AirPort

Management

System

By

­

# CONTENTS

1. PROBLEM DEFENITION 1
2. PROBLEM ANALYSIS 2
3. HARDWARE AND SOFTWARE REQUIEMENTS 7
4. FUTURE ENHANCEMENTS 8
5. SOURCE CODE 9
6. OUTPUT 30
7. BIBLIOGRAPHY 36

# PROBLEM DEFENITION:

An airport is a very busy place which handles a number of passengers and planes on a daily basis. So, management systems play a vital role in handling planes.

In today’s world management systems play a vital role in all the major organisations. It helps in effective allocation and deallocation of a variety of things such as food counters, bus for staff, seats and appointment timings. It helps establish a link between various departments in an organisation.

An airport management system plays a similar role. It helps in managing the arrival and departure of planes. it ensures proper allocation of buses, baggage terminals, gates and food for each plane.

Take an example of an arrival at the airport. The management system allocates baggage terminals, gates and food for that particular plane. Similarly, for a departure the management system arranges for counters, gates and fuels for the plane.

The aim of our project is to create an interactive menu driven C++ program that functions as an airport management system. It involves the following specifications mentioned below in the project analysis.

**PROBLEM ANALYSIS:**

An airport is a very busy place which handles a number of passengers and planes on a daily basis. It is important to manage the available resources in the airport to use them effectively and avoid any possible collision between planes.

Typically airport need to handle planes from multiple airlines, their arrival, departure, fuelling, food, baggage, security and passenger amenities.

This program is designed with two modules, admin module and user module.

Purpose of admin module is to feed required data to system. This is a onetime activity where an administrator or group of administrators are tasked with adding/removing necessary data into the system. This module also handles system maintenance and update. It allows the administrator to view details regarding the planes in the airport.This module is protected by username and password to prevent unauthorised access.

Purpose of user module is to provide necessary user interface for end users. This program allocates gate, counters, buses and baggage terminalsfor a given plane to use the airport resources effectively. Program will calculate the cost of food, fuel and then provides the final cost for the given plane. This does not require any password for authentication.

There are two categories of planes that need to be served in airport namely arrival and departure

Arrival planes require counters, gate, baggage terminal and buses while Departure planes require gate andbuses.

Data is stored in files on local disk of computer instead of database. This program uses the following files:

* **planedetails.dat**: This file is used to store details of planes that are added/removed by administrator.
* **accounts.dat**: This file is used to store details regarding how much food and fuel are required and then store the final cost computed for each plane.
* **allocation.dat**:This file will store data regarding allocation of counters,gates,buses and baggage terminals of a given plane.

Features available in user module and admin module are explained below

**User Module:**

**Add Plane Details**: Allows user to add details regarding new plane. Data is stored in **planedetails.dat**file.

**Modify Plane Details**: Allows user to modify details regarding existing plane. Data is stored in **planedetails.dat**file.

**Allocation**: Allocates counters, gates, buses and baggage terminals to planes based on whether it is an arrival or departure. Data is stored in **allocation.dat** file.

**Accounting**: Takes details regarding number of units of food required and distance to be travelled. It then calculates the amount of fuel required and total cost. Data is stored in **accounts.dat** file.

## Admin Module:

**Total Revenue**: Calculates the total revenue and displays it. Data is obtained from **accounts.dat**

**Search By Flight Number and Display**: Allows the admin the search and display the details regarding a specific plane. Data retrieved from planedetails.dat, allocation.dat and accounts.dat

**Search By Airlines Name And Display**: Allows the admin the search planes of a given airlines and display the details . Data retrieved from planedetails.dat, allocation.dat and accounts.dat

**Display All**: Displays the details of all the planes. Data retrieved from plane details.

**Allocation Status**: Provides details regarding availability of counters, gates, buses, baggage terminals.

# CALCULATION:

Cost of food=Number of food units x 200

Amount of Fuel (in l) = distance (in km) x 10 + 5000

Cost of fuel = amount of fuel x 50

Total cost = 500000 (Airport service cost) + cost of fuel +cost of food

Total Revenue = total amount paid by all aeroplanes.

**HARDWARE REQUIREMENTS:**

* 64 MB RAM
* Pentum-1 and Above Processor
* Mouse
* Keyboard

**SOFTWARE REQUIREMENT:**

* Operating System – Windows or any other supported
* Turbo C++ Compiler
* IDE (Integrated Development Environment) – Turbo C++

**FUTURE ENHANCEMENTS:**

1. Involving time factor in the program: The program does not take into consideration the time of arrival of the planes and the time of their departure. It limits itself to allocation of data as required by the admin and the user. The deallocation of memory space allocated for a plane is done manually and is not done automatically once the plane has departed. (i.e. once system time has crossed the departure time of the plane. Such modifications could make the program model the real world better.
2. Creating login for users: Each user can be given a login id and a unique password without which the user cannot access the user module.
3. Creating multiple admin logins: In a real airport various admins may access the system at a given time. So, creating multiple logins and ensuring that the memory space allocated for different planes do not overlap will make the program a more useful one. It will allow more than one admin to access the program at the same time.
4. Runway is an important resource in airport that needs to be managed, that is not covered in the existing program. This can be covered in future.
5. Program can be integrated with database instead of storing data in files locally. This will help in improving performance of the program.
6. This program can be extended to handle the staff of airport.
7. This program can be extended to handle vendors and suppliers of airport and integrate with appropriate security systems.

# SOURCE CODE:

/\* This is the Airport Management System created

Created by RohithVarmaBuddaraju, M.KruthicVignesh,

NileshBalu,Narasimhan S of class 12A5\*/

#include <stdio.h>

#include <iostream.h>

#include <fstream.h>

#include <string.h>

#include <conio.h>

intgater[15], countr[30], busr[15], bgr[5];

class aeroplane

{

charFlNo[6];

charAname[50];

char AD;

char org[50];

chardest[50];

public:

voiddisp()

{

cout<< "Flight Number: " <<FlNo<<endl;

cout<< "Airlines Name: " <<Aname<<endl;

cout<< "Origin : " << org <<endl;

cout<< "Destination : " <<dest;

}

char\* retflno()

{

returnFlNo;

}

charretAD()

{

return AD;

}

char\* retaname() {

returnAname;

}

void write();

void modify();

}aero;

void aeroplane::write() //writes onto file

{

ofstreamfout("planedetails.dat", ios::binary | ios::app);

if (!fout) { cout<< "FNF"; return; }

char rep;

do {

aeroplane ob;

cout<< "Enter Plane Details \n";

cout<< "Flight Number:";

gets(ob.FlNo);

cout<< "Airlines Name:";

gets(ob.Aname);

cout<< "Arrival(A)or Departure(D):";

cin>>ob.AD;

if (ob.AD != 'A' && ob.AD != 'D') {

cout<< "Error please provide code again:";

cin>> ob.AD;

}

cout<< "Origin :";

gets(ob.org);

cout<< "Destination :";

gets(ob.dest);

fout.write((char\*)&ob, sizeof(ob));

cout<< "Continue writing?";

cin>> rep;

} while (rep == 'y');

}

void aeroplane::modify() //allows user to modify

{

charflno[6];

aeroplane ob, ob1;

cout<< "Enter Flight Number:";

gets(flno);

fstream fin("planedetails.dat", ios::binary | ios::in | ios::out | ios::beg);

int f = 0;

while (fin.read((char\*)&ob, sizeof(ob)))

{

if (!strcmpi(ob.retflno(), flno))

{

cout<< "Enter New Details\n";

strcpy(ob1.FlNo, flno);

cout<< "Airlines Name:";

gets(ob1.Aname);

cout<< "Arrival(A)or Departure(D):";

cin>> ob1.AD;

if (ob1.AD != 'A' && ob1.AD != 'D') {

cout<< "Error please provide code again:";

cin>> ob1.AD;

}

cout<< "Origin:";

gets(ob1.org);

cout<< "Destination:";

gets(ob1.dest);

fin.seekp(-1 \* ((int)sizeof(ob)), ios::cur);

fin.write((char\*)&ob1, sizeof(ob1));

break;

}

else

f++;

}

if (f != 0)

cout<< "Recd not found";

}

classallo

{

int NC, AC[10], AG, NB, AB[10], ABG, stat;

charfno[6];

public:

allo();

void assignment(); //Funtion that calls all the below functions

int gate(); //To allocate gate

int bus(int[], int&); //To allocate buses

int baggage(); //To allocate baggage terminal

int counter(int[], int&); //To allocate counters

voiddeassignment(); //To deallocate

void output();

void reassign();

char\* retflnoallo()

{

returnfno;

}

intretstat() {

return stat;

}}al;

allo::allo()

{

strcpy(fno, "NULL");

stat = 0;

NC = 0;

AG = 0;

NB = 0;

ABG = 0;

for (int i = 0; i < 10; i++)

{

AC[i] = 0;

AB[i] = 0;

}}

voidallo::assignment()

{

alloob;

ob.stat = 0;

aeroplane r;

cout<< "Enter Flight no\n";

gets(ob.fno);

ifstream a("planedetails.dat", ios::binary | ios::beg);

ofstream b("allocation.dat", ios::binary|ios::app);

while (a.read((char\*)&r, sizeof(r)))

{

if (strcmpi(ob.fno, r.retflno()) == 0)

{

ob.stat++;

break;

}}

if (ob.stat == 0)

{

cout<< "Flight not found\n";

return;

}

ob.AG = gate();

cout<< "Enter the no of buses to be alloted\n";

cin>>ob.NB;

bus(ob.AB, ob.NB);

if (r.retAD() == 'A')

{

ob.ABG = baggage();

ob.NC = 0;

}

if (r.retAD() == 'D')

{

cout<< "Enter the no of counters to be alloted\n";

cin>> ob.NC;

counter(ob.AC, ob.NC);

ob.ABG = 0;

}

b.write((char\*)&ob, sizeof(ob));

}

intallo::gate()

{

int i, flag = 0;

for (i = 0; i<15; i++)

{

if (gater[i] == 0)

{

gater[i] = 1;

flag = 1;

return i + 1;

}}

if (flag == 0)

{

cout<< "No gates are available\n";

}

return 0;

}

intallo::bus(int z[], int&y) //To allocate buses

{

int i, flag = 0;

for (i = 0; i<y; i++)

for (int j = 0; j<15; j++)

{

if (z[i] > 0)

continue;

if (busr[j] == 0)

{

busr[j] = 1;

z[i] = j + 1;

flag++;

}}

if (flag != y)

{

cout<< "Not enough buses\n";

y = flag;

}

return 0;

}

intallo::baggage()

{

int i, flag = 0;

for (i = 0; i<5; i++)

{

if (bgr[i] == 0)

{

bgr[i] = 1;

flag = 1;

return i + 1;

}}

if (flag == 0)

{

cout<< "No baggage terminals are available\n";

return 0;

}

return 0;

}

intallo::counter(int d[], int&e)

{

int i, flag = 0;

for (i = 0; i<e; i++)

for (int j = 0; j<30; j++)

{

if (d[i] > 0)

continue;

if (countr[j] == 0)

{

countr[j] = 1;

d[i] = j + 1;

flag++;

}}

if (flag != e)

{

cout<< "Not enough counters\n";

e = flag;

}

return 0;

}

voidallo::deassignment()

{

fstream a("allocation.dat", ios::binary | ios::in | ios::out | ios::beg);

alloob;

cout<< "Enter the flight no\n";

gets(fno);

while (a.read((char\*)&ob, sizeof(ob)))

{

if (strcmpi(ob.fno,fno) == 0)

{

if (ob.stat == 0)

{

cout<< "Flight already deallocated\n";

return;

}

else

{

gater[ob.AG - 1] = 0;

if (ob.ABG != 0) bgr[ob.ABG - 1] = 0;

ob.ABG = 0;

ob.AG = 0;

for (int i = 0; i < ob.NC; i++)

{

countr[ob.AC[i] - 1] = 0;

}

for (int j = 0; j < ob.NC; j++)

{

ob.AC[j] = 0;

}

for (int k = 0; k <ob.NB; k++)

{

busr[ob.AB[k] - 1] = 0;

}

for (int l = 0; l <ob.NB; l++)

{

ob.AB[l] = 0;

}

ob.NC = 0;

ob.NB = 0;

ob.stat = 0;

}

a.seekp(-1 \* ((int)sizeof(ob)), ios::cur);

a.write((char\*)&ob, sizeof(ob));

break;

}}}

voidallo::output()

{

if (NC != 0)

{

cout<< "\nNo of alloted counters:" << NC;

cout<< "\nCounters Allocated:";

for (int i = 0; AC[i] != '\0'; i++)

cout<< AC[i] <<" ";

}

cout<< "\nGATE:" << AG << "\nNo of alloted buses:" << NB;

cout<< "\nBuses Allocated: ";

for (int i = 0; AB[i] != '\0'; i++)

cout<< AB[i] <<" ";

if (ABG != 0)

{

cout<< "\nBaggage terminal" << ABG;

}}

voidallo::reassign()

{

alloob;

ifstream fin("allocation.dat", ios::binary | ios::beg);

while (fin.read((char\*)&ob, sizeof(ob)))

{

if (ob.stat != 0)

{

gater[ob.AG - 1] = 1;

if (ob.ABG != 0)bgr[ob.ABG - 1] = 1;

for (int i = 0; i<ob.NB; i++)

{

busr[ob.AB[i] - 1] = 1;

}

for (int j = 0; j<ob.NC; j++)

{

countr[ob.AC[j] - 1] = 1;

}}}}

class accounting {

charFlno[20];

int NF;

float CF, DT, AF, CFU, TC;

public:

char\* retflno() {

returnFlno;

}

voidassigncost();

voidtotalrevenue();

void display() { //This is the function to display details from account.dat when flno is passed as arguement

cout<< "\nNumber of units of food:" << NF;

cout<< "\nCost of food:" << CF;

cout<< "\nTotal distance travelled in km:" << DT;

cout<< "\nAmount of fuel:" << AF;

cout<< "\nCost of fuel:" << CFU;

cout<< "\nTotal cost:" << TC;

}

voidmodcost();

}acco;

void accounting::assigncost(){ //this is the function to assign cost

accounting ob1;

charstr[6];

int flag = 0;

cout<< "Enter the flight number";

gets(str);

aeroplane ob; //object of aeroplane is created here

ifstream fin("planedetails.dat", ios::binary | ios::beg);

if (!fin) {

cout<< "FNF"; return;

}

while (fin.read((char\*)&ob, sizeof(ob)))

{

if (strcmpi(ob.retflno(), str) == 0) {

flag = 1; break;

}}

fin.close();

if (flag == 0) {

cout<< "Flight not found \n"; return;

}

strcpy(ob1.Flno, str);

cout<< "Enter number of unit of food required:";

cin>> ob1.NF;

cout<< "Enter the distance to be travelled:";

cin>> ob1.DT;

ob1.CF = ob1.NF \* 200.0; //Calculation occur here

ob1.AF = ob1.DT \* 5 + 5000;

ob1.CFU = ob1.AF \* 10;

ob1.TC = 500000 + ob1.CF + ob1.CFU;

ofstreamfout("accounts.dat", ios::app | ios::binary);

fout.write((char\*)&ob1, sizeof(ob1));

fout.close();

}

void accounting::totalrevenue(){ //This is the funtion to find the total revenue for admin

accounting ob;

ifstream fin("accounts.dat", ios::binary | ios::beg);

if (!fin) {

cout<< "FNF"; return;

}

float sum = 0;

while (fin.read((char\*)&ob, sizeof(ob))) {

sum += ob.TC;

}

cout<< "\nThe total revenue is " << sum;

fin.close();

}

void accounting::modcost(){

accounting ob1;

charstr[6];

int flag = 0;

cout<< "Enter the flight number";

gets(str);

fstream fin("accounts.dat", ios::binary | ios::beg | ios::in | ios::out);

if (!fin) {

cout<< "FNF"; return;

}

while (fin.read((char\*)&ob1, sizeof(ob1)))

{

if (strcmpi(ob1.Flno, str) == 0) {

flag = 1; break;

}}

if (flag == 0) {

cout<< "Flight not found in accounts file\n"; return;

}

strcpy(ob1.Flno, str);

cout<< "Enter nunber of unit of food required:";

cin>> ob1.NF;

cout<< "Enter the distance to be travelled:";

cin>> ob1.DT;

ob1.CF = ob1.NF \* 200.0; //Calculation occur here

ob1.AF = ob1.DT \* 5 + 5000;

ob1.CFU = ob1.AF \* 10;

ob1.TC = 500000 + ob1.CF + ob1.CFU;

fin.seekg((-1) \* sizeof(ob1), ios::cur);

fin.write((char\*)&ob1, sizeof(ob1));

fin.close();

}

class admin

{

public:

voidshowall();

voidsearchplane(char a[10]);

voidallostatus();

voiddeallocateall();

voidserairline();

}adm;

void admin::showall()

{

aeroplane ob;

accounting ob1;

allo ob2;

int flag1 = 0, flag2 = 0;

ifstream fin("planedetails.dat", ios::binary);

ifstream fin1("accounts.dat", ios::binary);

ifstream fin2("allocation.dat", ios::binary);

fin.seekg(0, ios::beg);

while (fin.read((char\*)&ob, sizeof(ob)))

{

flag1 = 0; flag2 = 0;

ob.disp(); //function to display the datamembers of object of aeroplane

fin1.seekg(0, ios::beg);

while (fin1.read((char\*)&ob1, sizeof(ob1)))

{

if (!strcmpi(ob.retflno(), ob1.retflno()))

{{

ob1.display(); /\*to print the corresponding account details of the same flight\*/

flag1++;

break;

}}}

if (flag1 == 0) {

cout<< "Flight not found in accounts file \n";

}

fin2.seekg(0, ios::beg);

while (fin2.read((char\*)&ob2, sizeof(ob2)))

{

if (!strcmpi(ob.retflno(), ob2.retflnoallo()))

{

flag2++;

if(ob2.retstat()==0)flag2--;

else

{ob2.output(); /\*to print the corresponding allocation details (like counters, baggages) of the same flight\*/

break;}}}

if (flag2 == 0) {

cout<< "Flight not found in allocation file\n";

}}

fin.close();

fin1.close();

fin2.close();

}

void admin::searchplane(char a[10]) /\*the flight number is given as an argument to search and display the details\*/

{

int flag = 0, flag1 = 0, flag2 = 0;

aeroplaneobj;

accounting obj1;

allo obj2;

ifstream f("planedetails.dat", ios::binary);

ifstream f1("accounts.dat", ios::binary);

ifstream f2("allocation.dat", ios::binary);

f.seekg(0, ios::beg);

while (f.read((char\*)&obj, sizeof(obj)))

{

flag1 = 0; flag2 = 0;

if (!strcmpi(a, obj.retflno()))

{

obj.disp(); /\*function to display the datamembers (of class aeroplane) of the object that is being searched\*/

flag++;

f1.seekg(0, ios::beg);

while (f1.read((char\*)&obj1, sizeof(obj1)))

{

if (!strcmpi(a, obj1.retflno()))

{ obj1.display(); /\*to print the corresponding account details of the same flight\*/

flag1++;

break;

}}

if (flag1 == 0) {

cout<< "Flight not found in accounts file\n";

}

f2.seekg(0, ios::beg);

while (f2.read((char\*)&obj2, sizeof(obj2)))

{

if (!strcmpi(a, obj2.retflnoallo()))

{

flag2++;

if(ob2.retstat()==0)flag2--;

else

{ob2.output(); /\*to print the corresponding allocation details (like counters, baggages) of the same flight\*/

break;}}}

if (flag2 == 0) {

cout<< "Flight not found in allocation file\n";

}

break;}}

if (flag == 0)

cout<< "plane details not found\n";

f.close();

f1.close();

f2.close();

}

void admin::deallocateall()

{

for (int i = 0; i<30; i++)

{

countr[i] = 0;

}

for (int j = 0; j<15; j++)

{

busr[j] = 0;

}

for (int k = 0; k<15; k++)

{

gater[k] = 0;

}

for (int l = 0; l<5; l++)

{

bgr[l] = 0;

}}

void admin::allostatus()

{

int c = 0, g = 0, bt = 0, b = 0; /\*c stands for number of counters, g for number of gates, bt for baggage terminals, b for buses\*/

for (int i = 0; i<30; i++)

{

if (countr[i] == 0)

c++;

}

for (int j = 0; j<15; j++)

{

if (busr[j] == 0)

b++;

}

for (int k = 0; k<15; k++)

{

if (gater[k] == 0)

g++;

}

for (int l = 0; l<5; l++)

{

if (bgr[l] == 0)

bt++;

}

cout<< "Number of counters unoccupied:" << c << "\n";

cout<< "Number of gates unoccupied:" << g << "\n";

cout<< "Number of baggage terminals unoccupied:" <<bt<< "\n";

cout<< "Number of buses unoccupied:" << b << "\n";

}

void admin::serairline()

{

int flag = 0, flag1 = 0, flag2 = 0;

aeroplaneobj;

accounting obj1;

allo obj2;

charstr[10];

cout<< "Enter Airlines Name";

gets(str);

ifstream f("planedetails.dat", ios::binary);

ifstream f1("accounts.dat", ios::binary);

ifstream f2("allocation.dat", ios::binary);

f.seekg(0, ios::beg);

while (f.read((char\*)&obj, sizeof(obj)))

{

if (!strcmpi(str, obj.retaname()))

{

flag1 = 0; flag2 = 0;

obj.disp(); /\*function to display the datamembers (of class aeroplane) of the object that is being searched\*/

flag++;

f1.seekg(0, ios::beg);

while (f1.read((char\*)&obj1, sizeof(obj1)))

{

if (!strcmpi(obj.retflno(), obj1.retflno()))

{ obj1.display(); /\*to print the corresponding account details of the same flight\*/

flag1++;

break;

}}

if (flag1 == 0) {

cout<< "Flight not found in accounts file\n";

}

f2.seekg(0, ios::beg);

while (f2.read((char\*)&obj2, sizeof(obj2)))

{

if (!strcmpi(obj.retflno(), obj2.retflnoallo()))

{

flag2++;

if(ob2.retstat()==0)flag2--;

else

{

ob2.output(); /\*to print the corresponding allocation details (like counters, baggages) of the same flight\*/

break;}}}

if (flag2 == 0) {

cout<< "Flight not found in allocation file\n";

}

break;}}

if (flag == 0)

cout<< "plane details not found\n";

f.close();

f1.close();

f2.close();

}

voidusermodule() {

intch;

char rep = 'y';

cout<< "User Module";

do {

cout<< "\nEnter Code\n1.Add Plane Details\n2.Modify Plane Details \n3.Assign gate,buses,baggageterminals,counters\n4.Deassign Gate,buses,baggageterminals,counters\n5.Enter cost details\n6.Modify Cost Details\n7.Return to Main Menu:";

cin>>ch;

if (ch == 1)

aero.write(); //Funtion to add detials from class aeroplane

else if (ch == 2)

aero.modify(); //Function to modify plane detial in class aeroplane

else if (ch == 3)

al.assignment(); //Funtion to assign Gate,buses,baggageterminals,counters form class allocat

else if (ch == 4)

al.deassignment(); //Funtion to deassignGate,buses,baggageterminals,counters form class allocat

else if (ch == 5)

acco.assigncost();//Funtion to assign cost

else if (ch == 5)

acco.modcost();

else if (ch == 7)

return;

elsecout<< "Invalid Choice";

cout<< "Do you want to continue:";

cin>> rep;

} while (rep == 'y');

return;

}

voidadminmodule() {

intch;

char rep = 'y';

cout<< "Admin Module";

do {

cout<< "\nEnter Code\n1.Total Revenue\n2.Search And Display by Plane Number \n3.Search And Display By Airlines Name \n4.Diplay Details of all Planes\n5.Allocation Status\n6.Deassign all Gate,buses,baggageterminals,counters\n7.Return to Main Menu:";

cin>>ch;

if (ch == 1)

acco.totalrevenue(); //Funtion to to give sum all total costs from class accounting

else if (ch == 2)

{

charstr[6];

cout<< "Enter Flight Number:";

gets(str);

adm.searchplane(str); //Function to search and display details of a given class in class admin

}

else if (ch == 3) {

adm.serairline();

}

else if (ch == 4)

adm.showall(); //Funtion to display details of all planes form class allocat

else if (ch == 5)

adm.allostatus(); //Funtion to number of Gate,buses,baggageterminals,counters available form class admin

else if (ch == 6)

adm.deallocateall();//Funtion to deassign all Gate,buses,baggageterminals,counters

else if (ch == 7)

return;

elsecout<< "Invalid Choice";

cout<< "Do you want to continue?";

cin>> rep;

} while (rep == 'y');

return;

}

int main()

{

clrscr();

intch;

al.reassign();

char rep = 'y';

cout<< "Welcome to Airport Management System\n";

do {

cout<< "Enter code \n1.User\n2.Admin";

cin>>ch;

if (ch == 1)usermodule();

else if (ch == 2) {

int pass;

cout<< "Enter Password:";

cin>> pass;

if (pass == 1234) adminmodule();

else

cout<< "Password Is Incorrect\n";

}

else if (ch == 3) {

ofstream fin1("planedetails.dat", ios::binary | ios::trunc);

if (!fin1) {

cout<< "File Plane details not found";

}

ifstream fin2("allocation.dat", ios::binary);

if (!fin2) {

cout<< "File allocation not found";

}

alloob;

while (fin2.read((char\*)&ob, sizeof(ob))) {

if (ob.retstat() == 1)ob.deassignment();

}

ofstream fin3("accounts.dat", ios::binary | ios::trunc);

if (!fin3) {

cout<< "File accounts not found";

}

fin2.close();

fin2.open("allocation.dat", ios::binary | ios::trunc);

fin1.close();

fin2.close();

fin3.close();

}

elsecout<< "Invalid Choice";

cout<< "Do you want to continue?";

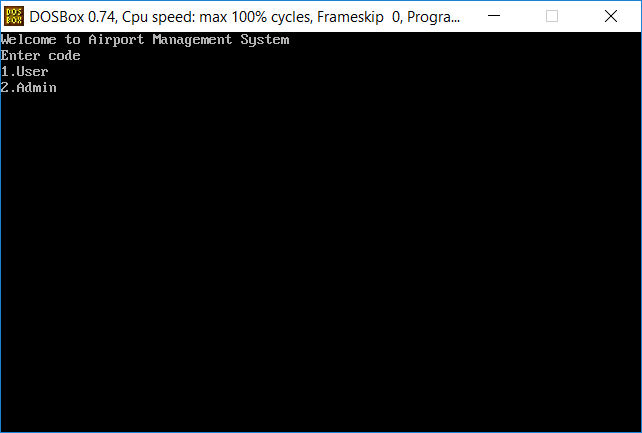
cin>> rep;

} while (rep == 'y');

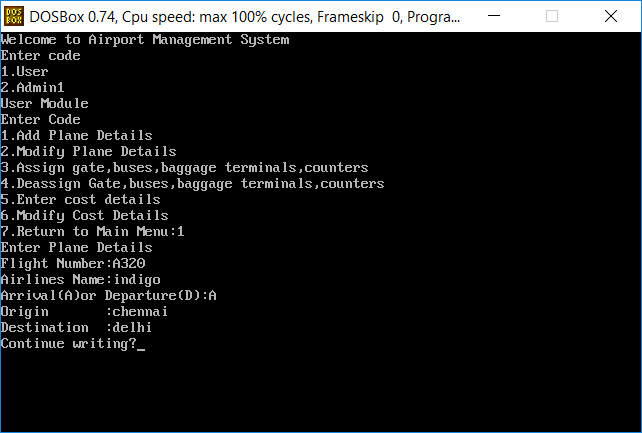
return 0;}

**OUTPUTS:**

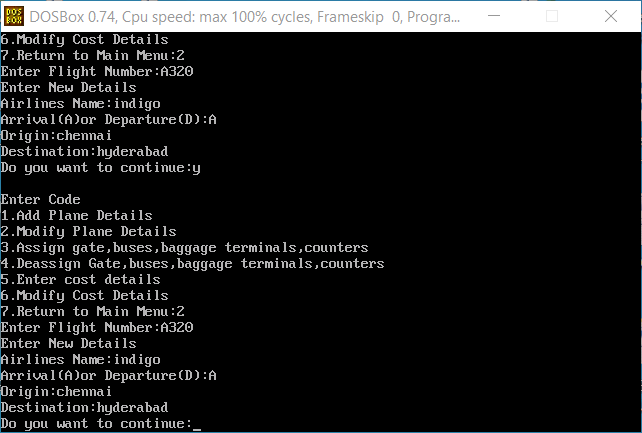
**1.MAIN MENU**



**2.ADDING PLANE DETAILS IN USER**

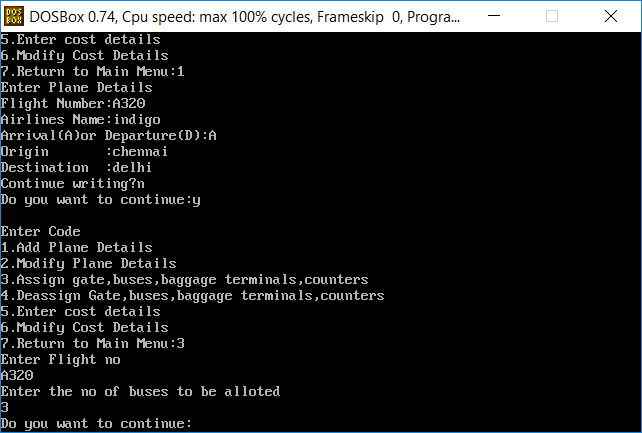


**3.MODIFY PLANE DETAILS**

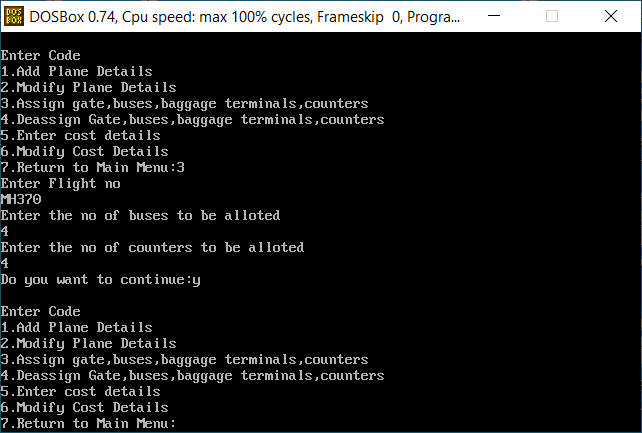


**4.ALLOCATION**

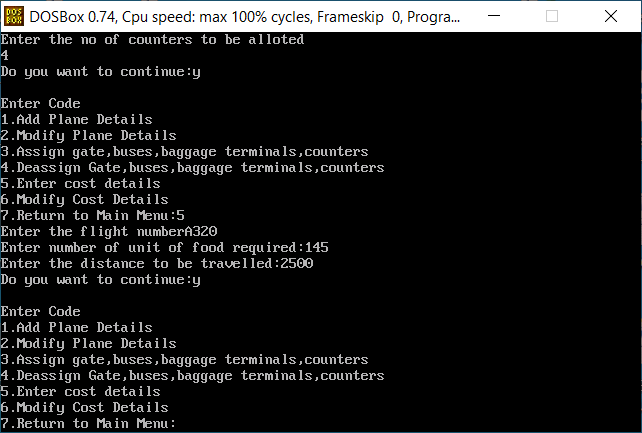
**For Arrivals**



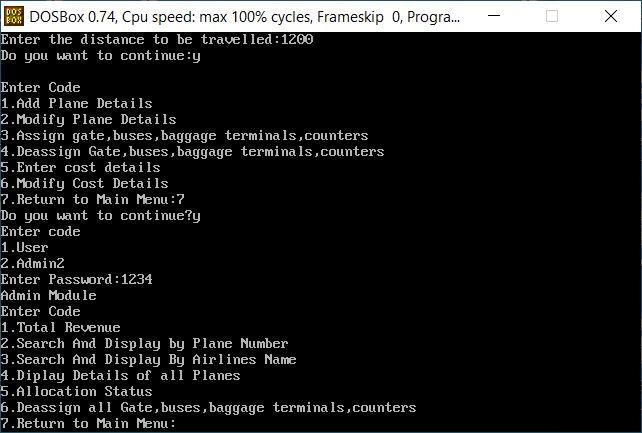
**For Departure**



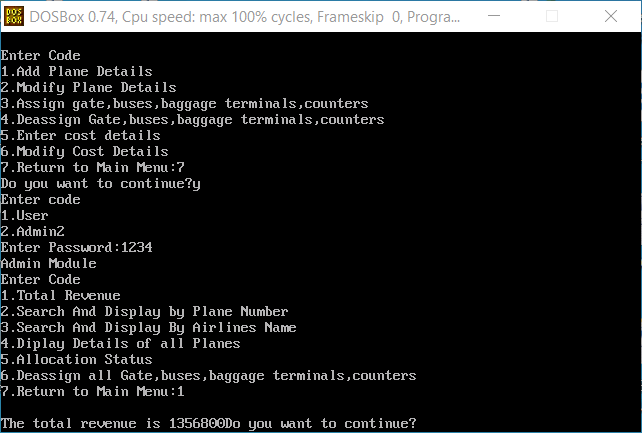
**5.COST**



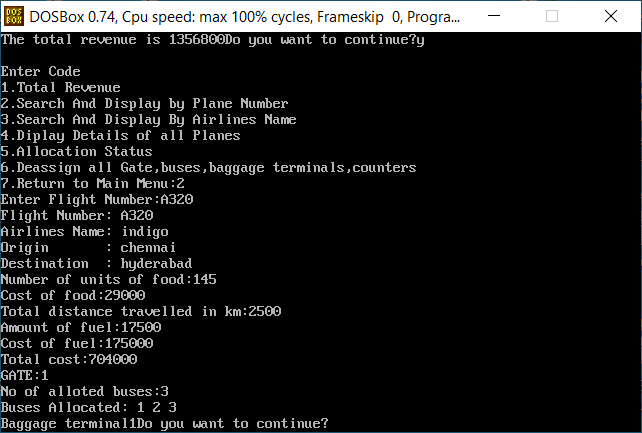
**6.ADMIN MODULE**



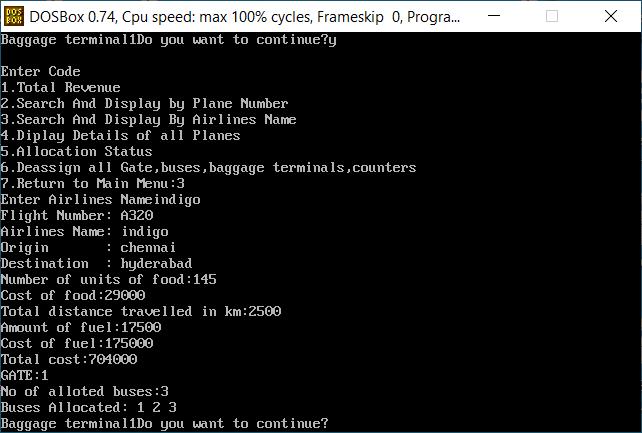
**7.TOTAL REVENUE**



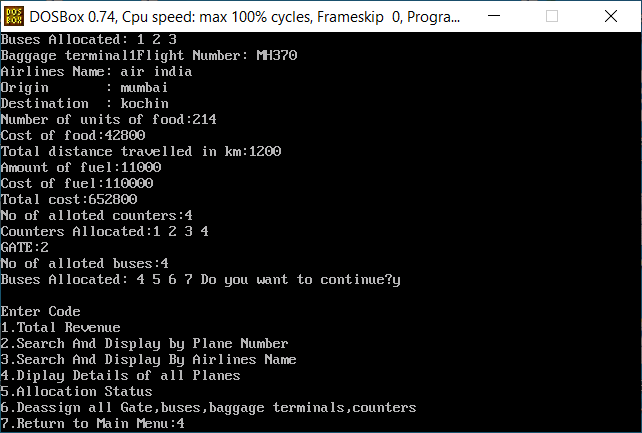
**8.SEARCH BY PLANE NUMBER**

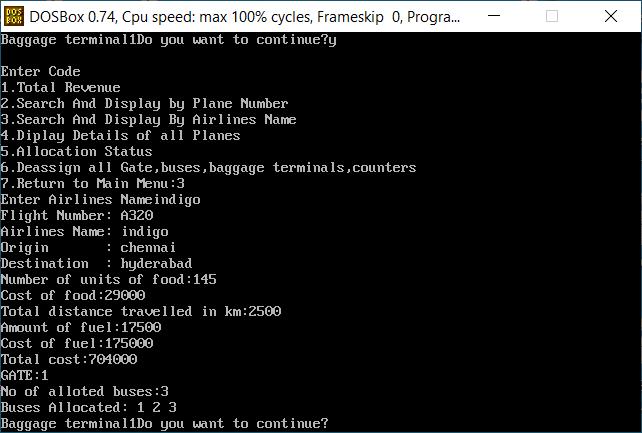
****

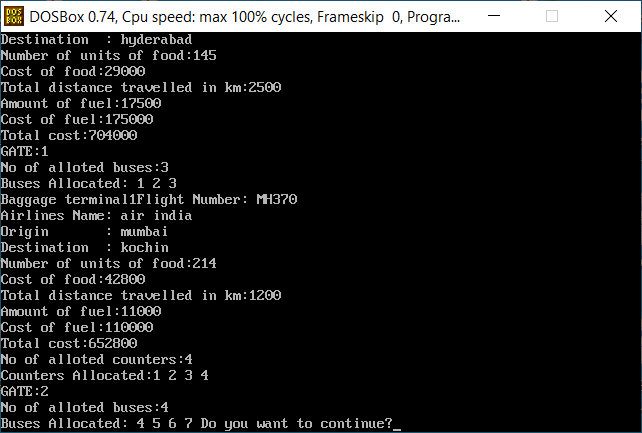
**9.SEARCH BY AIRLINE NAME**



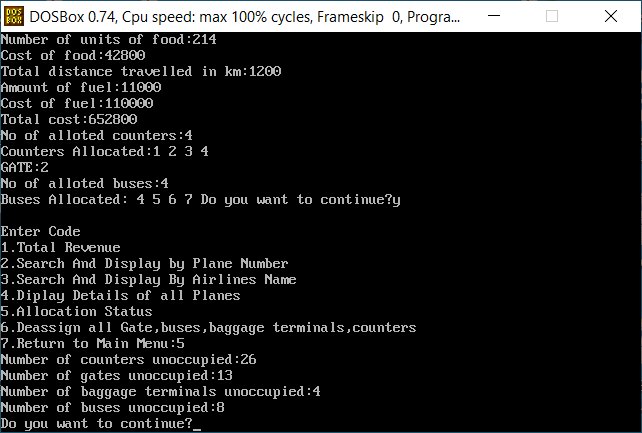
**10.DISPLAY ALL**

****

****

****

**11.ALLOCATION STATUS**



**BIBLIOGRAPHY:**

1. Computer Science With C++ by Sumita Arora
2. Online references

<https://www.geeksforgeeks.org/c-plus-plus/>

<https://www.tutorialspoint.com/reading-and-writing-binary-file-in-c-cplusplus>

<https://en.wikipedia.org/wiki/Total_Airport_Management_Systems>