**Project Report:** **Ticket Booking System using Singly Linked List**

**Introduction:**

The ticket booking system is a command-line application written in the C language and is used to book, cancel, perform, search and count tickets. This system uses single linked lists to store ticket information (such as ticket ID, name and seat number). The program allows users to interact with the system through a simple menu-powered interface.

**Objective:**

The main objectives of this project are:

* To implement a basic ticket booking system using single linked lists.
* To allow users to operate tickets such as booking, canceling and displaying tickets.
* To provide facilities like finding and counting the number of tickets booked.

**Facilities:**

The program includes the following characteristics:

* Ticket booking: A new ticket is added to the system that contains a unique ID, name and seat number.
* Ticket cancellation: The user can cancel the ticket by providing ticket ID.
* Ticket Performance: All the tickets currently booked in the system are displayed.
* Ticket Search: User can find tickets by ticket ID.
* Ticket count: The total number of tickets booked is displayed.

**System design:**

The system is designed to store and manage ticket data using a single linked list. Each ticket is shown as a node in the linked list, the composition of which is as follows:

struct Ticket {

int id;

char name[50];

int seat;

struct Ticket \*next;

};

The linked list allows for dynamic joints and deletion of tickets, which is useful in real -time ticket management.

**Program work:**

**bookTicket ()**

* Purpose: Add a new ticket to the list.
* Input: Ticket ID, Customer Name, Seat Number.
* Output: The ticket is added to the linked list and a confirmation message is displayed.

**cancelTicket ()**

* Objective: Cancel a ticket, remove it from the linked list.
* Input: Ticket ID for cancellation.
* Output: The ticket is removed from the linked list and a confirmation message is displayed.

**displayTickets ()**

* Objective: Display all tickets in the linked list.
* Output: ID, name and seat number of all booked tickets are displayed.

**searchTicket ()**

* Objective: Search for tickets by ticket ID.
* Input: to search for ticket ID.
* Output: If the ticket is found, its location comes back, otherwise a message is displayed that the ticket was not found.

**countTickets ()**

* Objective: Calculate the total number of booked tickets.
* Output: The total number of tickets is displayed in the linked list.

**Flowchart:**

The main flow of the following Flowchart program has a simple diagram:

Start -> Display menu -> user choice

-> Option 1: Book ticket -> Add tickets -> Display confirmation

-> Option 2: Cancel tickets -> Remove tickets -> Display confirmation

-> Option 3: Display tickets -> Print all tickets

-> Option 4: Search Ticket -> Display Position or not found

-> Option 5: Counting Ticket -> Show Total Tickets

-> Options 6: Exit -> End

**Code clarification:**

The code begins by defining the structure for a ticket. This includes the function prototype for various activities of the program. The main function runs a menu-powered loop that allows users to choose and perform various activities. Each activity is associated with a specific function that manipulates the linked list and does the necessary activity.

* **Ticket Booking:** bookTicket () function connects a new node at the end of the linked list.
* **Ticket cancellation:** cancelTicket () function searches for a ticket ID and removes it from the linked list.
* **Ticket Performance:** displayTickets () function detect the list and display all tickets.
* **Ticket Search:** searchTicket () function discovers tickets by ticket ID and returns its place.
* **Ticket count:** countTickets () function count the number of tickets in the linked list.

**Sample input and output:**

***Sample input:***

1. Book ticket

Enter ticket ID: 101

Enter the name: Sohan

Enter seat number: 12

***Sample output:***

Tickets booked: ID = 101, Name = Sohan, Seat = 12

***Sample input:***

3. Display tickets

***Sample output:***

Book tickets:

ID: 101, Name: Sohan, Seat: 12

***Sample input:***

5. Count tickets

***Sample output:***

Total tickets: 1

**Boundaries and Improvements Boundaries:**

***Boundaries***:

* The program does not handle input verification (for example, check whether the ticket ID is already present or whether the seat number is correct).
* The implementation of a linked list does not allow random access to make it disabled for large datasets.

***Improvements Boundaries***:

* Apply input verification to seat number, ticket ID and customer names.
* Use dynamic memory allocation more efficiently and check for memory allocation failures.
* Improve the user interface, allowing it to be more interactive and user friendly.
* Apply a feature that allows a booked ticket to be modified (eg. changing seat numbers).

**Conclusion:**  
The ticket booking system is a basic, functional command-line application that implements the ticket booking system using single linked lists. The project provides a good example of using linked lists to manage dynamic data structures. While space for improvement, the system offers a strong foundation