Department of Information Technology

RAILWAY RESERVATION USING C

By:
K. Padma Priyanka
P. S. L. S. Chandana
G. V. L. Spoorthi

CONTENTS

Introduction
Objective of the project
Data Structure
System requirements
Implementation modules
Results
Advantages
Future Enhancements
References

OBJECTIVE OF THE PROJECT

All the manual work should be converted into computerized so that the load of employees should decrease.

The data should be stored in computer rather than in register manually.

Booking can be done by sitting at your home only, no need to visit the booking counter.

Introduction

This system is basically concerned with the reservation of railway tickets to the passengers.

In this we are discussing that how the reservation is done with the feature of cancelling and waiting list.

In the project we are going to include entities like
Reservation
Cancellation
Display reserved and waiting list passengers.

Data Structure

Singly Linked List

Linked List is a sequential collection of nodes. Which is faster than array in terms of deletion of nodes. It's memory is dynamically allocated in runtime. This saves time and space.

Each node consists of four different data field :

#Name

#Age

#Registration Number

#Link to the next node

Data Structure

Queue

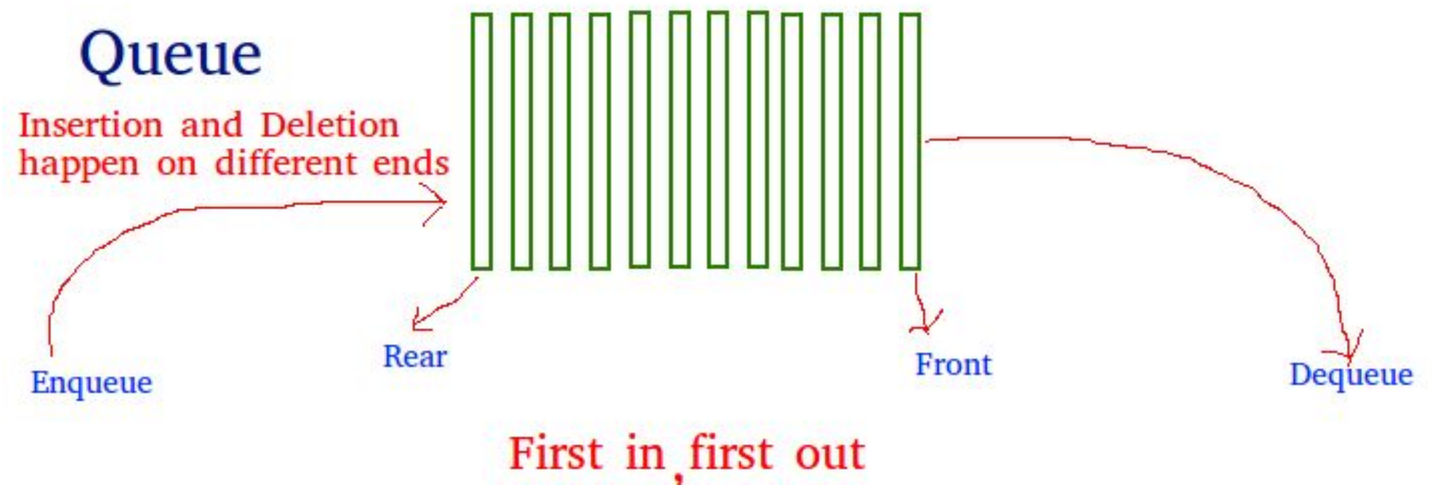
Queue is a data structure in which insertion and deletion takes place from the ends. It follows First In First Out Principle.

Queue data structure is used here to store the waiting list passengers. If anyone cancels their ticket then that seat is allocated to the first passenger in the queue.

Data Structure

LINEAR QUEUE

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO) .In a queue, we remove the item the least recently added.



SYSTEM REQUIREMENTS

SOFTWARE SPECIFICATION

• Operating System	:	OSx
• Frontend	:	C programming
• Backend	:	C programming
• IDE	:	Visual Studio Code

Implementation modules

```
node* deq();
int create();
int reserve(node*);
int cancel(int);
void enq(node*);
void display();
```

Implementation modules

```
typedef struct NODE
{
    int reg_no;
    int age;
    char name[20];
    struct NODE *next;
} node;
```

Implementation modules

```
int create( )
{
        node *new;
        new=(node *)malloc(sizeof(node));
        new->reg_no=1;
        printf("Name: ");
        scanf("%s", new->name);
        printf("Age : ");
        scanf("%d", &new->age);
    if(new->age>=90 || new->age<=10) {
        free(new);
        return -1;
    }
        start=new;
        new->next=NULL;
        num++;
    return 1;

}
```

Implementation modules

```
int reserve(node *start){
        int temp;
        if(start==NULL){
                  temp = create(start);
                   return temp;
        }
        else {
        node *temp, *new_node;
        temp=start;
        while(temp->next!=NULL){
          temp=temp->next;
        }
        new_node=(node *)malloc(sizeof(node));
        printf("Name: ");
        scanf("%s", new_node->name);
        printf("Age : ");
scanf("%d", &new_node->age);
if(new_node->age >=90 ||
      new_node>age<=10) {
        return -1;
    }
new_node->next=NULL;
        if(num<=size){
                  num++;
                  new_node->reg_no=num;
                  temp->next=new_node;
                  return 1;
        }
        else{
                  enq(new_node);
                  return 0;
        }
}
}
```

Implementation modules

```
int cancel(int reg)
{
    node *ptr, *preptr, *new;
    ptr=start;
    preptr=NULL;
    if(start==NULL)
    return -1;
    if(ptr->next==NULL &&
   ptr->reg_no==reg)
        {
        start=NULL;
        num--;
        free(ptr);
        return 1;
        }
    else{
while(ptr->reg_no!=reg && ptr->next!=NULL)
        {
            preptr=ptr;
            ptr=ptr->next;
        }
    if(ptr==NULL && ptr->reg_no!=reg)
        return -1;
        else
        preptr->next=ptr->next;
        free(ptr);
        new=deq();
        while(preptr->next!=NULL)
            preptr=preptr->next;
        preptr->next=new;
        num--;
        return 1;
    }
}
```

Implementation modules

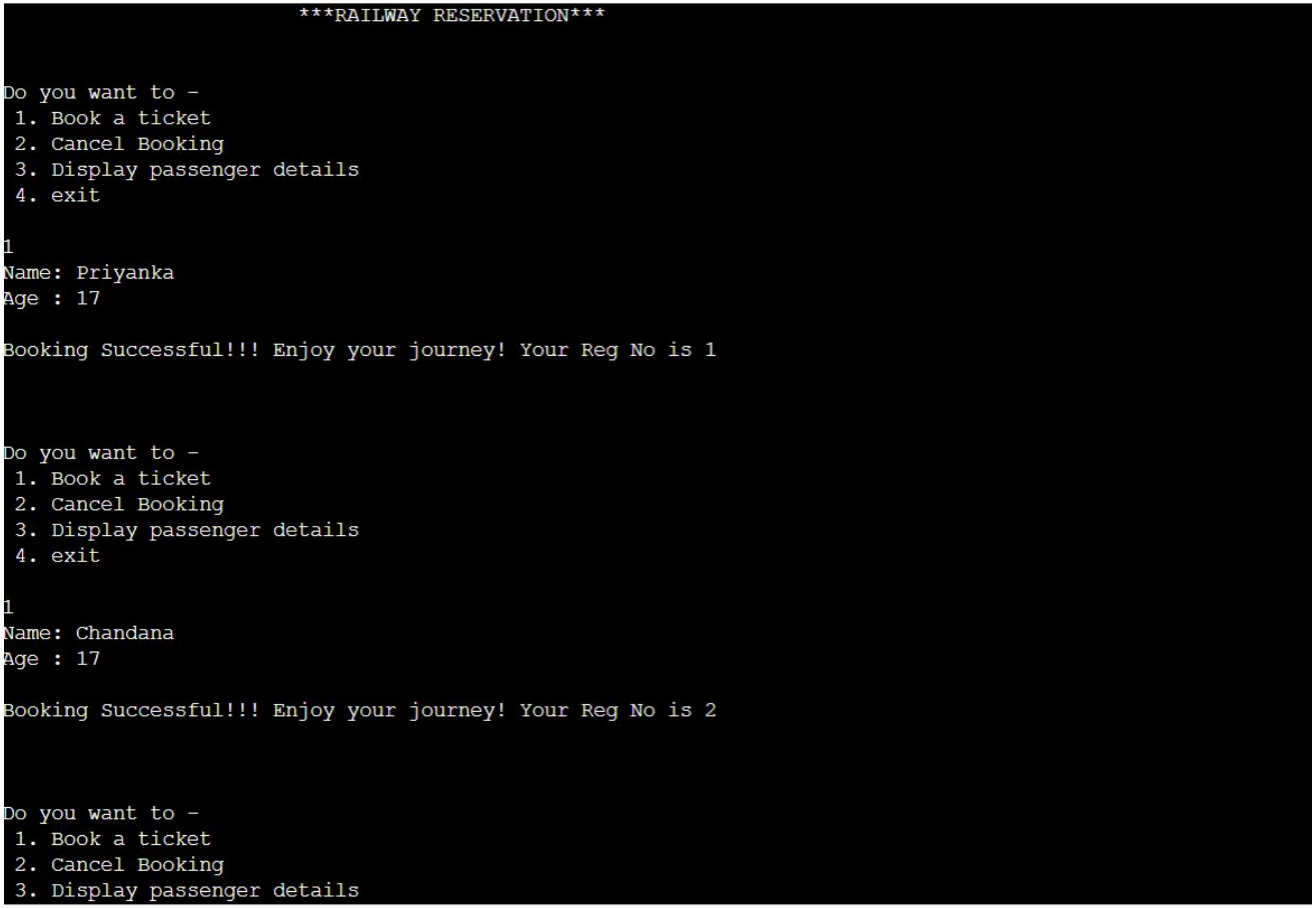
```
void enq(node *new_node)
{
    if(rear==NULL)
    {
        rear=new_node;
        rear->next=NULL;
        front=rear;
    }
    else
    {
        node *temp;
        temp=new_node;
        rear->next=temp;
        temp->next=NULL;
        rear=temp;
    }
    count++;
}
```

```
node* deq(){
    node *front1;
    front1=front;
    if(front==NULL)
        return NULL;
    else{
      count-- ;
        if(front->next!=NULL){
            front=front->next;
            front1->next=NULL;
            return front1;
        }
        else{
            front=NULL;
            rear=NULL;
            return front1;
}}}
```

Implementation modules

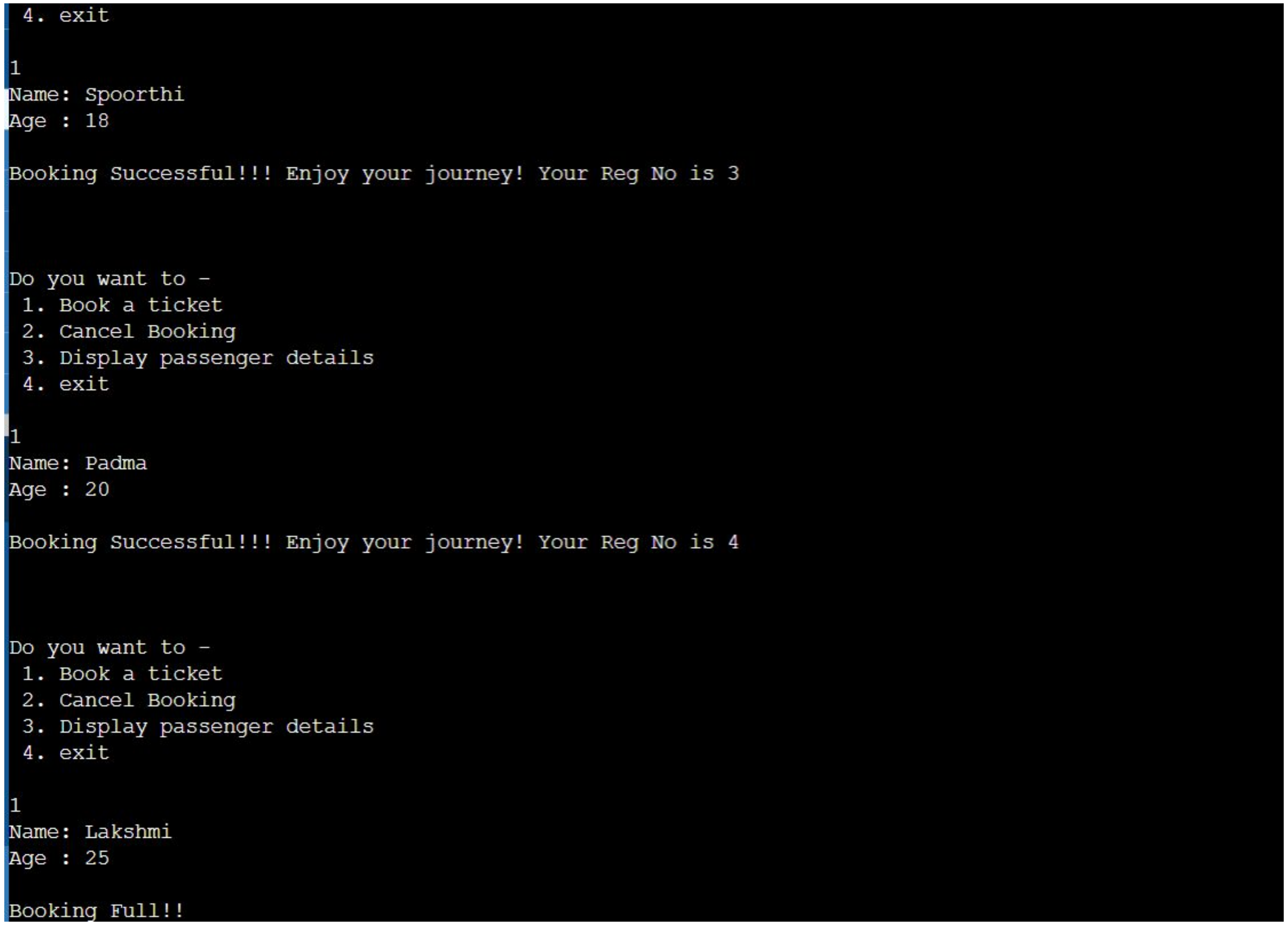
```
void display()
{
        node *temp;
        temp=start;
        while(temp!=NULL)
        {
                printf("\nRegistration Number:
      %d\n", temp->reg_no);
                printf("Name : %s\n\n",
      temp->name);
                temp=temp->next;
    }

}
```

Results



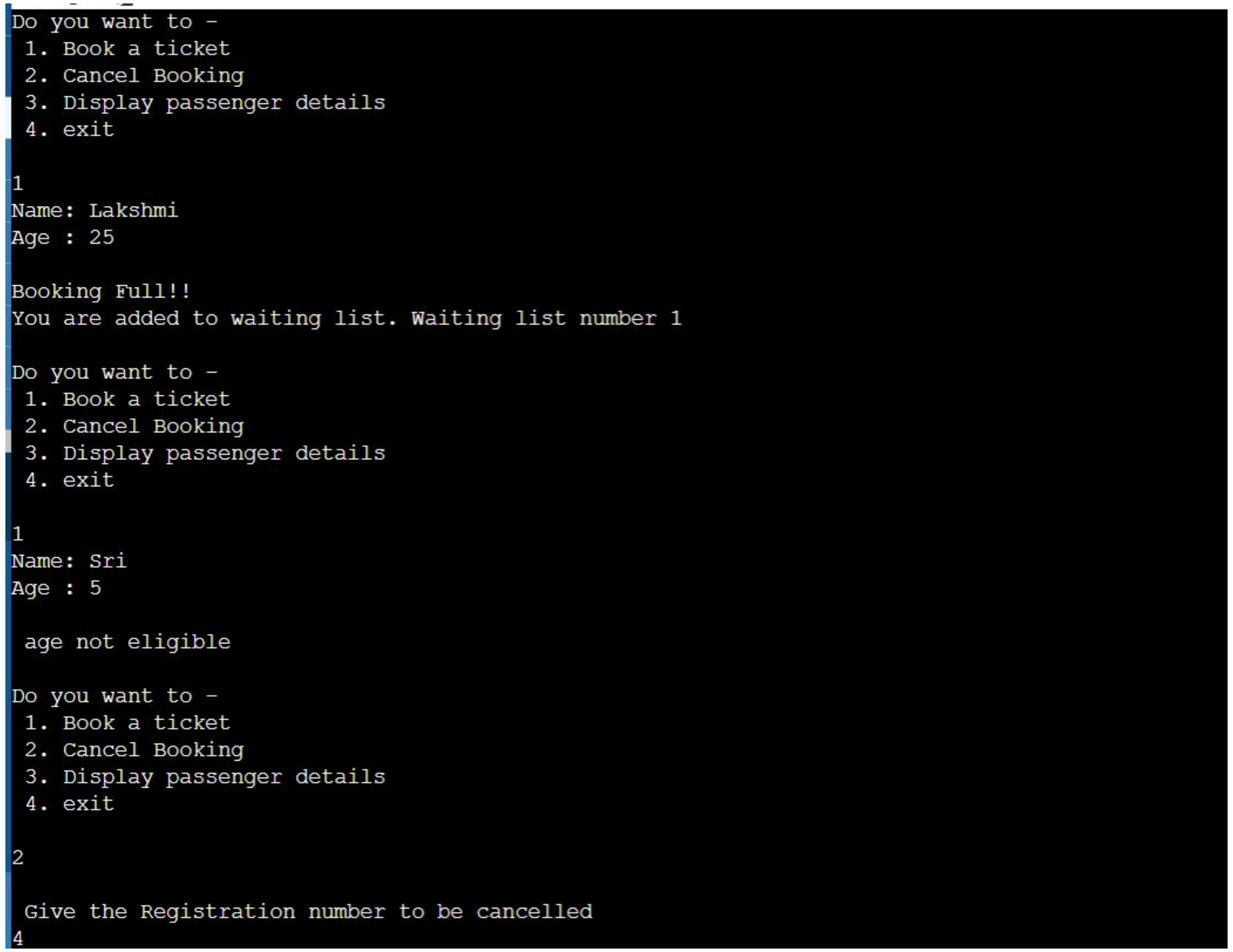
```
                        ***RAILWAY RESERVATION***


Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

1
Name: Priyanka
Age : 17

Booking Successful!!! Enjoy your journey! Your Reg No is 1



Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

1
Name: Chandana
Age : 17

Booking Successful!!! Enjoy your journey! Your Reg No is 2



Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
```

Results



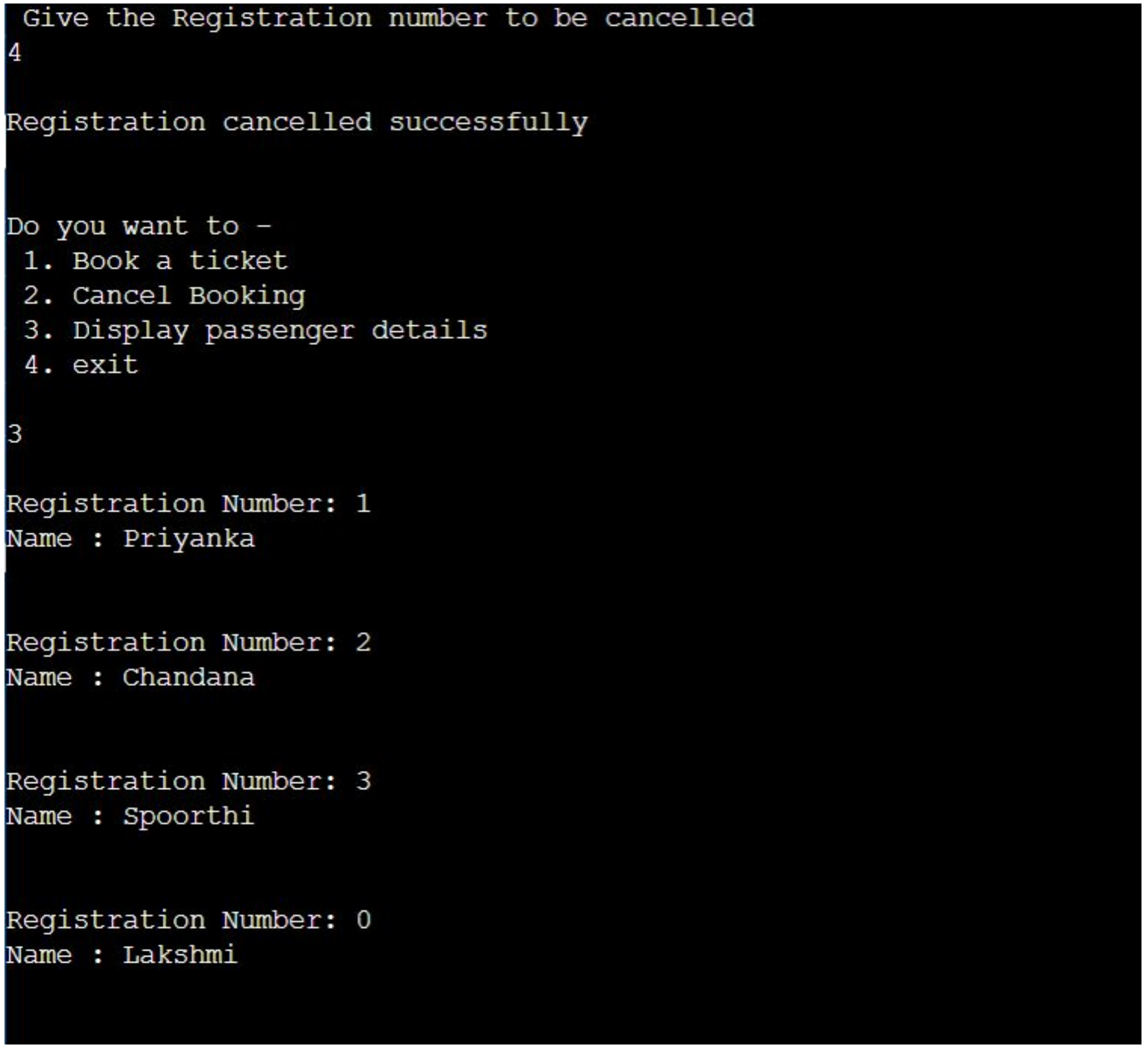
```
 4. exit

1
Name: Spoorthi
Age : 18

Booking Successful!!! Enjoy your journey! Your Reg No is 3



Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

1
Name: Padma
Age : 20

Booking Successful!!! Enjoy your journey! Your Reg No is 4



Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

1
Name: Lakshmi
Age : 25

Booking Full!!
```

Results



```
Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

1
Name: Lakshmi
Age : 25

Booking Full!!
You are added to waiting list. Waiting list number 1

Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

1
Name: Sri
Age : 5

 age not eligible

Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

2

 Give the Registration number to be cancelled
4
```

Results



```
 Give the Registration number to be cancelled
4

Registration cancelled successfully


Do you want to -
 1. Book a ticket
 2. Cancel Booking
 3. Display passenger details
 4. exit

3

Registration Number: 1
Name : Priyanka


Registration Number: 2
Name : Chandana


Registration Number: 3
Name : Spoorthi


Registration Number: 0
Name : Lakshmi
```

ADVANTAGES

Reduces the burden of traveler waiting in the booking counter.

User-friendly.

Convenient.

Time savings.

Future Enhancements

We can optimise our time complexity using some different data structure.

We can add features such as prioritising on the basis of age or railway employees and gender.

We can add feature of tatkal reservation.

We can provide this solution on online portal.

References

- www.youtube.com
- www.tutorialspoint.com
- www.greeksforgreek.org

THANK YOU