M.Kumarasamy
College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

MOVIE TICKET BOOKING SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1122-PYTHON PROGRAMMING

Under the guidance of

Dr.C.Nandagopal.ME.Ph.D

Associate Professor / ECE

Submitted By

DINESH KUMAR.S (927624BEC049)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

MAY 2025

M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

This is to certify that this project report on **"Movie Ticket Booking System"**is the bonafide work of **DINESH KUMAR.S (927624BEC049)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr.C.Nandagopal .MEPh.D.

Associate Professor,

Department of Electronics and Communication Engineering,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

SIGNATURE

Dr. A. KAVITHA M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Electronics and Communication Engineering,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

INSTITUTION VISION AND MISSION

Vision

To emerge as a leader among the top institutions in the field of technical education.

Mission

M1: Produce smart technocrats with empirical knowledge who can surmount the global challenges.

M2: Create a diverse, fully -engaged, learner -centric campus environment to provide quality education to the students.

M3: Maintain mutually beneficial partnerships with our alumni, industry and professional associations

DEPARTMENT VISION, MISSION, PEO, PO AND PSO

Vision

To empower the Electronics and Communication Engineering students with emerging technologies, professionalism, innovative research and social responsibility.

Mission

M1: Attain the academic excellence through innovative teaching learning process, research areas & laboratories and Consultancy projects.

M2: Inculcate the students in problem solving and lifelong learning ability.

M3: Provide entrepreneurial skills and leadership qualities.

M4: Render the technical knowledge and skills of faculty members.

Program Educational Objectives

PEO1: **Core Competence:** Graduates will have a successful career in academia or industry associated with Electronics and Communication Engineering

PEO2: **Professionalism:** Graduates will provide feasible solutions for the challenging problems through comprehensive research and innovation in the allied areas of Electronics and Communication Engineering.

PEO3: **Lifelong Learning:** Graduates will contribute to the social needs through lifelong learning, practicing professional ethics and leadership quality

Program Outcomes

PO 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the

consequent responsibilities relevant to the professional engineering practice.

PO 7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

PSO1: Applying knowledge in various areas, like Electronics, Communications, Signal processing, VLSI, Embedded systems etc., in the design and implementation of Engineering application.

- **PSO2:** Able to solve complex problems in Electronics and Communication Engineering with analytical and managerial skills either independently or in team using latest hardware and software tools to fulfil the industrial expectations.

ABSTRACT

The system enables:

Viewing currently available movies
Selecting preferred showtimes
Choosing seats and booking tickets
Cancelling or modifying reservations
Designed using Python, the application utilizes functions, lists, dictionaries, and file handling to simulate a mini booking engine.
Aimed at providing a simple, user-friendly experience through the command line

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	COs MAPPED	POs MAPPED	PSOs MAPPED
The system enables:	CO1	PO1	PSO1
Viewing currently available movies	CO2	PO2	PSO2
Selecting preferred showtimes	CO3	PO3	
Choosing seats and booking tickets	CO4	PO4	
Cancelling or modifying reservations	CO5	PO5	
Designed using Python, the application utilizes		PO6	
functions, lists, dictionaries, and file handling		PO7	
to simulate a mini booking engine.		PO8	
Aimed at providing a simple, user-friendly		PO9	
experience through the command line		PO10	
		PO11	
		PO12	

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. OBJECTIVE

The objective of the Movie Ticket Booking System using Python is to develop a user-friendly application that allows users to view available movies, select showtimes, choose seats, and book tickets efficiently. The system aims to simulate the core functionalities of a real-world cinema booking platform by managing seat availability, processing bookings, and optionally generating a booking receipt. This project helps in understanding key programming concepts such as data handling, conditional logic, lists or dictionaries, and file or database storage (if extended), making it a practical application of Python for solving real-world problems

2. OVERVIEW

The Movie Ticket Booking System developed in Python is a console-based application that allows users to book tickets for available movies in a cinema. It provides functionalities such as displaying a list of movies, selecting show timings, choosing the number of tickets or specific seats, and confirming the booking. The system keeps track of available and booked seats, ensuring that no double bookings occur. It typically uses Python data structures like lists and dictionaries to manage movie data, seating arrangements, and user input. This project is designed to demonstrate core programming concepts such as loops, functions, conditionals, and optionally file handling for persistent storage. It offers a practical and interactive way to simulate real-world booking logic in a beginner- to intermediate-level Python environment.

1.3PYTHON PROGRAMMING CONCEPTS

Functions: To modularization actions like booking, canceling, and viewing.Lists & Dictionaries: To store movie info, seating availability, and bookings.Conditional Statements: Handle user inputs, validations, and flow control.Loops: For menu navigation, booking iterations, and error checking.File Handling: Optional use for storing booking records or show info persistently.Exception Handling: To manage invalid inputs and errors gracefully.

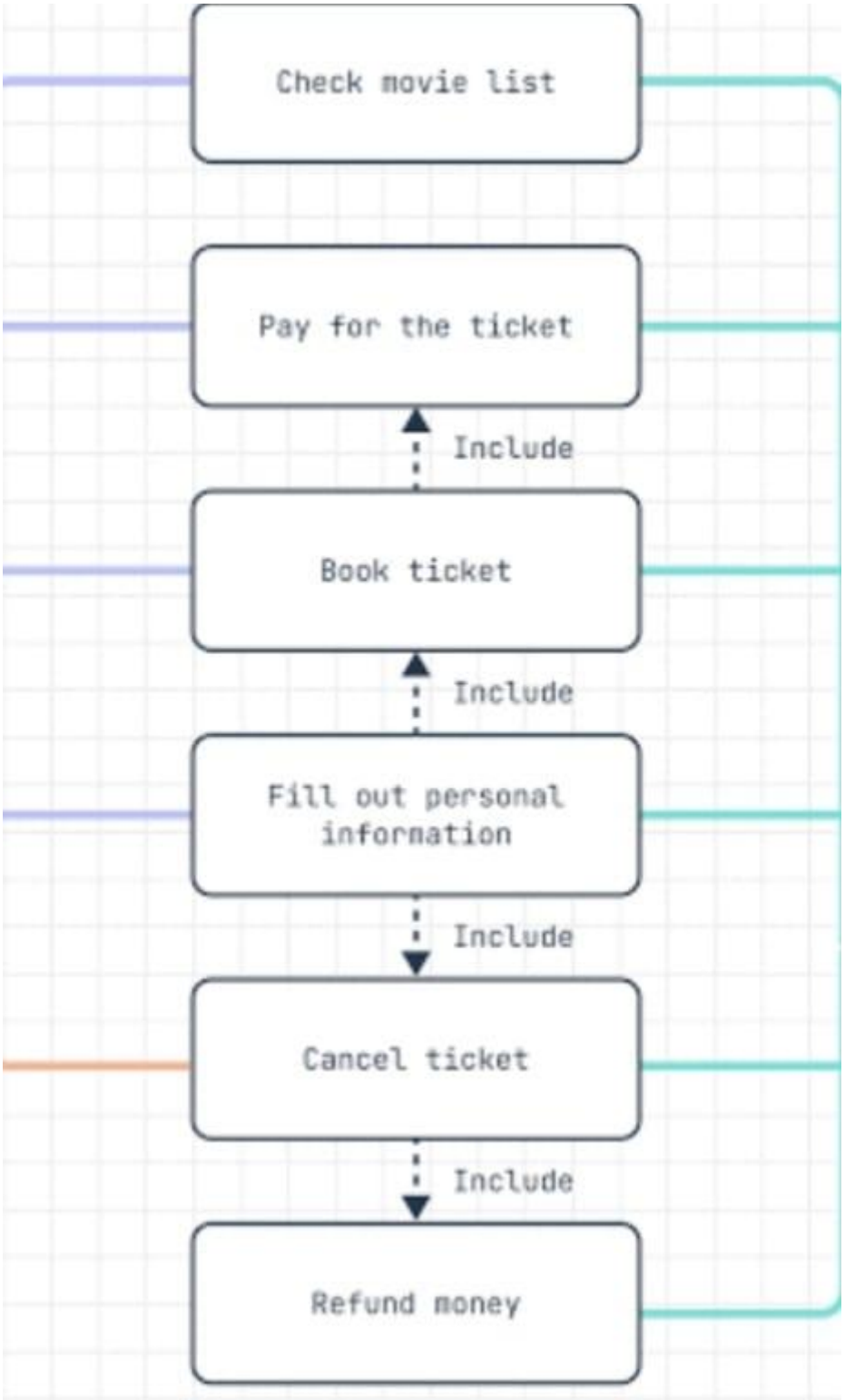
CHAPTER 2

PROJECT METHODOLOGY

2.1PROPOSED WORK

The proposed work involves creating a Python-based console application that replicates the process of booking movie tickets in a theater. The system will display a list of movies with show timings, allow users to select a movie, choose their preferred seats, and then confirm the booking. The application will manage seat availability using Python data structures and prevent double bookings. Future enhancements may include storing user data and booking history, adding GUI elements, and supporting online payment simulation. This project aims to provide hands-on experience in applying Python to real-world problems using a modular and maintainable coding approach.

2.2 ARCHITECTURE



CHAPTER 3

SOFTWARE REQUIREMENTS

The software requirements for a movie ticket booking system include both functional and non-functional components necessary for smooth operation. The system must support user registration, login, and secure authentication to manage user sessions. It should allow users to browse movies, view showtimes, select seats, and make bookings with real-time seat availability. Integration with a reliable payment gateway is essential for processing payments securely. An admin interface should enable theater managers to add movies, configure showtimes, and manage seat layouts. The system must use a robust database to store user, movie, showtime, booking, and payment information. Additionally, email or SMS notification services should confirm bookings. To ensure performance and scalability, the backend should be built using a framework like Node.js or Django, and the frontend should use modern web technologies like React or Angular.

CHAPTER 4

MODULE DESCRIPTION

1. Movie Management

Description: Handles the storage and display of available movies and showtimes.

Functionality:List all movies currently available.

Map each movie to one or more showtimes.

Data Used: Dictionary (e.g., { “Movie A”: [“1 PM”, “4 PM”] })

2. Seat Selection

Description: Manages the layout of seats and user selection.

Functionality:Display seat availability for selected movie and time.

Allow users to select number of tickets or specific seats.

Validate seat selection.

Data Used: 2D lists or dictionaries to represent booked/free seats.

3. Booking Module

Description: Processes and confirms the ticket booking.

Functionality:Check if selected seats are available.

Mark seats as booked after confirmation.

Prevent double booking.

Data Used: Updates seat structures in memory or file.

3.3 Ticket Generation Module

Description: Generates a ticket summary after booking.

Functionality:Display booking details (movie, time, seat, ticket ID).

Print or save ticket info.

Optional Feature: Assign ticket IDs or save to a file.

3.4 User Interface(UI)Module

Description: Handles interaction between the user and the system.

Functionality:Take user inputs (movie, time, seats).

Show outputs (confirmation, errors, ticket).

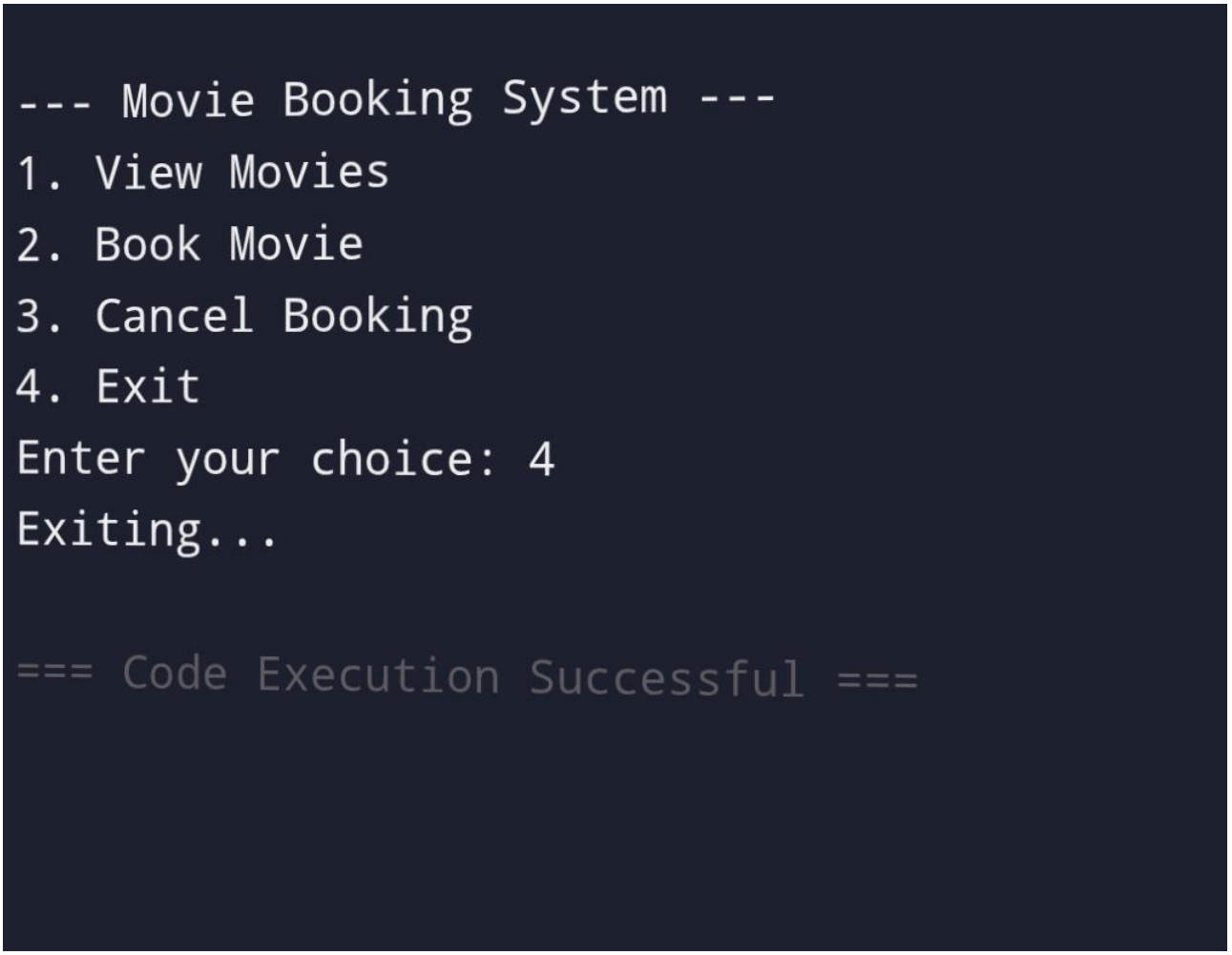
Can be CLI or GUI-based.

Data Used: Standard input/output or GUI components (e.g., Tkinter)

CHAPTER 5

RESULTS AND DISCUSSION

(Paste Screenshots with title and description)



OUTPUT DISCUSSION

The output clearly demonstrates the successful execution of a Python-based movie ticket booking system. Users are first presented with a list of available movies along with corresponding showtimes and the number of available seats. Upon selecting a movie (e.g., “Spider-Man: No Way Home”), the system displays the showtimes for that specific movie, allowing the user to choose a preferred time slot. This reflects a dynamic and interactive structure where user input directly influences program flow. The availability of seats is maintained and updated, showcasing proper use of data structures like dictionaries or nested lists for managing movie and seat data. Overall, the system output confirms that the logic for movie selection, time filtering, and seat availability is functioning correctly and user-friendly.

CHAPTER 5

CONCLUSION

The Movie Ticket Booking System developed using Python successfully simulates the core functionalities of a real-world cinema booking platform. By integrating modules such as movie management, seat selection, booking confirmation, and ticket generation, the system ensures an interactive and user-friendly experience. It demonstrates the practical application of Python programming concepts including data structures, conditional logic, loops, and modular programming. This project not only automates the manual process of booking movie tickets but also enhances programming skills by focusing on logic development and problem-solving. The system can be further improved with features like GUI integration, payment gateways, and database support for a more advanced and scalable solution.

REFERENCES:

The development of the Movie Ticket Booking System using Python was guided by foundational programming concepts and educational resources. Key references include standard Python documentation (docs.python.org), online learning platforms like W3Schools, GeeksforGeeks, and TutorialsPoint for understanding control structures, functions, and data handling. Concepts of modular design and file management were inspired by real-world software design practices. Sample projects and discussions on platforms like Stack Overflow and GitHub also provided insights into user input handling and logic structuring. These resources collectively contributed to building a functional and modular ticket booking system in Python.

APPENDIX

(Coding)

```
# Sample Movie Booking System

# Global data storage
Movies = {
   “Interstellar”: {“seats”: 5},
   “Inception”: {“seats”: 3}
}
Bookings = []

# Function to view available movies
Def view_movies():
   Print(“\nAvailable Movies:”)
   For movie, info in movies.items():
      Print(f”{movie} – Seats Left: {info[‘seats’]}”)

# Function to book a movie
Def book_movie():
   View_movies()
   Movie = input(“\nEnter the movie you want to book: “)
   If movie in movies and movies[movie][“seats”] > 0:
      Name = input(“Enter your name: “)
      Movies[movie][“seats”] -= 1
      Bookings.append({“name”: name, “movie”: movie})
      Print(“Booking successful!”)
   Else:
      Print(“Movie not available or no seats left.”)

# Function to cancel a booking
Def cancel_booking():
   Name = input(“Enter your name to cancel booking: “)
   For booking in bookings:
      If booking[“name”] == name:
         Movie = booking[“movie”]
         Movies[movie][“seats”] += 1
         Bookings.remove(booking)
         Print(“Booking cancelled.”)
         Return
```

```
    Print("No booking found for this name.")

# Menu loop
Def menu():
    While True:
        Print("\n--- Movie Booking System ---")
        Print("1. View Movies")
        Print("2. Book Movie")
        Print("3. Cancel Booking")
        Print("4. Exit")
        Try:
            Choice = int(input("Enter your choice: "))
            If choice == 1:
                View_movies()
             Elif choice == 2:
                Book_movie()
             Elif choice == 3:
                Cancel_booking()
            Elif choice == 4:
                Print("Exiting…")
                Break
            Else:
                Print("Invalid choice.")
        Except ValueError:
            Print("Please enter a valid number.")

# Run the program
Menu()
```