independent developers, there exists a strong need for a **modern, open, and extensible platform** that addresses these shortcomings. The **Django web framework** presents an ideal foundation for such a solution. With its secure and scalable architecture, support for modular development, and rich built-in features such as user authentication, ORM-based database handling, and template rendering, Django enables the creation of robust systems suitable for both academic learning and practical deployment. A movie ticket booking system built on Django can help **bridge the gap** by providing a **customizable, well-documented, and extensible solution**.

## **Chapter 4**

### **PROPOSED METHODOLOGY**

The proposed system employs a modular and layered architecture aimed at ensuring a clear separation of responsibilities, enhanced maintainability, and improved scalability. This platform is built using Django, a high-level Python web framework recognized for its ease of use, security features, and robust development capabilities.

The development process begins with gathering and analyzing requirements, which is followed by system design, implementation, testing, and eventual deployment. The system is organized into functional modules—including Accounts, Movies, Theatre, Bookings, Payments, Dashboard, and Reviews—each implemented as an independent Django app. These apps communicate with each other through Django's model-view-template (MVT) structure.

The Accounts module manages user authentication, registration, and session handling utilizing Django's integrated authentication system. The Movies module allows administrators to upload and control movie information, such as showtimes and trailers. The Bookings module enables users to choose from available seats and finalize their bookings, while being closely connected to the Payments module, which can link to either test or live payment gateways for processing transactions.

The Dashboard module presents an administrative interface where theatre owners or platform administrators can oversee data management, track bookings, and create reports. The Reviews module provides users the capability to submit feedback and view movie ratings, thereby boosting user interaction.

The system features a responsive frontend designed with Django Templates and Bootstrap. All backend activities are overseen by Django ORM, which interacts with a relational database (defaulting to SQLite but easily substitutable with PostgreSQL or MySQL if required). The application incorporates form validation, error remediation, and security precautions such as CSRF protection and input sanitization.

The project adopts an iterative development methodology, enabling ongoing testing and improvements based on feedback. Future upgrades may include the integration of external APIs, support for mobile applications, and deployment to cloud services like Heroku or AWS for enhanced scalability.

## System Architecture Overview

The system is designed following a **modular and layered architecture**, which is crucial for maintaining separation of concerns and promoting scalability and reusability.

### 1. Architectural Model: Layered Architecture

The system consists of the following layers:

#### a. Presentation Layer (Frontend)

- Built using **Django Templates** and styled with **Bootstrap** for responsiveness.
- Handles user interaction: login, movie listings, seat selection, reviews, etc.

#### b. Business Logic Layer

- Implemented through Django **views** and **forms**.
- Contains core application logic such as booking verification, seat availability, payment validation, and review moderation.

#### c. Data Access Layer

- Managed by **Django ORM** (Object Relational Mapper).
- Handles all interactions with the database, ensuring secure, optimized, and maintainable access to data.

### 2. Modular Design (Apps in Django)

Each of the system's major features is encapsulated as a Django app:

Module	Description
<b>Accounts</b>	User registration, login/logout, session handling, role management (admin/theatre owner/user).
<b>Movies</b>	CRUD for movie info: titles, genres, showtimes, trailers, etc.
<b>Theatres</b>	Manage theatre profiles, screens, seating layout, and show schedules.
<b>Bookings</b>	Seat selection, real-time availability, booking confirmation.
<b>Payments</b>	Integration with payment gateways (e.g., Razorpay test mode).
<b>Dashboard</b>	Admin tools for reports, analytics, revenue monitoring, and theatre management.
<b>Reviews</b>	User ratings, feedback submission, and display.



## Architecture and Workflow

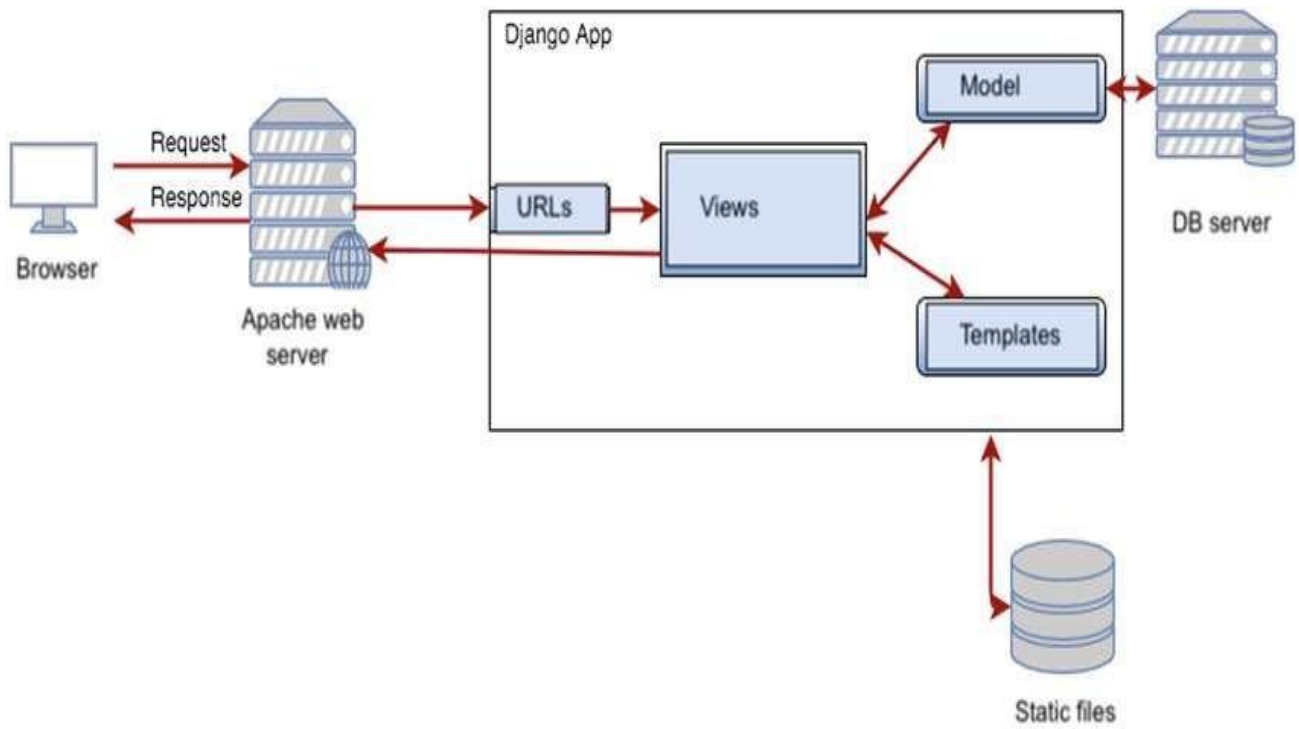


Figure 4.1

## Activity Diagram

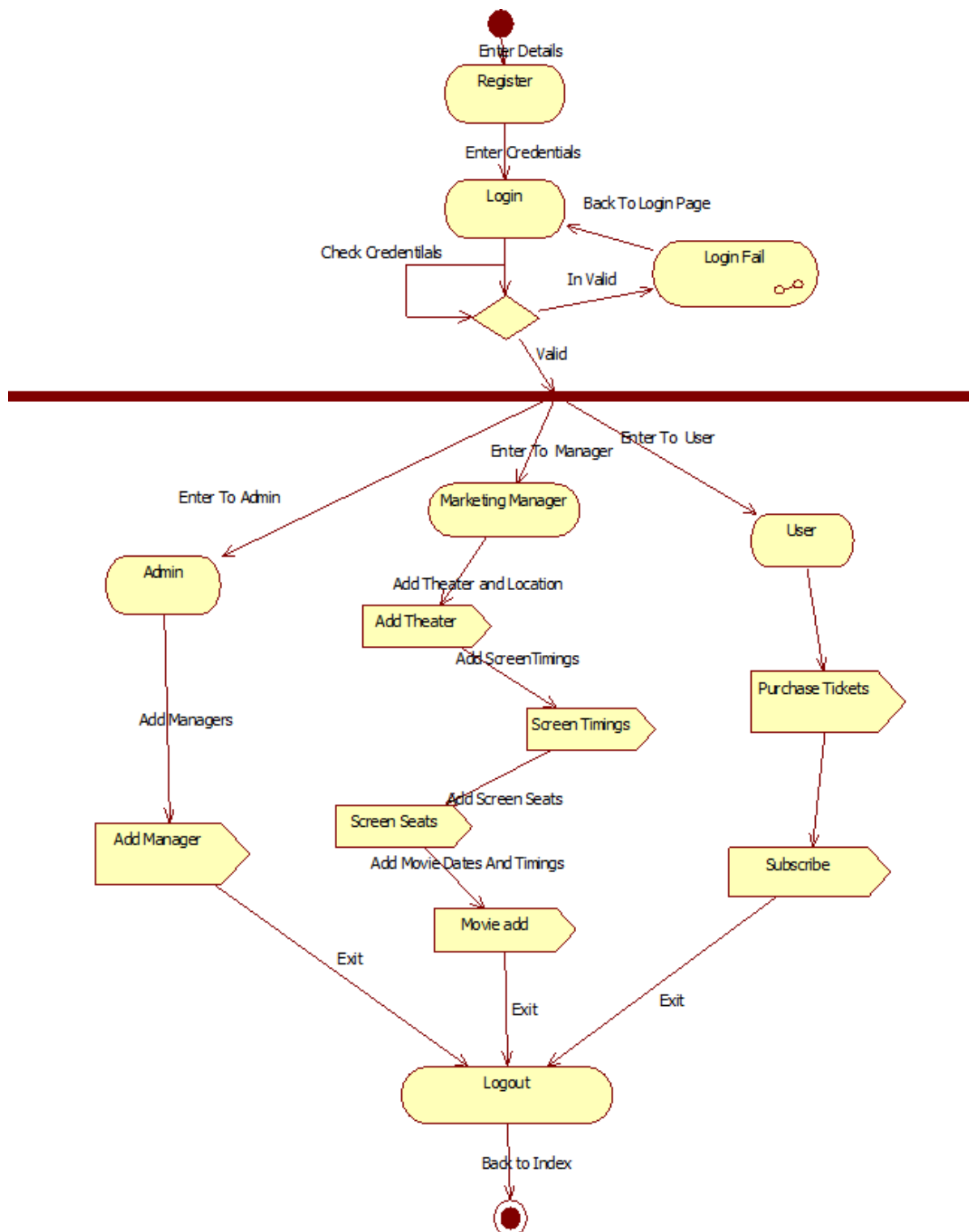


Figure 4.2

## Class Diagram

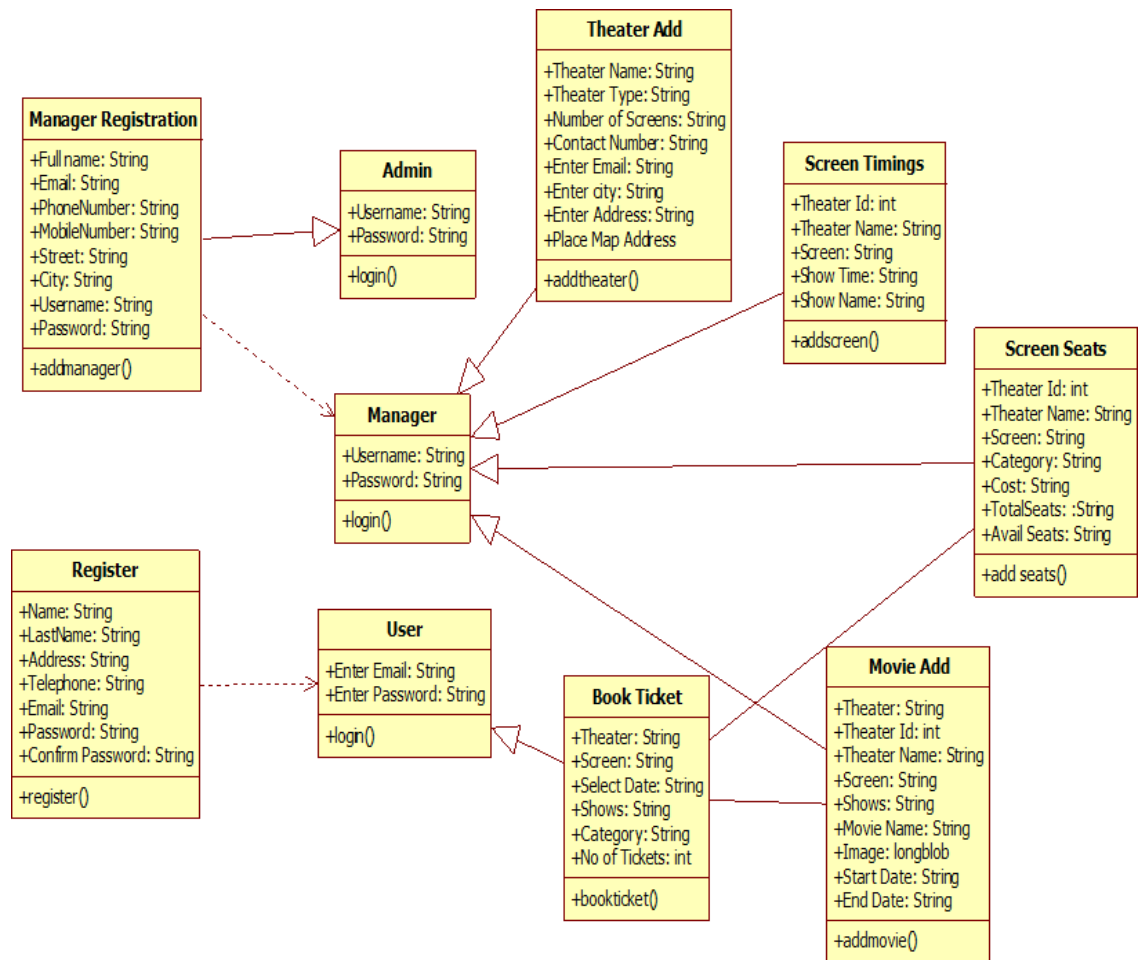


Figure 4.3

## Chapter 5

### OBJECTIVES

- The main aim of this project is to create and implement a web-based system that simplifies and improves the movie ticket booking process for users, theater managers, and system operators. This platform intends to provide a digital solution to traditional ticketing, thereby removing long lines, booking delays, and human errors.
- To create an intuitive interface for users to explore movies, check showtimes, view seating arrangements, and book tickets in real time.
- To offer secure user registration, login, and session management utilizing Django's built-in authentication features.
- To equip theater administrators with a centralized dashboard for overseeing movie listings, show schedules, seat assignments, and pricing.
- To incorporate a dynamic seat selection feature that displays real-time availability and prevents overlapping bookings.
- To add a payment system that accommodates transactional capabilities, whether in test mode or through actual payment processors.
- To guarantee accessibility on both mobile and desktop platforms through a responsive user interface designed with Bootstrap and Django Templates.
- To enhance user interaction via feedback and review features.
- To ensure secure and efficient backend processes using Django ORM and relational databases.
- These objectives not only enhance the system's functionality but also contribute to its scalability, maintainability, and relevance in both academic and real-life contexts.

## Chapter 6

# SYSTEM DESIGN & IMPLEMENTATION

The Online Movie Ticket Booking Platform is built upon Django's Model-View-Template (MVT) architecture, a powerful framework that enforces a clear separation of concerns. This structure not only streamlines the development process but also ensures long-term maintainability, scalability, and ease of testing. By integrating secure backend processes, a responsive user interface, and modular app design, the system provides a seamless experience for both users and administrators.

### 6.1 System Architecture

The platform is designed using a layered architectural approach with distinct responsibilities for each layer:

- **Presentation Layer:** This layer includes all user-facing components, such as HTML pages and CSS-styled forms, built using Django Templates and styled with Bootstrap. It ensures mobile responsiveness and accessibility across devices.
- **Application Logic Layer:** Managed via Django Views, this layer handles business rules, processes user inputs, and communicates between models and templates. It includes logic for seat availability checks, booking workflows, session handling, and payment processing.
- **Data Layer:** Django's Object-Relational Mapper (ORM) provides a high-level abstraction over the database. Instead of writing raw SQL, developers interact with Python objects, enabling faster development and safer database access.

This architecture promotes modularity, where each Django app is responsible for a specific function. These apps are logically grouped and follow reusable code practices.

### 6.2 Functional Modules Overview

The system is divided into independent functional modules, each implemented as a dedicated Django app:

- **Accounts Module:** Handles user registration, login/logout, password management, and user role definition (e.g., customer, admin, or theatre manager). It leverages Django's built-in authentication system.
- **Movies Module:** Allows admins to manage movie entries, including metadata such as

title, genre, duration, cast, language, synopsis, poster images, and trailers. Movies can be scheduled for multiple screens across theatres.

- **Bookings Module:** Users can view real-time seat layouts, select available seats, and confirm bookings. Backend logic ensures seat locking during transactions and prevents duplicate bookings.
- **Theatre Module:** Enables theatre owners to add and manage venue details including name, location, screen count, and seating layout. Each screen can be linked with specific movie schedules.
- **Payments Module:** This app facilitates financial transactions. Payment APIs like Razorpay or Stripe can be plugged into this module to process bookings securely. It also records payment status and order tracking IDs.
- **Dashboard Module:** Accessible to administrators and theatre managers, this dashboard displays booking statistics, user trends, sales reports, and allows manual updates to movie or theatre data.
- **Reviews Module:** Users can rate and review movies they've watched. Admins have moderation capabilities to ensure quality and appropriateness of content.

Each module interacts with others where needed, such as connecting user profiles with booking history or associating payment entries with movie shows.

### 6.3 Database Design

The relational database schema is structured to reflect real-world relationships between users, shows, theatres, and transactions. It is optimized for performance and data integrity.

Key tables include:

- **User** (id, username, email, password, role)
- **Movie** (id, title, genre, duration, language, poster)
- **Theatre** (id, name, location, screens)
- **Screen** (id, theatre\_id, screen\_number, seat\_layout)
- **Show** (id, movie\_id, screen\_id, start\_time, end\_time)
- **Booking** (id, user\_id, show\_id, seats\_selected, status, timestamp)
- **Payment** (id, booking\_id, amount, payment\_gateway, status, transaction\_id)
- **Review** (id, user\_id, movie\_id, rating, comment, timestamp)

Foreign keys ensure referential integrity and make complex queries efficient. The use of Django's ORM ensures portability across database systems

## 6.4 Implementation Strategy

The system is developed using Agile methodology, with short development sprints focusing on one or two modules at a time. Key strategies include:

- **Version Control:** Git is used for managing code versions and collaborative development via GitHub.
- **Continuous Testing:** Django's built-in Testcase framework is used to write unit and integration tests. Coverage includes model validations, view responses, and form handling.
- **Code Modularity:** Each app is independent but communicates via Django's built-in app system, making the entire platform easy to extend or maintain.
- **Security Measures:** The system includes CSRF protection, password hashing, input validation, and secure session management. Django's middleware handles most web vulnerabilities by default.

To simulate real-world deployment, configurations are prepared for cloud platforms such as Heroku and AWS, ensuring scalability and high availability.

## 6.5 Testing and Deployment

### Testing Phases:

- **Unit Testing:** All models and views are tested with Django's testing tools to ensure each function behaves as expected.
- **Integration Testing:** Modules such as booking and payments are tested end-to-end, especially seat-locking and transaction flows.
- **UI Testing:** Templates are tested manually and with tools like Selenium to verify mobile responsiveness and browser compatibility.

### Deployment Preparation:

- Static files are collected and served via White Noise or CDN.
- Environment variables (e.g., API keys) are secured using .env files.
- Database migrations are handled via manage.py migrate, and cloud storage options are configured if media content is large.

## 6.6 Summary and Future Work

The Django-based system offers a modular, secure, and user-friendly movie booking platform.

With its MVT-based architecture, clearly defined modules, and extensible backend, it stands as a capable solution for small theatres and educational institutions. It is scalable enough for commercial deployment and clear enough for academic learning.

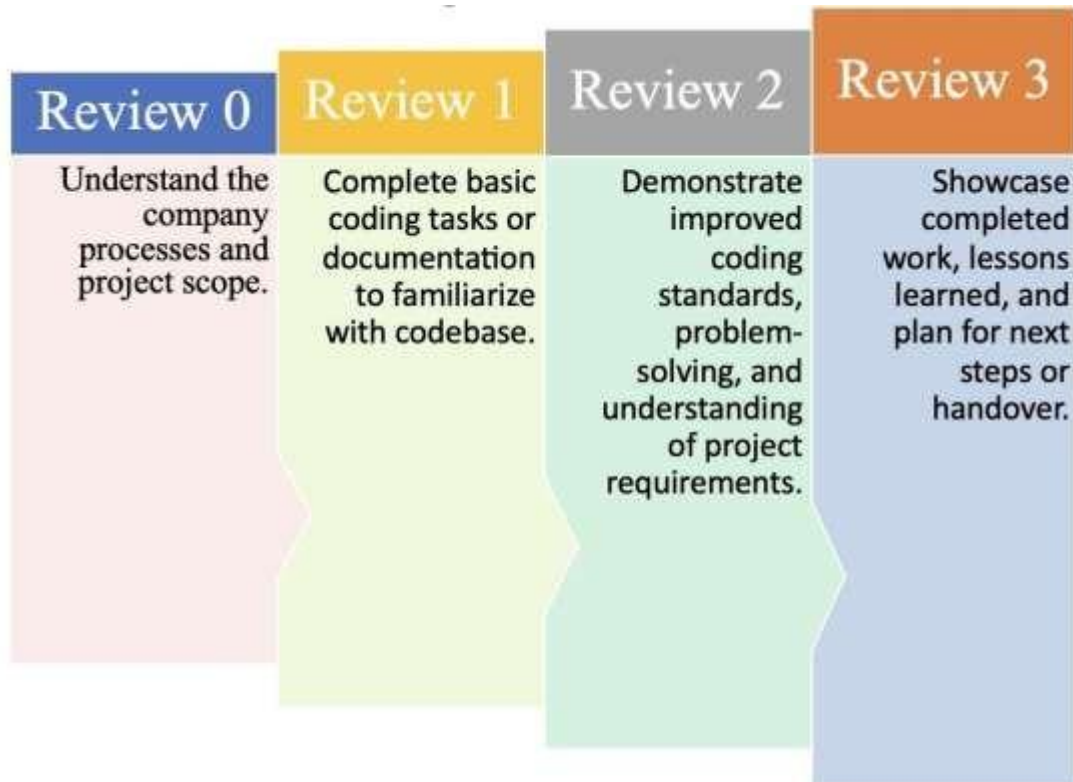
**Future Enhancements:**

- Mobile application integration using Flutter or React Native.
- AI-based movie recommendation system.
- Integration with SMS/Email marketing tools.
- Loyalty and reward programs for frequent users.
- Real-time notifications using WebSocket's or Django Channels.



## Chapter 7

### TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



## **Chapter 8**

### **OUTCOMES**

- The creation and deployment of the Online Movie Ticket Booking Platform resulted in several significant outcomes and achievements, both from a technical standpoint and user experience perspective.
- A fully operational web-based solution was developed using Django, accommodating multi-role access for users, administrators, and theater managers.
- Users can register, explore movie options, choose their seats, and finalize bookings through a user-friendly and responsive interface.
- Theater administrators received a dashboard that allows comprehensive management of movie listings, showtimes, and seating arrangements.
- The payment process was designed and integrated with placeholders, allowing for future implementation of a real payment gateway.
- An adaptive booking process prevents double bookings and ensures that seat availability is updated in real time.
- The platform incorporates a review module that fosters user feedback, thereby enhancing user engagement and trust.
- Code modularity and Django's MVT architecture were utilized to facilitate easier future maintenance and development.
- The project showcased effective application of agile methodology, Git version control, and best practices in web development.
- In summary, the project effectively achieved its objectives, delivering a practical, educational, and scalable solution for managing movie ticket bookings.

## Chapter 9

# RESULTS AND DISCUSSIONS

The Online Movie Ticket Booking Platform underwent comprehensive testing throughout its development to ensure that all features operated as intended. The system was subjected to unit testing, integration testing, and user acceptance testing to confirm its reliability, performance, and user-friendliness.

Throughout the testing phase, the platform efficiently managed user sessions, handled simultaneous bookings without issues, and processed simulated payments. The seat selection feature provided real-time updates and effectively prevented overbooking. Administrative features like adding or removing movies, adjusting showtimes, and Key discussion points include:

- **Performance:** The application demonstrated good performance with minimal delay, even under simulated concurrent usage.
- **Scalability:** Although the platform is developed using SQLite, it can be upgraded to PostgreSQL or MySQL in production environments for better scalability.
- **Security:** Fundamental security measures, such as input validation, session protection, and CSRF tokens, were effectively implemented.
- **Limitations:** Currently, the project relies on a placeholder payment system and lacks integration with third-party APIs, representing potential areas for future enhancement.

Overall, the system delivered results in line with its goals and is ready for further enhancement or real-world deployment with minimal modifications.

## Chapter 10

# CONCLUSION

The **Movie Ticket Booking Platform** represents a pivotal step in the digital transformation of the entertainment industry, particularly in how audiences access cinema experiences. This system delivers a streamlined, intuitive interface that allows users to explore available movies, select convenient showtimes, check real-time seat availability, and securely complete bookings—all from the comfort of their homes or mobile devices. By eliminating the need for physical queues and manual ticketing, the platform offers a hassle-free and time-saving alternative for modern moviegoers.

From the perspective of end-users, the platform introduces a highly personalized and efficient ticketing process. Features such as filtered movie searches, genre-based recommendations, and user reviews allow audiences to make informed choices. Real-time seat visualization not only enhances transparency but also ensures that users can select preferred seating options without ambiguity. Additionally, digital ticket generation—complete with QR codes—simplifies theater entry while reducing the reliance on paper-based processes. These innovations lead to greater user satisfaction, improved engagement, and a more enjoyable cinema experience.

For **theatre administrators and operators**, the system significantly improves operational efficiency and accuracy. Tasks that were traditionally performed manually—such as seat assignment, booking ledger maintenance, and revenue tracking—are now automated through intelligent backend processes. This reduces human error, eliminates overbooking risks, and ensures accurate scheduling. Furthermore, administrative dashboards offer real-time insights into ticket sales, audience trends, and occupancy rates, which are valuable for making data-driven decisions regarding show timings, promotions, and movie selection.

The platform's backend infrastructure is deliberately built to be both **modular and scalable**, making it easy to integrate future enhancements. Payment gateways can be extended to support various providers such as Razorpay, Stripe, or PayPal, ensuring greater flexibility in transaction handling. As user expectations evolve, additional features—like personalized notifications, AI-driven movie recommendations, or dynamic pricing models—can be incorporated without extensive reengineering.

In terms of **security and privacy**, the system emphasizes safe data handling practices. Secure payment processing is ensured through SSL encryption, CSRF protection, and tokenized transactions, safeguarding both financial and personal information. Authentication systems

are designed using Django's robust framework, incorporating password hashing, session management, and user role validation to prevent unauthorized access and ensure compliance with data protection standards.

The platform's design also supports **future technological growth**. Mobile application integration using cross-platform frameworks such as Flutter or React Native can extend accessibility to a wider audience. Support for **regional languages** and **multi-currency options** can make the system more inclusive, catering to users across different geographies and demographics. Loyalty programs, user badges, and referral systems could also be introduced to promote customer retention and brand engagement.

In a broader context, the Movie Ticket Booking Platform not only addresses the functional needs of digital ticketing but also aligns with global trends in automation, convenience, and personalized service. It demonstrates how technology can bridge the gap between traditional industries and the expectations of a tech-savvy audience. Whether deployed for small, independent cinemas or scaled to serve multi-location theater chains, the platform provides a reliable, efficient, and expandable foundation.

## APPENDIX-A

### PSEUDOCODE

#### Settings.py

```
"""
```

Django settings for movie\_ticket\_booking project.

Generated by 'django-admin startproject' using Django 5.2.

For more information on this file, see

<https://docs.djangoproject.com/en/5.2/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/5.2/ref/settings/>

```
"""
```

```
from pathlib import Path
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
```

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
# Quick-start development settings - unsuitable for production
```

```
# See https://docs.djangoproject.com/en/5.2/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = 'django-insecure-2do7%mybrmx&=*0uvow@#z-
```

```
vp#h(4_6pi^jr5tgyq#p8v=_15'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

#Application definition

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
  
    'accounts',  
    'movies',  
    'bookings',  
    'dashboard',  
    'theatre',  
    'reviews',  
    'payments',  
  
    'crispy_forms',  
    'crispy_bootstrap4',  
    'rest_framework',  
  
    'allauth',  
    'allauth.account',  
    'allauth.socialaccount',  
    'allauth.socialaccount.providers.google',  
]  
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',
```

```
'django.middleware.clickjacking.XFrameOptionsMiddleware',

'allauth.account.middleware.AccountMiddleware',

]

ROOT_URLCONF = 'movie_ticket_booking.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]

WSGI_APPLICATION = 'movie_ticket_booking.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```



```
}
```

```
# Password validation
```

```
# https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
```

```
{
```

```
    'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
```

```
},
```

```
{
```

```
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
```

```
},
```

```
{
```

```
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
```

```
},
```

```
{
```

```
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
```

```
},
```

```
]
```

```
# Internationalization
```

```
# https://docs.djangoproject.com/en/5.2/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/5.2/howto/static-files/
```

```
STATIC_URL = 'static/'
STATICFILES_DIRS = [BASE_DIR / 'static']

# Default primary key field type
# https://docs.djangoproject.com/en/5.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# Crispy Forms Settings
CRISPY_ALLOWED_TEMPLATE_PACKS = 'bootstrap4'

CRISPY_TEMPLATE_PACK = 'bootstrap4'

AUTH_USER_MODEL = 'accounts.User'

SOCIALACCOUNT_PROVIDERS = {
    'google': {
        # For each OAuth based provider, either add a ``SocialApp``
        # (``socialaccount`` app) containing the required client
        # credentials, or list them here:
        'APP': {
            'client_id': '1098107227522-
blo8vliu813bljysi22vvobron3tcehl.apps.googleusercontent.com',
            'secret': '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',
            'key': ""
        }
    }
}

SITE_ID = 1

SOCIALACCOUNT_LOGIN_ON_GET=True

LOGIN_REDIRECT_URL = '/home/'
```

```
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'vishnunandanar901@gmail.com'
EMAIL_USE_TLS = True
DEFAULT_FROM_EMAIL = 'vishnunandanar901@gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_PASSWORD = 'izzbioehxfmmzrrd'

from django.contrib import messages

MESSAGE_TAGS = {
    messages.ERROR: 'danger'
}

MEDIA_ROOT = BASE_DIR / 'media'
MEDIA_URL = 'media/'

# Stripe settings
STRIPE_PUBLIC_KEY =
'pk_test_51BTUDGJAJfZb9HEBwDg86TN1KNprHjkfipXmEDMb0gSCassK5T3ZfxsAbcg
KVmAIXF7oZ6IdZZbXO6idTHE67IM007EwQ4uN3'
STRIPE_SECRET_KEY = 'sk_test_tR3PYbcVNZZ796tH88S4VQ2u'
```

### **views.py**

```
from django.shortcuts import render, redirect, HttpResponseRedirect
from theatre.models import theatre, showtimes, seats
from movies.models import movies
from accounts.models import User
from datetime import datetime, timedelta
from .models import bookings, bookingseats
from django.contrib import messages
import json
```

```
from django.conf import settings
import stripe
from django.contrib.auth.decorators import login_required

stripe.api_key = settings.STRIPE_SECRET_KEY

# Create your views here.

def theatre_show_time_view(request, slug):
    today = datetime.today().date()
    start_date = today-timedelta(days=0)
    week=[]
    for i in range(7):
        day=start_date+timedelta(days=i)
        week.append({
            'name': day.strftime('%a').upper(),
            'day': day.day,
            'month': day.strftime('%b').upper(),
            'date': day
        })

    if movies.objects.filter(slug=slug).exists():
        movie = movies.objects.get(slug=slug)
        theatre_showtimes=[
            showtimes.objects.filter(movie=movie, theatre=theatre_obj).order_by('show_time')
            for theatre_obj in theatre.objects.all() if showtimes.objects.filter(movie=movie,
theatre=theatre_obj).exists()]
        context = {
            'theatre_showtimes': theatre_showtimes,
            'm': movie,
            'week': week,
            'today': today,
```

```
    }

    return render(request, 'theatre/theatre_show_time.html', context)
return render(request, 'movies/404.html' )

def theatre_show_time_selected_date_view(request, slug, date_str):
    try:
        selected_date = datetime.strptime(date_str, '%Y-%m-%d').date()
    except ValueError:
        selected_date = datetime.today().date()

    today = datetime.today().date()
    week = []
    for i in range(7):
        day = today + timedelta(days=i)
        week.append({
            'name': day.strftime('%a').upper(),
            'day': day.day,
            'month': day.strftime('%b').upper(),
            'date': day
        })

    movie = movies.objects.filter(slug=slug).first()
    if not movie:
        context = {
            'week': week,
            'today': selected_date,
            'theatre_showtimes': [],
            'm': None,
        }
        return render(request, 'theatre/theatre_show_time.html', context)

    theatre_showtimes = []
    for theatre_obj in theatre.objects.all():
```

```
shows = showtimes.objects.filter(
    movie=movie,
    theatre=theatre_obj,
    show_time__date=selected_date
).order_by('show_time')
if shows.exists():
    theatre_showtimes.append(shows)

context = {
    'theatre_showtimes': theatre_showtimes,
    'week': week,
    'today': selected_date,
    'm': movie,
}

return render(request, 'theatre/theatre_show_time.html', context)

def seat_selection_view(request, slug, showtime_id):
    showtime = showtimes.objects.get(id=showtime_id)

    # Get all seats for this showtime's screen
    all_seats = seats.objects.filter(
        theatre=showtime.theatre,
        screen_number=showtime.screen_number
    ).order_by('row_label', 'seat_number')

    # Get IDs of booked seats
    booked_seats = bookingseats.objects.filter(
        booking_showtime=showtime,
        booking_booking_status='confirmed'
    )
    booked_seat_ids = [seat.seat.id for seat in booked_seats]

    # Group ALL seats (not just available) by row
```

```
seat_rows = {}
for seat in all_seats:
    row = seat.row_label
    if row not in seat_rows:
        seat_rows[row] = []
    seat_rows[row].append(seat)

vip_rows = ['A', 'B', 'C']
gold_rows = ['D', 'E', 'F']
silver_rows = [row for row in seat_rows if row not in vip_rows + gold_rows]

context = {
    'showtime': showtime,
    'slug': slug,
    'seat_rows': seat_rows,
    'booked_seats': booked_seat_ids, # Just the IDs of booked ones
    'vip_rows': vip_rows,
    'gold_rows': gold_rows,
    'silver_rows': silver_rows,
}

return render(request, 'theatre/seating.html', context)
```

```
@login_required
def book_ticket_view(request, showtime_id):
    if request.method == 'POST':
        selected_seats = json.loads(request.POST.get('selected_seats')) # ➡ use () not []
        total_amount = int(request.POST.get('total_amount'))

        showtime = showtimes.objects.get(id=showtime_id)
        user = request.user
        booking = bookings.objects.create(
            user=user,
            showtime=showtime,
```

```
        total_amount=total_amount,
        booking_status='pending'
    )

    for seat in selected_seats:
        seat_key = seat['key']
        row = seat_key[0]
        number = seat_key[1:]
        seat_obj =
seats.objects.get(row_label=row,seat_number=number,screen_number=showtime.screen_number,theatre=showtime.theatre)
        bookingseats.objects.create(booking=booking, seat=seat_obj)

    context = {
        'convenience_fee': 49,
        'total_amount': total_amount,
        'booking': booking,
        'tickets': [s['key'] for s in selected_seats],
        'showtime': showtime,
        'subtotal': total_amount + 49,
        'stripe_public_key': settings.STRIPE_PUBLIC_KEY,

    }

    return render(request, 'payments/proceed_payments.html', context)

return HttpResponse("Invalid Request", status=400)

@login_required
def cancel_ticket(request, booking_id):
    booking = bookings.objects.get(id=booking_id)
    booking.booking_status = 'cancelled'
    booking.save()
    qs = bookingseats.objects.filter(booking=booking)
```



```
qs.delete()
messages.error(request, 'Your booking has been cancelled')
return redirect('your_orders')
```

## **models.py**

```
from django.db import models
from accounts.models import User
from movies.models import movies

# Create your models here.

class theatre(models.Model):

    name=models.CharField(max_length=255)
    city=models.CharField(max_length=255)
    address=models.CharField(max_length=255)
    manager=models.ForeignKey(User, on_delete=models.SET_NULL, null=True,
blank=True)

    def __str__(self):
        return self.name

    def generate_seats(self, screen_number, rows=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'],
seats_per_row=10):
        from .models import seat
        for row in rows:
            if row in ['A', 'B', 'C']:
                seat_type = 'VIP'
            elif row in ['D', 'E', 'F']:
                seat_type = 'Gold'
            else:
```

```
        seat_type = 'Regular'

    for number in range(1, seats_per_row + 1):
        seats.objects.get_or_create(
            theatre=self,
            screen_number=screen_number,
            row_label=row,
            seat_number=number,
            defaults={'seat_type': seat_type}
        )

class showtimes(models.Model):
    movie=models.ForeignKey(movies, on_delete=models.SET_NULL, null=True,
blank=True)
    theatre=models.ForeignKey(theatre, on_delete=models.CASCADE)
    show_time=models.DateTimeField()
    screen_number=models.IntegerField(null=True, blank=True)

    def save(self, *args, **kwargs):
        is_new = self.pk is None
        super().save(*args, **kwargs)

        if is_new:
            self.theatre.generate_seats(screen_number=self.screen_number)

class seats(models.Model):
    theatre=models.ForeignKey(theatre, on_delete=models.CASCADE)
    screen_number = models.IntegerField(null=True, blank=True)
    row_label = models.CharField(max_length=255)
    seat_number = models.IntegerField()
    seat_type = models.CharField(max_length=255,choices=[['VIP', 'VIP'], ['Gold', 'Gold'],
['Regular', 'Regular']])
```

## APPENDIX-B

## SCREENSHOTS

**CineVerse** Home My Bookings Account

### Register

Username\*

Required: 160 characters or fewer. Letters, digits and @/./+/-/\_ only.

Phone

Email address

Password\*

Your password can't be too similar to your other personal information.  
Your password must contain at least 8 characters.  
Your password can't be a commonly used password.  
Your password can't be entirely numeric.

Password confirmation\*

Enter the same password as before, for verification.

Is theatre manager ☐

Submit

Sign in with Google

Already a member?

Login Here

**CineVerse** Your ultimate destination for movie discovery and entertainment.

**Quick Links**  
Home  
Movies  
TV Shows  
Celebrities

**Legal**  
Terms of Use  
Privacy Policy  
Cookie Policy  
DMCA

**Connect With Us**  
Subscribe to our newsletter  
Your email

Figure. A1

**CineVerse** Home My Bookings Account

### Login

Username\*

heryn

Password\*

••••••••

Login

Sign in with Google

Not a member?

Register Here

**CineVerse** Your ultimate destination for movie discovery and entertainment.

**Quick Links**  
Home  
Movies  
TV Shows  
Celebrities

**Legal**  
Terms of Use  
Privacy Policy  
Cookie Policy  
DMCA

**Connect With Us**  
Subscribe to our newsletter  
Your email

© 2023 CineVerse. All rights reserved. Made with ❤️ for movie lovers.

Figure. A2

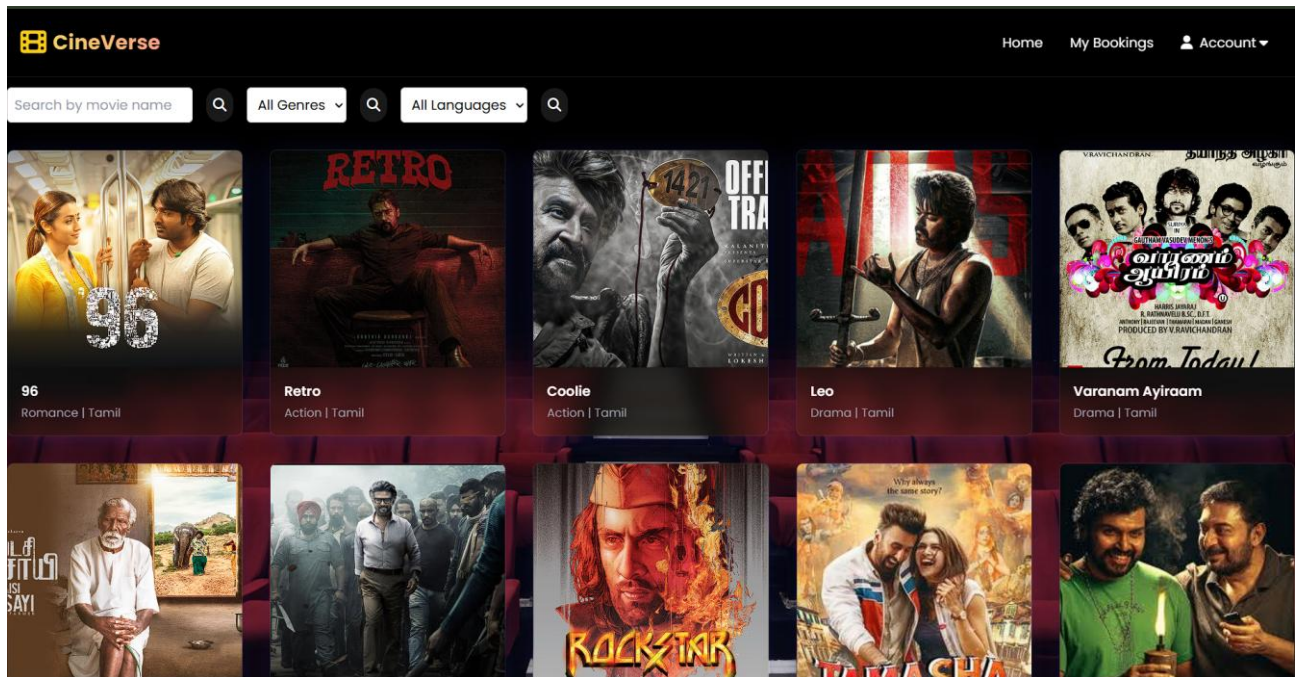


Figure. A3

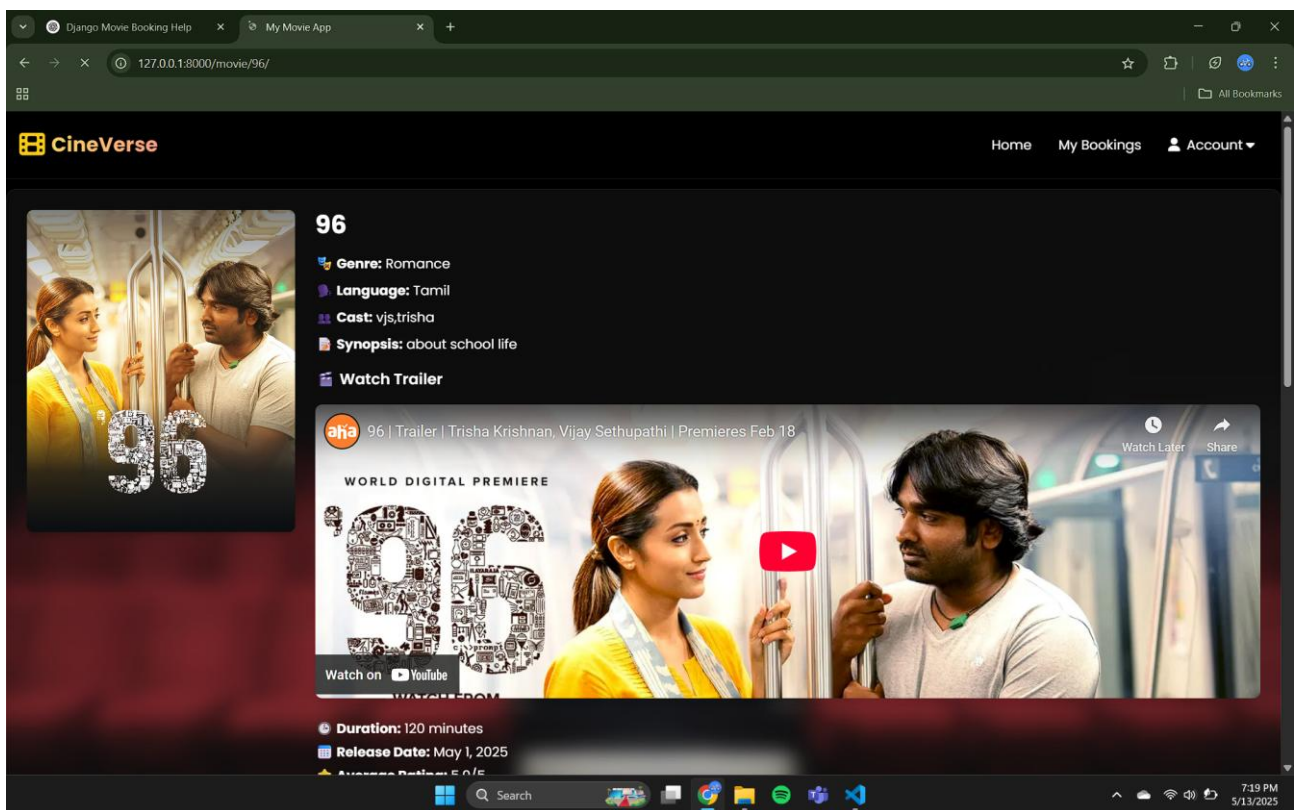


Figure. A4

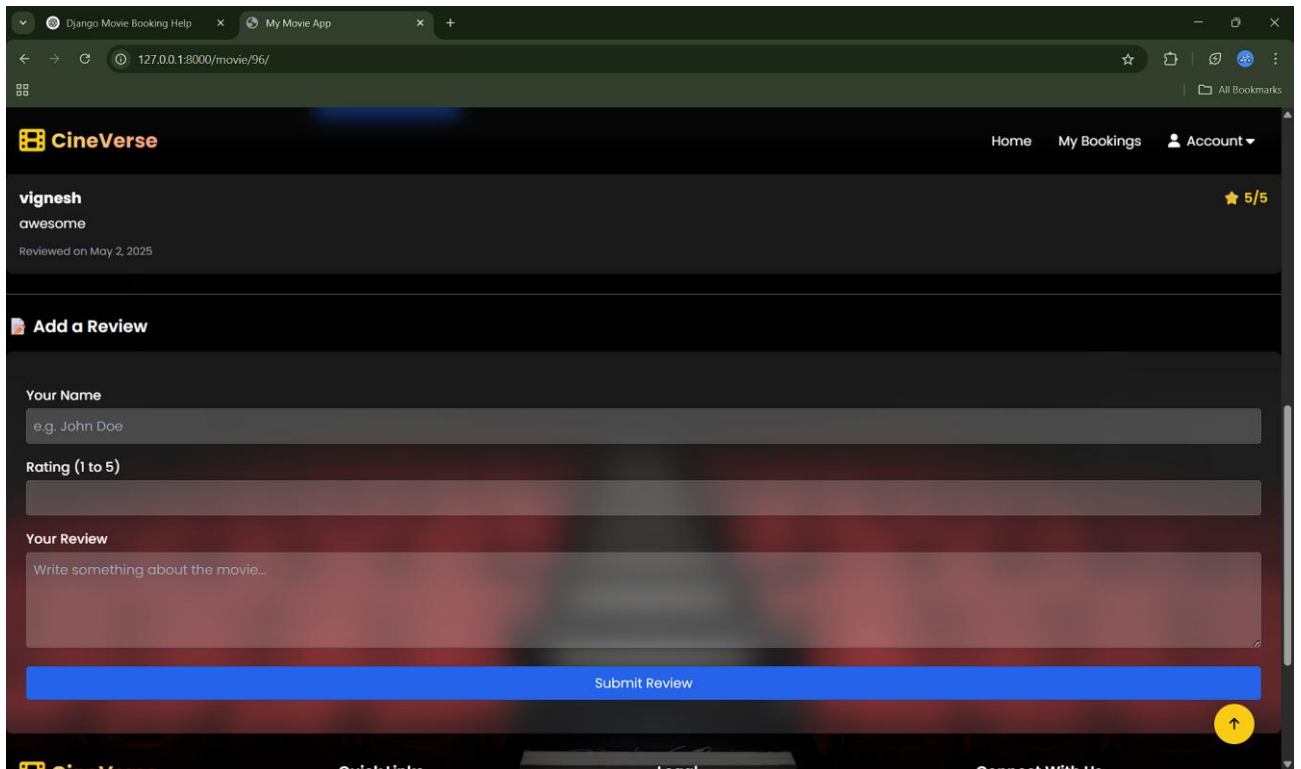


Figure. A5

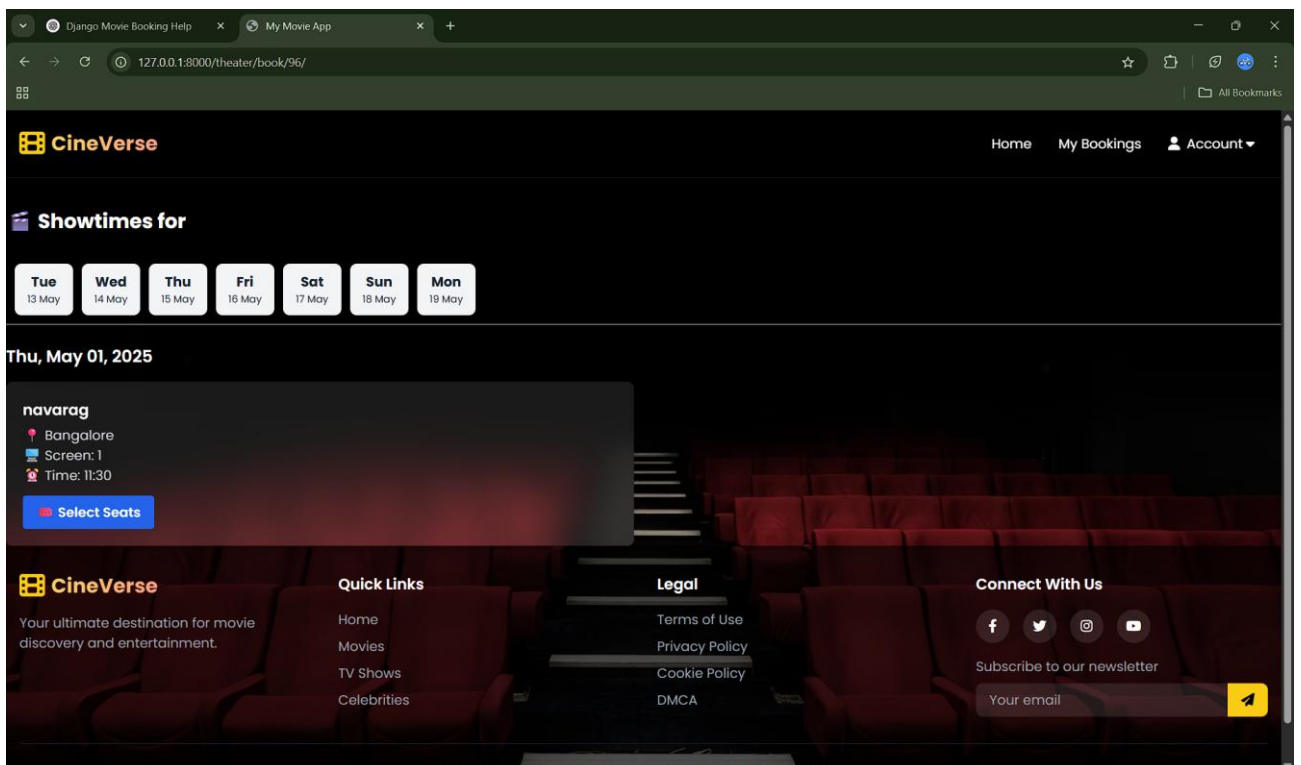


Figure. A6

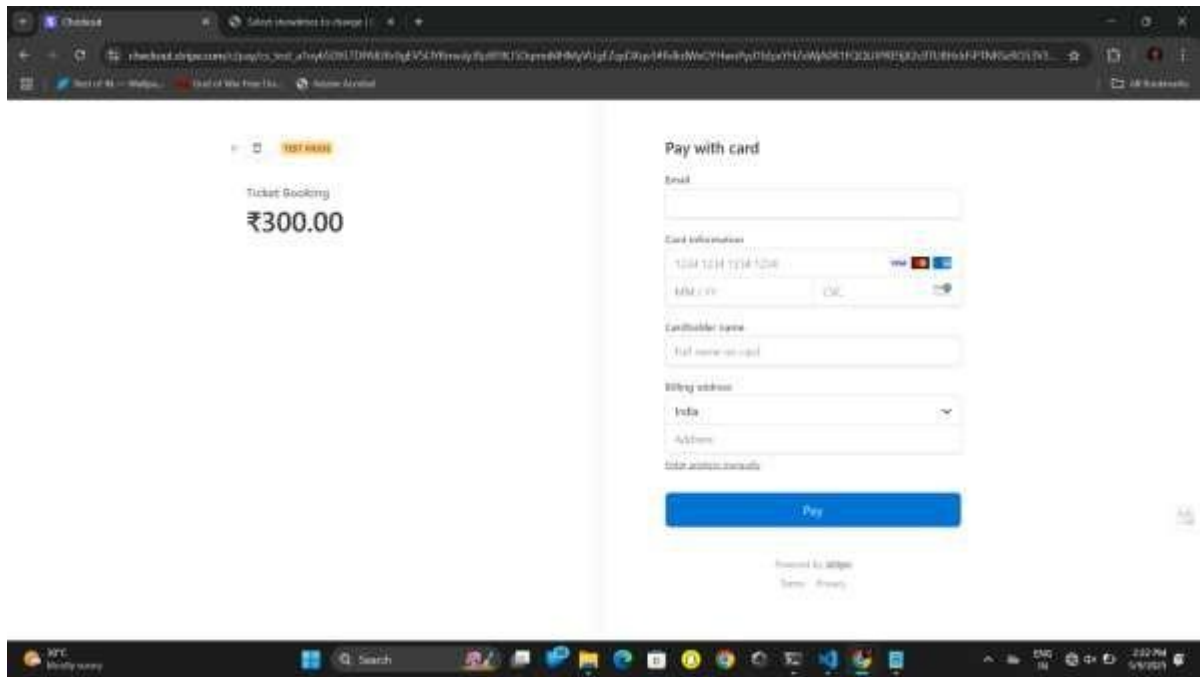


Figure. A7

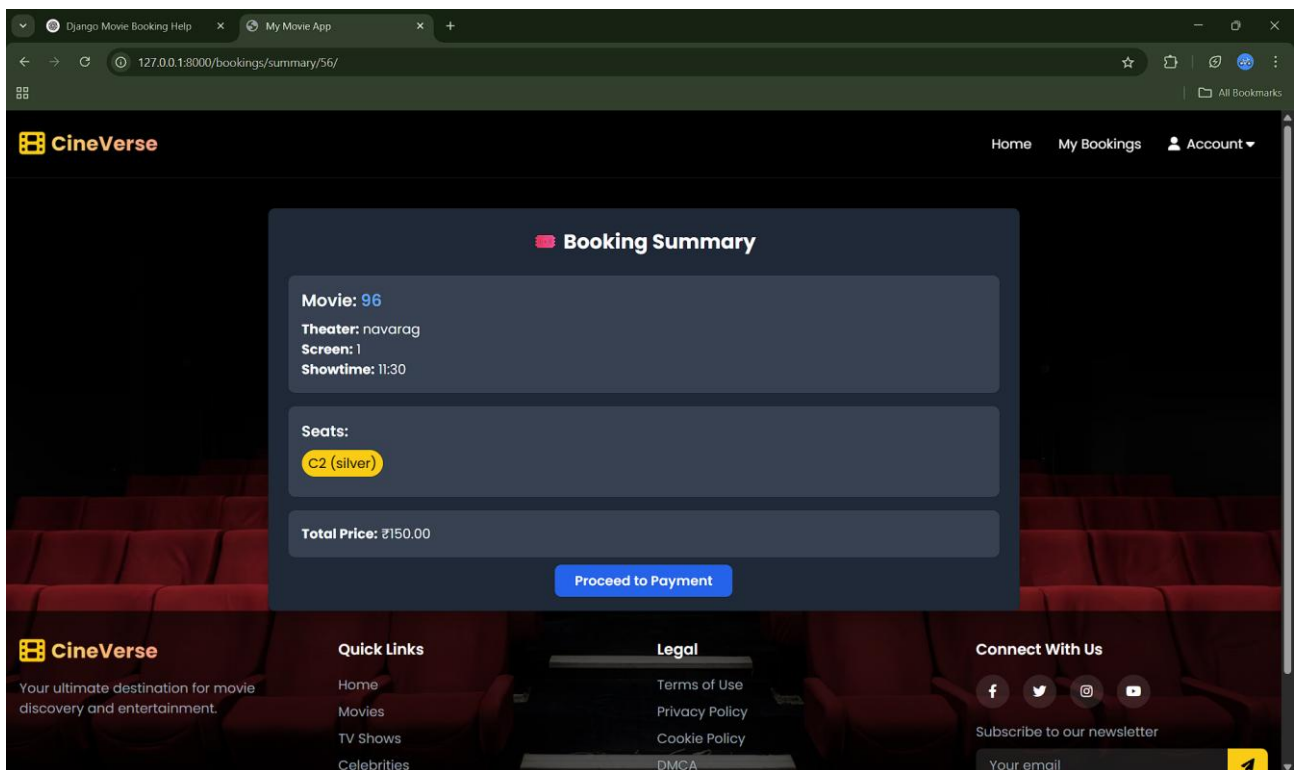



Figure. A8



## APPENDIX-C

### ENCLOSURES

#### 1. Similarity Index / Plagiarism Check report


Page 2 of 45 - Integrity Overview
Submission ID trn:oid=1:3249651718





### 16% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




#### Filtered from the Report

- Bibliography

#### Match Groups


- 
**41 Not Cited or Quoted 16%**  
Matches with neither in-text citation nor quotation marks
- 
**0 Missing Quotations 0%**  
Matches that are still very similar to source material
- 
**0 Missing Citation 0%**  
Matches that have quotation marks, but no in-text citation
- 
**0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

#### Top Sources

- 9%  Internet sources
- 9%  Publications
- 15%  Submitted works (Student Papers)


#### Integrity Flags

1 Integrity Flag for Review

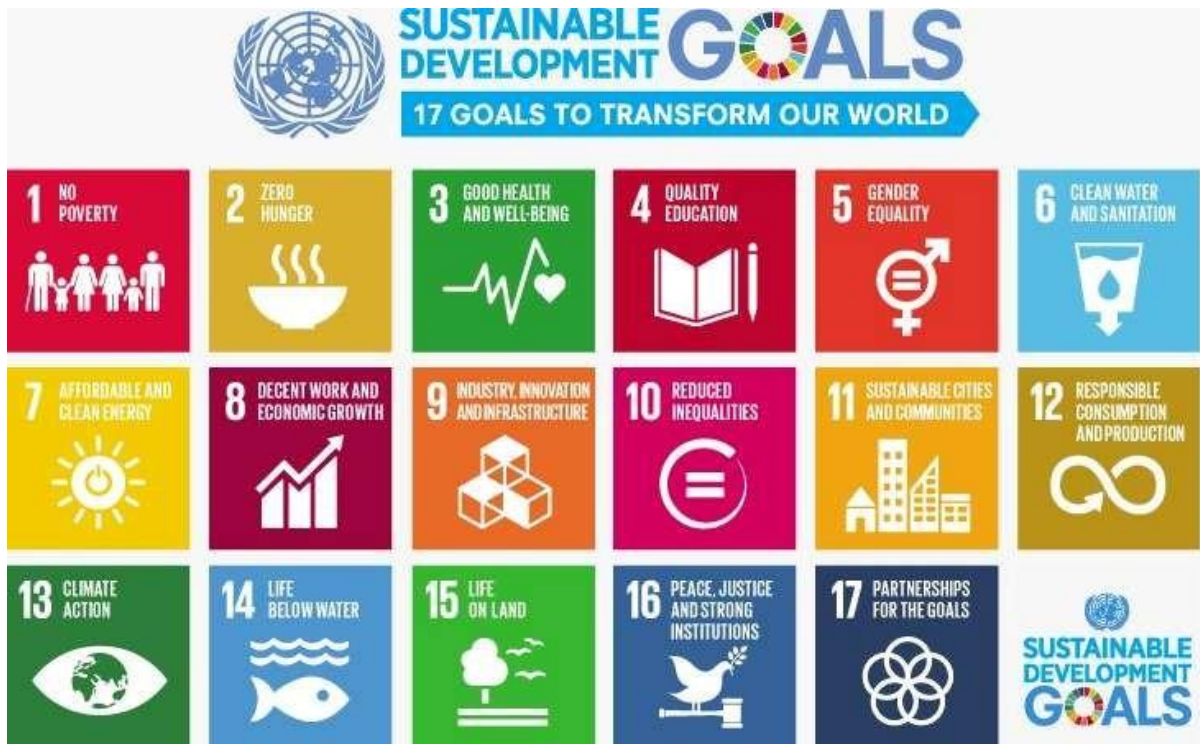
- 
**Hidden Text**  
1 suspect characters on 1 page  
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.


Page 2 of 45 - Integrity Overview
Submission ID trn:oid=1:3249651718

## 2. Details of mapping the project with the Sustainable Development Goals (SDGs).



### 1. SDG 8: Decent Work and Economic Growth

- **How it maps:**
  - Supports local cinemas and theater staff by increasing ticket sales through digital convenience.
  - Enables data-driven scheduling and targeted promotions, enhancing business sustainability.
- **Features that help:**
  - Digital ticketing reduces overhead costs.
  - Promotions for off-peak shows improve seat utilization and revenue.

### 2. SDG 9: Industry, Innovation and Infrastructure

- **How it maps:**
  - Encourages digital transformation of the entertainment industry.
  - Promotes innovation through real-time booking systems, mobile apps, and digital payments.
- **Features that help:**
  - Integration of tech like seat selection, QR code entry, etc.



- Cloud-based or scalable backend systems improve access and reliability.

### 3. SDG 11: Sustainable Cities and Communities

- **How it maps:**
  - Reduces physical queues and congestion at theaters.
  - Helps manage crowd flow during popular shows or festivals.
- **Features that help:**
  - Seat selection and pre-booking reduce overcapacity issues.
  - Can integrate location-based alerts or safety notifications.

### 4. SDG 12: Responsible Consumption and Production

- **How it maps:**
  - Digital tickets reduce paper use.
  - Analytics help theaters avoid overproduction of refreshments, optimizing resource use.
- **Features that help:**
  - E-ticketing and receipts via email/SMS.
  - Optional no-print mode at kiosks.

### 5. SDG 10: Reduced Inequalities

- **How it maps:**
  - Makes access to entertainment more inclusive by offering language filters, regional cinema listings, and potentially special needs seating.
- **Features that help:**
  - Language preferences.
  - Options for differently-abled seating or support.

