

# **M-Book: Revolutionizing Movie Booking**

**A PROJECT REPORT**

**for**

**Project (KCA451)**

**Session (2023-24)**

**Submitted by**

**Kunal Sharma**

**2200290140082**

**Submitted in partial fulfillment of the  
Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATIONS**

**Under the Supervision of**

**Dr. Amit Kumar**

**Assistant Professor**



**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**(MAY 2024)**

# CERTIFICATE

Certified that , **Kunal Sharma (2200290140082)** has/ have carried out the project work having “**M-Book: Revolutionizing Movie Booking.**” (**Project-KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Kunal Sharma**  
**2200290140082**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

**Dr. Amit Kumar**

**Assistant Professor**

**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

**Dr. Arun Tripathi**

**Head**

**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**



**M-Book: Revolutionizing Movie Booking**  
**Kunal Sharma**

**ABSTRACT**

M-Book is a revolutionary movie booking application designed to streamline the movie booking process for users. With the increasing demand for online services, there is a need for a convenient and user-friendly movie booking platform. M-Book aims to address this need by providing a seamless and efficient platform for users to book movie tickets, check show timings, and access detailed movie information, all from the convenience of their smartphones.

The key features of M-Book include user authentication, movie information and booking, search functionality, ticket management, and user reviews and ratings. These features allow users to easily browse through a wide selection of movies, check show timings, select seats, and book tickets with just a few taps on their smartphones. Additionally, M-Book offers a user-friendly interface, real-time updates, and secure payment options, ensuring a hassle-free and enjoyable movie booking experience for users.

**Keywords:** Movie Booking, Online Booking, Mobile Application, User-Friendly Interface, Convenience.

## ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Amit Kumar** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Kunal Sharma**

# TABLE OF CONTENT

Section	Page No.
Certificate	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Abbreviations	vii
List of Tables	viii
List of Figures	ix
<b>1 Introduction</b>	1-4
1.1 Overview	1
1.1.1 Application Features	1
1.1.1.1 User Registration and Authentication	1
1.1.1.2 Movie Listings and Categories	2
1.1.2 Booking and Payment Integration	2
1.2 Application Architecture	3
1.2.1 Frontend Development	3
1.2.2 Backend Integration	3
1.3 Technology Stack	4
1.4 User Experience	4
1.5 Future Enhancements	4
<b>2 Feasibility Study</b>	5-6
2.1 Technical Feasibility	5
2.2 Operational Feasibility	5
2.3 Economic Feasibility	5
2.4 Legal Feasibility	6

2.5 Schedule Feasibility	6
<b>3 Design</b>	4-23
3.1 Software Engineering Paradigm	7
3.2 System Analysis	8
3.2.1 Introduction	8
3.2.2 Data-Flow Diagram (DFD)	8
3.2.2.1 Symbols Used in DFD	8
3.2.2.2 Context Level DFD	9
3.2.2.3 Show Maintenance Process	10
3.2.2.4 User Login & Management Process	10
3.2.3 E-R Diagram	13
3.2.3.1 E-R Diagram for "Online Movie Tickets Booking Application"	13
3.3 Structure of the Project	14
3.3.1 Number of Modules and Their Description	14
3.3.1.1 User Module	14
3.3.1.2 Administrator Module	14
3.4 Data Structure	14
3.5 Report Generation	18
3.6 Software Interface	18
3.6.1 Login Process	18
3.6.2 Transaction	19
3.6.3 Booking	19
3.7 Security Mechanisms	21
3.7.1 Application Security	21
3.7.2 Language Security	21
3.7.3 Database Security	21
3.7.4 Backup Software Security	21

3.8 Future Scope and Further Enhancement of the Project	22
3.8.1 Future Scope	22
3.8.2 Further Enhancement	22
3.9 Codes	23-37
Bibliography	38



# List of Abbreviations

1. M-Book - The name of the movie booking application.
2. UI - User Interface.
3. UX - User Experience.
4. API - Application Programming Interface.
5. SDK - Software Development Kit.
6. IDE - Integrated Development Environment.
7. SQL - Structured Query Language.
8. NoSQL - Not only SQL.
9. CRUD - Create, Read, Update, Delete.
10. REST - Representational State Transfer.
11. JSON - JavaScript Object Notation.
12. CSV - Comma-Separated Values.
13. DBMS - Database Management System.
14. ORM - Object-Relational Mapping.
15. HTTP - Hypertext Transfer Protocol.
16. HTTPS - Hypertext Transfer Protocol Secure.
17. UI/UX - User Interface/User Experience.
18. QA - Quality Assurance.
19. CI/CD - Continuous Integration/Continuous Deployment.
20. OS - Operating System.
21. URL - Uniform Resource Locator.
22. APIs - Application Programming Interfaces.
23. OOP - Object-Oriented Programming.
24. JWT - JSON Web Token.
25. HTML - HyperText Markup Language.
26. CSS - Cascading Style Sheets.
27. DNS - Domain Name System.
28. SSL - Secure Sockets Layer.
29. TLS - Transport Layer Security.
30. AJAX - Asynchronous JavaScript and XML

## LIST OF TABLES

<b>Table No.</b>	<b>Name of Table</b>	<b>Page</b>
2.1	Admin	9
2.2	User	15
5.1	Show	16
5.2	Show	16
5.3	Booking	17

## LIST OF FIGURES

Figure No.	Name of Figure	Page No.
1.	Waterfall Model of App	7
2	Symbols Used in DFD	9
3	Context Diagram of App	9
4	Maintenance Process	10
5	Level User Login & Management Process	12
6	E-R Diagram for M-Book	13
7.1	Database of M-Book	15
7.2	Database of M-Book Connectivity	16
8	Login page	18
9	Movie Choices	19
10	PLACE AND TICEKET SELECTION	20
11	PLACE AND SEAT SELECTION	20
12	BOOKING OF SHOW	20

# CHAPTER 1

## INTRODUCTION

### 1.1 OVERVIEW

The M-Book application is a comprehensive movie booking platform developed using Flutter and Dart. It provides users with the ability to book movie tickets, access detailed information about movies and showtimes, and search for movies by various categories. The application aims to enhance the convenience and enjoyment of movie-goers by offering a seamless and user-friendly interface. The primary goal is to streamline the movie booking process, making it more accessible and efficient for users. By leveraging modern technologies, M-Book ensures that users can quickly find and book tickets for their favorite movies, access up-to-date information, and enjoy a personalized experience. The application's design prioritizes ease of use, ensuring that even users who are not tech-savvy can navigate it effortlessly. Furthermore, the app's responsive design means that it performs optimally across various devices, from smartphones to tablets, providing a consistent and enjoyable user experience.

#### 1.1.1 Application Features

The M-Book application boasts a range of features designed to enhance the movie booking experience for users. These features ensure that users have access to a wide variety of movies, can easily book tickets, and have a pleasant overall experience. The features include a streamlined registration and authentication process, detailed movie listings and categorization, personalized recommendations, showtime and seat selection, and secure booking and payment integration. Each feature is meticulously designed to provide maximum convenience and value to the user. The integration of these features into a cohesive platform allows M-Book to stand out in the competitive market of movie booking applications. By continually updating and refining these features based on user feedback and technological advancements, M-Book ensures it remains at the forefront of innovation and user satisfaction.

##### 1.1.1.1 User Registration and Authentication

Users can register for an account and log in to the application using their credentials. This feature ensures that only authenticated users can book tickets and access

personalized content. The registration process is straightforward, requiring basic information such as name, email address, and password. Additionally, email verification is implemented to ensure security and user integrity. The authentication system employs secure protocols to protect user data and prevent unauthorized access. Multi-factor authentication can be added for an extra layer of security. The user interface for registration and login is designed to be intuitive, guiding users through the process with clear instructions and feedback. This feature not only enhances security but also enables personalized services, such as saving favorite movies, viewing booking history, and receiving customized recommendations. By prioritizing user security and ease of use, M-Book aims to provide a reliable and personalized movie booking experience.

#### **1.1.1.2 Movie Listings and Categories**

The application offers a comprehensive listing of movies, categorized by genre, release date, and popularity. Users can easily browse and select movies based on their preferences. Each movie listing includes detailed information such as synopsis, cast, director, runtime, and user ratings, helping users make informed decisions. The categorization feature allows users to filter and search for movies, making it easier to find desired content quickly. Movie listings are regularly updated to include the latest releases, ensuring that users have access to current information. The detailed movie pages also include trailers, reviews, and related movies, providing users with a holistic view of their movie options. This comprehensive approach to movie listings and categorization not only enhances the user experience but also increases user engagement and satisfaction. By providing detailed and organized movie information, M-Book empowers users to make well-informed choices about their movie-going experience.

#### **1.1.2 Booking and Payment Integration**

The booking process in M-Book is streamlined and user-friendly, with integrated payment gateways to facilitate secure transactions. Users can choose their preferred payment method, including credit/debit cards, digital wallets, and net banking, ensuring flexibility and convenience in the booking process. The payment integration is designed to be secure, supporting various payment options to accommodate user preferences. Advanced encryption methods are used to protect user data during transactions, ensuring a safe and secure payment environment. The application also supports the storage of payment information for faster future transactions, with user consent and security protocols in place. Users receive instant confirmations and digital receipts for their bookings, which can be accessed anytime from their account. By providing a smooth and secure booking and payment process, M-Book aims to enhance user trust and satisfaction, making the movie ticket purchasing experience as seamless as possible.

## **1.2 Application Architecture**

The M-Book application is built on a robust architecture that ensures scalability, performance, and ease of maintenance. The architecture is designed to handle a large number of users simultaneously, providing a smooth and responsive experience. The use of a microservices architecture allows for modular development, where each component of the application can be developed, tested, and deployed independently. This modular approach enhances the scalability and maintainability of the application. The architecture also includes load balancing to distribute traffic evenly across servers, ensuring optimal performance even during peak usage times. Regular monitoring and performance testing are conducted to identify and address any bottlenecks or issues. By employing a robust and scalable architecture, M-Book ensures that it can accommodate growth and provide a consistently high-quality user experience.

### **1.2.1 Frontend Development**

The frontend of the application is developed using Flutter, which allows for a responsive and visually appealing user interface. Flutter's widget-based architecture enables the creation of a highly customizable and interactive user experience, ensuring consistency across different devices and screen sizes. The use of Flutter also speeds up the development process due to its hot-reload feature, which allows for instant updates and debugging. The user interface is designed with a focus on usability, incorporating intuitive navigation, clear visuals, and engaging animations. Accessibility features are also integrated to ensure that the application is usable by individuals with disabilities. By leveraging the capabilities of Flutter, M-Book provides a high-quality and visually appealing frontend that enhances the overall user experience. The frontend development process involves continuous testing and user feedback to refine and improve the interface, ensuring it meets the needs and expectations of users.

### **1.2.2 Backend Integration**

The backend services, including user authentication, movie data management, and payment processing, are seamlessly integrated using Dart. The backend is designed to be secure and efficient, with robust APIs that handle data transactions and communications between the server and the client. The use of a RESTful API architecture ensures that the backend services are scalable and can be easily extended with new features. Data storage is managed using a relational database, which provides reliable and efficient data management. The backend also includes features such as data caching and load balancing to enhance performance and reliability. Security is a top priority, with measures such as encryption, secure communication protocols, and regular security audits in place to protect user data. By implementing a secure and efficient backend, M-Book ensures that it can provide a reliable and high-performance service to users.

### **1.3 Technology Stack**

The choice of technology in developing M-Book plays a crucial role in delivering a high-quality application. Flutter and Dart were chosen for their modern features, performance benefits, and strong community support, enabling rapid development and deployment of the application. Flutter provides a rich set of pre-built widgets and tools that streamline the development process, while Dart offers a robust programming language that enhances performance and maintainability. Additional tools and frameworks used include Firebase for backend services, Stripe for payment processing, and various APIs for movie data. Firebase offers real-time database capabilities, authentication services, and cloud storage, providing a comprehensive backend solution. Stripe is integrated for secure and efficient payment processing, supporting a variety of payment methods. The use of these modern technologies ensures that M-Book is built on a solid and reliable foundation, capable of delivering a high-quality user experience.

### **1.4 User Experience**

M-Book is designed with a focus on user experience, ensuring that the interface is intuitive and easy to navigate. User feedback is continuously gathered and analyzed to make improvements, ensuring that the application meets user needs and expectations effectively. The design principles emphasize simplicity, accessibility, and responsiveness. The user interface is designed to be visually appealing, with clear and concise navigation elements. Interactive elements such as buttons and icons are designed to be user-friendly and responsive. Accessibility features are integrated to ensure that the application is usable by individuals with disabilities. Regular updates and improvements are made based on user feedback and usability testing, ensuring that the application remains relevant and user-friendly. By prioritizing user experience, M-Book aims to provide a seamless and enjoyable movie booking experience for all users.

### **1.5 Future Enhancements**

Plans for future enhancements include adding support for multiple languages, expanding movie listings, and incorporating social features for user interaction. Additional features like movie reviews, recommendations based on viewing history, and loyalty programs are also considered to enhance user engagement. Future updates aim to keep the application relevant and competitive in the market. Language support will make the application accessible to a global audience, while expanded movie listings will provide users with a wider range of options. Social features will allow users to interact with each other, share reviews and ratings, and engage more deeply with the application. Loyalty programs will reward frequent users with discounts and special offers,

## CHAPTER 2

### FEASIBILITY STUDY

#### 2.1 Technical Feasibility

An analysis of the technical requirements and the feasibility of developing the M-Book application using Flutter and Dart. This includes evaluating the development tools, libraries, and frameworks needed to build and maintain the application efficiently. The technical feasibility study also assesses the scalability and performance of the application, ensuring that it can handle a large number of users simultaneously. The choice of Flutter and Dart is based on their modern features, performance benefits, and strong community support. Additional tools and frameworks such as Firebase and Stripe are evaluated for their compatibility and integration capabilities. The technical feasibility study also considers the maintainability of the application, ensuring that it can be easily updated and extended with new features. By conducting a thorough technical feasibility study, M-Book ensures that it is built on a solid and reliable foundation, capable of delivering a high-quality user experience.

#### 2.2 Operational Feasibility

Evaluating the operational aspects, including user adoption, ease of use, and maintenance requirements of the M-Book application. This involves assessing the readiness of the operational environment and the ability of the organization to support the application post-deployment. The operational feasibility study also considers the availability of resources such as technical support, infrastructure, and personnel. User adoption is assessed through market research and user surveys, ensuring that the application meets the needs and expectations of its target audience. The ease of use is evaluated through usability testing and user feedback, ensuring that the application is intuitive and user-friendly. Maintenance requirements are assessed, ensuring that the application can be easily updated and maintained with minimal disruption. By conducting a thorough operational feasibility study, M-Book ensures that it is ready for deployment and can provide a high-quality user experience.

#### 2.3 Economic Feasibility

A cost-benefit analysis to determine the economic viability of the project, including development costs and potential revenue streams. This section explores various revenue models such as ticket booking fees, advertisements, and partnerships with movie theaters.



The economic feasibility study also considers the potential for growth and expansion, ensuring that the project is financially sustainable in the long term. Development costs include expenses for software development, infrastructure, marketing, and maintenance. Potential revenue streams are evaluated based on market research and industry trends, ensuring that the project can generate sufficient revenue to cover costs and generate profit. Additional revenue can be generated through premium features, in-app purchases, and exclusive content. The economic feasibility study provides a detailed analysis of the costs and benefits of the project, ensuring that it is financially viable and sustainable. By conducting a thorough economic feasibility study, M-Book ensures that it can achieve its financial goals and provide a high-quality user experience.

## **2.4 Legal Feasibility**

An assessment of any legal considerations, such as compliance with data protection regulations and licensing agreements for movie content. This ensures that the application adheres to relevant laws and regulations, protecting both the users and the developers from legal issues. The legal feasibility study also considers the intellectual property rights and copyright issues related to movie content. M-Book must comply with data protection regulations, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). This includes implementing measures to protect user data and ensuring transparency in data collection and usage. Licensing agreements with movie theaters and distributors are necessary to provide access to movie content. These agreements must be negotiated and adhered to, ensuring that M-Book has the legal right to display and sell tickets for the listed movies. By conducting a thorough legal feasibility study, M-Book ensures that it is compliant with all relevant laws and regulations, protecting both the users and the developers from legal issues.

## **2.5 Schedule Feasibility**

A timeline for the development and deployment of the M-Book application, ensuring that the project milestones are achievable within the specified timeframe. This includes planning for each phase of the project, from initial development to testing and final deployment. The schedule feasibility study also considers the availability of resources and the potential risks that may impact the project timeline. The project timeline should include key milestones such as initial planning, development, testing, and deployment. Each phase should have specific objectives and deliverables, ensuring that the project stays on track and within budget. Identifying potential risks and developing strategies to mitigate them is essential for the successful completion of the project. This includes risks related to technical challenges, resource constraints, and external factors such as regulatory changes. Allocating resources effectively is crucial for meeting the project timeline. This includes assigning tasks to team members based on their skills and expertise, and ensuring that adequate resources are available for each phase of the project.

## CHAPTER 3

### DESIGN

#### 3.1 Software Engineering Paradigm

Software Engineering combines process, methods, and tools to develop high-quality software in a systematic manner. The "Waterfall Model" has been chosen for developing the M-Book application due to its structured and sequential approach. Each phase of the Waterfall Model has well-defined starting and ending points, with clearly identifiable deliverables to the next phase, ensuring a methodical development process.

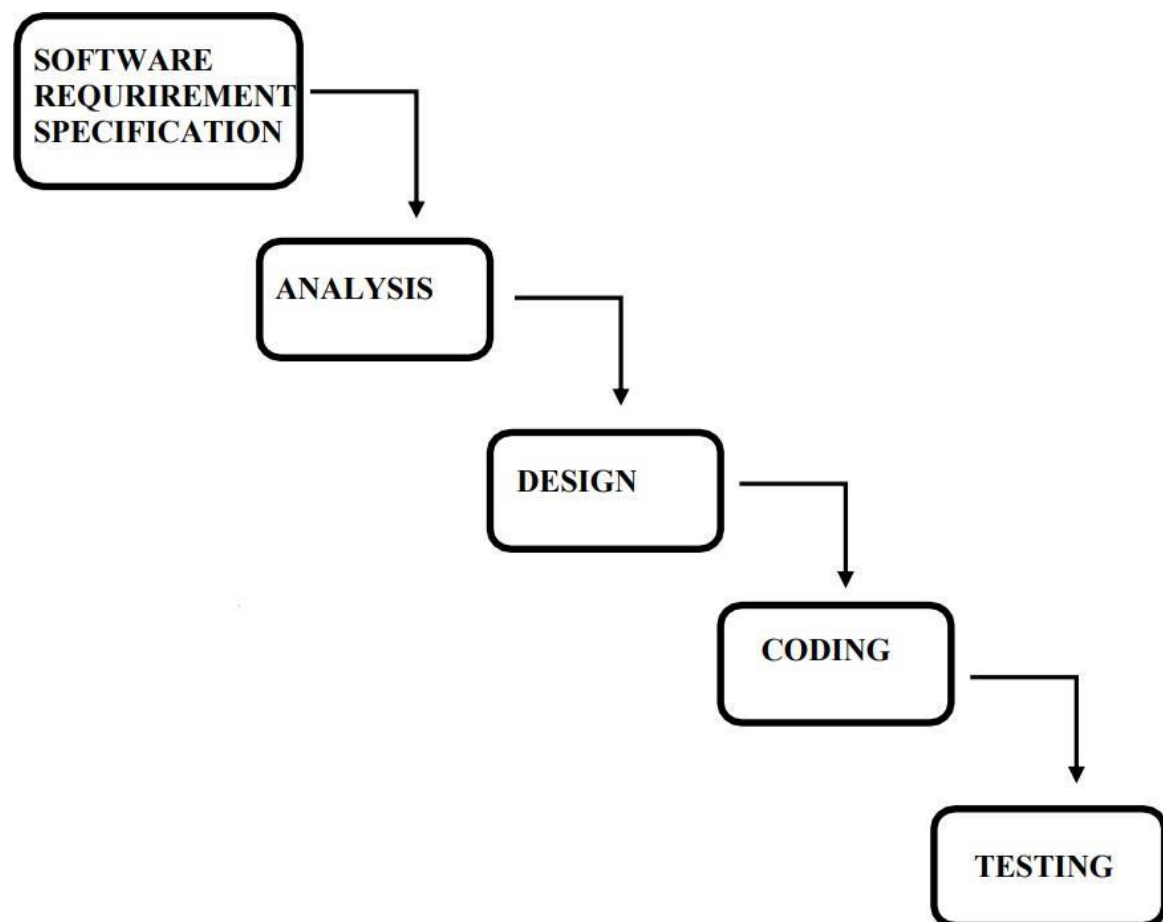


fig 1: Waterfall Model of App

## **3.2 System Analysis**

### **3.2.1 Introduction**

System analysis involves surveying and planning the system and project, studying and analyzing existing business and information systems, and defining business requirements and priorities for a new or improved system. This phase provides a comprehensive understanding of the project requirements and lays the foundation for designing an efficient and effective system that meets user needs.

### **3.2.2 Data-Flow Diagram (DFD)**

A Data-Flow Diagram (DFD) graphically represents the flow of data through an information system. It visualizes data processing by showing how data moves from an external source or internal data store to an internal data store or external sink via an internal process. DFDs illustrate what data will be input and output, where it will come from and go to, and where it will be stored.

#### **3.2.2.1 Symbols Used in DFD**

In a Data Flow Diagram (DFD), several key symbols are used to represent different components of the system. External entities, also known as terminators, represent sources or destinations of data outside the system. These are typically depicted as squares or rectangles. Processes, which denote functions or activities that transform data, are usually represented by circles or rounded rectangles. Data stores act as repositories for data storage and are depicted as open-ended rectangles or parallel lines. Data flows, represented by arrows, indicate the movement of data between entities, processes, and data stores. These arrows are crucial for illustrating how data travels through the system, from input to processing to output. Collectively, these symbols provide a visual representation of the system's data flow and processing, making it easier to understand how information is handled within the system. By clearly delineating these elements, a DFD helps stakeholders and developers visualize the interactions and data dependencies within the system, facilitating better design and communication.


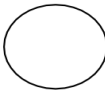


Symbol	Name	Description
	Data	Movement of data in the system.
	Flow	
	Process	Transform of incoming data flow(s) to outgoing flow(s).
	External Entity	Sources of destinations outside the specified system boundary.
	Data Store	Data repositories for data those are not moving.

fig 2: Symbols Used in DFD

### 3.2.2.2 Context Level DFD

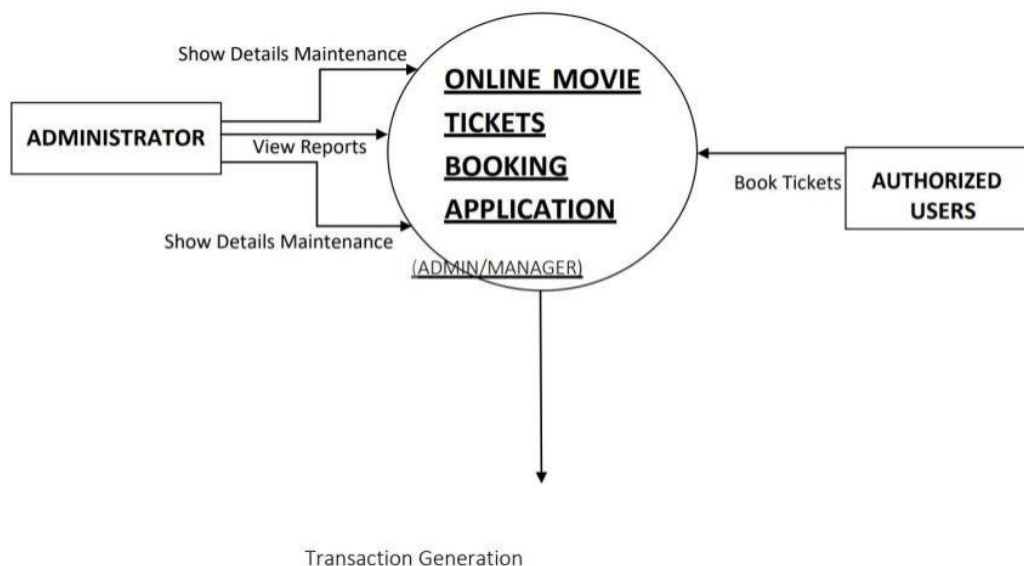


fig 3: Context Diagram of App

The context level DFD provides an overview of the entire system, showing the system as a single process with its interactions with external entities. This high-level diagram sets the stage for more detailed DFDs, illustrating the major data flows and interactions within the system.

### 3.2.2.3 Show Maintenance Process

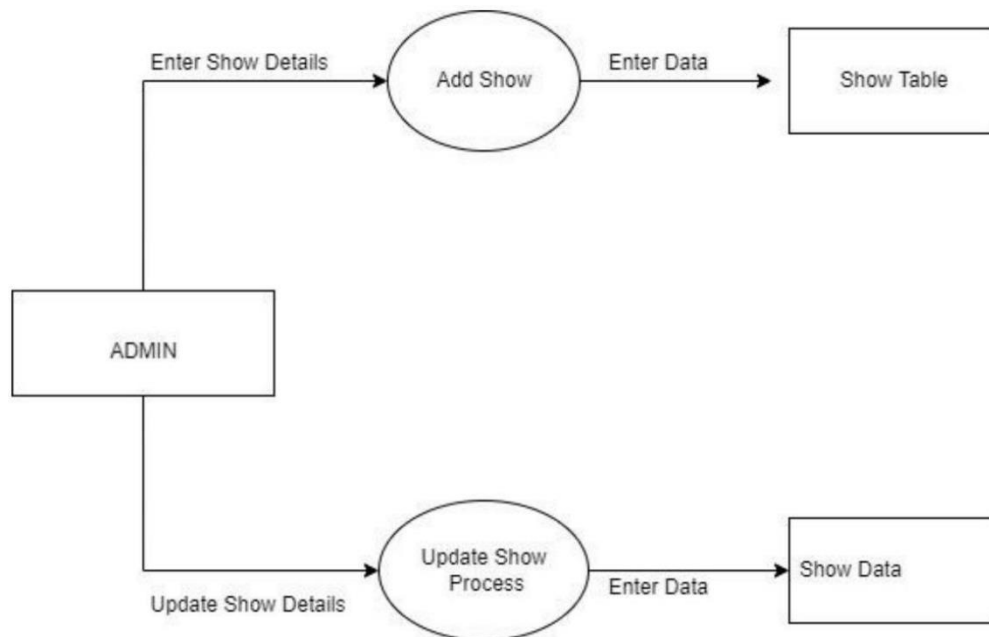


fig 4: Maintenance Process

The Show Maintenance Process DFD details the flow of data related to maintaining show information. It illustrates how administrators input show details, how this data flows through the system, and how it is stored and accessed by users. This process ensures that the show information is up-to-date and accurate.

### 3.2.2.4 User Login & Management Process

The User Login & Management Process in the M-Book application is designed to ensure secure and efficient access to the platform's features while maintaining user data integrity and security. This process involves several key steps:

1. **User Registration:** New users must first register an account. This involves providing basic information such as name, email address, and password. Additional security measures, such as CAPTCHA verification, are used to prevent automated registrations. Upon submission, an email verification link is sent to the user to confirm their email address. This step is crucial for validating user identity and preventing fraudulent accounts.

2. **Login:** Registered users can log in using their email and password. The application uses secure authentication protocols to protect user credentials. If the login information is correct, the user is granted access to the application. For enhanced security, M-Book supports multi-factor authentication (MFA), where users must provide an additional verification code sent to their registered email or phone number.
3. **Password Management:** Users have the ability to reset their passwords if they forget them. This process involves clicking a "Forgot Password" link, entering their registered email address, and receiving a password reset link via email. The link directs the user to a secure page where they can set a new password. Strong password guidelines are enforced to ensure user accounts remain secure.
4. **Profile Management:** Once logged in, users can manage their profiles, including updating personal information, changing passwords, and setting preferences. This section also allows users to view their booking history, saved movies, and payment methods. All profile changes are securely processed and stored to ensure data integrity.
5. **Session Management:** User sessions are managed to maintain security. Sessions are automatically logged out after a period of inactivity to prevent unauthorized access. Users can also manually log out from their accounts. Session tokens are used to maintain user state across different pages of the application.
6. **Role-Based Access Control:** The application supports different user roles, such as regular users and administrators. Administrators have access to additional features, such as user management and system settings. Role-based access control (RBAC) ensures that users can only access the functionalities relevant to their role, enhancing security and data protection.
7. **Audit and Monitoring:** All login attempts and profile changes are logged and monitored for security purposes. This audit trail helps in identifying suspicious activities and taking timely action to protect user accounts. Regular security audits are conducted to ensure the system's robustness against potential threats.

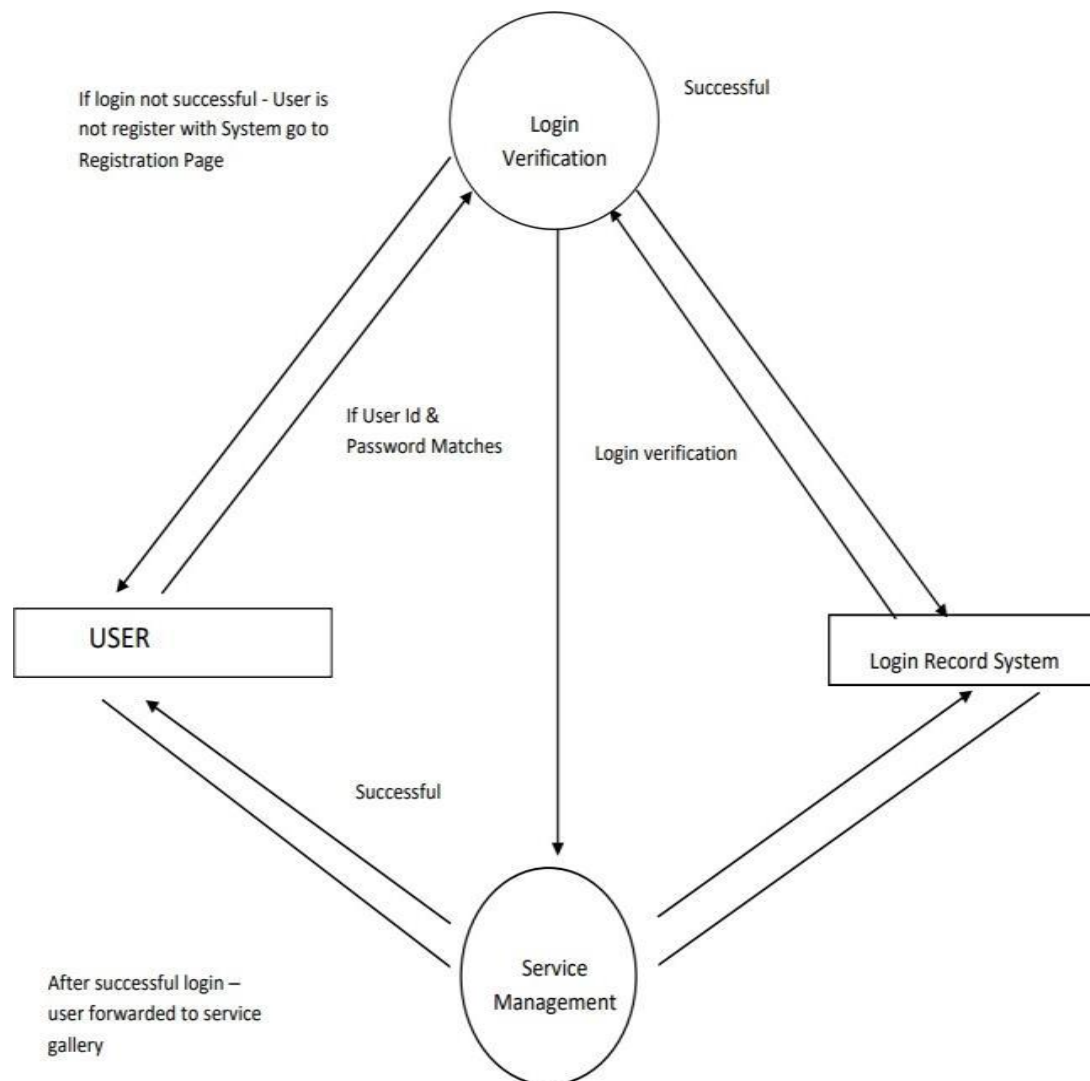


fig 5: Level User Login & Management Process

The User Login & Management Process DFD outlines the data flow involved in user authentication and management. It shows how user credentials are processed for authentication, how user data is managed, and how authenticated users interact with the system. This process is crucial for ensuring secure access and personalized user experiences.

### 3.2.3 E-R Diagram

An Entity-Relationship (E-R) diagram visually represents the entities within the system and their relationships. It helps in understanding the system's data requirements and designing the database structure. The E-R diagram for the M-Book application identifies key entities such as users, movies, shows, and bookings, and illustrates the relationships between these entities.

#### 3.2.3.1 E-R Diagram for “ONLINE MOVIE TICKETS BOOKING APPLICATION”

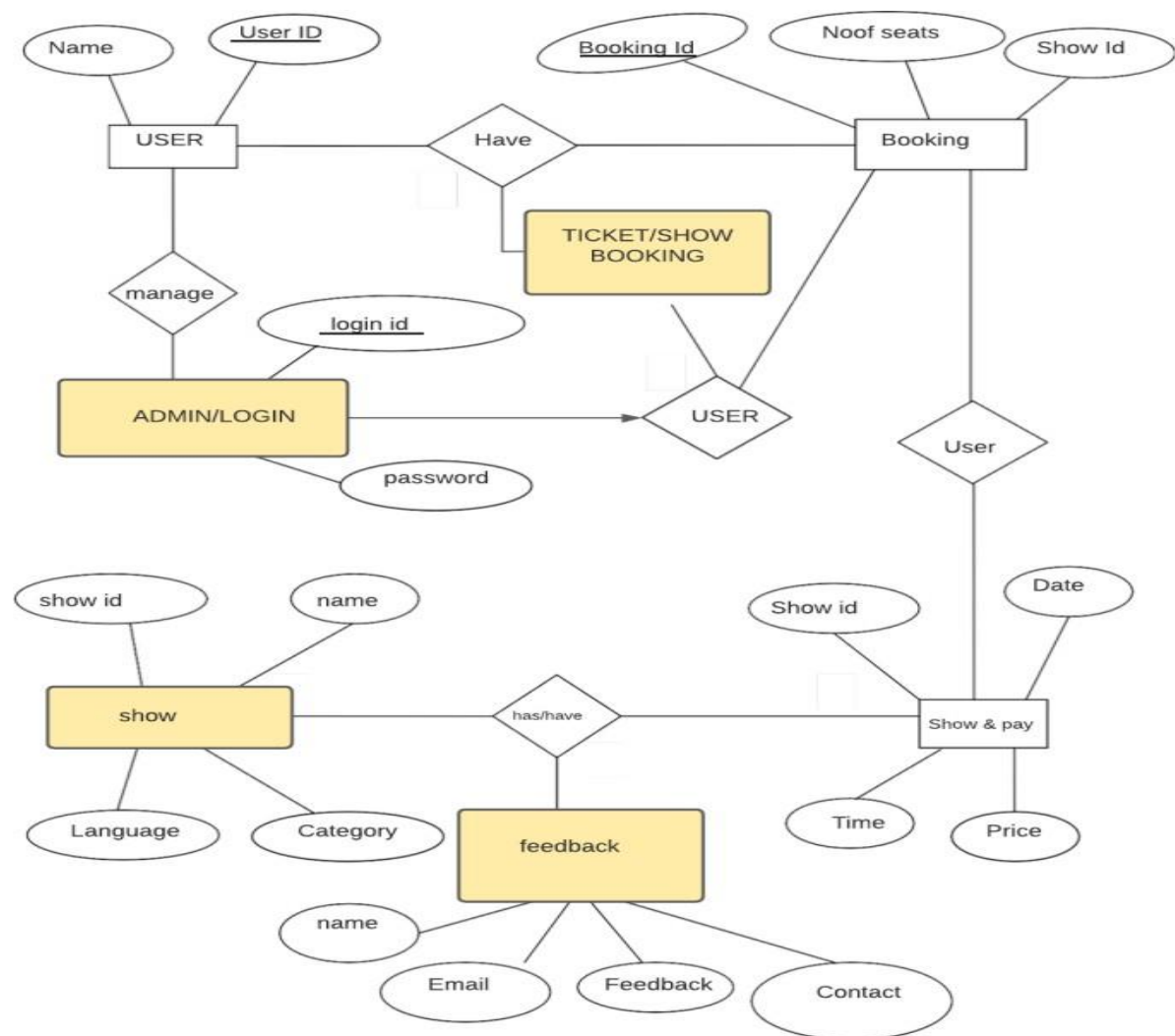


fig 6: E-R Diagram for M-Book



The E-R diagram for the M-Book application provides a detailed visual representation of the system's entities and their relationships. It defines the attributes of each entity and the connections between them, ensuring a clear and organized database structure that supports efficient data retrieval and management.

### **3.3 Structure of the Project**

#### **3.3.1 Number of Modules and Their Description**

The M-Book application consists of two main modules: the User Module and the Administrator Module. Each module is designed to handle specific functionalities within the application, ensuring that both end-users and administrators have the tools they need to perform their respective tasks efficiently.

##### **3.3.1.1 User Module**

The User Module manages functionalities related to end-users. It includes the login process, where users enter their credentials, which are then authenticated by the system. Upon successful authentication, users are redirected to their personalized homepage. The module also handles transactions, where users input show details, check ticket availability, and proceed to the booking page if tickets are available. For the booking process, users enter payment details, which are validated by the system, and a booking ID is generated upon successful payment.

##### **3.3.1.2 Administrator Module**

The Administrator Module handles administrative functions necessary for maintaining the application. Administrators log in using their credentials, which are authenticated by the system, granting them access to their dashboard. The module also includes the show details maintenance process, where administrators input new show details, which are stored in the Firebase database. Upon successful data entry, administrators are redirected to a confirmation page, ensuring that show information is accurately maintained and up-to-date.

### **3.4 Data Structure**

The data structure of the M-Book application is designed to support its functionality efficiently. The database tables are structured to store user information, movie details, show schedules, and booking information. Each table is designed to ensure data integrity and optimize data retrieval, supporting the smooth operation of the application. Detailed diagrams and schema definitions are included to illustrate the relationships between different data entities.

The below table which will be helpful on the functionality of the proper project.

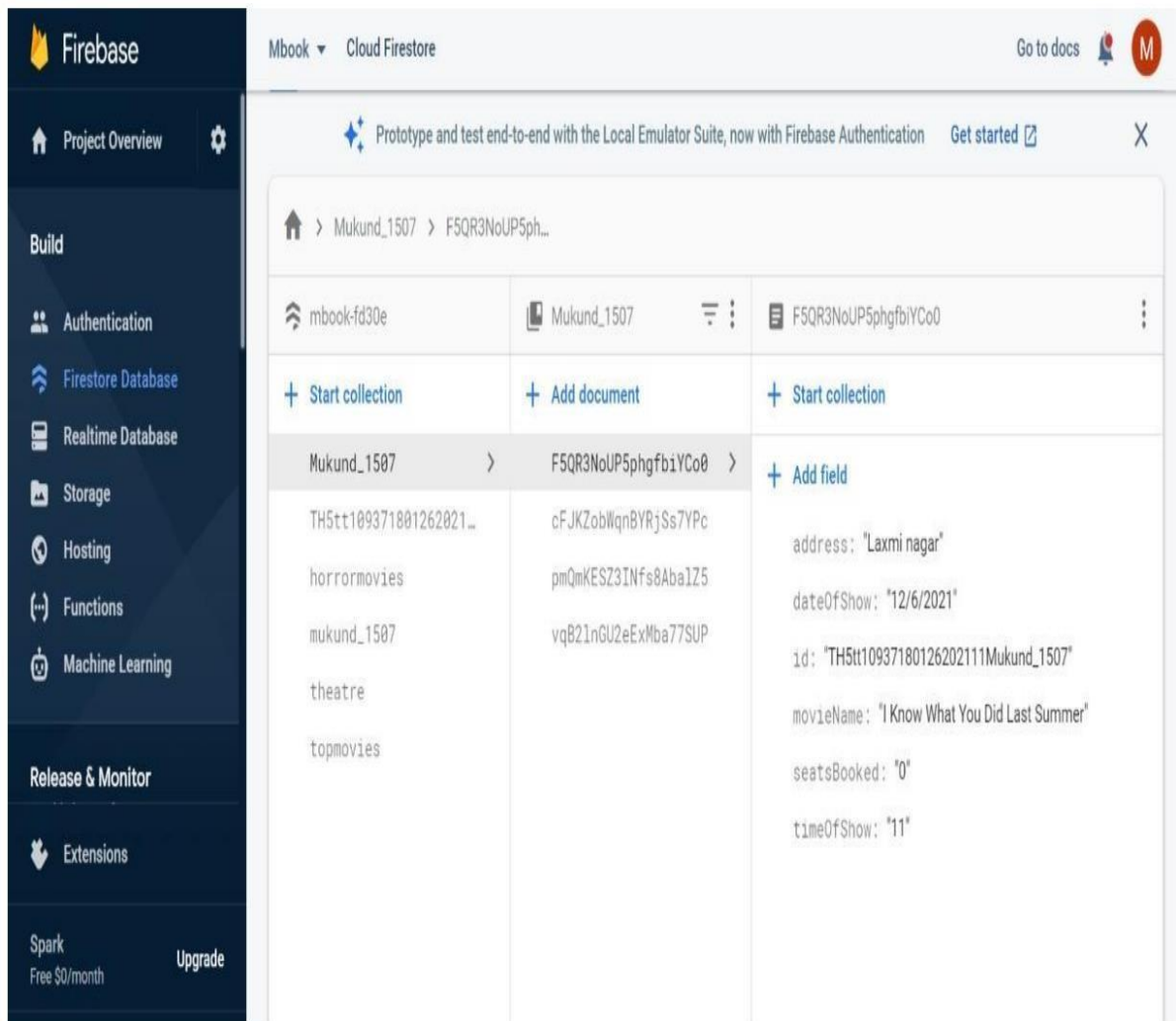


fig 7: Database of M-Book

Table Name: Admin

Field	Type	Null	Description
AdminEmailId	varchar(50)	No	UserEmailId
AdminPassword	varchar(50)	No	UserPassword

Table Name: User

Field	Type	Null	Description
UserId	varchar(50)	No	UserEId
UserPassword	varchar(50)	No	UserPassword

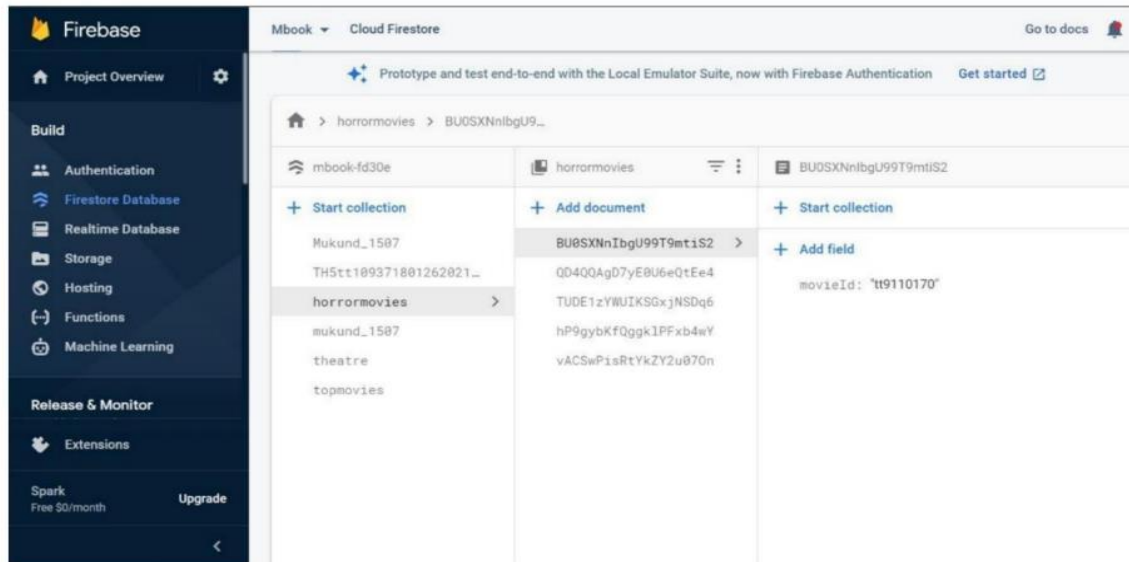


Table Name: Show

Field	Type	Null	Description
Show_Id	Int	No	Show_Id
Show_Name	varchar(50)	No	Show_Name
Category	varchar(50)	No	Category

fig 7.1: Database of M-Book

Table Name: Show

Field	Type	Null	Key	Description
Show_Id	Int	No		Show_Id
Show_Id	Int	No	FK	Show_Id
Audi_No	Int	No		Audi_No
Show_Date	Datetime	No		Show_Date
Show_Timings	Datetime	No		Show_Timings
Ticket_Price	Money	No		Ticket_Price

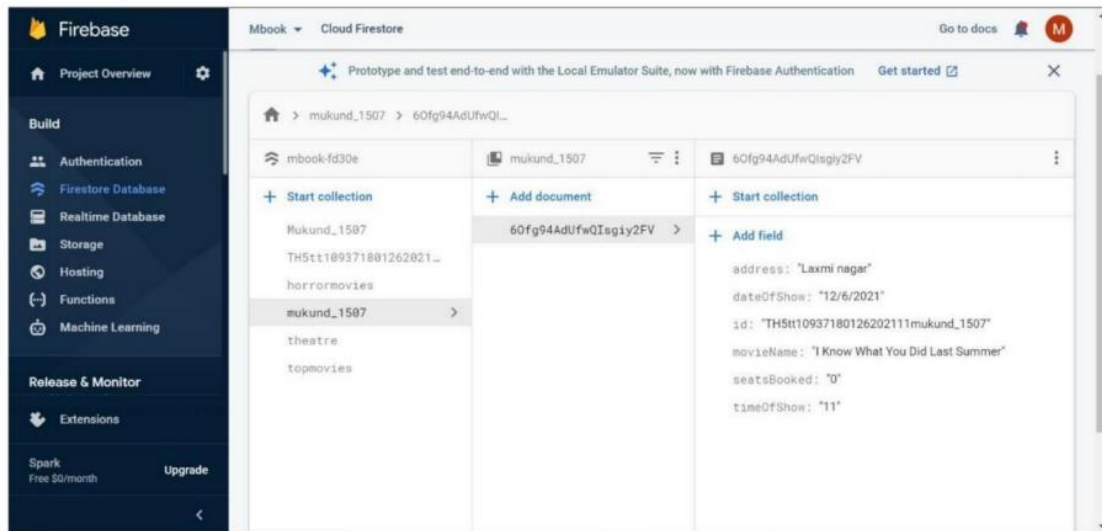


Table Name: Booking

Field	Type	Null	Description
Booking_Id	Int	No	sequence
Show_Id	Int	No	Show_Id
Audi_No	Int	No	Audi_No
Seat_No	Int	No	Seat_No

fig 7.2: Database of M-Book

### 3.5 Report Generation

The M-Book application generates several reports to provide insights into the system's operations and user activities. These reports include:

- List of Registered Users: This report displays the total number of users registered on the platform, providing insights into user growth and engagement.
- List of Shows: This report lists all the shows available in the database, helping administrators manage and update show schedules.
- List of Shows by Cinema Hall: This report organizes shows by cinema hall, offering a detailed view of show distribution and scheduling across different venues. These reports help in monitoring system performance and making informed decisions.

### 3.6 Software Interface

The software interface of the M-Book application is designed to be intuitive and user-friendly. The interface includes screens for user login, show browsing, seat selection, and payment processing. Each screen is designed to guide the user through the process smoothly, with clear instructions and feedback at each step. The design emphasizes ease of use and accessibility, ensuring a positive user experience.

#### 3.6.1 Login Process:

The process is:

- Take the user ID and Password.
- Check For the authentication.
- Produce the desire web page

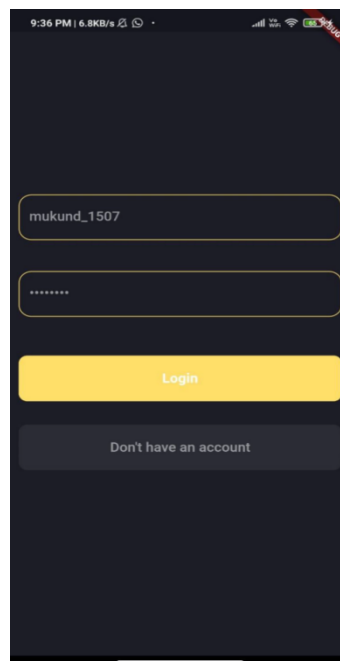


fig 8: Login page

### 3.6.2 Transaction:

- Enter the necessary details of show.
- Check for the availability of the tickets.
- Produce the desire page.

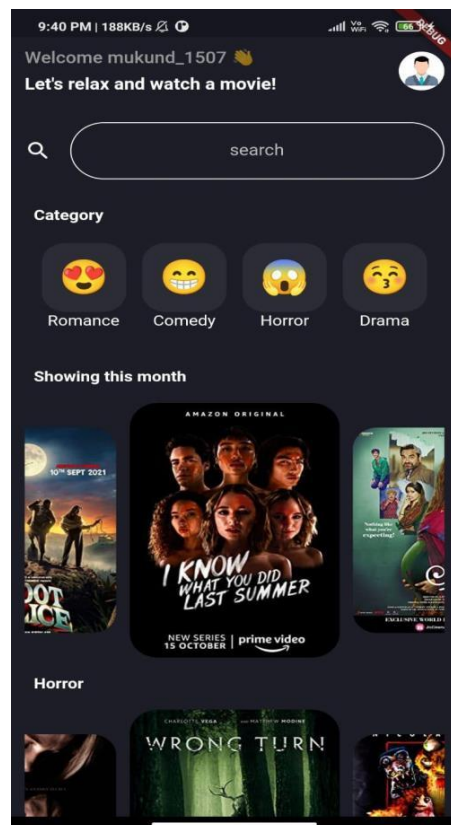


fig 9: Movie Choices

### 3.6.3 Booking:

The process is:

- Enter the credit/debit card number.
- Check for the validity.
- Give the Booking ID

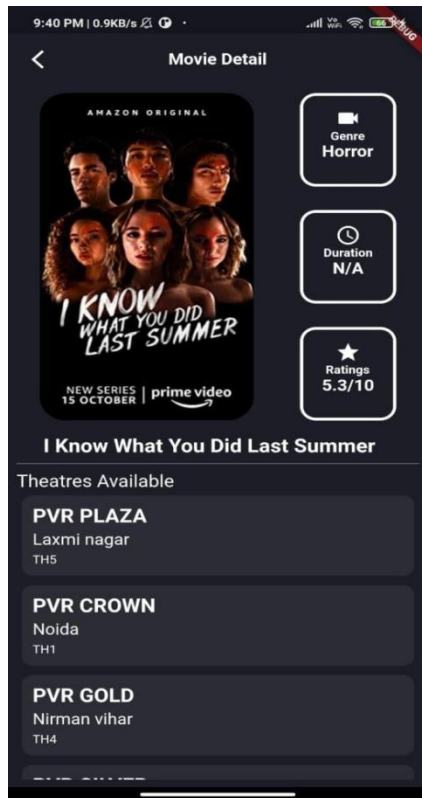


fig 10: PLACE AND TICEKET SELECTION

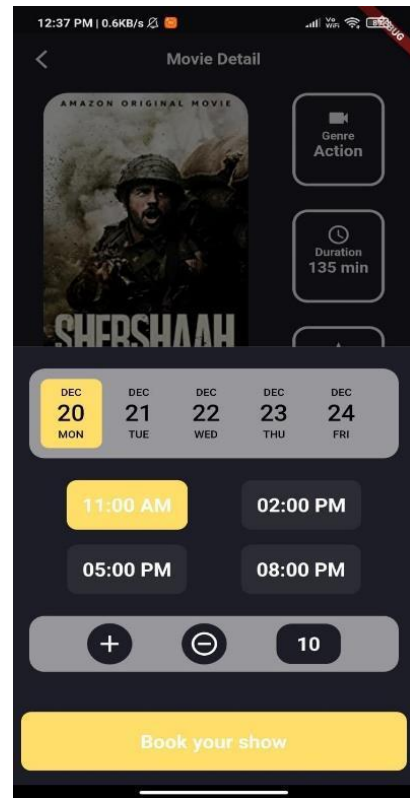


fig 11: PLACE AND SEAT SELECTION

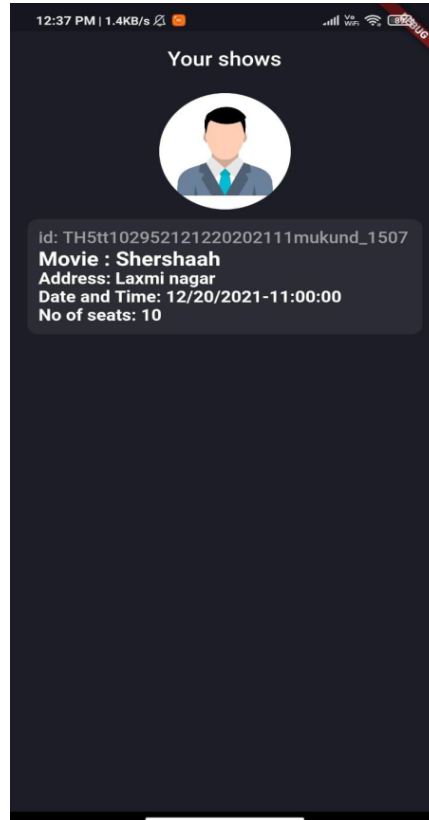


fig 12: BOOKING OF SHOW

### **3.7 Security Mechanisms**

Security is a critical aspect of the M-Book application, ensuring the system's protection from damage, errors, and unauthorized access.

#### **3.7.1 Application Security**

To prevent unauthorized access, a secure login form is implemented. Users must provide valid usernames and passwords to enter the application, ensuring that only authorized individuals can access the system. This login mechanism helps in safeguarding user data and maintaining the integrity of the application by restricting access to authenticated users only.

#### **3.7.2 Language Security**

The software is developed in Dart, which compiles source code to byte code. This machine-independent byte code is difficult to reverse-engineer, enhancing the security of the application's source code and preventing unauthorized modifications. This layer of security ensures that the underlying code remains protected from potential threats, making it harder for malicious users to manipulate the system.

#### **3.7.3 Database Security**

Firebase is used as the backend database, accessible only with administrator permission. This ensures that sensitive data is protected, and only authorized users can interact with the database, maintaining data integrity and security. Firebase provides robust security features, such as authentication and real-time data monitoring, ensuring that the application's data is safe from unauthorized access and breaches.

#### **3.7.4 Backup Software Security**

To protect against data loss due to hardware failures, a backup security facility is implemented. This ensures that in the event of a hard disk crash, data can be recovered, safeguarding important customer information and maintaining system reliability. Regular backups and data recovery plans are crucial for minimizing downtime and ensuring business continuity, providing users with a reliable and resilient system.



### **3.8 Future Scope and Further Enhancement of the Project**

The M-Book application is designed with future enhancements in mind, providing opportunities for additional features and improvements.

#### **3.8.1 Future Scope**

The proposed software offers significant benefits, such as the convenience of booking cinema tickets online, saving users time. However, the project is not complete and will undergo further development to enhance its capabilities and user experience. Future scope includes expanding payment options, integrating more sophisticated features, and enhancing the user interface to improve usability and accessibility.

#### **3.8.2 Further Enhancement**

Future enhancements include enabling payments via debit cards and integrating bank servers to facilitate direct bank account access, which is currently not implemented due to cost constraints. Additionally, improvements in database security and the ability for users to watch show demos are planned. These enhancements will further improve the functionality and user experience of the M-Book application. By continuously updating and expanding the application's features, the development team aims to meet evolving user needs and industry standards, ensuring that M-Book remains a leading solution for online movie ticket bookings.

### 3.9 Codes

#### 3.9.1 Database(Firebase)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:mbook/models/theatre.dart';

class DbServices {
  final FirebaseFirestore _firebaseFirestore = FirebaseFirestore.instance;
  Future<List<Theatre>> getTheatres() async {
    List<Theatre> listOfTheatres = [];
    final theatres = await _firebaseFirestore.collection('theatre').get();
    final th = theatres.docs.asMap();
    for (int i = 0; i < theatres.docs.length; i++) {
      Theatre temp = Theatre(
        id: th[i]['id'], name: th[i]['name'], address: th[i]['address']);
      listOfTheatres.add(temp);
    }
    return listOfTheatres;
  }
}
```

### 3.9.2 Movie model

```
import 'networking.dart';

const String apiKey = 'd79cb85d';

class MovieModel {
  Future<dynamic> getMovieDetails(String movieName, String movieYear) async {
    NetworkHelper networkHelper = NetworkHelper(
      'https://www.omdbapi.com/?t=$movieName&y=$movieYear&apikey=$apiKey');
    var movieData = await networkHelper.getData();
    return movieData;
  }
  Future<dynamic> getMovieDetailsByName(String movieName) async {
    int movieYear = DateTime.now().year;
    NetworkHelper networkHelper = NetworkHelper(
      'https://www.omdbapi.com/?t=$movieName&y=$movieYear&apikey=$apiKey');
    var movieData = await networkHelper.getData();
    return movieData;
  }
}
```

### 3.9.3 Networking

```
import 'package:http/http.dart' as http;
import 'dart:convert';

class NetworkHelper{
  NetworkHelper(this.url);
  final String url;

  Future getData () async {
    http.Response response = await http.get(Uri.parse(url));
    if (response.statusCode == 200) {
      String data = response.body;
      var decodeData = jsonDecode(data);
      return decodeData;
    } else {
      print(response.statusCode);
    }
  }
}
```

### 3.9.4 Main file

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'screens/welcomeScreen.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        scaffoldBackgroundColor: const Color(0xff1d1d28),
        primaryColor: const Color(0xffffde6a),
        secondaryHeaderColor: const Color(0xff2b2c34),
      ),
      home: const WelcomeScreen(),
    );
  }
}
```

### 3.9.5 Category

```
import 'package:flutter/material.dart';
import 'package:mbook/constants.dart';
import '../reusables/reusableemojibox.dart';

class Category extends StatelessWidget {
  const Category({ Key? key }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            'Category',
            style: kTextStyle3.copyWith(
              fontSize: 15,
            ),),
          Const SizedBox(
            Height: 20,
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: const [
              ReusableEmojiBox(emoji: '💕', text: 'Romance'),
              ReusableEmojiBox(emoji: '😄', text: 'Comedy'),
              ReusableEmojiBox(
                emoji: '😱',
                text: 'Horror',
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

```

        ReusableEmojiBox(
          emoji: '☐',
          text: 'Drama',
        ),
      ],
    ),
    const SizedBox(
      height: 40,
    ),
    Text(
      'Showing this month',
      style: kTextStyle3.copyWith(
        fontSize: 15,
      ),
    ),
  ],
),
);
}
}

```

### 3.9.6 loginscreen

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import '../constants.dart';
import '../reusables/roundedtextbutton.dart';
import '../screens/registrationscreen.dart';
import 'homescreen.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  late String userName;
  late String password;
  FirebaseAuth auth = FirebaseAuth.instance;
  late User user;

  @override
  Widget build(BuildContext context) {
    OutlineInputBorder roundBorder = OutlineInputBorder(
      borderRadius: BorderRadius.circular(15),
      borderSide: BorderSide(
        color: Theme.of(context).primaryColor,
      ),
    ),
```



```

);
return Scaffold(
  resizeToAvoidBottomInset: false,
  body: Padding(
    padding: const EdgeInsets.all(10.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        TextField(
          onChanged: (value) {
            userName = value;
          },
          style: kTextStyle2,
          decoration: InputDecoration(
            hintText: 'Username',
            hintStyle: kTextStyle2,
            enabledBorder: roundBorder,
            focusedBorder: roundBorder,
          ),
        ),
        const SizedBox(
          height: 40,
        ),
        TextField(
          onChanged: (value) {
            password = value;
          },
          style: kTextStyle2,
          obscureText: true,
          decoration: InputDecoration(
            hintText: 'Password',

```

```

        hintStyle: kTextStyle2,
        enabledBorder: roundBorder,
        focusedBorder: roundBorder,
    ),
),
const SizedBox(
    height: 50,
),
RoundedTextButton(
    text: 'Login',
    textStyle: kTextStyle3,
    color: Theme.of(context).primaryColor,
    onPressed: () async {
        try {
            await auth.signInWithEmailAndPassword(
                email: userName + '@gmail.com', password: password);
            //user = auth.currentUser!;
            Navigator.push(
                context,
                CupertinoPageRoute(builder: (context) {
                    return HomeScreen(userName);
                })),
        );
    } catch (e) {
        print(e);
    }
},
),
const SizedBox(
    height: 30,
),
RoundedTextButton(

```

```

        text: 'Don\'t have an account',
        textStyle: kTextStyle2,
        color: Theme.of(context).secondaryHeaderColor,
        onPressed: () {
            Navigator.pop(context);
            Navigator.push(
                context,
                CupertinoPageRoute(builder: (context) {
                    return const RegistrationScreen();
                }),
            );
        },
    ),
],
),
);
}
}

```

### 3.9.6 moviedetailscreen

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:mbook/screens/bookingsheet.dart';
import '../models/movie.dart';
import '../models/theatre.dart';
import '../services/dbservices.dart';
import '../reusables/posterbox.dart';
import '../constants.dart';
import '../reusables/infobox.dart';
import '../reusables/theatretab.dart';

class MovieDetailScreen extends StatefulWidget {
  const MovieDetailScreen({ Key? key, required this.movie }) : super(key: key);

  final Movie movie;

  @override
  State<MovieDetailScreen> createState() => _MovieDetailScreenState();
}

class _MovieDetailScreenState extends State<MovieDetailScreen> {
  DbServices Db = DbServices();
  List<Theatre> theatres = [];

  @override
  void initState() {
    super.initState();
    setThings();
  }
}
```

```

void setThings() async {
    theatres = await Db.getTheatres();
    setState(() {});
}

List<Widget> getTheatre(Movie selectedMovie) {
    List<TheatreTab> theatresTab = [];
    for (var currentTheatre in theatres) {
        TheatreTab tab = TheatreTab(
            name: currentTheatre.name,
            address: currentTheatre.address,
            id: currentTheatre.id,
            onPressed: (){
                showModalBottomSheet(context: context, builder: (BuildContext context){
                    return BookingSheet(selectedMovie: selectedMovie, selectedTheatre:
currentTheatre,);
                });
            },
        );
        theatresTab.add(tab);
    }
    return theatresTab;
}

```

@override

```

Widget build(BuildContext context) {
    var genre = widget.movie.genre;
    return Scaffold(
        appBar: AppBar(
            elevation: 0.0,
            centerTitle: true,
            backgroundColor: Theme.of(context).scaffoldBackgroundColor,

```

```

title: const Text(
  'Movie Detail',
  style: kTextStyle3,
),
leading: IconButton(
  onPressed: () {
    Navigator.pop(context);
  },
  icon: const Icon(Icons.arrow_back_ios_outlined),
),
),
body: SafeArea(
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          SizedBox(
            height: 350,
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: [
                PosterBox(
                  imageString: widget.movie.poster,
                  onPress: () {},
                ),
                Column(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    InfoBox(
                      icon: Icons.videocam,

```

```

        label: 'Genre',
        text: genre.substring(
            0,
            (!genre.contains(','))
                ? genre.length
                : genre.indexOf(','),
        ),
    ),
    InfoBox(
        icon: Icons.access_time,
        label: 'Duration',
        text: widget.movie.duration,
    ),
    InfoBox(
        icon: Icons.star,
        label: 'Ratings',
        text: '${widget.movie.rating}/10',
    ),
  ],
),
],
),
),
const SizedBox(
  height: 15,
),
Padding(
  padding: const EdgeInsets.only(left: 25.0),
  child: Text(
    widget.movie.title,
    style: kTextStyle1,
  ),
),

```

```

    ),
    const Divider(
      color: Colors.white,
    ),
    Text(
      'Theatres Available',
      style: kTextStyle3.copyWith(
        fontWeight: FontWeight.normal,
        fontSize: 18,
      ),
    ),
    ...getTheatre(widget.movie),
  ],
),
),
),
),
);
}
}

```

## CHAPTER 4



## BIBLIOGRAPHY

1. J. Liu and J. Yu, "Research on Development of Android Applications," 2011 4th International Conference on Intelligent Networks and Intelligent Systems, Kunming, 2011, pp. 69-72, doi: 10.1109/ICINIS.2011.40.
2. Simform, "How to choose the right database," available at: <https://www.simform.com/mobile-app-developers-database-selection/>
3. Android Developers, "Ways to connect to a network," available at: <https://developer.android.com/training/basics/network-ops/connecting>
4. Li Ma, Lei Gu, and Jin Wang, "Research and Development of Mobile Application for Android Platform," International Journal of Multimedia and Ubiquitous Engineering, vol. 9, no. 4, pp. 187-198, 2014, doi: 10.14257/ijmue.2014.9.4.20.
5. Deepali Bajaj, Asha Yadav, Bhawna Jain, Deeksha Sharma, Diksha Tewari, Dinika Saxena, Disha Sahni, Preetanjali Ray, "Android Based Nutritional Intake Tracking Application for Handheld Systems," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, India.
6. Kin Chi Chan, Tak Leung Cheung, Siu Hong Lai, Kin Chung Kwan, Hoyin Yue, Wai-Man Pang, "Where2Buy: A Location-based Shopping App with Products-wise Searching," 2017 IEEE International Symposium on Multimedia (ISM), Taichung, Taiwan.