S.D.M.E Society's

SDM COLLEGE OF ENGINEERING AND TECHNOLOGY, DHAVALGIRI, DHARWAD-580002

(An Autonomous institute affiliated to Visvesvaraya technological university.)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Software Requirements Specification Document

"MOVIE TICKET BOOKING SYSTEM"

Under guidance of

"Prof. Varsha S Jadhav"

Submitted by

YASASWINI B	**2SD23IS064**
VIDYA DHANANJAY PAI	**2SD23IS060**
VAISHNAVI G M	**2SD23IS059**
ARUN H	

3rd Semester B.E **Academic Year 2024-25**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to outline the functionality, features, and requirements of the Movie Ticket Booking System. This system streamlines ticket reservations by allowing customers to browse movies, view showtimes, select seats, and book tickets online, while also providing cinema operators with tools to efficiently manage movie listings, showtimes, and seat availability. The intended audience includes developers, who will use the document for implementation guidance; testers, who will reference it for creating test cases; project managers and stakeholders, to ensure alignment with business objectives; and end users, to verify the system's usability and satisfaction of user needs.

1.2 Scope

The Movie Ticket Booking System is an online platform designed to simplify the reservation process for cinema-goers and to assist cinema operators with managing schedules and seat availability. Users can browse available movies, select showtimes, choose seats, and complete their reservations, all through a streamlined, user-friendly interface.

For cinema operators, the system provides administrative tools for managing movie listings, showtimes, and monitoring seat availability in real time to prevent overbooking. It is designed primarily for small to medium-sized cinemas and does not handle any financial transactions, nor does it include third-party payment integrations. Key benefits include enhanced convenience for customers, improved accuracy, and reduced operational workload for cinema staff.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:

Term	Description
User	An individual who interacts with the Movie Ticket Booking System to browse movies, select showtimes, and book tickets.
Admin	A cinema operator who manages movie listings, showtimes, and seat availability within the Movie Ticket Booking System.
Booking	The process by which a user reserves a seat for a specific movie and showtime.
Cancellation	The action taken by a user to void a previously made booking, making the reserved seat available for other users.
Showtime	A scheduled time at which a movie is screened in a cinema.
Seat Availability	The status indicating whether a seat is free for booking or has already been reserved.

Acronyms and Abbreviations:

SRS - Software Requirements Specification

DFD - Data Flow Diagram

1.4References:

[1] *Data Structures using C", Aaron M Tenenbaum, YedidyahLangsam and Moshe J.Augenstein, Pearson education, Thirteenth Impression 2019.*

[2] *E Balagurusamy, "Programming in ANSI C", 6th Edition, Tata McGraw Hill, 2012*

1.5 Overview

This SRS document provides a complete overview of the Movie Ticket Booking System's functionality, requirements, and intended use. **Section 2, Overall Description**, presents a high-level summary of the system's features, user roles, and main functions, offering customers and potential users a clear understanding of the platform's capabilities and benefits.

For developers and testers, **Section 3: Specific Requirements** provides the detailed functional and non-functional specifications necessary for system development. This section includes the technical requirements, performance standards, and constraints guiding the system's design and testing. This structure ensures each audience can easily access the information most relevant to their roles.

2. Overall Description

2.1 Product Perspective

Here's a refined version that removes future integration details and maintains focus on the current scope:

The Movie Ticket Booking System is a standalone, self-contained application designed to streamline movie ticket reservations for cinema-goers. While it shares similarities with online ticketing platforms like BookMyShow and Fandango—such as allowing users to browse movies, select showtimes, and reserve seats—this system is built for local, console-based use, emphasizing simplicity over the feature richness of larger, web-based applications. It is particularly suited to small and medium-sized cinema operators who require straightforward functionality rather than extensive digital infrastructure.

Comparison with Existing Products

1. Online Ticketing Platforms:

Unlike platforms with integrated payment gateways and extensive online features, this system operates solely with file handling, focusing on core functions such as seat selection, showtime management, and ticket reservations.

2. Traditional Box Office Systems:

Similar to traditional box offices in that it provides ticket reservations, this system modernizes the experience by offering a digital alternative, reducing in-person queues and automating seat management for enhanced efficiency.

Product Scope and Interfaces

This application is fully self-contained, with no external integrations like payment gateways or third-party APIs. Its design centres on core booking functionalities and administrative controls, making it ideal for straightforward, local use.

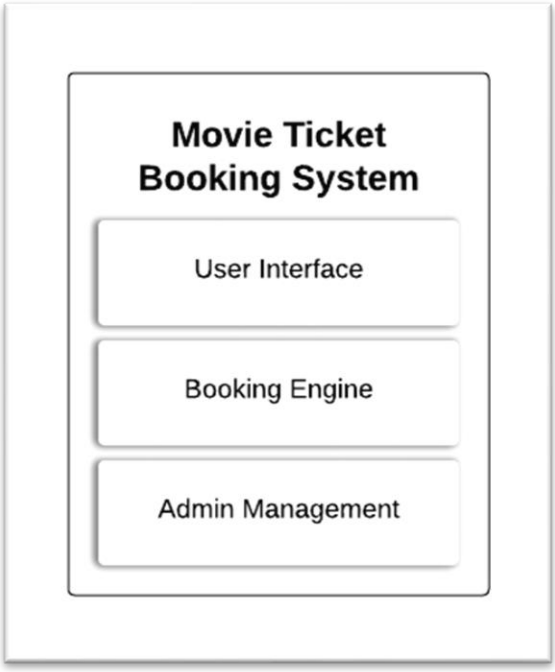

Fig 1.Block Diagram

This diagram highlights the main components of the Movie Ticket Booking System, clarifying its independent, file-based functionality without suggesting external connections or future versions.

2.1.1 System Interfaces

The Movie Ticket Booking System currently functions as a standalone, console-based application without external integrations. Future versions could include interfaces to integrate payment gateways for secure online transactions and a database management system to improve data handling. Additionally, user authentication services could be added for enhanced security and user personalization. These potential integrations would allow the system to scale for larger cinema operations, but they are beyond the current scope.

2.1.2 Interfaces

The Movie Ticket Booking System will use a simple console-based interface for both users and administrators, relying on text-based commands for navigation.

Logical Characteristics of the Interface

1. Command-Line Interface (CLI): Users interact through typed commands, following prompts for registration, movie selection, seat booking, and ticket cancellation.

2. Input/Output: Users input data (e.g., movie choice, seat numbers), and the system outputs confirmations or notifications of errors (e.g., "Seat already booked").

3. Error Handling: Clear error messages assist users with incorrect inputs, allowing easy retry options or navigation back to menus.

Optimizing the User Interface

1. User-Friendly: Designed for simplicity, the interface minimizes complexity, supporting users of all computer literacy levels.

2. Accessibility: Although limited as a console-based application, text clarity and intuitive prompts enhance usability for all users.

2.1.3 Hardware Interfaces

The Movie Ticket Booking System is a software-only console application designed to operate on standard PCs or workstations without requiring specialized hardware. It utilizes basic input/output operations, relying on the keyboard for user commands and the monitor to display outputs. Data, such as movie listings and bookings, is managed through file handling, stored locally on internal or external drives in text or CSV formats. The application interacts with the system's storage through C's file-handling libraries, following standard file system protocols to read and write data. No additional hardware configurations or devices, such as printers or network components, are necessary for operation.

2.1.4 Software Interfaces

The Movie Ticket Booking System is a standalone, C-based application that operates using core C libraries and basic OS functions without dependencies on external software or APIs. It relies on the C Standard Libraries (ANSI C89/C99) for essential functions like file handling and input/output operations (e.g., fopen, fwrite, fread from stdio.h) to manage movie listings and bookings in CSV or text format. Developed in Visual Studio Code (or a similar C IDE such as GCC), the IDE supports coding, compiling, and debugging, enabling efficient testing and deployment of the application.

2.1.5 Communications Interfaces

The Movie Ticket Booking System is a standalone console-based application that operates locally without the need for communication or networking interfaces. It does not connect to external systems or networks, nor does it require any internet or server connections.

All data processing and storage occur on the local machine, relying only on basic file handling for data management. Therefore, there are no requirements for protocols, network communication, or interfacing with external systems, making the application self-contained and isolated from other software environments or networks.

2.1.6 Memory Constraints

The Movie Ticket Booking System is a console-based application designed to run efficiently on machines with a minimum of 1GB of RAM, although most modern systems typically have at least 4GB, ensuring smooth operation even with background applications. The system requires minimal installation disk space, estimated at around 50MB for application files and necessary resources. Overall, the application is optimized for typical user environments, providing good performance without imposing strict memory constraints

2.1.7 Operations

Movie Ticket Booking System Operations Overview

1. Modes of Operation:

- User Mode: Users can search for movies, book tickets, and view booking history.
- Admin Mode: Administrators manage movie listings, schedules, ticket availability, and access sales reports.

2. Interactive Operations:

- Real-Time Interaction: The system functions primarily in real-time, requiring user input for ticket bookings and other tasks.
- Admin Maintenance: Administrators can perform scheduled maintenance tasks during non-peak hours.

3. Data Processing Support Functions:

- The system processes real-time user inputs such as search queries, seat selections, and booking confirmations, with no batch processing involved.

4. Backup and Recovery Operations:

- Backup: Manual backups of critical files (e.g., movie listings and booking records) are necessary, with periodic backups encouraged on local storage.
- Recovery: In case of system failure, log files can be restored from the latest backup, relying on manual file-based recovery due to the absence of automated systems.

5. Operational Constraints:

- Downtime for Maintenance: The system can be taken offline during non-operational hours for maintenance or backups, with minimal expected downtime due to its console-based nature.

2.1.8 Site Adaptation Requirements

Installation and Initialization Requirements for the Movie Ticket Booking System

1. Data and Initialization Sequences:

- Movie Listings and Showtimes: Prior to system activation, administrators must input the initial set of movie listings, including titles, showtimes, and ticket prices. This can be performed through the admin mode during the setup phase.
- Seating Configuration: Each theatre's seating arrangement must be defined and initialized to reflect the actual layout and availability for each movie, ensuring accurate booking capabilities.
- Pricing Policies: Pricing details for different categories (e.g., regular, premium, discounts) need to be established and configured based on site-specific policies before user access is granted.

2. Site or Mission-Related Features:

- Theater-Specific Customization: The software should be tailored to each theater's unique operational requirements, such as location details, seating capacity, and specific promotional pricing or packages.
- Log File Management: Procedures for manual backups of critical data logs must be established, including instructions for administrators on how to perform these backups regularly to avoid data loss.

3. Required Modifications to Customer Work Area:

- Computer Specifications: The customer must ensure that the computer running the application meets the minimum memory and processing requirements to support smooth operations. A standard PC with at least 1GB of RAM is recommended.
- Console Access: The system requires access to a terminal or console; thus, a suitable workspace with an appropriate computer setup should be provided.
- No Significant Hardware Modifications: No major equipment changes (e.g., backup power generators, cooling systems) are necessary for installation, but the existing environment should be conducive to maintaining operational efficiency.

2.2 Product Functions

The Movie Ticket Booking System is a console-based application designed to streamline the ticket booking process for moviegoers. It allows users to view available movies, check showtimes, select seats, and book tickets without the need for APIs or databases. Below are the major functions of the system, highlighting user experience:

1. Movie Listings Display:

- Users can view a list of currently showing movies, including titles, showtimes, available seats, brief descriptions, genres, and durations.

2. Showtime and Availability Checking:

- Users can select a movie to view corresponding showtimes, with seat availability displayed for each option. This allows users to choose based on personal preference.

3. Seat Selection:

- The system presents an interface for users to select their preferred seats from available options. Once booked, seats become unavailable for others. The system validates seat selections to ensure they are within available constraints.

4. Ticket Booking:

- After choosing the movie, showtime, and seats, users proceed to book tickets. The system provides a summary of selected options and prompts for confirmation. Upon confirmation, the booking is finalized, and seats are marked as reserved.

5. Ticket Confirmation:

- Upon successful booking, the system generates a confirmation message detailing the movie name, showtime, number of seats, and a unique booking ID. Users can save or print this confirmation for reference.

6. Admin Mode:

- Administrators can manage movie listings, showtimes, and seat availability. They have the ability to add, update, or remove movies and showtimes, ensuring the system is current.

7. System Exit:

- Users can exit the system at any time, with the current state (e.g., booked tickets, updated seat availability) saved for consistency when reopened.

2.3 User Characteristics

The Movie Ticket Booking System targets a diverse user base, primarily consisting of general audience members and administrative users. Understanding the general characteristics of these users is essential, as it influences both the user interface (UI) design and the internal design of the system.

1. Educational Level:

- Users come from varied educational backgrounds, ranging from high school graduates to those with advanced degrees. This diversity implies that the system must be intuitive and accessible, avoiding complex terminology that could alienate less experienced users. The UI will be designed with clear, straightforward language to ensure that all users can navigate the system without confusion.

2. Experience with Technology:

- Most users are expected to possess basic computer skills, as they will likely have experience with simple applications. This suggests that the system should prioritize a user-friendly console interface that minimizes the learning curve. Given that many users may not have extensive experience with technology, the design will focus on straightforward workflows, clear instructions, and helpful prompts to guide users through the ticket booking process.

3. Technical Expertise:

- The intended users do not require programming knowledge or advanced technical skills. This characteristic informs the design by emphasizing simplicity and ease of use. For example, the system will feature effective error handling to assist users in correcting mistakes, ensuring that users with minimal technical expertise can navigate the application with confidence.

Design Implications

The characteristics of the user base will significantly impact the system's design:

- User Interface Design: The UI will be crafted to be intuitive and navigable, with clear labels and options that reflect the user's familiarity with basic applications. The design will incorporate visual cues and prompts that enhance usability for those with limited technical experience.
- Internal System Design: The design will account for user interactions that may not conform to expected patterns, necessitating robust error handling and user feedback mechanisms. This ensures that the system remains resilient and user-friendly, even in cases of user error.
- Training and Support: While the system aims for self-sufficiency, the variety in user experience may lead to a need for supplementary materials, such as user guides or quick reference tips, to enhance user confidence and satisfaction.

2.4 Constraints

The Movie Ticket Booking System operates under various constraints that shape its design and development:

1. Hardware Limitations: The system must be optimized to run on basic computer hardware with minimal RAM and storage, ensuring compatibility with legacy systems.

2. Single-User Operation: The system is designed for single-user operation at any given time, which limits the need for multi-threading or concurrent processing.

3. Regulatory Policies: Although there are no specific regulatory policies for a console-based internal system, general best practices regarding data privacy and security must be adhered to.

4. No External Interfaces: The system does not interface with any external APIs or databases, limiting integration options with other software.

5. Simplicity of Functionality: The focus on a straightforward user experience constrains the complexity of features and the overall system architecture.

6. Reliability Expectations: The system must ensure correct ticket management to prevent issues like double bookings, placing a premium on stability and reliability during development.

7. Basic Security Measures: While advanced security features like encryption are not necessary, there are basic security requirements, such as preventing unauthorized access to the admin panel.

These constraints define the operational environment and functionality, guiding the design and development process while limiting the scope of implementation options.

2.5 Assumptions and Dependencies

The development of the Movie Ticket Booking System relies on several assumptions and external dependencies. These factors, though not directly constraining the design, can impact the requirements in the SRS if they change or are not met.

1. Operating System Compatibility: If the target platform does not support the required operating systems (Windows, Linux, macOS), the SRS would need to be updated to accommodate a different OS.

2. Console Access: Changes in the availability of a console or terminal environment may necessitate a revision of the SRS to include alternative user interaction methods.

3. Integration with APIs or Databases: Should there be a future requirement for integrating with external APIs or databases, the SRS would need to specify additional data handling and interaction requirements.

4. Hardware Specifications: If the system is deployed on hardware that does not meet the basic processing, memory, or storage requirements, the SRS would need to be adjusted to reflect these limitations.

5. User Load: The initial design assumes a single-user operation. If there is a shift to support multiple users or simultaneous operations, the SRS would require modifications to include multi-user capabilities.

6. User Technical Skills: A change in the expected technical proficiency of the user base may necessitate enhancements to the user interface or user instructions within the SRS.

7. Data Storage Needs: If future requirements dictate more complex data storage solutions (e.g., relational databases), the SRS would need to be revised to address these new data management requirements.

8. Network Requirements: If the system is required to support online features or remote data access in the future, the SRS would need to include specifications for network dependencies.

9. Feature Expansion: Any new requirements for advanced features (e.g., graphical user interfaces, cloud integration, or analytics) would necessitate updates to the SRS to accommodate these functionalities.

10. Regulatory Changes: If any legal or regulatory requirements regarding data privacy or user information security emerge, the SRS may need to be modified to ensure compliance.

2.6 Apportioning of Requirements

Due to time constraints, resource limitations, and customer prioritization, several features of the Movie Ticket Booking System may be postponed for future iterations. These features, while beneficial, are not critical for the initial release and can be planned for later versions.

1. Multi-User Support

- Current Version: Supports single-user operations.
- Future Version: Introduction of multi-user functionality, including user authentication and access control.

2. Graphical User Interface (GUI)

- Current Version: Console-based application with a command-line interface.
- Future Version: Development of a GUI to enhance user-friendliness for a wider audience.

3. Integration with Online Payment Gateways

- Current Version: No online payment functionality included.
- Future Version: Integration with payment gateways to facilitate online transactions.

4. Database Integration

- Current Version: Data stored in text files.
- Future Version: Transition to a database-backed system for improved data handling and reporting.

5. Report Generation

- Current Version: Basic records of bookings and cancellations.
- Future Version: Advanced reporting features, such as sales summaries and booking trends.

6. Online Booking Capabilities

- Current Version: Designed for offline use.
- Future Version: Implementation of online booking features for greater versatility.

7. Mobile Application Support

- Current Version: Desktop console application only.
- Future Version: Development of a mobile application for ticket booking via smartphones.

8. Email and SMS Notifications

- Current Version: No notifications sent to users.
- Future Version: Addition of notifications for booking confirmations and reminders.

9. Dynamic Pricing and Offers

- Current Version: Fixed pricing model.

- Future Version: Introduction of dynamic pricing and promotional offers.

These deferred features will be prioritized by the customer based on their relevance and importance, enabling an iterative development process that focuses on delivering core functionalities first, while planning for advanced features in subsequent releases.

3. Specific Requirements

This section outlines the detailed, verifiable requirements for the Movie Ticket Booking System to ensure that designers and developers can build a system that meets customer needs. The requirements are divided into functional and non-functional categories, and every requirement is specific, unambiguous, and traceable. Each requirement is numbered for reference, and they align with the constraints, assumptions, and apportioning described earlier.

3.1 External Interfaces

This section provides a detailed technical description of all inputs into and outputs from the Movie Ticket Booking System. It includes the content and format details necessary for developers to understand how the system will interact with external entities.

Here's the revised version of the external interfaces for the Movie Ticket Booking System with the Payment Input section removed:

3.1.1 User Registration Input

- Name of Item: User Registration Input
- Description of Purpose: Captures new user information for account creation.
- Source of Input: User input via a console-based form.
- Valid Range, Accuracy, and/or Tolerance:
- Username: 3-20 alphanumeric characters.
- Password: Minimum of 8 characters, must include at least one number and one special character.
- Email: Must follow valid email format (e.g., example@domain.com).
- Units of Measure: N/A
- Timing: Immediate input during the registration process.
- Relationships to Other Inputs/Outputs: Successful registration creates a new user entry in the system database.
- Data Formats:
  - Username: String
  - Password: String (hashed for security)
  - Email: String
  - Command Formats: register [username] [password] [email]
- End Messages:
  - "Registration successful."
  - "Error: Invalid input. Please try again."

3.1.2 Login Input

- Name of Item: User Login Input

- Description of Purpose: Authenticates an existing user and grants access to the system.
- Source of Input: User input via the console login form.
- Valid Range, Accuracy, and/or Tolerance:
  - Username: 3-20 alphanumeric characters.
  - Password: Minimum of 8 characters.
- Units of Measure: N/A
- Timing: User submits credentials during the login process.
- Relationships to Other Inputs/Outputs: Successful login provides access to movie selection, booking, and payment functionalities.
- Data Formats:
  - Username: String
  - Password: String
- Command Formats: login [username] [password]
- End Messages:
  - "Login successful."
  - "Error: Invalid credentials."

3.1.3 Movie Search Input

- Name of Item: Movie Search Input
- Description of Purpose: Allows users to search for available movies based on title, genre, or showtime.
- Source of Input: User input in the console search field.
- Valid Range, Accuracy, and/or Tolerance:
- Search Query: Up to 100 characters (movie title or genre).
- Units of Measure: N/A
- Timing: Search results are returned immediately after query submission.
- Relationships to Other Inputs/Outputs: Query results are matched against the available movies database.
- Data Formats:
  - Search Query: String
  - Command Formats: search [title/genre]
- End Messages:
  - "Displaying search results."
  - "No movies found matching your query."

3.1.4 Seat Selection Input

- Name of Item: Seat Selection Input
- Description of Purpose: Allows users to select seats for a specific movie and showtime.
- Source of Input: User input in the seat selection screen.
- Valid Range, Accuracy, and/or Tolerance:
- Seat Numbers: Defined by the theater layout (e.g., A1, B2).

- Units of Measure: N/A
- Timing: Real-time selection process to avoid overbooking.
- Relationships to Other Inputs/Outputs: The selected seats are marked as booked in the system, reducing availability for other users.
- Data Formats:
  - Seat Number: String
  - Command Formats: select [seat number(s)]
- End Messages:
  - "Seats selected successfully."
  - "Error: Seat(s) unavailable."

3.1.5 Cancellation Input

- Name of Item: Booking Cancellation Input
- Description of Purpose: Cancels a booked ticket within the allowable time frame.
- Source of Input: User input via console.
- Valid Range, Accuracy, and/or Tolerance:
- Booking Reference Number: Alphanumeric (e.g., BK12345).
- Units of Measure: N/A
- Timing: Immediate update to the system and user notification.
- Relationships to Other Inputs/Outputs: Updates the booking database to mark tickets as cancelled and processes refunds if applicable.
- Data Formats:
  - Booking Reference Number: String
- Command Formats: cancel [booking reference number]
- End Messages:
  - "Booking cancelled successfully."
  - "Error: Unable to cancel booking."

This structured description of external interfaces will guide developers in understanding the required inputs and outputs for the Movie Ticket Booking System, ensuring consistency and accuracy across various operations.

Here's a summarized version of the functional requirements for the Movie Ticket Booking System, organized according to your guidelines:

3.2 Functions

This section outlines the functional requirements of the Movie Ticket Booking System, specifying the core functionality and operations the system must perform to meet business and user needs.

3.2.1 User Registration

- The system shall allow users to register by providing a username, password, and email, and validate inputs for uniqueness and format.
- The system shall hash and securely store user passwords and confirm successful registration with a message.

3.2.2 User Login

- The system shall authenticate users by verifying their username and password, granting access upon successful login.
- The system shall deny access and lock the user out after three failed attempts, displaying appropriate error messages.

3.2.3 Movie Search

- The system shall enable users to search for movies by title, genre, or showtime and retrieve results from the database.
- The system shall display relevant movie details and indicate when no matches are found.

3.2.4 Seat Selection

- The system shall allow users to select available seats, updating seat availability in real-time.
- The system shall confirm seat selection with details and total cost while providing error messages for unavailable seats.

3.2.5 Booking Confirmation

- The system shall generate a booking confirmation with movie details, send it to the user, and store it in their booking history.

3.2.6 Ticket Cancellation

- The system shall allow users to cancel bookings by validating the booking reference number and processing refunds as applicable.
- The system shall confirm successful cancellations with appropriate messages.

3.2.7 Error Handling and Recovery

- The system shall handle invalid inputs and log errors for administrative review while attempting to recover from errors by reinitializing processes.
- The system shall provide fallback options for seat selection issues.

3.2.8 Data Validation

The system shall validate all user inputs, ensuring correct formats to prevent vulnerabilities and ensure data integrity. This partitioning of functional requirements ensures the system processes inputs reliably, validates user actions, handles exceptions, and provides a robust, user-friendly experience.

3.3 Performance Requirements

This section outlines the static and dynamic numerical requirements for the Movie Ticket Booking System.

3.3.1 Static Numerical Requirements

- The system shall support 50 terminals for user access.
- The system shall accommodate up to 1,000 simultaneous users.
- The system shall manage up to 10,000 movie listings.
- The system shall store information for up to 10,000 registered users.

3.3.2 Dynamic Numerical Requirements

- 95% of transactions shall be processed in less than 2 seconds under normal conditions.
- The system shall handle up to 500 transactions per minute during peak hours.
- 90% of search results shall be displayed in less than 1 second.
- Users shall complete bookings in under 5 minutes for 90% of transactions.
- The system shall resolve 90% of errors in less than 1 hour.
- Data backup operations shall complete in under 10 minutes during off-peak hours.

3.4 Logical Database Requirements

This section specifies the logical requirements for information to be placed into the database, covering various aspects necessary for efficient data management.

Types of Information

- User Information: User ID, Username, Hashed Password, Email, Phone Number
- Movie Listings: Movie ID, Title, Genre, Director, Cast, Duration, Rating, Showtimes, Available Seats
- Bookings: Booking ID, User ID (FK), Movie ID (FK), Showtime, Selected Seats, Total Amount Paid, Booking Status, Timestamp
- Payments: Payment ID, Booking ID (FK), Payment Method, Payment Status, Payment Amount, Timestamp

Frequency of Use

- User information: Frequently accessed during login and booking processes.
- Movie listings: Accessed multiple times per session for searches and selections.
- Booking information: Accessed for booking confirmation and management.
- Payment information: Accessed upon transaction completion.

Accessing Capabilities

- Users can view their profile information and booking history.
- Available movies, showtimes, and seat selections are accessible without authentication.
- Admin users have elevated access rights to manage movie listings and bookings.

Data Entities and Their Relationships

- User (1) <--> (N) Booking: A user can make multiple bookings.
- Movie (1) <--> (N) Booking: A movie can have multiple bookings.
- Booking (1) <--> (1) Payment: Each booking corresponds to one payment.

Integrity Constraints

- User ID, Movie ID, and Booking ID must be unique.
- Email addresses must be unique across users.
- Seat selections must validate against available seats for the chosen showtime.

Data Retention Requirements

- User information and booking history: Retained for 3 years after the last booking.
- Payment records: Retained for 7 years for auditing purposes.
- Movie listings: Archived after 2 years of inactivity but retrievable upon request.

By adhering to these logical database requirements, the Movie Ticket Booking System will ensure efficient data management and support necessary operations while maintaining data integrity.

3.5 Design Constraints

This section outlines the design constraints for the Movie Ticket Booking System, which may arise from external standards, hardware limitations, and other factors that must be considered during the system's development.

3.5.1 Standards Compliance

1. Report Format: Reports must follow a standard format with headers, timestamps, and be exportable (e.g., CSV).

2. Data Naming: Files should follow consistent naming conventions for clarity and easy access.

3. File Handling: Standardized procedures for reading, writing, deleting, and regular backups are required for data integrity.

4. Audit Tracing: The system must log all file operations, with before/after values for critical changes, stored securely with restricted access.

3.6 Software System Attributes

The Movie Ticket Booking System must meet quality standards for usability, reliability, performance, maintainability, portability, security, and scalability. These attributes ensure effective operation, high user satisfaction, and long-term adaptability of the software.

3.6.1 Reliability

To meet reliability requirements at the time of delivery, the software system must adhere to these factors:

1. Mean Time Between Failures (MTBF):

The software shall achieve an MTBF of at least 500 hours, indicating stable operation before failure.

2. Error Handling:

Robust error handling will ensure unexpected inputs are managed without crashes, logging errors with details for later analysis.

3. Data Integrity:

Data integrity checks (e.g., checksum verification) and backup protocols will prevent data loss during file handling.

4. Testing Requirements:

The software shall undergo unit, integration, and system tests with 90% code coverage to validate reliability.

5. Performance Monitoring and User Notifications:

Built-in performance monitoring will detect reliability issues early, with clear notifications guiding users on errors and corrective actions.

Together, these factors establish a foundation for consistent and dependable software performance.

3.6.2 Availability

The following factors are required to ensure the system's defined availability:

1. Availability Level: The system shall operate with a minimum availability of 99% each month, maintaining accessibility during expected operational hours.

2. Checkpointing: Regular checkpoints shall save the current state every 5 minutes, allowing users to resume from the last saved state after an interruption.

3. Recovery and Restart: The system shall provide recovery and restart capabilities that allow users to resume after a failure with a loss of no more than 12 characters of input.

4. Graceful Restart: Upon restart, users should return to their previous workflow with minimal disruption, preserving recent actions to ensure continuity.

3.6.3 Security

To protect the software from unauthorized access and ensure data integrity, the following security measures are specified:

1. User Authentication:

Limit system access to registered users only, requiring user login with password verification.

2. Password Hashing:

Use secure hashing (e.g., SHA-256) for passwords, storing only hashed values to protect credentials in case of data exposure.

3. Access Control for Sensitive Operations:

Restrict critical operations to verified users only, enforcing access control on essential functions.

4. Data Integrity Checks:

Implement checksums or hash validations on critical data fields to detect any tampering.

5. Logging for Registered Users:

Maintain a basic log for user actions like logins and sensitive operations to monitor and review access patterns.

These measures ensure only verified users access the system and help secure critical data against unauthorized access or tampering.

3.6.4 Maintainability

To ensure effective maintenance and updates by future developers, the software shall include:

1. Modularity: Organized into distinct modules for easier debugging and updates.

2. Clear Interface Definitions: Well-defined input and output specifications for each module.

3. Comprehensive Documentation: Detailed comments and external documentation for modules and functions.

4. Error Handling Protocols: Structured error management and logging mechanisms.

5. Automated Testing: A suite of automated tests to verify functionality and prevent new issues.

3.6.5 Portability

This section defines key characteristics that enhance the software's ability to be ported across different platforms with minimal effort.

1. Host-Dependent Code Percentage (H):

Limit host-dependent code to no more than 20% of the total codebase to reduce complexity in porting.

2. Use of Portable Language (H):

Develop the software in C to maximize portability across various operating systems.

3. Compiler Compatibility (M):

Use widely accepted compilers like GCC and Clang to promote compatibility and simplify the build process.

4. Operating System Independence (H):

Ensure the application operates on both Windows and Linux without significant changes to the code.

5. Standard Library Usage (M):

Utilize only standard libraries available on all target operating systems to reduce compatibility issues.

ID	Characteristic	H/M/L	1	2	3	4	5	6	7	8	9	10	11	12
1	Correctness	H												
2	Efficiency	M												
3	Flexibility	M												
4	Integrity/Security	H												
5	Interoperability	M												
6	Maintainability	H												
7	Portability	H												
8	Reliability	H												
9	Reusability	M												
10	Testability	H												
11	Usability	M												
12	Availability	H												

3.7 Organizing the Specific Requirements

3.7.1 System Mode

The Movie Ticket Booking System can operate in several distinct modes, each serving different user needs. The primary modes include:

- User Mode: This mode is designed for customers looking to book tickets. In this mode, users can browse movies, select showtimes, choose seats, and make payments. The interface should prioritize ease of navigation and clarity of information, allowing users to complete bookings with minimal steps.
- Admin Mode: Administrators or theater staff operate in this mode to manage the backend of the system. Tasks include adding new movies, scheduling showtimes, managing seat availability, and viewing sales reports. This mode may require a more complex interface with advanced functionalities for inventory and data management.

3.7.2 User Class

The Movie Ticket Booking System supports various user classes, each with unique roles and permissions. These user classes may include:

- Regular Customers: These users can browse movies, book tickets, and manage their bookings. Requirements for this class should emphasize user-friendly navigation and straightforward booking processes.
- Administrators: Admins have access to all functionalities within the system, including backend management. They can add or remove movies, manage user accounts, and generate sales reports. Requirements for this class should focus on data integrity, security, and efficient management tools.

3.7.3 Objects

The key objects within the Movie Ticket Booking System include:

- Users: Represents customers and administrators. Attributes may include user ID, name, email, and booking history. Functions related to this object include creating accounts, logging in, and managing personal information.
- Movies: Represents films available for booking. Attributes may include movie title, genre, duration, and rating. Functions associated with movies include adding new titles, updating movie details, and removing old films.
- Theaters: Represents physical locations where movies are screened. Attributes may include theater name, location, and list of showtimes. Functions related to theaters include managing seating arrangements and displaying available showtimes.
- Bookings: Represents individual ticket reservations. Attributes may include booking ID, user ID, movie ID, showtime, seat selection, and payment status. Functions related to bookings include creating new reservations, cancelling bookings, and processing payments.

3.7.4 Feature

The Movie Ticket Booking System includes several key features, each representing an externally desired service. Features can be described using sequences of stimulus-response pairs. Some important features include:

- Ticket Booking: Users select a movie and showtime, choose seats, and confirm the booking. The system should respond by confirming the reservation and providing a booking reference number.
- Seat Selection: Users can view available seats and select their preferred options. The system should update seat availability in real-time and provide visual feedback on the selected seats.
- Booking Confirmation: After a successful booking, the system sends a confirmation email or notification to the user with details of their reservation. This feature should also allow users to view their booking history.

3.7.5 Stimulus

In this organizational scheme, the Movie Ticket Booking System can be described by its functions based on user stimuli, which initiate specific actions. Examples of stimuli include:

- Selecting a Movie: When a user selects a movie from the list, the system retrieves showtimes and seat availability for that movie.
- Choosing Seats: When a user clicks on a seating layout, the system highlights available seats and allows the user to make a selection.
- Submitting a Booking: Upon clicking the 'Book Now' button, the system processes the booking request and displays confirmation or error messages based on the success of the operation.

3.7.6 Response

Conversely, the system can also be organized by the responses generated from user interactions. For example:

- Booking Confirmation: After a user submits a booking, the system provides a confirmation message along with a booking reference number.
- Error Notifications: If a user tries to book an unavailable seat, the system should provide immediate feedback, indicating that the selection is invalid.

3.7.7 Functional Hierarchy

When other organizational schemes are insufficient, the overall functionality of the Movie Ticket Booking System can be structured into a hierarchy based on common inputs, outputs, or internal data access. This hierarchy can be visualized using data flow diagrams (DFDs) that illustrate how data moves through the system and how different functions interact with each other. For example:

- Booking Management: Includes functionalities for searching movies, selecting showtimes, and processing bookings.
- User Management: Covers account creation, authentication, and user profile management.
- Payment Processing: Encompasses features for handling different payment methods and confirming transactions.

Utilizing functional hierarchies clarifies the relationships between various functions and ensures that all aspects of the system are accounted for in the requirements. Data dictionaries can further elaborate on the data structures involved, enhancing overall understanding and implementation.

3.8 Additional Comments

1. Combining Approaches

- Mode and User Class: Outline requirements based on user roles and operational modes. For example, user management can vary by user type (e.g., admin vs. regular user) and operational context (e.g., setup mode vs. normal operation).

- Feature and Functional Hierarchy: Detail stimulus-response sequences for critical features alongside a functional hierarchy that groups related functions, benefiting both users and developers.

Notations and Tools

Several notations and tools can effectively support requirement documentation:

- Finite State Machines: Useful for visualizing state transitions based on inputs in systems with distinct operational modes.
- Object-Oriented Analysis: Methods like UML organize requirements around objects and interactions, enhancing clarity.
- Data Flow Diagrams (DFDs): Illustrate data flows between processes, inputs, and outputs, clarifying system operations.
- Stimulus-Response Sequences: Clarify how the system reacts to specific inputs, particularly in user-facing applications.

Requirement Specification

"Functional Requirement i" can be structured in various ways:

- Native Language: Accessible descriptions for non-technical stakeholders.
- Pseudocode: Structured logic outlines that avoid specific programming languages.
- System Definition Language: Formal specification of functions and behaviors for precision.
- Four Subsections Approach:
- Introduction: Purpose and importance of the requirement.
- Inputs: Required inputs, formats, and valid ranges.
- Processing: How inputs are processed to generate outputs, including algorithms.
- Outputs: Outputs produced, including formats and expected results.

Using multiple organizational techniques enriches the SRS, catering to diverse stakeholder needs and ensuring clarity in essential functions. This approach enhances communication and collaboration, contributing to the success of the software development process.

4. Change Management Process

The change management process for this project is designed to systematically identify, log, evaluate, and update the Software Requirements Specification (SRS) to reflect any changes in project scope and requirements. This structured approach ensures that all modifications are carefully considered and documented to maintain project integrity and alignment with stakeholder expectations.

4.1 Change Identification

Changes may arise from various sources, including client feedback, stakeholder requests, or identified gaps in requirements. Each change request must be formally documented to ensure clarity and facilitate evaluation.

4.2 Change Logging

- All change requests will be logged in a Change Request Document. This document will include:
- Request ID: Unique identifier for each request.
- Date Submitted: When the request was made.
- Requester: The individual or organization making the request.
- Description of Change: Detailed explanation of the requested change.
- Rationale: Reasoning behind the change.

4.3 Change Evaluation

Each logged change request will undergo an evaluation process to assess its impact on:

- Project Scope: Will the change affect the overall project deliverables?
- Timeline: How will the change impact the project schedule?
- Resources: Will additional resources be required?
- Cost: Will the change affect the budget?

The evaluation will involve the relevant stakeholders, including project managers, team leads, and, when necessary, the customer.

4.4 Change Approval Process

1. Consensus Requirement: Changes will require consensus among the project team. Team members will review the evaluated change request and discuss its implications.

2. Customer Communication: The customer cannot simply call to request changes without following the formal process. Change requests must be submitted in writing, either through email or a formal Change Request Document. This ensures that all requests are documented and can be referenced later.

3. Final Approval: Once a consensus is reached, the change request will be formally approved by the project manager. The approved request will then be added to the Change Request Document for tracking purposes.

4.5 Implementation of Changes

Upon approval, the team will update the SRS to reflect the changes. The updated document will be circulated among stakeholders to ensure everyone is informed of the new requirements.

4.6 Communication of Changes

All stakeholders will be notified of changes to requirements via:

- Email Notification: A summary of the changes will be sent to all relevant parties.
- Document Revision: Updated versions of the SRS will be distributed, highlighting changes made.

This change management process provides a structured approach to handling modifications in project scope and requirements. By requiring formal submissions and ensuring team consensus, we maintain the integrity of the project while accommodating necessary adjustments.