

A Course Based Project Report on
FLIGHT MANAGEMENT SYSTEM

Submitted to the

Department of Information Technology

in partial fulfillment of the requirements for the completion of course
DATA STRUCTURES LABORATORY (22ES2CS102)

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

RAM DAS	24071A1281
SATHWIK	24071A1282
PRANEETH	24071A1283
SATHWIK	24071A1284
NITHIN	24071A1285

Under the guidance of

Mrs. M. Susmitha

(Course Instructor)

Assistant Professor, Department of IT, VNRVJIET



DEPARTMENT OF INFORMATION TECHNOLOGY

**VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI
INSTITUTE OF ENGINEERING & TECHNOLOGY**

An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

May 2025

**VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI
INSTITUTE OF ENGINEERING AND TECHNOLOGY**

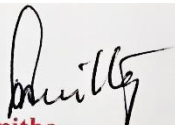
An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses, Approved by AICTE, New Delhi, Affiliated to JNTUH, Recognized as "College with Potential for Excellence" by UGC, ISO 9001:2015 Certified, QS I GUAGE Diamond Rated
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

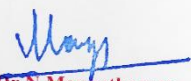
DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project report entitled "**Flight Management System**" is a bonafide work done under our supervision and is being submitted by **MR. RAMDAS (24071A1281), MR. D.SATHWIK (24071A1282), MR. PRANEETH (24071A1283), MR. G.SATHWIK (24071A1284), MR. NITHIN (24071A1285)** in partial fulfilment for the award of the degree of **Bachelor of Technology** in Information Technology, of the VNRVJIET, Hyderabad during the academic year 2024-2025.


M.Susmitha
Assistant Professor, IT


Dr N Mangathayaru
Professor &HOD
Department of IT


Course based Projects Reviewer

**VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI
INSTITUTE OF ENGINEERING AND TECHNOLOGY**

An Autonomous Institute, NAAC Accredited with 'A++' Grade,
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

DEPARTMENT OF INFORMATION TECHNOLOGY



DECLARATION

We declare that the course based project work entitled “**FLIGHT MANAGEMENT SYSTEM**” submitted in the Department of Information Technology, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in Information Technology** is a bonafide record of our own work carried out under the supervision of **M.Susmitha , Assistant Professor, Department of IT, VNRVJiet**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad.

D.RAMDAS

24071A1281

D.SATHWIK

24071A1282

G.PRANEETH

24071A1283

G.SATHWIK

24071A1284

G.NITHIN

24071A1285

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our beloved President, Sri. D. Suresh Babu, VNR Vignana Jyothi Institute of Engineering & Technology for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we record our deep sense of gratitude to our beloved Principal, Dr. C.D Naidu, for permitting us to carry out this project.

We express our deep sense of gratitude to our beloved Professor **Dr N Mangathayaru**, Associate Professor and Head, Department of Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad-500090 for the valuable guidance and suggestions, keen interest and through encouragement extended throughout the period of project work.

We take immense pleasure to express our deep sense of gratitude to our beloved Guide, **M.Susmitha**, Assistant Professor in Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad, for his/her valuable suggestions and rare insights, for constant source of encouragement and inspiration throughout my project work.

We express our thanks to all those who contributed for the successful completion of our project work.

MR.D.RAMDAS	(24071A1281)
MR.D.SATHWIK	(24071A1282)
MR.PRANEETH	(24071A1283)
MR.G.SATHWIK	(24071A1284)
MR.G. NTHIN	(24071A1285)

TABLE OF CONTENTS

S.NO.	CONTENTS	PAGE NO.
1.	ABSTRACT	2
2.	INTRODUCTION	3-4
3.	SOURCE CODE	5-9
4.	TEST CASES / OUTPUT	10-11
5.	CONCLUSIONS	12
6.	REFERENCES	13

ABSTRACT

This C program implements a basic flight and passenger management system using the concepts of structures and singly linked lists. The primary purpose of the system is to allow users to add flights, book passengers onto specific flights, and view all flights along with their passenger details. Each flight is represented by a structure that holds the flight number, source city, destination city, and a linked list of passengers associated with that flight. Similarly, each passenger is represented by a structure containing the passenger's name, source, and destination. The program ensures data consistency by validating that the source and destination entered while booking a passenger must match the route of the flight. If the route does not match, the passenger cannot be added, which simulates real-world flight booking constraints. The use of dynamic memory allocation and linked lists allows the program to handle an arbitrary number of flights and passengers without pre-defining limits, making the system flexible and memory efficient.

INTRODUCTION

1.1 PROBLEM DEFINITION

Create A Linked List Program for Flight Management System

1.2 OBJECTIVE

- **Flight Addition**

Allow users to add new flights by entering a unique flight number along with source and destination cities. Each flight is stored in a linked list for efficient management and future retrieval.

- **Passenger Booking**

Enable users to book passengers by associating them with an existing flight. Each passenger's name, source, and destination are stored in a linked list under the corresponding flight node.

- **Display Flight and Passenger Details**

Provide functionality to display all flights along with their complete details and a list of passengers booked on each flight.

- **Dynamic Memory Management**

Utilize `malloc` for dynamic memory allocation to manage flights and passengers. This ensures memory is only used when necessary, allowing the system to grow without predefined limits.

- **Linked List Implementation**

Use singly linked lists to model the one-to-many relationship between flights and their passengers. This allows flexible insertion, deletion, and traversal as the data grows.

- **Simple Menu-Driven Interface**

Offer a user-friendly, text-based interface that guides users through different operations such as adding flights, booking passengers, and viewing flight details.

- **Basic Error Handling and Validation**

Implement checks to ensure that passengers can only be booked on existing flights, and that the passenger's travel route matches the flight's defined route. Users are notified if invalid input is provided.

- **Memory Efficiency**

Ensure that memory allocated for flights and passengers is used efficiently and safely. While the current version dynamically allocates memory, future versions may also include proper memory deallocation.

- **Extendability for Future Features**

Design the program structure in a modular way, making it easy to add new features like deleting flights or passengers, searching flights, sorting data, or storing flight and passenger records in files.

SOURCE CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Passenger structure

struct Passenger {

    char name[50];

    char source[50];

    char destination[50];

    struct Passenger* next;

};


// Flight structure

struct Flight {

    char flightNumber[10];

    char source[50];

    char destination[50];

    struct Passenger* passengers;

    struct Flight* next;

};


struct Flight* flightHead = NULL;
```

```

// Add new flight

void addFlight() {

    struct Flight* newFlight = (struct Flight*)malloc(sizeof(struct Flight));

    printf("Enter flight number: ");

    scanf("%s", newFlight->flightNumber);

    printf("Enter flight source: ");

    scanf("%s", newFlight->source);

    printf("Enter flight destination: ");

    scanf("%s", newFlight->destination);

    newFlight->passengers = NULL;

    newFlight->next = flightHead;

    flightHead = newFlight;

    printf("Flight added.\n");

}

```

```

// Book a passenger

void bookPassenger() {

    char flightNum[10];

    char name[50];

    char src[50], dest[50];

    printf("Enter flight number: ");

    scanf("%s", flightNum);

    struct Flight* temp = flightHead;

```

```

while (temp != NULL && strcmp(temp->flightNumber, flightNum) != 0) {

    temp = temp->next;

}

if (temp == NULL) {

    printf("Flight not found.\n");

    return;

}

printf("Enter passenger name: ");

scanf("%s", name);

printf("Enter passenger source: ");

scanf("%s", src);

printf("Enter passenger destination: ");

scanf("%s", dest);

// Validate passenger's route is within the flight route

if (strcmp(src, temp->source) != 0 || strcmp(dest, temp->destination) != 0) {

    printf("Passenger route must match the flight's source and destination (%s to
%s).\n", temp->source, temp->destination);

    return;

}

struct Passenger* newPass = (struct Passenger*)malloc(sizeof(struct Passenger));

strcpy(newPass->name, name);

```

```

strcpy(newPass->source, src);

strcpy(newPass->destination, dest);

newPass->next = temp->passengers;

temp->passengers = newPass;

printf("Passenger added.\n");
}

// Show all flights and passengers

void showFlights() {

    struct Flight* f = flightHead;

    if (f == NULL) {

        printf("No flights added yet.\n");

        return;

    }

    while (f != NULL) {

        printf("\nFlight: %s\n", f->flightNumber);

        printf("Route: %s -> %s\n", f->source, f->destination);

        struct Passenger* p = f->passengers;

        if (p == NULL) {

            printf(" No passengers.\n");

        } else {

            printf(" Passengers:\n");

            while (p != NULL) {

```

```

        printf(" - %s (From %s to %s)\n", p->name, p->source, p->destination);

        p = p->next;

    }

}

f = f->next;

}

}

int main() {

    int choice;

    while (1) {

        printf("\n1. Add Flight\n2. Book Passenger\n3. Show Flights\n0. Exit\n");

        printf("Choose: ");

        scanf("%d", &choice);

        if (choice == 1) addFlight();

        else if (choice == 2) bookPassenger();

        else if (choice == 3) showFlights();

        else if (choice == 0) break;

        else printf("Invalid choice.\n");

    }

    return 0; }

```

TEST CASES/ OUTPUT

3.1 Test case 1:

```
1. Add Flight
2. Book Passenger
3. Show Flights
0. Exit
Choose: 1
Enter flight number: 22E
Enter flight source: DEL
Enter flight destination: BLR
Flight added.

1. Add Flight
2. Book Passenger
3. Show Flights
0. Exit
Choose: 2
Enter flight number: 22E
Enter passenger name: SATHWIK
Enter passenger source: DEL
Enter passenger destination: BLR
Passenger added.
```

3.2 Test case 2:

```
1. Add Flight
2. Book Passenger
3. Show Flights
0. Exit
Choose: 2
Enter flight number: 22E
Enter passenger name: KIRAN
Enter passenger source: DEL
Enter passenger destination: HYD
Passenger route must match the flight's source and destination (DEL to BLR
).
```

3.3 Test case 3:

```
1. Add Flight
2. Book Passenger
3. Show Flights
0. Exit
Choose: 3

Flight: 22E
Route: DEL -> BLR
  Passengers:
    - SATHWIK (From DEL to BLR)

1. Add Flight
2. Book Passenger
3. Show Flights
0. Exit
Choose: 0

=== Code Execution Successful ===
```

CONCLUSION

The Flight and Passenger Management System utilized in C is a simple but efficient method of managing flight bookings employing structures and linked lists. The program supports the dynamic addition of flights, passengers, and the display of flight and passenger details for each flight. By employing linked lists, the system promotes flexibility and scalability such that an arbitrary number of flights and passengers without size restrictions are permitted.

Some of the key features like route validation, dynamic memory allocation, and a menu-driven simple interface make the program easy to use and instructive for learning fundamental programming concepts. Moreover, by bifurcating data from logic through structures, the program has clean, neat code, which is an excellent foundation for incorporating future additions like data persistence, deletion operations, or more complex search capabilities.

Overall, this project illustrates basic data structure manipulation in C and is a good stepping stone for constructing more intricate data management systems.

REFERENCES

- [1]. International Journal of Computer Mathematics
- [2]. "Data Structures Using C" by Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidiah
Langsam
- [3]. "Programming in C" by Stephen G. Kochan
- [4]. Journal of Computer Science and Technology
- [5]. <https://www.geeksforgeeks.org/courses/dsa-self-paced>