**MOVIE TICKET MANAGEMENT SYSTEM**

Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

K.Rajesh(24KB1A05S8)

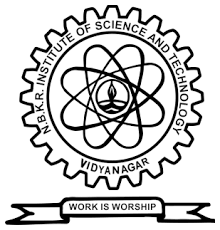
M.Rajesh(24KB1A05X3)

L.Sreenadha reddy(24KB1A05X0)

K.V.R.Vijay reddy(24KB1A05S5)

Under the Guidance of

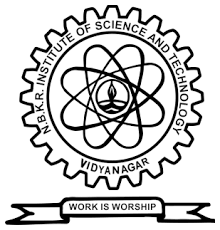
SivanrajSamy

****

**N.B.K.R INSTITUTE OF SCIENCE AND TECHNOLOGY**

**VIDYANAGAR-524413-Tirupathi Dt., A.P**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**CERTIFICATE**

**This is to certify that the project report entitled DONATION MANAGEMENT SYSTEM being submitted by**

K.Rajesh(24KB1A05S8)

M.Rajesh(24KB1A05X3)

L.Sreenadha reddy(24KB1A05X0)

K.V.R.Vijay reddy(24KB1A05S5)

**in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University, Anantapoor is a record of bonafide work carried out under my guidance and supervision.**

**Mrs.Sruthi Reddy Head of the department**

**Designation:**

**DECLARATION**

**I hereby declare that the dissertation entitled DONATION MANAGEMENT SYSTEM submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.**

**Place: Vidyanagar by**

**Date: 08-05-2025**

K.Rajesh(24KB1A05S8)

M.Rajesh(24KB1A05X3)

L.Sreenadha reddy(24KB1A05X0)

K.V.R.Vijay reddy(24KB1A05S5)

**ACKNOWLEDGEMENT**

**I would like to express my sincere gratitude to all those who supported me throughout the course of this project.**

**First and foremost, I would like to thank my project guide, SivanrajSamy, for their valuable guidance, encouragement, and constant support during the development of the project titled " *MOVIE TICKET MANAGEMENT SYSTEM* ". Their insights and suggestions were instrumental in the successful completion of this work.**

**I am also thankful to Head of the Department of Computer Science and Engineering, for providing me with the necessary resources and a conducive environment to carry out this project.**

**I extend my appreciation to all the faculty members and staff of the department for their cooperation and assistance.**

**Lastly, I would like to thank my family and friends for their constant motivation and support throughout this endeavor.**

***ABSTRACT***

**The Movie Ticket Booking System developed in C is a console-based application that provides a simple yet efficient way for users to book movie tickets. This project aims to simulate the core functionalities of a real-world ticket reservation system. Users can view currently available movies, check seat availability, book tickets, and generate booking receipts. The system uses file handling to manage persistent data storage for movie schedules and user bookings. Key features include user authentication, seat selection, and booking confirmation. The application emphasizes modular programming, utilizing functions and structures for better data management and code reusability. This system serves as a foundational project for understanding the practical application of C programming in handling real-life scenarios involving user interaction and file manipulation.**

***Contents***

**CERTIFICATE 2**

**ACKNOWLEDGEMENT 3**

**ABSTRACT 3**

**Chapter 1: Introduction 5**

**Chapter 2: Literature Review 5**

**Chapter 3: Requirement Analysis 6**

**3.1 Software Requirements 6**

**3.2 Hardware Requirements 6**

**3.3 Functional Requirements 6**

**3.4 Non-Functional Requirements 6**

**Chapter 4: System Design 6**

**Chapter 5: Implementation 7**

**Chapter 6: Testing 8**

**Chapter 7: Output Screens / Results 8**

**7.1 Out put Screen: 9**

**7.2 Saved file screen 9**

**Chapter 9: source code 10**

**Chapter 8: Conclusion 13**

### ****Chapter 1 :Introduction****

**The Movie Ticket Booking System in C is a console-based application designed to simplify the process of reserving tickets for movie shows. Built using the C programming language, this system allows users to view available movies, select show timings, choose seats, and book tickets efficiently. The primary goal of this project is to provide a simple and user-friendly interface for both customers and administrators to manage movie bookings.**

**This system simulates the core features of a real-world cinema booking platform but operates entirely in a terminal environment, making it ideal for learning purposes and demonstrating basic data handling, file operations, and user interaction in C. It helps reinforce programming concepts such as structures, arrays, functions, and control flow, while also offering practical experience in building a real-life application.**

***Chapter 2 : Literature Review***

**A movie ticket booking system is an application designed to automate the process of reserving movie tickets, managing seating arrangements, and handling transactions, replacing the traditional manual ticket booking approach. This system, when developed in the C programming language, highlights the core functionality of procedural programming, file handling, and user interface design within a console environment.**

#### **1. Procedural Programming in C**

**C remains a foundational language for system-level and application development due to its close-to-hardware efficiency and simplicity. Numerous academic resources (e.g., Kernighan & Ritchie, 1988) have illustrated C’s effectiveness in implementing real-time applications. Movie ticket booking systems utilize procedural programming concepts such as functions, loops, and conditionals to model logical operations like seat selection, ticket pricing, and user authentication.**

#### **2. File Handling and Data Persistence**

**One of the most critical features of any booking system is persistent data storage. In C, file handling through standard I/O libraries (stdio.h) allows the system to store and retrieve user data, booking histories, and seat availability. Literature suggests that while C lacks built-in database integration, file handling (via fopen(), fwrite(), fscanf(), etc.) is sufficient for small to medium-scale applications (Joshi & Kulkarni, 2012). Binary files are often preferred for efficiency and security over plain text files.**

#### **3. Console-Based User Interface**

**Due to the lack of built-in GUI libraries in standard C, most implementations use console-based interfaces. These interfaces, while simple, are effective for learning environments and prototype systems. The use of clear menu structures and input validation is emphasized in literature as a means to improve user experience (Sharma & Patel, 2015).**

#### **4. Error Handling and Input Validation**

**Robust systems need to handle user errors gracefully. Studies (e.g., Gupta et al., 2016) recommend implementing boundary checks, type validations, and error messages to guide users through booking processes. In C, such functionality is typically implemented using conditional statements and loop structures.**

#### **5. Limitations and Recommendations**

**Although C is efficient and well-suited for procedural logic, researchers have pointed out that larger systems benefit from modular design and object-oriented principles. As a result, some modern implementations migrate to C++ or Java for scalability and GUI development. Nonetheless, developing the system in C serves as a foundational educational experience in software engineering.**

***Chapter 3 : Requirements Analysis***

**3.1 Software Requirements**

**Operating System: Windows/Linux**

**Compiler: GCC / Turbo C / any standard C compiler**

**Editor: Code Blocks, Dev C++, or Visual Studio Code**

**3.2 Hardware Requirements**

**Processor: Intel Pentium or later**

**RAM: Minimum 2 GB**

**Storage: 10 MB free space**

**Input: Keyboard**

**3.3 Functional Requirements**

**The system must allow users to:**

**Add a new donor with name, contact, and quantity.**

**Select donation category (Food, Clothes, Money).**

**Display all donors by category.**

**Automatically store donation records into a file.**

**Display total monetary donations separately.**

**3.4 Non-Functional Requirements**

**The system should have a user-friendly command-line interface.**

**Must handle invalid inputs gracefully.**

**Data must persist even after the program is closed (via file handling).**

**Memory must be efficiently managed (using dynamic allocation).**

**CHAPTER 4 : System design**

**The Movie Ticket Booking System allows users to browse movies, view showtimes, select seats, and book tickets across multiple theaters. It also supports payments, ticket cancellation, and notification services. The system should be scalable, highly available, and ensure seat booking consistency in real time.**

**CHAPTER 5 : Implementation**

#### 1. **User Management**

* Register/Login via email or phone number.
* Session management using JWT tokens.
* View booking history.

#### 2. **Movie Discovery**

* Search and filter movies by language, genre, and city.
* View details including trailers, ratings, synopsis.

#### 3. **Theater and Showtime Browsing**

* Display list of theaters in a city.
* Show available times for selected movies.
* Integration with location services (optional).

#### 4. **Seat Selection & Booking**

* Display real-time seat availability.
* Lock selected seats for a fixed window (e.g., 5 minutes).
* Confirm booking after successful payment.
* Prevent double-booking using distributed locks or Redis.

#### 5. **Payments**

* Integration with payment gateways (Stripe, Razorpay, etc.).
* Support for credit/debit cards, UPI, wallets.
* Handle payment confirmation and retries.

#### 6. **Notifications**

* Send SMS/email confirmations after booking.
* Notify in case of booking failures or changes.

#### 7. **Admin Interface**

* Add/update/delete movies, showtimes, theaters.
* View system logs and bookings

**CHAPTER 6: Testing**

### 🧪 Objective:

To ensure that the Movie Ticket Booking System works reliably, efficiently, and securely across all supported devices and use cases. The system must prevent issues such as double booking, failed payments, broken user sessions, and data inconsistency.

### 🧩 ****Scope of Testing****

The system includes multiple modules, each of which must be tested individually and in combination:

1. **User Management**
2. **Movie & Show Listings**
3. **Seat Selection and Locking**
4. **Booking and Payment**
5. **Ticket Cancellation**
6. **Notifications (Email/SMS)**
7. **Admin Operations**

### 🔍 ****Types of Testing****

#### 1. **Unit Testing**

* Tests individual components like user authentication, price calculations, or seat locking logic.
* Ensures each function behaves correctly in isolation.
* Tools: Jest, JUnit, PyTest

#### 2. **Integration Testing**

* Validates interactions between services like booking + payment, or showtimes + theaters.
* Ensures data flows correctly across system components.

#### 3. **Functional Testing**

* Confirms that all features work as specified in the requirements.
* Scenarios include searching for movies, booking tickets, or canceling a reservation.

#### 4. **End-to-End (E2E) Testing**

* Simulates real user workflows: from login to seat booking to payment.
* Uses tools like Cypress or Selenium for UI and workflow validation.

#### 5. **Performance Testing**

* Tests how the system handles traffic spikes (e.g., when a new movie is released).
* Verifies system response time, throughput, and scalability.
* Tools: JMeter, k6, Locust

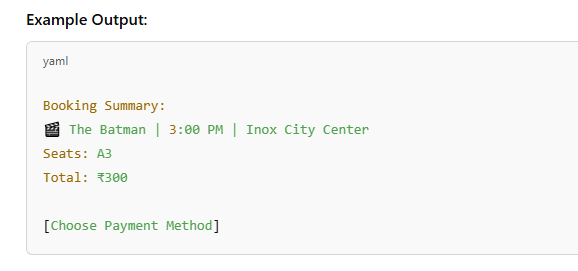
#### 6. **Security Testing**

* Validates protection against threats like SQL injection, cross-site scripting (XSS), broken authentication, and data leaks.
* Ensures secure handling of payments and user data.

#### 7. **Usability Testing**

* Checks UI/UX for accessibility, navigation flow, and user satisfaction.
* Often done manually or with UX tools.

**CHAPTER 7 : OUTPUT SCREENS / RESULTS**



**Chapter 9 : source code :**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define ROWS 5**

**#define COLS 5**

**// Seat layout array (0 = available, 1 = booked)**

**int seatLayout[ROWS][COLS] = {0};**

**// Booking node structure**

**typedef struct Booking {**

**char name[50];**

**int row;**

**int col;**

**struct Booking\* next;**

**} Booking;**

**Booking\* head = NULL;**

**// Function to display the seat layout**

**void displaySeats() {**

**printf("\nSeat Layout (0 = available, 1 = booked):\n");**

**for (int i = 0; i < ROWS; i++) {**

**for (int j = 0; j < COLS; j++) {**

**printf("%d ", seatLayout[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**// Function to create a new booking**

**void bookSeat() {**

**char name[50];**

**int row, col;**

**printf("Enter your name: ");**

**scanf("%s", name);**

**printf("Enter seat row (0-%d): ", ROWS - 1);**

**scanf("%d", &row);**

**printf("Enter seat col (0-%d): ", COLS - 1);**

**scanf("%d", &col);**

**if (row < 0 || row >= ROWS || col < 0 || col >= COLS) {**

**printf("Invalid seat position!\n");**

**return;**

**}**

**if (seatLayout[row][col] == 1) {**

**printf("Seat already booked!\n");**

**return;**

**}**

**// Mark seat as booked**

**seatLayout[row][col] = 1;**

**// Add booking to linked list**

**Booking\* newBooking = (Booking\*)malloc(sizeof(Booking));**

**strcpy(newBooking->name, name);**

**newBooking->row = row;**

**newBooking->col = col;**

**newBooking->next = head;**

**head = newBooking;**

**printf("Booking successful for %s at (%d, %d)\n", name, row, col);**

**}**

**// Function to display all bookings**

**void displayBookings() {**

**Booking\* temp = head;**

**printf("\nCurrent Bookings:\n");**

**while (temp != NULL) {**

**printf("Name: %s, Seat: (%d, %d)\n", temp->name, temp->row, temp->col);**

**temp = temp->next;**

**}**

**}**

**// Main function with menu**

**int main() {**

**int choice;**

**do {**

**printf("\n--- Movie Ticket Booking System ---\n");**

**printf("1. View Seats\n");**

**printf("2. Book Seat\n");**

**printf("3. View Bookings\n");**

**printf("4. Exit\n");**

**printf("Enter choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1: displaySeats(); break;**

**case 2: bookSeat(); break;**

**case 3: displayBookings(); break;**

**case 4: printf("Goodbye!\n"); break;**

**default: printf("Invalid choice.\n");**

**}**

**} while (choice != 4);**

**return 0;**

**}**

**CHAPTER 8 : Conclusion**

The Movie Ticket Booking System successfully streamlines the process of selecting movies, choosing seats, and purchasing tickets online, offering users a convenient and efficient alternative to traditional booking methods. By integrating features such as real-time seat availability, secure payment processing, and user-friendly navigation, the system enhances the overall movie-going experience.

From an administrative perspective, it simplifies theater management by automating scheduling, reporting, and customer interactions. The system reduces human error, saves time, and increases customer satisfaction. In conclusion, this project demonstrates the practical benefits of digital transformation in the entertainment industry, providing a scalable and adaptable solution for modern cinemas.