MOVIE BOOKING SYSTEM

A Project report submitted by

RAJEEV NAIR
(VAS23MCA - 2045)

to APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the degree of
Master of Computer Applications

Department of Computer Applications
Vidya Academy of Science & Technology
Thalakkottukara, Thrissur - 680 501
April 2025

Department of Computer Applications

Vidya Academy of Science & Technology

Thalakkottukara, Thrissur - 680 501

(http://www.vidyaacademy.ac.in)

CERTIFICATE

This is to certify that the report titled **MOVIE BOOKING SYSTEM** is a bona-fide record of the work related to the paper 20MCA246 MAIN PROJECT done by **RAJEEV NAIR (Reg. No. VAS23MCA - 2045)** of S4 MCA (2023 admissions) class of Vidya Academy of Science & Technology, Thrissur - 680501 in partial fulfillment of the requirement for the award of the Degree of Master of Computer Applications of APJ Abdul Kalam Technological University.

Guide/Supervisor

Name : Ms Siji KB

Signature : .....

Date : April 10, 2025

Head of Department

Name : Dr Reji C Joy

Signature : .....

Date : April 10, 2025

(Seal of Department of Computer Applications)

External Supervisor

Name : .....

Signature : .....

Date : .....

Declaration

I, **RAJEEV NAIR**, studying in Fourth Semester MCA (2023 admissions) class of Vidya Academy of Science & Technology, Thrissur – 680501, hereby declare that the project report ("MOVIE BOOKING SYSTEM") submitted by me for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of APJ Abdul Kalam Technological University, Kerala, is a bona fide work done by me under supervision of Ms Siji KB. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place	: Thrissur – 680501	Signature of student	: .....
Date	: April 10, 2025	Name of student	: RAJEEV NAIR

Acknowledgment

I wish to record my indebtedness and thankfulness to all who helped me prepare this report titled MOVIE BOOKING SYSTEM and present it in a satisfactory way. This report is part of my work related to the paper 20MCA246 MAIN PROJECT.

I am especially thankful to my guide and supervisor Ms Siji KB in the Department of Computer Applications for giving me valuable suggestions and critical inputs in the preparation of this report. I am also thankful to Dr Reji C Joy, the Head of Computer Applications for encouragement.

My friends in my class have always been helpful and I am grateful to them for their constant help and support.

RAJEEV NAIR
Reg. No. VAS23MCA - 2045
S4 MCA (2023 Admissions)
Vidya Academy of Science & Technology
Thrissur -680 501.

Synopsis

The Movie Booking System is a comprehensive, feature-rich web application designed to revolutionize the process of booking movie tickets online for both movie enthusiasts and cinema administrators. The system aims to eliminate the traditional hassles of ticket purchasing by offering a digital platform that is not only efficient but also intuitive and engaging. Through this application, users can browse available movies, select their preferred showtimes, choose specific seats from a dynamic seat layout, and complete their bookings seamlessly. One of the standout features of the system is the integrated snacks selection option, which allows users to order refreshments conveniently along with their movie tickets, enhancing the overall movie-going experience.

Another innovative addition is the ticket transfer feature, which empowers users to transfer their booked tickets to others in case they are unable to attend the show. This functionality adds a new layer of flexibility and user control to the system, making it more adaptable to real-life situations.

The platform is built using a robust technology stack, with Django serving as the backend framework, PostgreSQL as the relational database for secure and efficient data storage, and HTML, CSS, and JavaScript powering the frontend for a responsive and visually appealing user interface. In addition, the system integrates secure payment gateways, enabling users to complete their transactions safely and effortlessly. This integration ensures a seamless end-to-end booking process, from seat selection to payment confirmation.

Overall, the Movie Booking System is designed to offer a smooth, convenient, and enjoyable experience for moviegoers, while also simplifying and enhancing operational management for cinema administrators. By combining core ticketing functionalities with advanced features like snack ordering and ticket transfer, the platform aims to redefine the standard of online movie booking services.

Contents

Certificate from College 2

Declaration by Student 3

Acknowledgment 4

Synopsis 5

Table of Contents 6

List of Tables 9

List of Figures 10

1 INTRODUCTION 2

1.1 Project Overview 2

1.1.1 Objective 2

1.1.2 Key Features 3

1.1.3 Target Audience: 4

2 SYSTEM ANALYSIS 5

2.1 The Existing System 5

2.2 Proposed System 5

3 FEASIBILITY STUDY 6

3.1 Technical Feasibility 6

3.2 Economical Feasibility 6

3.3 Operational Feasibility 7

4 SYSTEM DESIGN 8

4.1 Input Design 8

4.2 Output Design 8
4.3 Database Design 9
4.3.1 Normalization 9
4.3.2 Database Tables and Keys 10

5 TESTING AND VALIDATION 12
5.1 Testing Methods 12
5.1.1 Unit Testing 12
5.1.2 Integration Testing 12
5.1.3 White Box Testing 13
5.1.4 Black Box Testing 13
5.2 Validation 13

6 SYSTEM SECURITY MEASURES 14
6.1 Key Security Features 14

7 CONCLUSION 15

8 SCOPE FOR FUTURE ENHANCEMENT 16

APPENDICES 17

A SCRUM PROCESS ARTIFACTS 18
A.1 Product Backlog 18
A.1.1 Functional Requirements 19
A.1.2 Non Functional Requirements 19
A.1.3 System Configuration 19
A.1.4 Conceptual Models 21
A.2 Sprint Details 23
A.3 Details of daily scrum meeting 24
A.4 Details of bi-weekly meeting 25

B VERSIONING 26

C DATA FLOW DIAGRAMS 27

D TABLE STRUCTURE 29

E SAMPLE SCREENSHOTS 32

F REPORTS **38**

F.0.1 Index Page 38
F.0.2 Login Page 43
F.0.3 Contact Page 46
F.0.4 About Page 47
F.0.5 Register Page 48
F.0.6 Movie Selection Page 50
F.0.7 Movie Detail Page 57
F.0.8 Seats Selection Page 62
F.0.9 Snacks Selection Page 70
F.0.10 Payment Summary Page 74
F.0.11 Booked History Page 81

Bibliography **90**

List of Tables

D.1 Registration Table 29
D.2 Movie Table 29
D.3 Location Table 29
D.4 Cast Table 30
D.5 Showtime Table 30
D.6 Snack Table 30
D.7 SelectedSnack Table 30
D.8 BookingDetail Table 31

List of Figures

A.1 Entity-Relationship (ER) Diagram of the System 20

C.1 DFD LEVEL 0 27
C.2 DFD LEVEL 1.1 Admin 28
C.3 DFD LEVEL- 1.2 User 28

E.1 index page 32
E.2 Registeration page 33
E.3 Login page 33
E.4 Home page 34
E.5 Showtime page 34
E.6 Seat Selection page 35
E.7 Snacks Selection page 35
E.8 Payment Summary 36
E.9 About page 36
E.10 Contact page 37
E.11 Booking History page 37

PROJECT REPORT

Chapter 1

INTRODUCTION

The Movie Booking System is a web-based platform designed to simplify and enhance the movie-going experience. Built with Django, it offers users a seamless way to browse movies, book tickets, manage bookings, and even handle ticket transfers. This system aims to eliminate the traditional hassles of cinema ticketing, providing a convenient and efficient solution for movie enthusiasts.

1.1 Project Overview

The Movie Booking System is a web-based application developed to facilitate online movie ticket booking and enhance cinema management. The system offers a user-friendly interface for moviegoers to browse movies, view showtimes, select seats, and purchase tickets and snacks. Additionally, it provides cinema administrators with tools to manage movie listings, schedules, pricing, and generate reports. A core objective of the Movie Booking System is to improve both the customer experience and cinema efficiency. To achieve this, the system incorporates features such as: online ticket booking, simplifying the ticket purchase process; interactive seat selection, allowing users to choose their preferred seats; a convenient snacks selection option, enabling users to order snacks concurrently with tickets; and a flexible ticket transfer feature, providing users with the ability to transfer tickets to others. By integrating these features, the Movie Booking System aims to provide a comprehensive and efficient solution for all stakeholders.

1.1.1 Objective

The Movie Booking System project aims to address the inefficiencies and limitations of traditional movie ticketing systems. The primary objective is to create a web-based platform that solves the problems of long queues, limited access to real-time information, and lack of flexibility in ticket management. Specific objectives include providing a convenient online platform for

users to book tickets from anywhere, offering an interactive seat selection feature with real-time availability, allowing users to pre-order snacks and refreshments online, implementing a ticket transfer mechanism to accommodate changes in user plans, and developing robust administrative tools to streamline cinema operations and improve management efficiency. Ultimately, the project aims to improve the overall movie-going experience and optimize cinema operations through the implementation of a user-centric and efficient online system.

1.1.2 Key Features

- Movie Browsing and Selection – Users can browse currently showing and upcoming movies, viewing details such as title, genre, synopsis, cast, and ratings.
- Showtime Management – The system displays available showtimes for each movie at different cinemas, allowing users to select their preferred time.
- Interactive Seat Selection – Users can view a visual representation of the theater layout and select their desired seats, with real-time availability updates.
- Online Ticket Booking – Users can book and purchase movie tickets online through a secure payment gateway.
- Snacks Selection– Users have the option to conveniently order snacks and refreshments during the ticket booking process.
- Ticket Transfer – The system allows users to transfer purchased tickets to other users, providing flexibility in case of changes.
- User Account Management – Users can create accounts, manage their booking history, save favorite movies, and store payment information.
- Booking History and Management – Users can view and manage their past and upcoming bookings, including the ability to cancel or modify reservations (within specified limits).
- Sales and Booking Reports – The system generates reports on ticket sales, booking statistics, and revenue, providing insights for cinema management.
- Secure Payment Gateway Integration – Integration with secure payment gateways ensures safe and reliable online transactions.
- Payment Summary – Ensures to give a summary of payment which shows the details of which seats and snacks has been selected.

- User Authentication and Authorization – The system provides secure user login and role-based access control to protect sensitive data and restrict access to administrative functions.
- Data Validation and Security – Implementation of data validation and security measures to prevent malicious attacks and ensure data integrity.

1.1.3 Target Audience:

The Movie Booking System is strategically designed to cater to a broad and diverse audience, encompassing both individual moviegoers seeking enhanced convenience and cinema operators aiming for streamlined efficiency. At its core, the system prioritizes the needs of contemporary moviegoers who value the accessibility and time-saving benefits of online platforms. This includes a wide spectrum of users, ranging from digitally native millennials and Gen Z individuals who are accustomed to conducting transactions and accessing entertainment online, to busy professionals and families who appreciate the ability to plan their movie outings in advance, avoiding the unpredictability of ticket availability and the inconvenience of waiting in physical queues. The system's user-friendly interface and intuitive navigation are crucial in accommodating users with varying levels of technical proficiency, ensuring that even those less familiar with online booking can easily browse movies, compare showtimes across different cinemas, and select their preferred seating arrangements. Furthermore, the inclusion of features like online snack ordering directly caters to the desire for a seamless and integrated entertainment experience, while the ticket transfer option addresses the dynamic nature of social planning, offering valuable flexibility to users.

Beyond individual consumers, the Movie Booking System provides significant value to cinema administrators and operational staff. By offering a centralized platform for managing movie schedules, ticket pricing, and real-time seat availability, the system empowers cinemas to optimize their resource allocation, reduce staffing needs at ticket counters, and minimize errors in booking management. The system's reporting and analytics capabilities provide valuable insights into sales trends, popular movies, and customer preferences, enabling data-driven decision-making for marketing strategies, inventory management, and overall business growth. In essence, the Movie Booking System is tailored to create a mutually beneficial ecosystem, enhancing the movie-going experience for customers while simultaneously improving the operational efficiency and profitability of cinemas.

Chapter 2

SYSTEM ANALYSIS

2.1 The Existing System

The existing online movie ticket booking systems, exemplified by platforms like BookMyShow and similar services, have significantly transformed the way users access and purchase movie tickets by offering a degree of convenience and efficiency. These systems generally enable users to browse movies, view showtimes, select from basic seating charts, and reserve tickets online, often through websites or mobile applications. Specifically, many existing platforms do not offer detailed user reviews or feedback mechanisms associated with specific showtimes, which could help users gauge the quality and audience reception of a particular screening. Furthermore, while online snack ordering can enhance the overall movie-going experience, it is not a standard feature in many existing systems. Similarly, the ability to easily transfer tickets to other users is also not commonly implemented.

2.2 Proposed System

The proposed Movie Booking System enhances the user experience and expands the functionality of existing online movie ticket booking platforms by incorporating key features that address identified limitations. This system integrates a comments section for each showtime, allowing users to view and contribute reviews and feedback specific to that screening. To elevate the user experience, it also includes an integrated pre-ordering option for snacks and beverages, enabling users to conveniently select and purchase refreshments alongside their tickets, streamlining the concession process. Furthermore, recognizing the dynamic nature of user plans, the system incorporates a flexible ticket transfer feature, allowing users to easily transfer their purchased tickets to other users. By integrating these enhancements, the proposed system aims to create a more engaging, informative, and user-friendly movie booking experience.

Chapter 3

FEASIBILITY STUDY

3.1 Technical Feasibility

The proposed Movie Booking System is technically feasible, utilizing standard web technologies such as HTML, CSS, JavaScript, and [Your Backend Framework/Language] (e.g., Python/Django, Node.js, Java/Spring) for the backend development. The system will rely on a robust database management system like PostgreSQL or MySQL to store and manage movie information, showtimes, user data, booking details, and snack inventory. The application can be hosted on cloud platforms such as AWS, Google Cloud, or Azure, offering scalability, high availability, and efficient maintenance. This cloud infrastructure will also ensure seamless integration and smooth performance as user demand grows and the system expands to accommodate more cinemas and users.

3.2 Economical Feasibility

Economically, the Movie Booking System is a viable solution as it reduces the time and effort required for users to book movie tickets and streamlines cinema operations. The automation of ticket booking, seat selection, and snack ordering minimizes errors and reduces the need for extensive staffing at ticket counters and concession stands. While the initial development will require investment in terms of time, resources, and hosting infrastructure, the long-term benefits include reduced operational costs for cinemas, improved customer satisfaction, and increased ticket and snack sales.

The application's potential to attract a broad user base—ranging from casual moviegoers to frequent cinema patrons—also presents opportunities for revenue generation through increased ticket sales and potential partnerships with food vendors. Additionally, the application's scalability ensures that it can handle increased traffic and transactions without significant additional costs.

The long-term savings, enhanced customer experience, and the ability to capitalize on the demand for online convenience make the project a sound investment with substantial financial returns for cinemas.

3.3 Operational Feasibility

The Movie Booking System is operationally feasible, as it automates and streamlines the movie ticket booking process, reducing manual work for both users and cinema staff. With an intuitive, user-friendly design, the application allows users to easily browse movies, select showtimes and seats, order snacks, and manage their bookings independently. The platform's real-time availability updates and secure payment processing ensure a smooth and efficient experience, enhancing user satisfaction.

On the administrative side, the system integrates seamlessly with existing cinema operations, providing tools for efficient management of movie schedules, ticket pricing, snack inventory, and booking records. The application's backend, built on [Your Backend Framework/Language], provides easy maintenance and minimal overhead for administrators. Additionally, the system's responsive design ensures that it can be accessed and operated across various devices (desktops, tablets, and smartphones), reducing the need for specialized hardware or software. The ease of use for both users and cinema staff ensures smooth adoption, with minimal training required for cinema personnel to manage the system effectively.

Chapter 4

SYSTEM DESIGN

4.1 Input Design

The input design of the Movie Booking System is focused on providing a user-friendly and efficient experience while ensuring data accuracy and security. Users register with a username and password, gaining access to personalized features like booking history and saved preferences. Users input their movie preferences by browsing available films, selecting showtimes and choosing seats from a theater layout. Users can also input their snack and beverage orders during the booking process.

The system validates user inputs, such as ensuring correct date and time formats, verifying seat availability, and confirming payment details. Users can modify their booking selections before final confirmation. At checkout, users enter payment information to complete transactions securely.

Cinema staff input movie details, showtimes, ticket prices, theater layouts, and snack inventory through an administrative interface. Input validation is crucial to ensure accurate movie information, showtimes, and pricing. Overall, the system simplifies the movie booking process, ensuring efficiency, accuracy, and ease of use for both customers and cinema staff.

4.2 Output Design

The output design of the Movie Booking System focuses on providing clear, accurate, and user-friendly presentation of information to both users and cinema staff. The system provides users with real-time feedback on seat availability, booking confirmations, and order summaries. Upon selecting a movie and showtime, the interface dynamically updates to display available seats and ticket prices, ensuring users are well-informed before finalizing their booking.

Users can view their booking history, upcoming reservations, and saved preferences in their

profile. The checkout page presents a summary of the selected movie, showtime, seats, snacks, and total cost, ensuring transparency before purchase. After completing a transaction, the system generates a booking confirmation with details such as movie title, showtime, seat numbers, snack order, payment information, and a booking ID.

4.3 Database Design

The database design of the Movie Booking System project is structured to efficiently manage and retrieve data related to movies, showtimes, bookings, users, and cinema operations. The system will use a relational database management system (RDBMS) such as PostgreSQL or MySQL to ensure data integrity and efficient querying.

The core tables include Movies, Showtimes, Theaters, Seats, Bookings, Users, and Snacks. The Movies table stores information about films, such as title, genre, synopsis, and cast. The Showtimes table stores details about movie screenings, including date, time, and theater. The Theaters table contains information about cinema locations and layouts. The Seats table manages seat availability and assignments within theaters. The Bookings table records ticket reservations, linking users, showtimes, seats, and payment information. The Users table stores user account details, including login credentials and booking history. The Snacks table stores information about available snacks and beverages.

Relationships between tables are carefully defined to ensure data consistency and enable efficient data retrieval. For example, a Movie can have multiple Showtimes, a Showtime is associated with a specific Theater, and a Booking involves a User, Showtime, and Seat. This structured approach facilitates accurate data management, efficient booking processes, and seamless user experiences, while the RDBMS's features ensure data integrity and optimized performance.

4.3.1 Normalization

Normalization is the process of organizing a relational database to reduce data redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships to ensure efficient storage, retrieval, and updating of data. The goal of normalization is to eliminate anomalies in insertion, deletion, and updating while maintaining consistency.

4.3.1.1 First Normal Form

A table is in First Normal Form (1NF) if:

All columns contain atomic (indivisible) values.

Each column contains only one type of data (no multiple values in a single field).

Each row has a unique identifier (primary key).

4.3.1.2 Second Normal Form

A table is in Second Normal Form (2NF) if:

It is already in First Normal Form (1NF) (all values are atomic).

It does not have partial dependencies, meaning every non-key attribute must be fully dependent on the entire primary key and not just part of it.

4.3.1.3 Third Normal Form

A table is in Third Normal Form (3NF) if:

It is already in Second Normal Form (2NF) (no partial dependencies).

It does not have transitive dependencies, meaning non-key attributes should depend only on the primary key and not on another non-key attribute.

4.3.2 Database Tables and Keys

A database table is a structured collection of related data organized into rows (records) and columns (fields). Each table typically represents an entity (e.g., Users, Orders, Products) and consists of attributes defining its properties.

4.3.2.1 Primary Key

A unique identifier for each record in a table.

Cannot have NULL values and must be unique.

4.3.2.2 Foreign Key

A column that establishes a relationship between two tables.

It references a Primary Key from another table.

Ensures referential integrity (no orphan records).

4.3.2.3 Candidate Key

A set of columns that can uniquely identify a record.

One of them is chosen as the Primary Key.

4.3.2.4 Composite Key

A key formed by two or more columns to uniquely identify a record.

Used when a single column is not enough to define uniqueness.

4.3.2.5 Alternate Key

A candidate key that was not selected as the primary key.

4.3.2.6 Super Key

A set of one or more attributes that can uniquely identify a record.

Includes the Primary Key and any additional columns.

4.3.2.7 Unique Key

Ensures that the values in a column remain unique but can have NULL values.

Chapter 5

TESTING AND VALIDATION

5.1 Testing Methods

5.1.1 Unit Testing

The unit testing for the Movie Booking System project covers key functionalities to ensure the system's reliability and correctness. This includes testing user authentication processes such as registration, login, logout, and password management. It also involves verifying the correct display of movie listings, the accuracy of movie details, and the proper functioning of search and filtering options. The entire booking process, from seat selection to payment processing, including ticket generation and booking confirmation, is also thoroughly tested. In addition, the snack ordering and ticket transfer features are validated to ensure they function as intended. Finally, the functionality of the admin dashboard, including movie management, showtime scheduling, and report generation, is tested. All core tests should pass successfully to confirm the system's stability, with future testing focusing on performance, security, and edge cases.

5.1.2 Integration Testing

Integration testing is conducted to ensure that the different modules of the Movie Booking System work together seamlessly after being independently tested. This testing validates the workflow, including movie selection, seat booking, snack selection, and payment processing. For example, when a user selects a movie and seats, the data must flow correctly to the snack ordering page and proceed accurately to the payment gateway. Integration testing ensures that the interaction between booking management, seat availability update, snack management, payment confirmation, and ticket generation modules happens without data loss or logical errors. This phase of testing helps identify any issues related to data exchange between components, ensuring that the system operates as a unified whole.

5.1.3 White Box Testing

White box testing focuses on the internal structure, logic, and flow of the code used in the Movie Booking System. Developers test individual functions, control flows, loops, and conditional statements to ensure they operate as expected. For instance, during seat selection, white box testing verifies whether the system correctly checks seat availability, prevents double bookings, and updates the seat status after booking. It also tests the payment processing logic, ensuring correct price calculations including ticket price, snacks, taxes, and discounts. Additionally, admin features like movie addition, showtime scheduling, and report generation undergo white box testing to ensure the correct execution of business rules and data handling. This approach helps detect hidden errors, unreachable code, and logic flaws in the system.

5.1.4 Black Box Testing

Black box testing evaluates the Movie Booking System based on its functionality without examining the internal code structure. Testers interact with the system through the user interface by entering inputs and checking if the expected outputs are produced. For example, black box testing is used to validate processes like user registration, movie search, seat selection, snack ordering, and payment confirmation. It also ensures that users can view their booking history, download tickets, and cancel bookings where allowed. Testing different user roles such as users and admin ensures that access rights and functionalities behave correctly according to user permissions. This method ensures that the system meets all specified requirements and delivers the intended outcomes from the user's perspective.

5.2 Validation

The validation system in the Movie Booking System project ensures data integrity, consistency, and user input accuracy. This includes validating user inputs in forms, such as registration details, payment information, and search queries, to ensure data is in the correct format and meets specified criteria. The system also verifies that seat availability is accurately tracked and updated to prevent double bookings. Booking data, including movie details, showtime, seat selection, and payment information, is validated to ensure accuracy and consistency. Payment transactions are validated to ensure secure and accurate processing, and snack orders are validated for accuracy and availability. These validation measures enhance the system's reliability, reduce errors, and improve the overall user experience.

Chapter 6

SYSTEM SECURITY MEASURES

The Movie Booking System project implements robust security measures to safeguard user data, transactions, and system integrity. User credentials are secured using Django's built-in authentication system with hashed passwords, and secure session management prevents unauthorized access. Cross-Site Request Forgery (CSRF) protection is implemented to prevent malicious form submissions, and input validation is used to mitigate risks of SQL injection and Cross-Site Scripting (XSS) attacks. The system enforces HTTPS for secure communication, ensuring that data transmitted between the user and the server is encrypted. Role-based access control restricts administrative functionalities, such as managing movie schedules and ticket pricing, to authorized cinema staff. Secure payment gateway integration ensures the confidentiality and integrity of financial transactions. Regular security updates, rate limiting for login attempts, and logging mechanisms are implemented to detect and mitigate potential threats.

6.1 Key Security Features

- User authentication using Django's built-in authentication system.
- Password hashing with Django's password hashing algorithms.
- Cross-Site Request Forgery (CSRF) protection enabled.
- Secure session management to prevent session hijacking.
- HTTPS enforcement for secure data transmission.
- Rate limiting on login attempts to prevent brute-force attacks.
- Input validation and sanitization to prevent XSS attacks.
- Secure payment gateway integration to protect financial data.

Chapter 7

CONCLUSION

The Movie Booking System project successfully delivers a comprehensive and dynamic platform that addresses the modern needs of both moviegoers and cinema administrators. By transitioning the traditional ticket purchasing process into a fully digital experience, the system offers users an intuitive, fast, and secure way to browse movies, select seats interactively, order snacks, and complete bookings from the comfort of their devices. One of the major achievements of the system is its focus on enhancing the user experience. Features such as real-time seat selection, integrated snack ordering, and seamless online payment create a smooth and engaging journey for users, minimizing the friction commonly associated with manual ticket bookings. Additionally, the ticket transfer option adds flexibility, empowering users to manage their bookings according to their changing needs.

Security has been given high priority in the development of the system. The integration of secure payment gateways and adherence to best practices in user authentication and data protection ensures that personal and financial information is safeguarded throughout the booking process. This builds trust among users and enhances the system's credibility.

In conclusion, the Movie Booking System not only streamlines the process of purchasing movie tickets but also enriches the overall movie-going experience. By combining powerful administrative tools with a user-centered design, the system successfully bridges the gap between convenience and functionality. It sets a strong foundation for future enhancements, such as mobile app integration, loyalty programs, and AI-driven movie recommendations, ensuring that it remains relevant and scalable in a rapidly evolving digital landscape.

Chapter 8

SCOPE FOR FUTURE ENHANCEMENT

The Movie Booking System project offers several avenues for future enhancements to expand its functionality and user experience. To cater to a wider audience, the database could be expanded with more diverse movie selections. User engagement could be improved by implementing personalized movie recommendations. To optimize the user's experience with concessions, the system could be enhanced to allow for dynamic snack and beverage options that change based on the specific showtime or day. For instance, offering breakfast items for morning shows or special deals on certain days.To enhance security around ticket sharing, the ticket transfer feature could be made more secure by adding verification steps, such as requiring confirmation from both the sender and receiver, or implementing a time limit for transfers.

Further enhancements could include integrating with social media platforms, supporting loyalty programs, and developing a mobile application. Advanced analytics dashboards and real-time inventory management for snacks could further benefit cinema operations. Exploring innovative technologies like VR/AR for virtual theater tours or enhanced movie information could also be considered.

APPENDICES

Appendix A

SCRUM PROCESS ARTIFACTS

A.1 Product Backlog

Priority	Product Backlog Items	User Story	Estimate (Hours)
1	Database Setup	As a developer, I want to create a structured database to store movie, user, booking, and snack data.	180
2	User Authentication	As a user, I want to register and log in securely to manage my bookings and preferences.	120
3	Movie Browsing	As a user, I want to browse available movies with details like title, genre and synopsis.	200
4	Showtime and Seat Selection	As a user, I want to view showtimes for a chosen movie.	250
5	Online Ticket Booking	As a user, I want to book and purchase movie tickets online through a secure payment gateway.	220
6	Snacks Selection	As a user, I want to order snacks and beverages along with my ticket purchase.	100
7	Booking History	As a user, I want to view and manage my past and upcoming booking history.	90
8	Ticket Transfer	As a user, I want to transfer my purchased tickets to another user.	110

A.1.1 Functional Requirements

The functional requirements of the Movie Booking System allow users to browse movies and showtimes, select seats and purchase tickets online. It provides an interface for users to manage their bookings, including viewing booking history and transferring tickets to others. The system enables users to order snacks and beverages along with their tickets. User authentication allows for account creation, login, and profile management. The system also facilitates secure payment processing for ticket purchases.

A.1.2 Non Functional Requirements

The non-functional requirements of the Movie Booking System project define system attributes that enhance usability, performance, security, and maintainability. The platform should ensure high availability and fast response times to provide a smooth booking experience. It must be scalable to handle a growing number of users, movies, and bookings without performance degradation. Security measures, including user authentication, data encryption, and protection against SQL injection and CSRF attacks, must be implemented to safeguard user data and transactions. The system should be compatible with major web browsers and responsive across different devices. Maintainability is crucial, requiring a modular code structure and clear documentation to facilitate future updates and debugging. Reliability should be ensured through error handling and system logs for tracking failures. Additionally, usability should be prioritized with an intuitive UI, simple navigation, and accessibility features to support a wide range of users.

A.1.3 System Configuration

The system configuration for the Movie Booking System project includes the hardware and software requirements needed for optimal performance. The backend is developed using Django with PostgreSQL as the database, ensuring reliability and efficient data handling. The frontend is built with HTML, CSS, and JavaScript, providing a responsive user interface. The server should have a minimum of 8GB RAM, a quad-core processor, and at least 50GB of disk space to handle product data, user transactions, and AI-driven assistance. The application is hosted on a cloud platform or dedicated server with SSL encryption for secure connections. Additionally, third-party libraries for data validation, authentication, and image processing are integrated to enhance functionality. The system requires Python 3.x, Django, PostgreSQL, and necessary dependencies for smooth operation.

A.1.3.1 ER Diagram

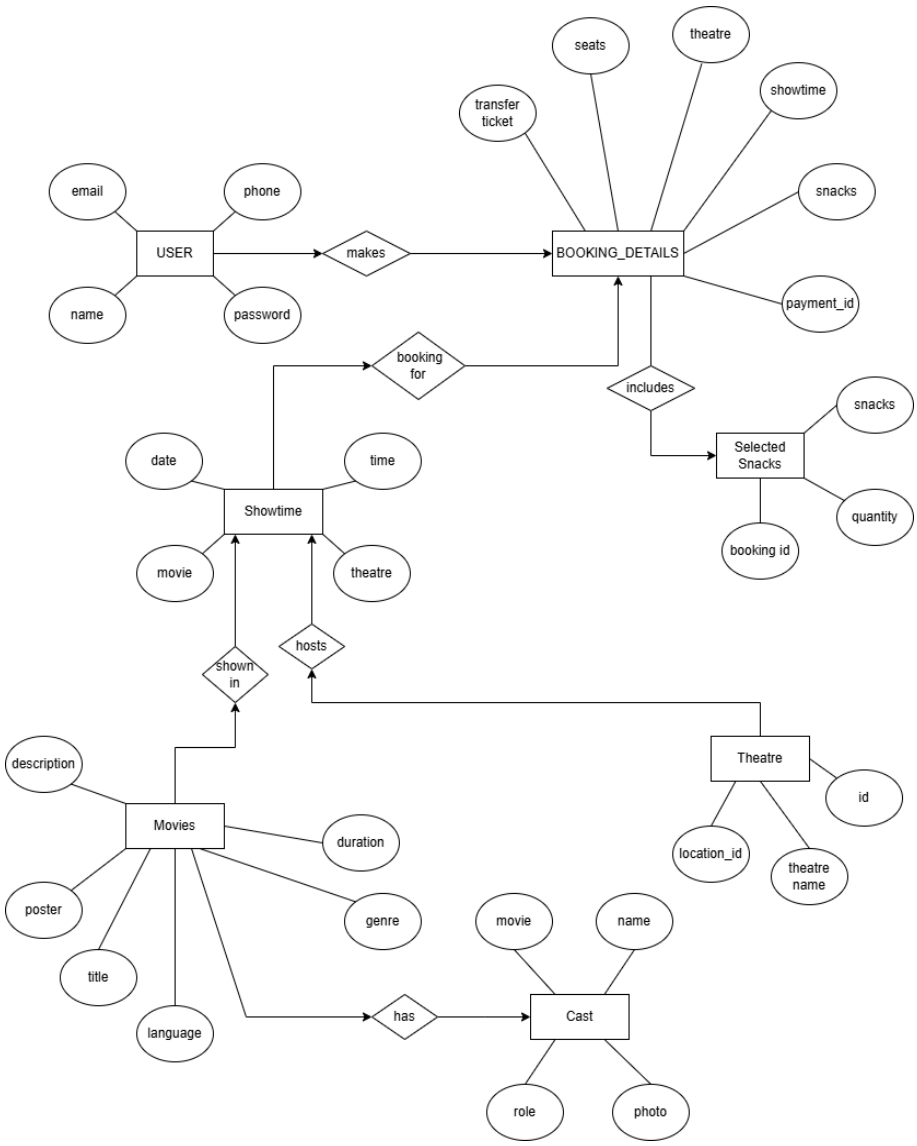

Figure A.1: Entity-Relationship (ER) Diagram of the System

A.1.3.2 Hardware Configuration

- Processor: Intel i5/AMD Ryzen 5 or higher
- RAM: Minimum 8GB (16GB recommended for optimal performance)
- Storage: At least 50GB SSD (preferably 100GB for better efficiency)
- GPU: Dedicated graphics card recommended for frontend rendering (optional)
- Network: Stable internet connection for database access and API calls
- Power Supply: Sufficient wattage to support continuous operation

A.1.3.3 Software Configuration

System Configuration defines the software and hardware requirements necessary to run the Movie Booking System project efficiently. It ensures optimal performance and compatibility across different platforms.

- Operating System: Windows 10/11, macOS, or Linux (Ubuntu 20.04 or higher)
- Backend: Django (Python-based framework)
- Database: PostgreSQL
- Frontend: HTML, CSS, JavaScript (with Bootstrap for responsiveness)
- Version Control: Git and GitHub
- Virtual Environment: Python venv for dependency management

A.1.4 Conceptual Models

Conceptual Models represent the high-level structure of the system, focusing on how different entities interact without diving into implementation details. This helps in understanding the core functionality and data flow of the Movie Booking System project.

- **Registration Model:** Stores user registration data.
- **Login Model:** Stores user login credentials.
- **Location Model:** Stores cinema location names.
- **Movie Model:** Stores movie details and relates to Location.
- **Cast Model:** Stores cast information and relates to Movie.

- **Comment Model:** Stores user comments and relates to Movie and Registration.
- **Theatre Model:** Stores theater details.
- **Showtime Model:** Stores showtime information and relates to Movie and Theatre.
- **Seat Model:** Stores seat details and relates to Showtime.
- **Snack Model:** Stores snack details.
- **SelectedSnack Model:** Stores selected snack information and relates to BookingDetail and Snack.
- **BookingDetail Model:** Stores booking details and relates to Registration, Movie, Theatre, Showtime, and Seat.

A.2 Sprint Details

Task Name	Description	Priority	Status	Duration (Days)
Sprint 1	User Authentication	High	Completed	5
Task 1	Implement user registration	High	Completed	3
Task 2	Implement user login/logout	High	Completed	2
Sprint 2	Movie Browsing	High	Completed	5
Task 1	Display movie listings	Low	Completed	3
Task 2	Implement movie details view	Medium	Completed	2
Sprint 3	Showtime Selection	High	Completed	4
Task 1	Display showtime availability	High	Completed	2
Task 2	Implement seat selection	High	Completed	2
Sprint 4	Booking	High	Completed	7
Task 1	Implement booking process	High	Completed	4
Task 2	Integrate payment gateway	High	Completed	3
Sprint 5	Booking Management	Medium	Completed	5
Task 1	View booking history	Medium	Completed	3
Task 2	Implement ticket transfer	High	Completed	2
Sprint 6	Snack Ordering	Medium	Completed	3
Task 1	Implement snack selection	Medium	Completed	2
Task 2	Add snack order to booking	Medium	Completed	1

A.3 Details of daily scrum meeting

In the development of the Movie Booking System project, daily scrum meetings were conducted to ensure smooth collaboration and progress tracking. These meetings took place in the development workspace at 9:30 am and were strictly time-boxed to 15 minutes, ensuring concise yet effective discussions.

All development team members were required to attend these meetings, including the Scrum Master and project owner, as they played crucial roles in guiding the team. Other stakeholders could attend as observers, making the scrum meetings an efficient way to share updates and maintain transparency.

The daily scrum was not a forum for in-depth problem-solving but rather a space for surfacing potential blockers. Any raised issues were taken offline and addressed by relevant team members after the meeting. Each participant answered the following three key questions, focusing on the progress made on Movie Booking System features:

The daily scrum was not a forum for in-depth problem-solving but rather a space for surfacing potential blockers. Any raised issues were taken offline and addressed by relevant team members after the meeting. Each participant answered the following three key questions:

- What did you work on yesterday related to the Movie Booking System?
- What are you planning to work on today?
- Are there any challenges or blockers in your development tasks for the Movie Booking System?

By focusing on completed and upcoming tasks related to booking functionality, user interface development and database integration, the team maintained a clear understanding of project progress. These meetings were not for status reporting to a superior but rather a space where developers made commitments to one another, fostering accountability and collaboration within the team.

A.4 Details of bi-weekly meeting

Bi-weekly Meeting No.	**Date**	**Details**
Bi-weekly Meeting 1	10-Jan-2025	Discussion on setting up the database to store movie, user, booking, and snack data.
Bi-weekly Meeting 2	24-Jan-2025	Planning and implementation of user authentication features, including registration and login/logout.
Bi-weekly Meeting 3	07-Feb-2025	Discussion on implementing movie browsing functionality, including displaying movie listings and details.
Bi-weekly Meeting 4	21-Feb-2025	Implementation of showtime and seat selection features, including displaying showtime availability and allowing users to select seats.
Bi-weekly Meeting 5	07-Mar-2025	Development of online ticket booking and integration of a secure payment gateway.
Bi-weekly Meeting 6	21-Mar-2025	Implementation of snack selection functionality to allow users to order snacks and beverages with their tickets.
Bi-weekly Meeting 7	28-Mar-2025	Development of booking history functionality and implementation of the ticket transfer feature.
Bi-weekly Meeting 8	02-Apr-2025	Finalizing the system and addressing any remaining bugs or issues before deployment.

Appendix B

VERSIONING

Sprint Number	Current Version Number
Sprint 1	V1.0
Sprint 2	V1.6
Sprint 3	V2.2
Sprint 4	V2.4
Sprint 5	V2.6
Sprint 6	V3.3

Appendix C

DATA FLOW DIAGRAMS

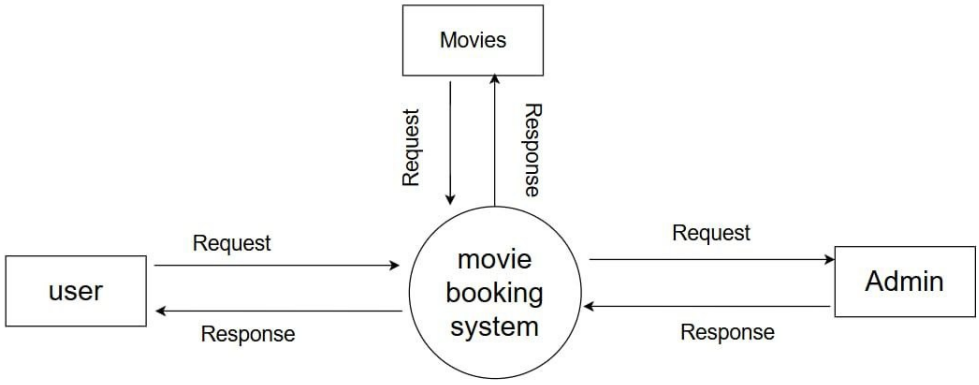

Figure C.1: DFD LEVEL 0

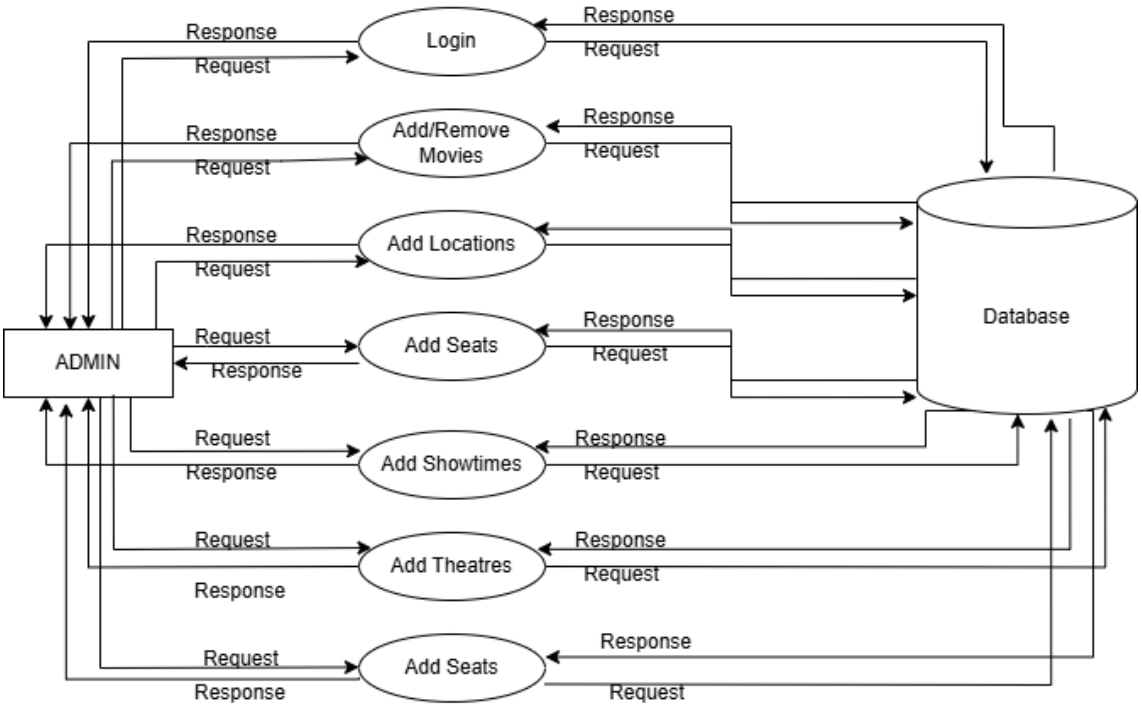

Figure C.2: DFD LEVEL 1.1 Admin

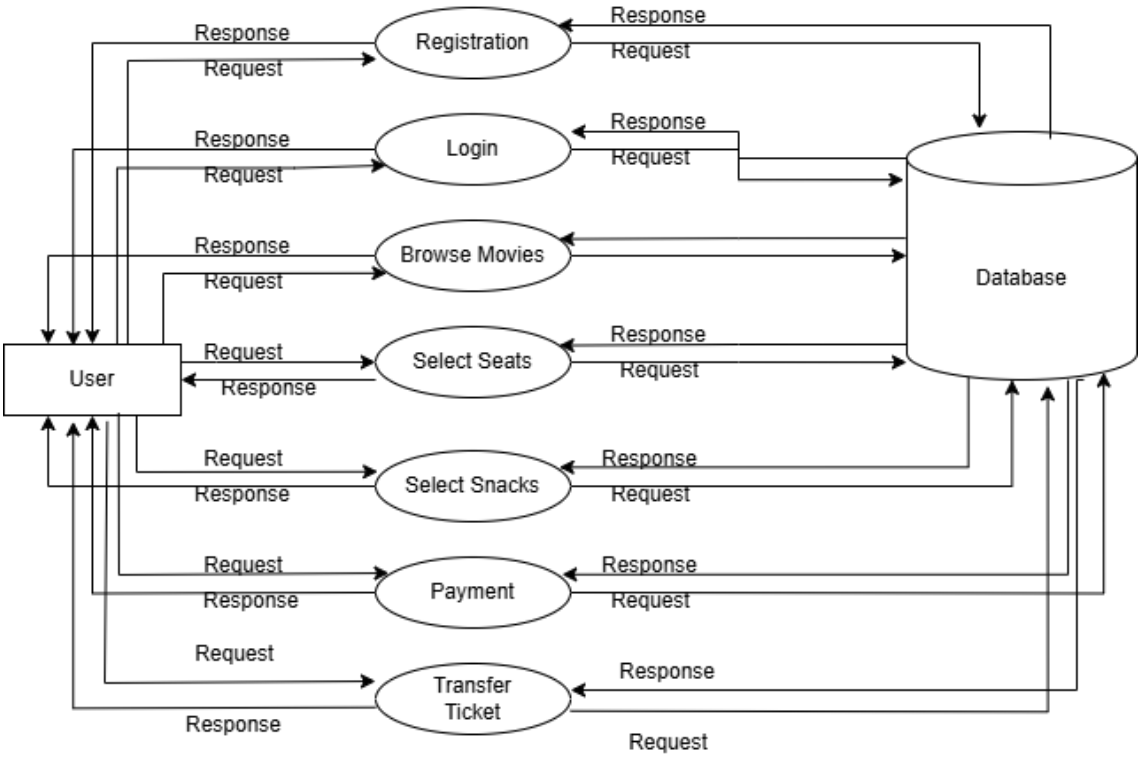

Figure C.3: DFD LEVEL- 1.2 User

Appendix D

TABLE STRUCTURE

Sl. No.	Field	Datatype	Constraint	Description
1	id	Integer	Primary Key, Auto-increment	Unique registration ID
2	fname	Varchar(100)	Not Null	Full name of user
3	email	Email	Not Null	User email address
4	username	Varchar(100)	Not Null	Unique username
5	Password	Varchar(100)	Not Null	Password
6	phone	Varchar(15)	Not Null	Contact number

Table D.1: Registration Table

Sl. No.	Field	Datatype	Constraint	Description
1	id	Integer	Primary Key, Auto-increment	Unique movie ID
2	title	Varchar(200)	Not Null	Movie title
3	description	Text	Nullable	Movie description
4	release_date	Date	Not Null	Date of release
5	genre	Varchar(100)	Not Null	Genre of the movie
6	star_rating	Float	Default 0.0	Star rating out of 5
7	language	Varchar(50)	Not Null	Movie language
8	duration	Duration	Not Null	Duration of the movie
9	poster	Image	Nullable	Poster image

Table D.2: Movie Table

Sl. No.	Field	Datatype	Constraint	Description
1	id	Integer	Primary Key, Auto-increment	Unique location ID
2	name	Varchar(100)	Not Null	Location name

Table D.3: Location Table

Sl. No.	Field	Datatype	Constraint	Description
1	id	Integer	Primary Key	Unique cast ID
2	movie_id	Integer	Foreign Key (Movie)	Linked movie
3	name	Varchar(100)	Not Null	Actor/actress name
4	role	Varchar(100)	Nullable	Role name
5	photo	Image	Nullable	Photo of actor/actress

Table D.4: Cast Table

Sl. No.	Field	Datatype	Constraint	Description
1	id	Integer	Primary Key	Unique showtime ID
2	movie_id	Integer	Foreign Key (Movie)	Movie showing
3	theatre_id	Integer	Foreign Key (Theatre)	Theatre showing it
4	date	Date	Not Null	Date of show
5	time	Time	Not Null	Time of show

Table D.5: Showtime Table

Sl. No.	Field	Datatype	Constraint	Description
1	snack_id	Integer	Primary Key, Auto-increment	Unique ID for each snack
2	name	Varchar(100)	Not Null	Name of the snack
3	price	Decimal(6,2)	Not Null	Price of the snack
4	image	Image path	Nullable	Image of the snack

Table D.6: Snack Table

Sl. No.	Field	Datatype	Constraint	Description
1	selected_snack_id	Integer	Primary Key, Auto-increment	Unique ID for each selected snack
2	booking_id	Integer	Foreign Key (BookingDetail)	Associated booking ID
3	snack_id	Integer	Foreign Key (Snack)	Selected snack ID
4	quantity	Integer	Default 1, Not Null	Quantity of snack selected

Table D.7: SelectedSnack Table

Sl. No.	Field	Datatype	Constraint	Description
1	booking_id	Integer	Primary Key, Auto-increment	Unique ID for each booking
2	user_id	Integer	Foreign Key (Registration)	User who made the booking
3	movie_id	Integer	Foreign Key (Movie), Nullable	Movie booked
4	theater_id	Integer	Foreign Key (Theatre), Nullable	Theater selected
5	showtime_id	Integer	Foreign Key (Showtime), Nullable	Show time selected
6	total_price	Decimal(10,2)	Not Null	Total price including seats/snacks
7	payment_id	Varchar(100)	Not Null	Payment reference ID
8	booking_time	Datetime	Auto Now Add	Timestamp of booking
9	status	Varchar(20)	Default 'Booked'	Booking status
10	transferred_to	Integer	Foreign Key (Registration), Nullable	User to whom the booking is transferred
11	transfer_time	Datetime	Nullable	Timestamp of transfer

Table D.8: BookingDetail Table

Appendix E

SAMPLE SCREENSHOTS

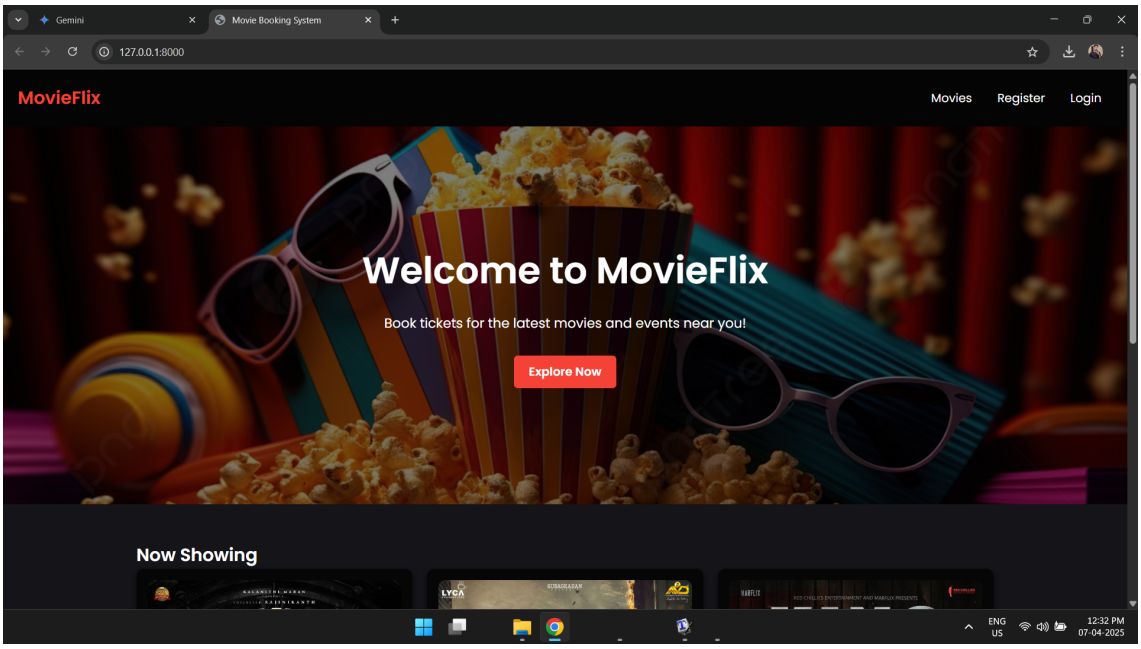

Figure E.1: index page

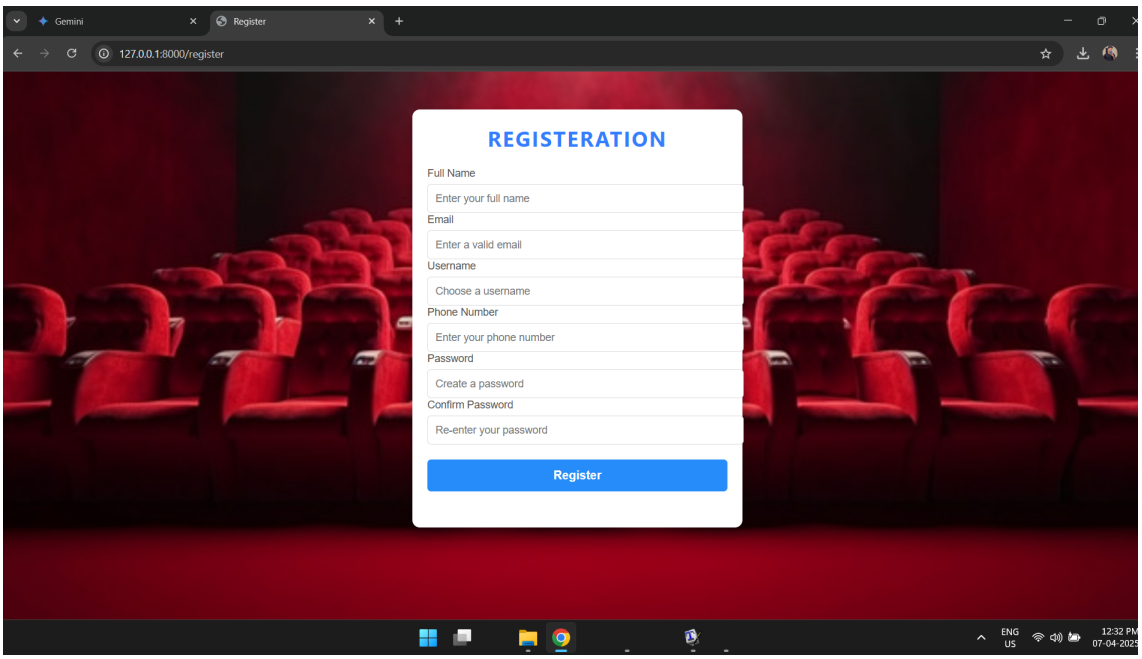

Figure E.2: Registeration page

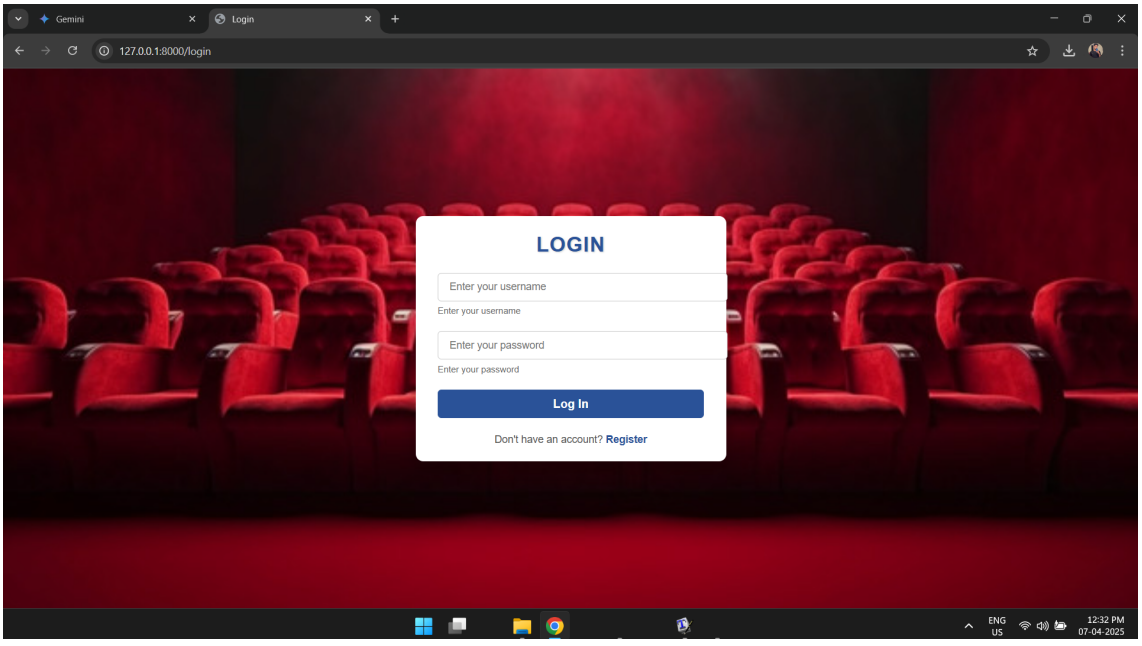

Figure E.3: Login page

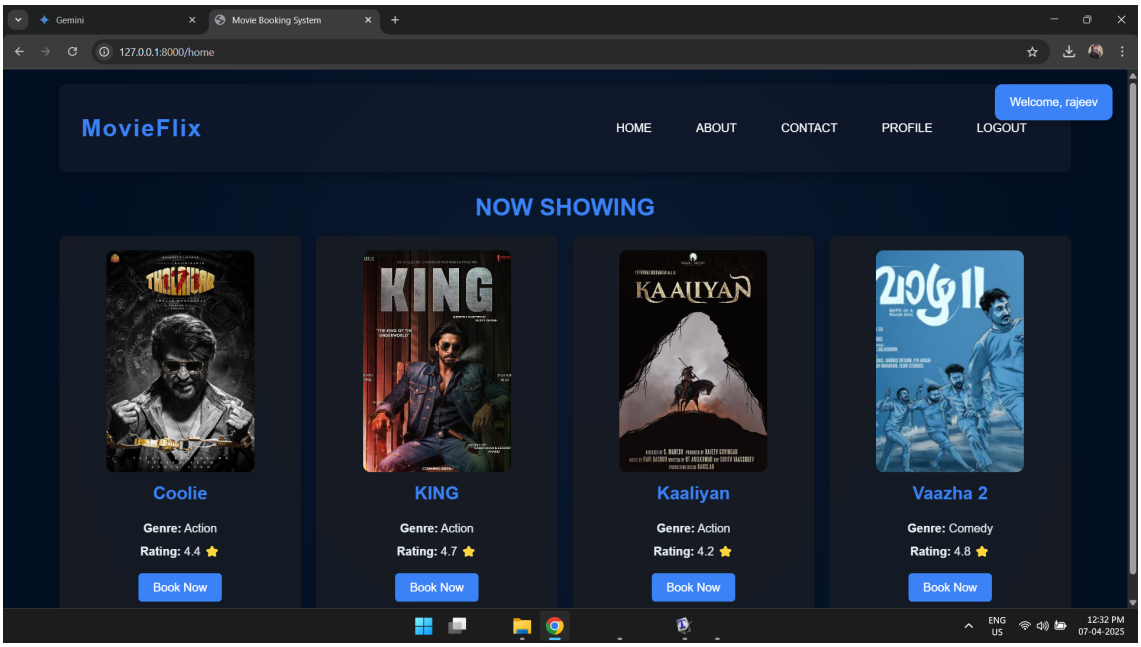

Figure E.4: Home page

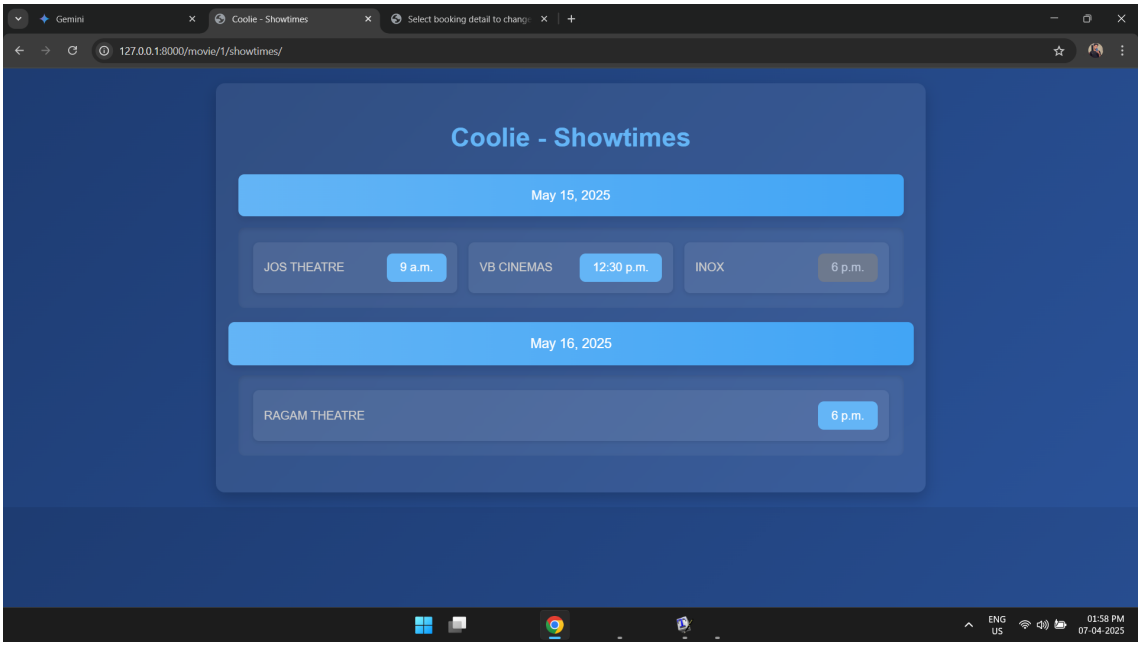

Figure E.5: Showtime page

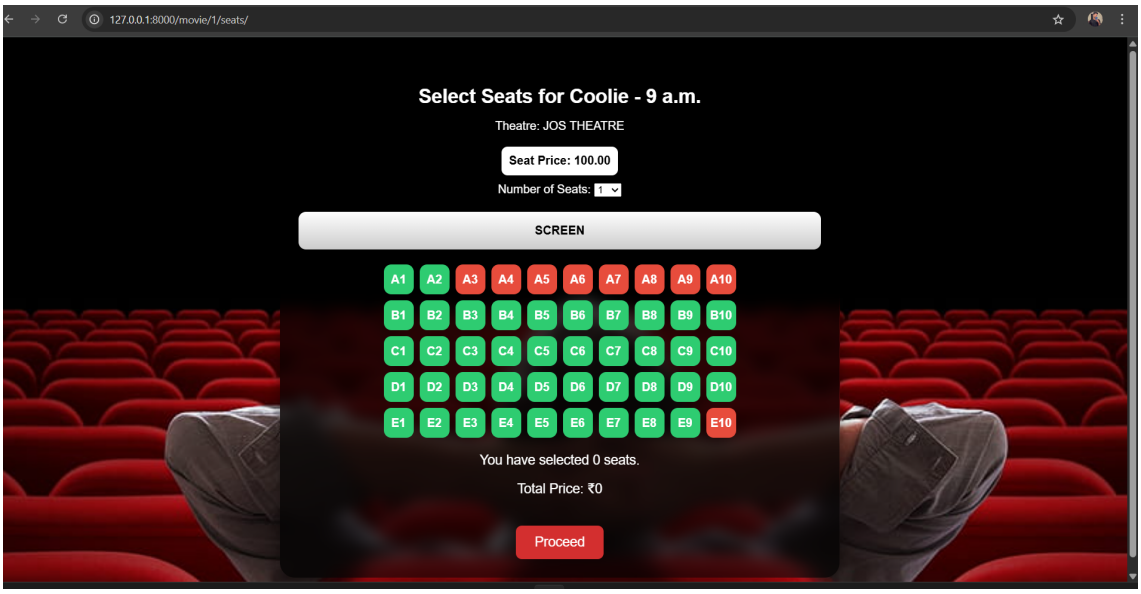

Figure E.6: Seat Selection page

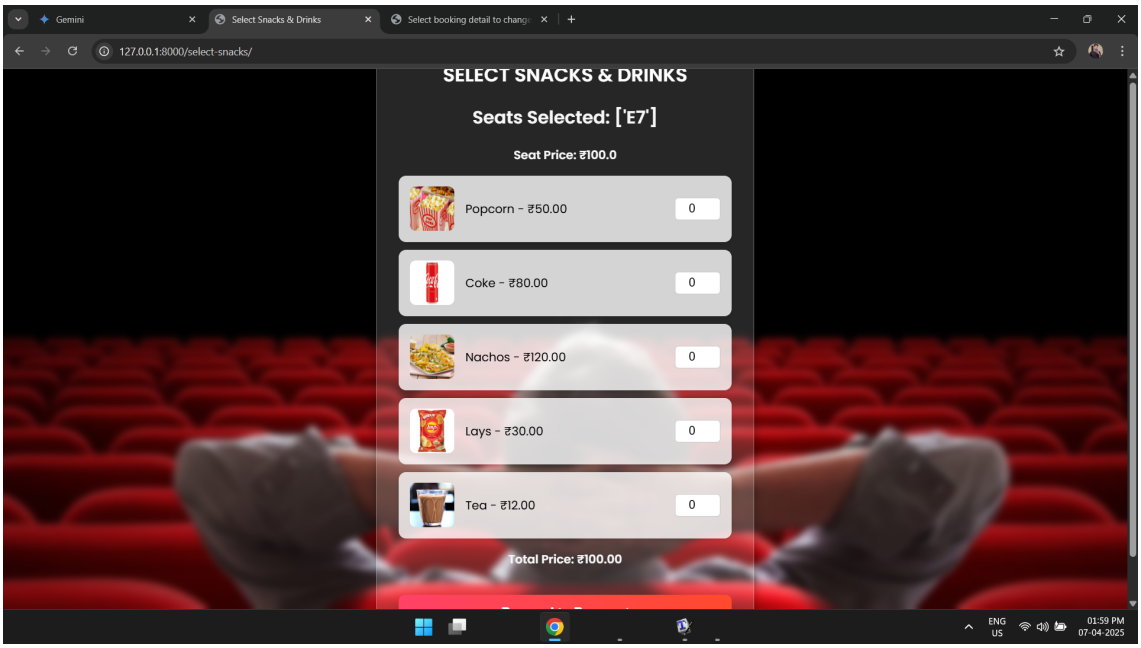

Figure E.7: Snacks Selection page

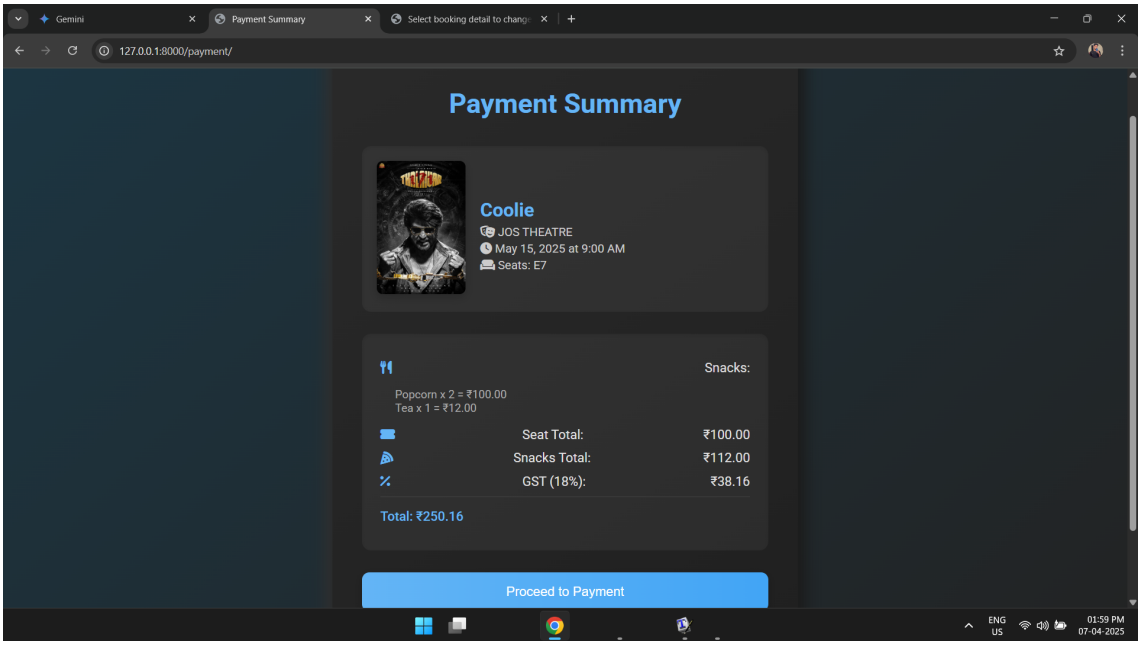

Figure E.8: Payment Summary

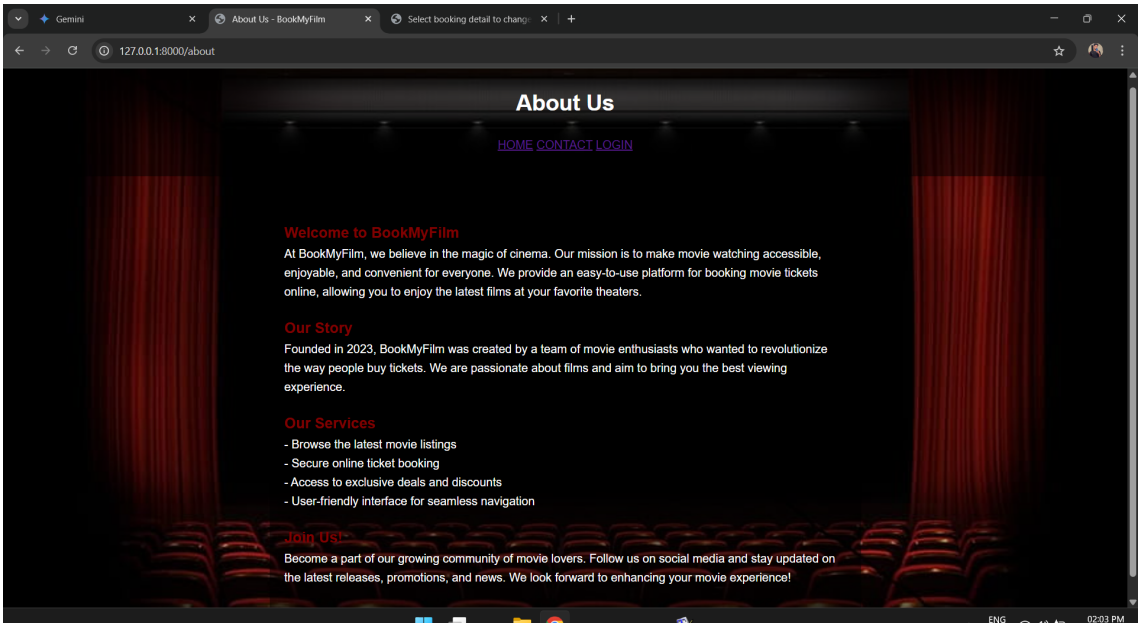

Figure E.9: About page

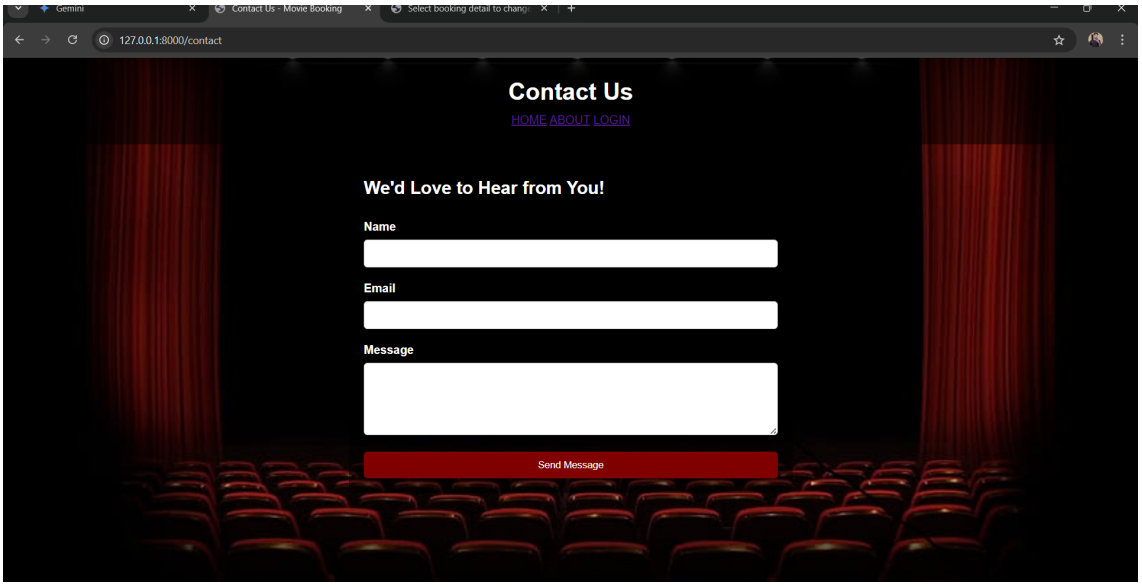

Figure E.10: Contact page

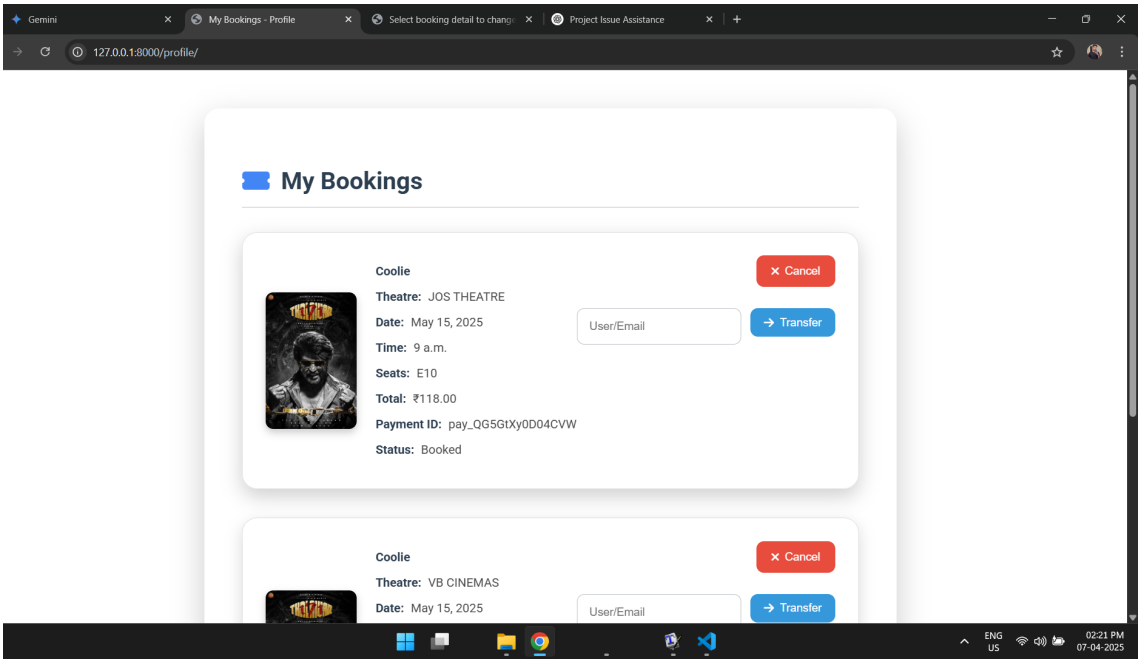

Figure E.11: Booking History page

Appendix F

REPORTS

F.0.1 Index Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Movie Booking System</title>
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }

        body {
            font-family: 'Poppins', sans-serif;
            background-color: #141419;
            color: #fff;
            margin: 0;
        }

        header {
```

```
        background-color: rgba(0, 0, 0, 0.8);
        padding: 20px;
        display: flex;
        justify-content: space-between;
        align-items: center;
        box-shadow: 0 2px 10px rgba(0, 0, 0, 0.7);
    }

    header .logo {
        font-size: 24px;
        font-weight: bold;
        color: #f44336;
    }

    nav a {
        color: #fff;
        text-decoration: none;
        margin: 0 15px;
        font-weight: 500;
        transition: color 0.3s ease;
        cursor: pointer;
    }

    nav a:hover {
        color: #f44336;
    }

    .hero {
        position: relative;
        background-image: url("https://png.pngtree.com/background/2023082
        background-size: cover;
        background-position: center;
        height: 70vh;
        display: flex;
        align-items: center;
        justify-content: center;
```

```
        text-align: center;
        color: #fff;
    }

    .hero::after {
        content: '';
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: rgba(0, 0, 0, 0.5);
        z-index: 1;
    }

    .hero-content {
        position: relative;
        z-index: 2;
    }

    .hero h1 {
        font-size: 50px;
        margin-bottom: 20px;
        font-weight: 600;
    }

    .hero p {
        font-size: 18px;
        margin-bottom: 30px;
    }

    .hero a {
        display: inline-block;
        padding: 10px 20px;
        font-size: 16px;
        color: #fff;
```

```
            background: #f44336;
            border-radius: 5px;
            text-decoration: none;
            font-weight: bold;
            transition: background 0.3s;
        }

        .movie-card h3 {
            font-size: 20px;
            margin-bottom: 10px;
            color: #f44336;
        }

        .movie-card p {
            font-size: 14px;
            color: #aaa;
        }

        footer {
            text-align: center;
            padding: 20px;
            background: #000;
            color: #aaa;
            margin-top: 20px;
        }

        footer a {
            color: #f44336;
            text-decoration: none;
        }

        footer a:hover {
            text-decoration: underline;
        }
    </style>
</head>
```

```
<body>
    <header>
        <div class="logo">MovieFlix</div>
        <nav>
            <a href="#" class="scroll-link" data-target="#movies-section">Mov
            <a href="register">Register</a>
            <a href="login">Login</a>
        </nav>
    </header>

    <div class="hero">
        <div class="hero-content">
            <h1>Welcome to MovieFlix</h1>
            <p>Book tickets for the latest movies and events near you!</p>
            <a href="#" class="scroll-link" data-target="#movies-section">Exp
        </div>
    </div>

    <div class="container" id="movies-section">
        <h2>Now Showing</h2>
        <div class="movies">
            <div class="movie-card">
                <img src="{% static 'img/coolie.jpeg' %}" alt="Movie Poster">
                <h3>COOLIE</h3>
                <p>A thrilling action you can't miss from LCU Universe.</p>
            </div>
            <div class="movie-card">
                <img src="{% static 'img/l2.jpg' %}" alt="Movie Poster">
                <h3>L2:Empuraan</h3>
                <p>Experience the mohanlal action on the big screen.</p>
            </div>
            <div class="movie-card">
                <img src="{% static 'img/king.jpg' %}" alt="Movie Poster">
                <h3>KING</h3>
                <p>Witness the power of greatest superstar in the world.</p>
            </div>
```

```
        </div>
    </div>

    <footer>
        <p>&copy; 2025 MovieFlix. All Rights Reserved.</p>
    </footer>

    <script>
        // JavaScript for Smooth Scrolling
        document.querySelectorAll('.scroll-link').forEach(link => {
            link.addEventListener('click', function (e) {
                e.preventDefault(); // Prevent default link behavior
                const target = document.querySelector(this.getAttribute('data
                if (target) {
                    target.scrollIntoView({
                        behavior: 'smooth', // Smooth scrolling
                        block: 'start' // Align to the top
                    });
                }
            });
        });
    </script>
</body>
</html>
```

F.0.2 Login Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
  <link rel="stylesheet" href="{% static 'css/style.css' %}">
  <style>
```

```
/* General Styles */
body {
  margin: 0;
  padding: 0;
  font-family: 'Arial', sans-serif;
  background: linear-gradient(to bottom right, #1e3c72, #2a5298);
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background: url("{% static 'img/bg.jpg' %}") no-repeat center center fi
  background-size: cover;
}

.wrapper {
  width: 360px;
  padding: 20px 30px;
  background: #fff;
  border-radius: 10px;
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3);
  text-align: center;
}

.input-field label {
  font-size: 12px;
  color: #666;
  margin-top: 5px;
  text-align: left;
}

button {
  width: 100%;
  padding: 10px 15px;
  background: #2a5298;
  color: #fff;
  font-size: 16px;
```

```
    font-weight: bold;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s ease;
}

button:hover {
    background: #1e3c72;
}

.register {
    margin-top: 20px;
    font-size: 14px;
    color: #555;
}

.register a {
    color: #2a5298;
    font-weight: bold;
    text-decoration: none;
}

.register a:hover {
    text-decoration: underline;
}

/* Responsive Design */
@media (max-width: 400px) {
    .wrapper {
        width: 90%;
        padding: 15px 20px;
    }

    h2 {
        font-size: 24px;
```

```
      }
    }
  </style>
</head>
<body>
  <div class="wrapper">
    <form method="post" action="">
      {% csrf_token %}
      <h2>Login</h2>
      <div class="input-field">
        <input type="text" name="username" id="username" placeholder="Enter y
        <label>Enter your username</label>
      </div>
      <div class="input-field">
        <input type="password" name="password" id="password" placeholder="Ent
        <label>Enter your password</label>
      </div>
      <button type="submit">Log In</button>
      <div class="register">
        <p>Don't have an account? <a href="register">Register</a></p>
      </div>
    </form>
  </div>
</body>
</html>
```

F.0.3 Contact Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Us - Movie Booking</title>
</head>
<body>
```

```
    <header>
        <h1>Contact Us</h1>
        <nav>
            <a href="{% url 'home' %}">HOME</a>
            <a href="{% url 'about' %}">ABOUT</a>
            <a href="{% url 'login' %}">LOGIN</a>
    </nav>
    </header>
    <div class="container">
        <h2>We'd Love to Hear from You!</h2>
        <form>
            <label for="name">Name</label>
            <input type="text" id="name" name="name" required>

            <label for="email">Email</label>
            <input type="email" id="email" name="email" required>

            <label for="message">Message</label>
            <textarea id="message" name="message" rows="5" required></textare

            <button type="submit">Send Message</button>
        </form>
    </div>

</body>
</html>
```

F.0.4 About Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Us - BookMyFilm</title>
</head>
```

```
<body>
    <header>
        <h1>About Us</h1>
        <nav>
                <a href="{% url 'home' %}">HOME</a>
                <a href="{% url 'contact' %}">CONTACT</a>
                <a href="{% url 'login' %}">LOGIN</a>
        </nav>
    </header>
    <div class="container">
        <h2>Welcome to BookMyFilm</h2>
        <p>
            At BookMyFilm, we believe in the magic of cinema. Our mission is
        </p>
        <h2>Our Story</h2>
        <p>
            Founded in 2023, BookMyFilm was created by a team of movie enthus
        </p>
        <h2>Our Services</h2>
        <p>
            - Browse the latest movie listings<br>
            - Secure online ticket booking<br>
            - Access to exclusive deals and discounts<br>
            - User-friendly interface for seamless navigation
        </p>
        <h2>Join Us!</h2>
        <p>
            Become a part of our growing community of movie lovers. Follow us
        </p>
    </div>

</body>
</html>
```

F.0.5 Register Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <title>Register</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
      /* General Styles */
      body {
        font-family: 'Arial', sans-serif;
        margin: 0;
        padding: 0;
        background: url("{% static 'img/bg.jpg' %}") no-repeat center center
        background-size: cover;
      }
        h2 {
          font-size: 24px;
        }
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h2>Registeration</h2>
      <div class="content">
        <form method="post" action="">
          {% csrf_token %}
          <div class="user-details">
            <div class="input-box">
              <span class="details">Full Name</span>
              <input type="text" name="fname" id="fname" placeholder="Enter y
            </div>
            <div class="input-box">
              <span class="details">Email</span>
```

```
            <input type="email" name="email" id="email" placeholder="Enter
          </div>
          <div class="input-box">
            <span class="details">Username</span>
            <input type="text" name="username" id="username" placeholder="C
          </div>
          <div class="input-box">
            <span class="details">Phone Number</span>
            <input type="text" name="phone" id="phone" placeholder="Enter y
          </div>
          <div class="input-box">
            <span class="details">Password</span>
            <input type="password" name="pass1" id="pass1" placeholder="Cre
          </div>
          <div class="input-box">
            <span class="details">Confirm Password</span>
            <input type="password" name="pass2" id="pass2" placeholder="Re-
          </div>
        </div>
        <div class="button">
          <input type="submit" value="Register">
        </div>
      </form>
    </div>
  </div>
</body>
</html>
```

F.0.6 Movie Selection Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Movie Booking System</title>
<style>
    /* General Styles */
    body {
        margin: 0;
        font-family: 'Poppins', sans-serif;
        background-color: #0a192f;
        color: #e6edf3;
        overflow-x: hidden;
        display: flex;
        flex-direction: column;
        min-height: 100vh; /* Ensure the body takes full viewport height
    }

    /* Background Animation */
    body::before {
        content: "";
        position: fixed;
        width: 100vw;
        height: 100vh;
        background: radial-gradient(circle, rgba(10, 25, 47, 0.9) 10%, rg
        z-index: -1;
    }

    #shell {
        width: 90%;
        margin: 0 auto;
        padding: 20px 0;
        flex-grow: 1; /* Allow the content to expand and fill available s
    }

    /* Header */
    #header {
        background: rgba(15, 23, 42, 0.9);
        backdrop-filter: blur(10px);
```

```
        color: #fff;
        display: flex;
        justify-content: space-between;
        align-items: center;
        padding: 20px 30px;
        border-radius: 10px;
        box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.3);
    }

    #navigation ul li a:hover {
        background: rgba(255, 255, 255, 0.2);
    }

    /* Welcome Popup */
    .popup {
        position: fixed;
        top: 20px;
        right: 20px;
        background: #3b82f6;
        color: #fff;
        padding: 15px 20px;
        border-radius: 10px;
        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
        font-size: 1rem;
        opacity: 0;
        transform: translateX(100px);
        transition: opacity 0.5s, transform 0.5s;
    }

    .popup.show {
        opacity: 1;
        transform: translateX(0);
    }

    /* Now Showing Section */
    .box {
```

```
    margin-top: 30px;
    text-align: center;
}

.box .head h1 {
    font-size: 2rem;
    margin-bottom: 20px;
    color: #3b82f6;
    text-transform: uppercase;
    animation: fadeInUp 1s ease-in-out;
}

.carousel {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); /* R
    gap: 20px;
    justify-content: center;
}

.movie {
    background: rgba(22, 27, 34, 0.9);
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.2);
    text-align: center;
    transition: transform 0.3s, box-shadow 0.3s;
    animation: fadeIn 1s ease-in-out;
    display: flex;
    flex-direction: column;
    align-items: center; /* Center content horizontally */
}

.movie:hover {
    transform: scale(1.05);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
}
```

```
.movie img {
    width: 200px; /* Set a fixed width */
    height: 300px; /* Set a fixed height */
    object-fit: cover; /* Crop and scale the image to fit */
    border-radius: 10px;
    margin-bottom: 15px;
}

.movie h2 {
    margin-top: 0; /* Remove default margin */
    font-size: 1.5rem;
    color: #3b82f6;
}

.movie p {
    margin: 5px 0;
}

.movie .button {
    margin-top: 15px;
    padding: 10px 20px;
    background: #3b82f6;
    color: #fff;
    border: none;
    border-radius: 5px;
    text-decoration: none;
    transition: background 0.3s ease;
}

.movie .button:hover {
    background: #2563eb;
}

@keyframes fadeInUp {
    from {
```

```
                opacity: 0;
                transform: translateY(20px);
            }
            to {
                opacity: 1;
                transform: translateY(0);
            }
        }

        @keyframes fadeIn {
            from {
                opacity: 0;
            }
            to {
                opacity: 1;
            }
        }
    </style>
</head>
<body>
    <div id="shell">
        <header id="header">
            <h1 id="logo">MovieFlix</h1>
            <nav id="navigation">
                <ul>
                    <li><a href="/">HOME</a></li>
                    <li><a href="{% url 'about' %}">ABOUT</a></li>
                    <li><a href="{% url 'contact' %}">CONTACT</a></li>
                    <li><a href="{% url 'profile' %}" class="button">PROFILE<
                    <li><a href="{% url 'login' %}" class="button">LOGOUT</a>
                </ul>
            </nav>
        </header>

        <div class="popup" id="welcomePopup">Welcome, {{ username }}</div>
```

```
        <main>
            <section class="box">
                <div class="head">
                    <h1>Now Showing</h1>
                </div>
                <div class="carousel">
                    {% for movie in movies %}
                        <div class="movie">
                            {% if movie.poster %}
                                <img src="{{ movie.poster.url }}" alt="{{ mov
                            {% else %}
                                <p>No poster available</p>
                            {% endif %}
                            <h2>{{ movie.title }}</h2>
                            <p><strong>Genre:</strong> {{ movie.genre }}</p>
                            <p><strong>Rating:</strong> {{ movie.star_rating
                            <a href="{% url 'moviedetail' movie.id %}" class=
                        </div>
                    {% empty %}
                        <p>No movies available.</p>
                    {% endfor %}
                </div>
            </section>
        </main>
    </div>

    <script>
        window.onload = function() {
            document.getElementById("welcomePopup").classList.add("show");
            setTimeout(() => {
                document.getElementById("welcomePopup").classList.remove("sho
            }, 3000);
        };
    </script>
</body>
</html>
```

F.0.7 Movie Detail Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ movie.title }} - Book Tickets</title>
    <style>
        /* General Styles */
        body {
            margin: 0;
            font-family: 'Poppins', sans-serif;
            background-color: #0a192f;
            color: #e6edf3;
            overflow-x: hidden;
        }

        /* Background Animation */
        body::before {
            content: "";
            position: fixed;
            width: 100vw;
            height: 100vh;
            background: radial-gradient(circle, rgba(10, 25, 47, 0.9) 10%, rg
            z-index: -1;
        }

        #shell {
            width: 90%;
            margin: 0 auto;
            padding: 20px 0;
        }

        /* Header */
        #header {
```

```
    background: rgba(15, 23, 42, 0.9);
    backdrop-filter: blur(10px);
    color: #fff;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px 30px;
    border-radius: 10px;
    box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.3);
}

#logo {
    font-size: 2rem;
    font-weight: bold;
    letter-spacing: 2px;
    color: #3b82f6;
}

#navigation ul {
    list-style: none;
    display: flex;
    align-items: center;
    padding: 0;
    margin: 0;
}

#navigation ul li {
    margin: 0 15px;
}

#navigation ul li a {
    color: #fff;
    text-decoration: none;
    font-size: 1rem;
    padding: 10px 15px;
    border-radius: 5px;
```

```
    transition: all 0.3s ease-in-out;
}

#navigation ul li a:hover {
    background: rgba(255, 255, 255, 0.2);
}

/* Movie Details Section */
.movie-header {
    display: flex;
    gap: 20px;
    align-items: center;
    background-color: #1e2a47;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}


.comment p {
    margin: 5px 0;
}

.comment-button {
    margin-top: 10px;
    padding: 10px 20px;
    background-color: #3b82f6;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.comment-button:hover {
    background-color: #2563eb;
}
```

```
        textarea {
            width: 100%;
            border-radius: 5px;
            padding: 10px;
            margin-top: 10px;
            background-color: #1e2a47;
            color: #e6edf3;
        }

        /* Footer */
        footer {
            margin-top: 50px;
            padding: 20px;
            background-color: #1e2a47;
            color: #e6edf3;
            text-align: center;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <div id="shell">
        <!-- Movie Header -->
        <div class="movie-header">
            <img src="{{ movie.poster.url }}" alt="{{ movie.title }}">
            <div class="movie-details">
                <h1 class="movie-title">{{ movie.title }}</h1>
                <p class="movie-info"><strong>Genre:</strong> {{ movie.genre
                <p class="movie-info"><strong>Language:</strong> {{ movie.lan
                <p class="movie-info"><strong>Duration:</strong> {{ movie.dur
                <p class="movie-info"><strong>Release Date:</strong> {{ movie
                <p class="rating"> {{ movie.star_rating }}</p>
                <a href="{% url 'showtime_selection' movie.id %}" class="book
            </div>
        </div>
```

```
<!-- About Movie -->
<section class="about-movie">
    <h2>About the Movie</h2>
    <p>{{ movie.description }}</p>
</section>

<!-- Comment Section -->
<section class="comment-section">
    <h2>Comments</h2>
    {% for comment in comments %}
    <div class="comment">
        <p><strong>{{ comment.user.fname }}</strong> ({{ comment.crea
        <p>{{ comment.content }}</p>
    </div>
    {% empty %}
        <p>No comments yet. Be the first to comment!</p>
    {% endfor %}
    {% if request.session.user_id %}
    <form method="POST">
        {% csrf_token %}
        <textarea name="comment" rows="4" placeholder="Write your com
        <button type="submit" class="comment-button">Post Comment</bu
    </form>
    {% else %}
        <p>You need to <a href="{% url 'login' %}">log in</a> to post
    {% endif %}
</section>

<!-- Cast Section -->
<section class="cast-section">
    <h2>Cast</h2>
    <div class="cast-list">
        {% for cast_member in movie.cast.all %}
        <div class="cast-item">
            <img src="{{ cast_member.photo.url }}" alt="{{ cast_membe
```

```
                    <p>{{ cast_member.name }}</p>
                    <p class="role">{{ cast_member.role }}</p>
                </div>
                {% empty %}
                <p>Cast information is not available.</p>
                {% endfor %}
            </div>
        </section>

        <!-- Footer -->
        <footer>
            <p>&copy; 2025 MovieFlix. All rights reserved.</p>
        </footer>
    </div>
</body>
</html>
```

F.0.8 Seats Selection Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Seat Booking - {{ showtime.movie.title }}</title>
    <link rel="stylesheet" href="{% static 'css/style1.css' %}">
    <style>
        body {
            font-family: 'Poppins', sans-serif;
            background: url("{% static 'img/bg.jpeg' %}") no-repeat center ce
            background-size: cover;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
```

```
        color: white;
        text-align: center;
    }

    .container {
        background: rgba(0, 0, 0, 0.7);
        padding: 25px;
        border-radius: 20px;
        box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
        backdrop-filter: blur(10px);
        width: 90%;
        max-width: 700px;
    }

    h1 {
        font-size: 26px;
        margin-bottom: 15px;
    }

    .screen {
        width: 100%;
        height: 50px;
        background: linear-gradient(to bottom, #fff, #ccc);
        color: black;
        font-weight: bold;
        line-height: 50px;
        margin: 20px 0;
        border-radius: 10px;
    }

    .seat-grid {
        display: grid;
        grid-template-columns: repeat(10, 40px);
        gap: 8px;
        justify-content: center;
        margin-bottom: 20px;
```

```
}

.seat {
    width: 40px;
    height: 40px;
    border-radius: 10px;
    cursor: pointer;
    background-color: #2ecc71;
    color: white;
    font-weight: bold;
    line-height: 40px;
    transition: 0.2s;
}

.seat.booked {
    background-color: #e74c3c;
    cursor: not-allowed;
}

.seat.selected {
    background-color: #f39c12;
}

.seat:hover {
    transform: scale(1.1);
}

.info-container {
    margin-top: 20px;
    font-size: 18px;
}

.btn {
    padding: 12px 25px;
    background: #d32f2f;
    color: white;
```

```
        border: none;
        cursor: pointer;
        font-size: 18px;
        border-radius: 8px;
        margin-top: 20px;
        transition: 0.3s;
    }

    .btn:hover {
        background: #b71c1c;
        transform: scale(1.05);
    }

    .container {
        background: rgba(0, 0, 0, 0.7);
        padding: 25px;
        border-radius: 20px;
        box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
        backdrop-filter: blur(10px);
        width: 90%;
        max-width: 700px;
    }

    h1 {
        font-size: 26px;
        margin-bottom: 15px;
    }

    .screen {
        width: 100%;
        height: 50px;
        background: linear-gradient(to bottom, #fff, #ccc);
        color: black;
        font-weight: bold;
        line-height: 50px;
        margin: 20px 0;
```

```
        border-radius: 10px;
    }

    .seat-grid {
        display: grid;
        grid-template-columns: repeat(10, 40px);
        gap: 8px;
        justify-content: center;
        margin-bottom: 20px;
    }

    .seat {
        width: 40px;
        height: 40px;
        border-radius: 10px;
        cursor: pointer;
        background-color: #2ecc71;
        color: white;
        font-weight: bold;
        line-height: 40px;
        transition: 0.2s;
    }

    .btn:hover {
        background: #b71c1c;
        transform: scale(1.05);
    }

    .seat-price-info {
        background-color: #ffffff;
        color: black;
        padding: 10px;
        border-radius: 8px;
        font-weight: bold;
        display: inline-block;
        box-shadow: 0px 0px 8px rgba(0, 0, 0, 0.2);
```

```
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Select Seats for {{ showtime.movie.title }} - {{ showtime.time }}
        <div class="theater-info">
            Theatre: <span id="theatre-name">{{ showtime.theatre.name }}</spa
        </div><br>

        <div class="seat-price-info">
            Seat Price: <span id="seat-price-display">{{ seats.0.price }}</sp
        </div>
        <br>

        <label for="numSeats">Number of Seats:</label>
        <select id="numSeats" onchange="updateSeatSelection()">
            {% for i in seat_range %}
                <option value="{{ i }}">{{ i }}</option>
            {% endfor %}
        </select>

        <div class="screen">SCREEN</div>

        <div class="seat-grid">
            {% for seat in seats %}
                <div class="seat {% if seat.is_booked %}booked{% else %}avail
                     id="seat-{{ seat.id }}"
                     data-seat-id="{{ seat.id }}"
                     data-seat-label="{{ seat.row }}{{ seat.number }}"
                     data-seat-price="{{ seat.price }}"
                     onclick="selectSeat(this)">
                    {{ seat.row }}{{ seat.number }}
                </div>
            {% empty %}
```

```
                <p>No seats available for this showtime.</p>
            {% endfor %}
        </div>

        <div class="info-container">
            <p>You have selected <span id="selected-count">0</span> seats.</p
            <p>Total Price: <span id="total-price">0</span></p>
            <button class="btn" onclick="proceedToPayment()">Proceed </button
        </div>
    </div>

    <form id="booking-form" method="post">
        {% csrf_token %}
        <input type="hidden" id="selected-seats" name="selected_seats">
    </form>

    <script>
        let selectedSeats = [];
        let maxSeats = 1;

        function updateSeatSelection() {
            maxSeats = parseInt(document.getElementById("numSeats").value);
            selectedSeats = [];
            document.querySelectorAll(".seat.selected").forEach(seat => seat.
            document.getElementById("selected-count").innerText = selectedSea
            updateTotalPrice();
        }

        function selectSeat(element) {
            if (!element.classList.contains("booked")) {
                const seatId = element.dataset.seatId;  // Store seat ID inst
                const seatPrice = parseInt(element.dataset.seatPrice);

                if (selectedSeats.includes(seatId)) {
                    element.classList.remove("selected");
                    selectedSeats = selectedSeats.filter(id => id !== seatId)
```

```
                } else {
                    if (selectedSeats.length < maxSeats) {
                        element.classList.add("selected");
                        selectedSeats.push(seatId);
                    } else {
                        alert("You can only select " + maxSeats + " seats.");
                    }
                }

                document.getElementById("selected-count").innerText = selecte
                updateTotalPrice();
            }
        }

        function updateTotalPrice() {
            let total = 0;
            document.querySelectorAll(".seat.selected").forEach(seat => {
                total += parseInt(seat.dataset.seatPrice);
            });
            document.getElementById("total-price").innerText = `{total}`;
        }

        function proceedToPayment() {
            if (selectedSeats.length > 0) {
                document.getElementById("selected-seats").value = selectedSea
                document.getElementById("booking-form").submit();
            } else {
                alert("Please select at least one seat.");
            }
        }
    </script>
</body>
</html>
```

F.0.9 Snacks Selection Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Select Snacks & Drinks</title>
    <link rel="stylesheet" href="{% static 'css/style1.css' %}">
    <style>
        /* Custom Fonts */
        @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@30

        body {
            font-family: 'Poppins', sans-serif;
            background: url('{% static "img/bg.jpeg" %}') no-repeat center ce
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            backdrop-filter: blur(5px);
        }

        .container {
            background: rgba(255, 255, 255, 0.15);
            backdrop-filter: blur(10px);
            padding: 30px;
            border-radius: 15px;
            box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
            width: 450px;
            text-align: center;
            border: 1px solid rgba(255, 255, 255, 0.3);
        }
```

```
h1 {
    font-size: 26px;
    color: #fff;
    text-transform: uppercase;
    font-weight: 600;
}

h2, p {
    color: #f8f8f8;
}

.item {
    display: flex;
    align-items: center;
    justify-content: space-between;
    background: rgba(255, 255, 255, 0.8);
    padding: 15px;
    margin: 10px 0;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    transition: transform 0.2s ease-in-out;
}

.item:hover {
    transform: scale(1.05);
}

.item img {
    width: 60px;
    height: 60px;
    border-radius: 8px;
    margin-right: 15px;
}

.item label {
```

```
            flex-grow: 1;
            text-align: left;
            font-size: 16px;
            font-weight: 500;
        }

        .quantity {
            width: 50px;
            padding: 5px;
            font-size: 16px;
            text-align: center;
            border-radius: 5px;
            border: 1px solid #ccc;
        }

        .btn {
            padding: 14px 25px;
            background: linear-gradient(45deg, #ff416c, #ff4b2b);
            color: white;
            border: none;
            cursor: pointer;
            font-size: 18px;
            border-radius: 8px;
            width: 100%;
            margin-top: 20px;
            transition: 0.3s ease-in-out;
            font-weight: 600;
        }

        .btn:hover {
            transform: scale(1.05);
            background: linear-gradient(45deg, #ff4b2b, #ff416c);
        }
    </style>
</head>
<body>
```

```
<div class="container">
    <h1>Select Snacks & Drinks</h1>
    <h2>Seats Selected: <span id="seat-details">{{ selected_seats }}</spa
    <p><strong>Seat Price: <span id="seat-price">{{ seat_price }}</span><

    <form id="snacks-form" method="post" action="{% url 'select_snacks' %


        {% csrf_token %}
        <input type="hidden" name="movie_id" value="{{ movie.id }}">
        <input type="hidden" name="selected_seats" id="selected_seats" va
        <input type="hidden" id="total_price" name="total_price" value="{

        {% for snack in snacks %}
        <div class="item">
            <img src="{{ snack.image.url }}" alt="{{ snack.name }}">
            <label for="snack-{{ forloop.counter }}">
                {{ snack.name }} - {{ snack.price }}
            </label>
            <input type="number" id="snack-{{ forloop.counter }}" name="s
        </div>
        {% endfor %}

        <p><strong>Total Price: <span id="total-amount">{{ seat_price }}<

        <button type="submit" class="btn">Proceed to Payment</button>
    </form>
</div>

<script>
    document.addEventListener("DOMContentLoaded", function () {
        updateTotal(); // Ensure total is updated when the page loads

        document.querySelectorAll(".quantity").forEach(input => {
            input.addEventListener("input", updateTotal);
        });
```

```
            document.getElementById("snacks-form").addEventListener("submit",
                let selectedSeats = document.getElementById("seat-details").i
                if (!selectedSeats || selectedSeats === "[]") {
                    alert("Please select at least one seat before proceeding
                    event.preventDefault();
                }
            });
        });

        function updateTotal() {
            let seatPrice = parseFloat(document.getElementById("seat-price").
            let total = seatPrice;

            document.querySelectorAll(".quantity").forEach(input => {
                let price = parseFloat(input.getAttribute("data-price")) || 0
                let quantity = parseInt(input.value) || 0;
                total += price * quantity;
            });

            document.getElementById("total-amount").innerText = total.toFixed
            document.getElementById("total_price").value = total;
        }
    </script>
</body>
</html>
```

F.0.10 Payment Summary Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Payment Summary</title>
```

```
<link rel="stylesheet" href="{% static 'css/style1.css' %}">
<script src="https://checkout.razorpay.com/v1/checkout.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
<style>
    @import url('https://fonts.googleapis.com/css2?family=Roboto:wght@400
    body {
        font-family: 'Roboto', sans-serif;
        background: linear-gradient(135deg, #1c3643, #232323);
        color: #e0e0e0;
        margin: 0;
        padding: 20px;
        min-height: 100vh;
        display: flex;
        justify-content: center;
        align-items: center;
        box-sizing: border-box;
    }

    .payment-card {
        background: linear-gradient(145deg, #2a2a2a, #202020);
        box-shadow: 10px 10px 20px #1a1a1a, -10px -10px 20px #303030;
        border-radius: 18px;
        padding: 40px;
        max-width: 550px;
        width: 100%;
        animation: cardFadeIn 0.8s ease-in-out;
    }

    @keyframes cardFadeIn {
        from { opacity: 0; transform: translateY(30px); }
        to { opacity: 1; transform: translateY(0); }
    }

    .payment-header {
        text-align: center;
        margin-bottom: 35px;
```

```
}

.payment-header h1 {
    font-size: 2.3rem;
    color: #64b5f6;
    margin-bottom: 10px;
    animation: headerFadeIn 0.8s ease-in-out;
}

@keyframes headerFadeIn {
    from { opacity: 0; transform: translateY(-20px); }
    to { opacity: 1; transform: translateY(0); }
}

.movie-info {
    display: flex;
    align-items: center;
    margin-bottom: 30px;
    background: rgba(255, 255, 255, 0.05);
    padding: 20px;
    border-radius: 12px;
    box-shadow: inset 0 2px 5px rgba(0, 0, 0, 0.1);
}

.movie-poster img {
    max-width: 120px;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
    margin-right: 20px;
    animation: posterScaleIn 0.8s ease-in-out;
}
@keyframes cardFadeIn {
    from { opacity: 0; transform: translateY(30px); }
    to { opacity: 1; transform: translateY(0); }
}
```

```
.payment-header {
    text-align: center;
    margin-bottom: 35px;
}

.payment-header h1 {
    font-size: 2.3rem;
    color: #64b5f6;
    margin-bottom: 10px;
    animation: headerFadeIn 0.8s ease-in-out;
}

@keyframes headerFadeIn {
    from { opacity: 0; transform: translateY(-20px); }
    to { opacity: 1; transform: translateY(0); }
}

.movie-info {
    display: flex;
    align-items: center;
    margin-bottom: 30px;
    background: rgba(255, 255, 255, 0.05);
    padding: 20px;
    border-radius: 12px;
    box-shadow: inset 0 2px 5px rgba(0, 0, 0, 0.1);
}

.summary-details p i {
    margin-right: 10px;
    color: #64b5f6;
}

.snack-list {
    list-style: none;
    padding-left: 20px;
}
```

```
.snack-list li {
    font-size: 0.95rem;
    color: #aaa;
}

.total-amount {
    font-size: 1.4rem;
    font-weight: 500;
    color: #64b5f6;
    margin-top: 25px;
    padding-top: 15px;
    border-top: 1px solid rgba(255, 255, 255, 0.1);
    text-align: right;
}

.payment-button {
    width: 100%;
    padding: 16px;
    font-size: 1.1rem;
    font-weight: 500;
    border: none;
    border-radius: 10px;
    background: linear-gradient(to right, #64b5f6, #42a5f5);
    color: white;
    cursor: pointer;
    transition: transform 0.3s ease;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
}

.payment-button:hover {
    transform: scale(1.03);
}

.secure-badge {
    font-size: 0.9rem;
```

```
            text-align: center;
            margin-top: 15px;
            color: #aaa;
        }
    </style>
</head>
<body>

<div class="payment-card">
    <div class="payment-header">
        <h1>Payment Summary</h1>
    </div>

    <div class="movie-info">
        <div class="movie-poster">
            <img src="{{ movie.poster.url }}" alt="{{ movie.title }}">
        </div>
        <div class="movie-details">
            <h2>{{ movie.title }}</h2>
            <p><i class="fas fa-theater-masks"></i> {{ theatre.name }}</p>
            <p><i class="fas fa-clock"></i> {{ showtime.date }} at {{ showtim
            <p><i class="fas fa-couch"></i> Seats: {{ selected_seats|join:",
        </div>
    </div>

    <div class="summary-details">
        <p><i class="fas fa-utensils"></i> Snacks:</p>
        {% if snacks_with_qty %}
            <ul class="snack-list">
                {% for item in snacks_with_qty %}
                    <li>{{ item.snack.name }} x {{ item.quantity }} = {{ item
                {% endfor %}
            </ul>
        {% else %}
            <p style="color: #ccc; margin-left: 10px;">No snacks selected.</p
        {% endif %}
```

```
            <p><i class="fas fa-ticket-alt"></i> Seat Total: <span>{{ seat_price|
            <p><i class="fas fa-pizza-slice"></i> Snacks Total: <span>{{ snacks_t
            <p><i class="fas fa-percent"></i> GST (18%): <span>{{ gst_amount|floa
            <p class="total-amount">Total: {{ final_amount|floatformat:2 }}</p>
        </div>

        <button type="button" class="payment-button" id="pay-button">Proceed to P
        <p class="secure-badge"><i class="fas fa-lock"></i> Secure Payment Gatewa
    </div>

    <script>
        document.getElementById("pay-button").onclick = function () {
            let confirmation = confirm("Are you sure you want to proceed with the
            if (!confirmation) return;

            var options = {
                "key": "{{ razorpay_key }}",
                "amount": "{{ final_amount|floatformat:0 }}00",
                "currency": "INR",
                "name": "Movie Booking",
                "description": "Movie Ticket Payment",
                "order_id": "{{ razorpay_order_id }}",
                "callback_url": "{% url 'payment_success' %}",
                "prefill": {
                    "name": "{{ user.fname }}",
                    "email": "{{ user.email }}",
                    "contact": "{{ user.phone }}"
                },
                "theme": {
                    "color": "#64b5f6"
                }
            };
            var rzp = new Razorpay(options);
            rzp.open();
        };
```

```
</script>

</body>
</html>
```

F.0.11 Booked History Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Bookings - Profile</title>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            background-image: url('https://source.unsplash.com/1920x1080/?mov
            background-size: cover;
            background-position: center;
            background-attachment: fixed;
            margin: 0;
            padding: 0;
            color: #333;
            display: flex;
            flex-direction: column;
            min-height: 100vh;
            overflow-x: hidden;
            transition: background-image 0.5s ease-in-out;
        }

        .container {
            max-width: 1100px;
            margin: 50px auto;
            background: rgba(255, 255, 255, 0.97);
```

```
    padding: 50px;
    border-radius: 20px;
    box-shadow: 0 15px 40px rgba(0, 0, 0, 0.2);
    flex-grow: 1;
    backdrop-filter: blur(12px);
    animation: fadeIn 0.9s ease-out;
}

@keyframes fadeIn {
    from { opacity: 0; transform: translateY(25px); }
    to { opacity: 1; transform: translateY(0); }
}

h2 {
    text-align: left;
    color: #2c3e50;
    margin-bottom: 40px;
    font-weight: 700;
    font-size: 32px;
    border-bottom: 2px solid #e0e0e0;
    padding-bottom: 15px;
    display: flex;
    align-items: center;
}

h2 i {
    margin-right: 15px;
    color: #4285f4;
}

.message {
    margin-bottom: 30px;
    padding: 20px;
    border-radius: 10px;
    font-size: 16px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.12);
```

```
    animation: slideInLeft 0.6s ease-out;
}

@keyframes slideInLeft {
    from { transform: translateX(-25px); opacity: 0; }
    to { transform: translateX(0); opacity: 1; }
}

.message.success {
    background-color: #e6f7ec;
    color: #27ae60;
    border: 1px solid #a5d6a7;
}

.movie-poster {
    width: 120px;
    height: 180px;
    object-fit: cover;
    border-radius: 10px;
    margin-right: 25px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.2);
    transition: transform 0.3s ease;
}

.movie-poster:hover {
    transform: scale(1.04);
}

.details {
    flex: 2.5;
}

.details p {
    margin: 8px 0;
    font-size: 16px;
    color: #4a4a4a;
```

```
    line-height: 1.6;
}

.details p strong {
    color: #2c3e50;
    font-weight: 600;
}

.details p span {
    font-weight: 600;
    color: #2c3e50;
    margin-right: 5px;
}

.actions {
    flex: 1;
    text-align: right;
}

.btn {
    padding: 12px 20px;
    font-size: 15px;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    color: white;
    transition: background-color 0.3s ease, transform 0.2s ease, box-
    display: inline-flex;
    align-items: center;
    margin-bottom: 10px;
}

.btn i {
    margin-right: 8px;
}
```

```
.btn-cancel {
    background-color: #e74c3c;
}

.btn-cancel:hover {
    background-color: #c0392b;
    transform: scale(1.05);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

.btn-transfer {
    background-color: #27ae60;
}

.btn-transfer:hover {
    background-color: #229954;
    transform: scale(1.05);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

.transfer-form {
    display: flex;
    gap: 12px;
    margin-top: 18px;
}

.transfer-form input {
    padding: 10px 16px;
    flex: 1;
    border: 1px solid #bdc3c7;
    border-radius: 10px;
    font-size: 15px;
}

.transfer-form button {
    padding: 10px 18px;
```

```
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    font-size: 15px;
    transition: background-color 0.3s ease, transform 0.2s ease, box-
    display: inline-flex;
    align-items: center;
}

.transfer-form button i {
    margin-right: 8px;
}

.transfer-form button:hover {
    background-color: #2980b9;
    transform: scale(1.05);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

.back-to-home {
    text-align: center;
    margin-top: 40px;
    animation: pulse 2s infinite ease-in-out;
}

@keyframes pulse {
    0% { transform: scale(1); }
    50% { transform: scale(1.04); }
    100% { transform: scale(1); }
}

.back-to-home a {
    padding: 16px 28px;
    background-color: #4285f4;
```

```
            color: white;
            text-decoration: none;
            border-radius: 12px;
            font-size: 17px;
            transition: background-color 0.3s ease, transform 0.2s ease, box-
            display: inline-flex;
            align-items: center;
        }

        .back-to-home a i {
            margin-right: 8px;
        }

        .back-to-home a:hover {
            background-color: #3367d6;
            transform: scale(1.04);
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div class="container">
        <h2><i class="fas fa-ticket-alt"></i> My Bookings</h2>
        {% if messages %}
            {% for message in messages %}
                <div class="message {{ message.tags }}">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}
        {% if bookings %}
            {% for booking in bookings %}
                <div class="booking-item">
                    <div class="movie-info">
                        <img src="{{ booking.show_time.movie.poster.url }}" a
                        <div class="details">
```

```
                        <p><strong>{{ booking.show_time.movie.title }}</s
                        <p><span>Theatre:</span> {{ booking.show_time.the
                        <p><span>Date:</span> {{ booking.show_time.date }
                        <p><span>Time:</span> {{ booking.show_time.time }
                        <p><span>Seats:</span> {% for seat in booking.boo
                        <p><span>Total:</span> {{ booking.total_price }}<
                        <p><span>Payment ID:</span> {{ booking.payment_id
                        <p><span>Status:</span> {{ booking.status }}</p>
                        {% if booking.selected_snacks.all %}
                            <p><span>Snacks:</span> {% for snack in booki
                        {% endif %}
                    </div>
                </div>
                <div class="actions">
                    {% if booking.status == "Booked" or booking.status ==
                        <form method="POST" action="{% url 'cancel_bookin
                            {% csrf_token %}
                            <button type="submit" class="btn btn-cancel">
                        </form>
                    {% else %}
                        <p style="color: #757575;">{{ booking.status }}</
                    {% endif %}
                    {% if booking.status == "Booked" %}
                        <form method="POST" class="transfer-form" action=
                            {% csrf_token %}
                            <input type="text" name="transfer_to" placeho
                            <button type="submit" class="btn btn-transfer
                        </form>
                    {% endif %}
                </div>
            </div>
        {% endfor %}
    {% else %}
        <p style="text-align: center; font-size: 19px; color: #757575;">N
    {% endif %}
    <div class="back-to-home">
```

```
            <a href="{% url 'home' %}"><i class="fas fa-home"></i> Back to Ho
        </div>
    </div>
    <script>
        function confirmCancellation(event, formElement) {
            event.preventDefault();
            if (confirm("Are you sure you want to cancel this booking? This a
                formElement.submit();
            }
        }
        function confirmTransfer(event, formElement) {
            event.preventDefault();
            if (confirm("Are you sure you want to transfer this ticket? This
                formElement.submit();
            }
        }
    </script>
</body>
</html>
```

Bibliography

[1] *First Lessons in LaTeX*, by Dr. V N Krishnachandran, Vidya Academy of Science & Technology, Thrissur - 680 501, 2011.

[2] *Django for Beginners: Build websites with Python and Django*, by William S. Vincent, 2022.

[3] *Full-Stack Web Development with Django and React*, by David S. Clinton and Riaz Ahmed, Packt Publishing, 2023.

[4] *Django RESTful Web Services*, by Gaston C. Hillar, Packt Publishing, 2018.

[5] *HTML and CSS: Design and Build Websites*, by Jon Duckett, Wiley, 2011.

[6] *JavaScript: The Good Parts*, by Douglas Crockford, O'Reilly Media, 2008.

[7] *Database System Concepts*, by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, McGraw-Hill, 2019.

[8] *Designing Data-Intensive Applications*, by Martin Kleppmann, O'Reilly Media, 2017.

[9] *Web Application Security: Exploitation and Countermeasures for JavaScript Apps*, by Andrew Hoffman, O'Reilly Media, 2020.

[10] *Law Relating to Women and Children*, by Mamta Rao, Eastern Book Company, 4th Edition, 2021.

Department of Computer Applications
Vidya Academy of Science & Technology
Thalakkottukara, Thrissur - 680 501
(http://www.vidyaacademy.ac.in)