# PROJECT WORK

**Date of Submission: 22-11-23**

**Ticket Reservation System**

**Project Definition:**

Write a C++ program to create a ticket reservation system. Program should facilitate the end user to create a separte file for each ticket record entered with ticket number as the file name. ticket record details should be :cname,tno, and destination- addr.Provide a menu driven o/p with following options for the program:

option 1->To enter ticket details and display separately each ticket file as o/p

option 2-> To edit tno of a ticket and also display the updated content to a ticket file & display appended content on screen

**CodeBase:**

#include<iostream> #include<fstream> #include<string.h> #include<stdlib.h> #include<stdio.h> #include<cstdlib> #include<ctime> #include<iomanip> #include<vector> using namespace std;

#define RESET\_CLR "\033[0m" #define BLACK\_TXT "\033[30m" #define RED\_TXT "\033[31m" #define GREEN\_TXT "\033[32m" #define YELLOW\_TXT "\033[33m" #define BLUE\_TXT "\033[34m" #define WHITE\_BG "\033[47m"

string intro = "+ +\n" "| "

"TICKET MANAGEMENT SYSTEM "

" |\n"

"| Developed by: Damodar Banaulikar (22B-CO-012) |\n" "| : Dattaraj Gawde (22B-CO-014) |\n"

"| : Divyam Redkar (22B-CO-016) |\n" "| : Ervin (22B-CO-018) |\n"

"| : Giselle Fernandes (22B-CO-019) |\n"

"| : Goraksh Naik (22B-CO-020) |\n"

"+ +\n\n\n";

class ticket; class BGT{ public:

static void serach\_ticket();

static void display\_details(ticket \*T); static string generate\_ticket\_no(); static string generate\_train\_no(); static string get\_names(int pas\_count);

static void serialize\_ticket(ticket \*T, fstream &ticketFile); static void deserialize\_ticket(ticket \*T, fstream &ticketFile); static void serialize\_string(string \*str, fstream &ticketFile); static void deserialize\_string(string \*str, fstream &ticketFile);

};

class ticket

{

public:

string cname; string ticket\_no; string train\_no; string source; string destination; string coach;

int no\_pas; float price;

ticket():price(0){}

//option 1

static void option1();

//puts the details from ptr to file

static int put\_details\_tofile(ticket \*ptr);

//option 2

static void option2();

//edit ticket number (tno)

static int edit\_tno\_infile(ticket \*old\_T, ticket \*new\_T);

//option 3

static void option3();

//add price to file

static int add\_price\_tofile(ticket \*ptr);

};

string BGT :: generate\_ticket\_no()

{

string ticket\_no; srand(time(0));

// Generate a random 6-digit ticket number

int ticket\_number = rand() % 900000 + 100000;

// Convert the ticket number to a string ticket\_no = to\_string(ticket\_number); return ticket\_no;

}

string BGT :: generate\_train\_no()

{

string train\_no; srand(time(0));

// Generate a random 6-digit train number

int ticket\_number = rand() % 9000 + 1000;

// Convert the train number to a string train\_no = to\_string(ticket\_number); return train\_no;

}

string BGT :: get\_names( int pas\_count )

{

string all\_names; string name;

int i = 1; cin.ignore();

while (i <= pas\_count) {

cout << "\nEnter passenger ["<<i++<<"] name : "; getline(cin, name);

all\_names += name + "$";

}

return all\_names;

}

void BGT :: serialize\_ticket(ticket \*T, fstream &ticketFile)

{

BGT::serialize\_string(&T->cname, ticketFile); BGT::serialize\_string(&T->ticket\_no, ticketFile); BGT::serialize\_string(&T->train\_no, ticketFile); BGT::serialize\_string(&T->source, ticketFile); BGT::serialize\_string(&T->destination, ticketFile); BGT::serialize\_string(&T->coach, ticketFile); ticketFile << ',';

ticketFile << T->no\_pas; ticketFile << ','; ticketFile << T->price;

}

void BGT :: deserialize\_ticket(ticket \*T, fstream &ticketFile)

{

char comma;

BGT::deserialize\_string(&T->cname, ticketFile); BGT::deserialize\_string(&T->ticket\_no, ticketFile); BGT::deserialize\_string(&T->train\_no, ticketFile); BGT::deserialize\_string(&T->source, ticketFile); BGT::deserialize\_string(&T->destination, ticketFile); BGT::deserialize\_string(&T->coach, ticketFile); ticketFile >> comma;

ticketFile >> T->no\_pas; ticketFile >> comma; ticketFile >> T->price;

}

void BGT :: serialize\_string(string \*str, fstream &ticketFile)

{

ticketFile << ','; ticketFile << str->size(); ticketFile << ','; ticketFile << str->c\_str();

}

void BGT :: deserialize\_string(string \*str , fstream &ticketFile)

{

int len;

char comma; ticketFile >> comma; ticketFile >> len; ticketFile >> comma; if (ticketFile && len) {

vector<char> tmp(len); ticketFile.read(tmp.data() , len); str->assign(tmp.data(), len);

}

}

void BGT :: serach\_ticket()

{

ticket T;

cout<<"Enter ticket number : "; cin.ignore(); getline(cin,T.ticket\_no); BGT::display\_details(&T);

}

void BGT :: display\_details(ticket \*T)

{

string ticketNo = T->ticket\_no + ".txt"; fstream ticketFile;

ticketFile.open(ticketNo, ios::in|ios::binary);

//checks if file exists ticketFile.seekg(0); if(ticketFile){

BGT::deserialize\_ticket(T,ticketFile); ticketFile.close();

}

else

{

ticketFile.close();

cout<<RED\_TXT<<"\nFile does not exist !\n"<<RESET\_CLR; return;

}

istringstream iss(T->cname); string name;

cout<<setfill(' ')<<"\n\n"<<WHITE\_BG<<BLACK\_TXT<<"+----------------------------------------------------

+"<<RESET\_CLR<<endl;

cout<<WHITE\_BG<<BLACK\_TXT<<"|Train Number "<<"Number of passengers "<<"Ticket Number|"<<RESET\_CLR<<endl;

cout<<WHITE\_BG<<BLACK\_TXT<<"|"<<left<<setw(16)<<T->train\_no<<setw(24)<<T-

>no\_pas<<setw(12)<<T->ticket\_no<<"|"<<RESET\_CLR<<endl; cout<<WHITE\_BG<<BLACK\_TXT<<"|Train Coach "<<"Boarding Station "<<"Destination

Station|"<<RESET\_CLR<<endl; cout<<WHITE\_BG<<BLACK\_TXT<<"|"<<setw(15)<<T->coach<<setw(18)<<T-

>source<<setw(19)<<T->destination<<"|"<<RESET\_CLR<<endl<<"\033[2K"; cout<<WHITE\_BG<<BLACK\_TXT<<"|Passenger Names : |"<<RESET\_CLR<<endl; while (getline(iss, name, '$')) {

cout<<RESET\_CLR<<WHITE\_BG<<BLACK\_TXT<<setw(53)<<name<<"|"<<RESET\_CLR<<endl;

}

if(T->price != 0){

cout<<WHITE\_BG<<BLACK\_TXT<<"|"<<" Price : "<<setw(40)<<T-

>price<<"|"<<RESET\_CLR<<endl;

}

cout<<WHITE\_BG<<BLACK\_TXT<<"+ +"<<RESET\_CLR;

}

void ticket :: option1()

{

ticket T;

cout<<"\nEnter details:\n"; cout<<"Enter Source : "; cin.ignore(); getline(cin,T.source); cout<<"Enter Destination : "; cin.ignore(); getline(cin,T.destination);

cout<<"Enter Train Coach (1A, 2A, 3A, CC, FC, SL, 2S) : "; cin.ignore();

getline(cin,T.coach);

cout<<"Enter number of passengers : "; cin>>T.no\_pas;

cout<<"Enter customer name : "; T.cname = BGT::get\_names(T.no\_pas); T.ticket\_no = BGT::generate\_ticket\_no(); cin.ignore();

T.train\_no = BGT::generate\_train\_no();

if(put\_details\_tofile(&T) == 0){ BGT::display\_details(&T);

}

}

int ticket :: put\_details\_tofile(ticket \*ptr)

{

string ticketNo = ptr->ticket\_no + ".txt";

fstream ticketFile; ticketFile.open(ticketNo, ios::in);

//checks if file exists if(!ticketFile){

//creates a new file if does not exist ticketFile.close();

ticketFile.open(ticketNo, ios::out|ios::binary); BGT::serialize\_ticket(ptr,ticketFile);

}

else{

//gives error if file exists

cout<<YELLOW\_TXT<<"\nTicket already exists!"<<RESET\_CLR;

}

if(ticketFile.fail()){

cout<<YELLOW\_TXT<<"\nError : I/O Error has occurred !\n"<<RESET\_CLR;

ticketFile.close(); return 1;

}

if(ticketFile.bad()){

cout<<RED\_TXT<<"\nFatal Error : Invalid operation attempted !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

ticketFile.close(); return 0;

}

void ticket :: option2()

{

ticket old\_T, new\_T;

cout<<"\nEnter ticket no to be edited : "; cin>>old\_T.ticket\_no;

new\_T.ticket\_no = BGT::generate\_ticket\_no(); if(ticket::edit\_tno\_infile(&old\_T,&new\_T) == 0)

BGT::display\_details(&new\_T); getchar();

}

int ticket :: edit\_tno\_infile(ticket \*old\_T, ticket \*new\_T)

{

string Ticket\_old = old\_T->ticket\_no + ".txt"; string Ticket\_new = new\_T->ticket\_no + ".txt";

//creating updated file fstream ticketFile;

ticketFile.open(Ticket\_old,ios::in); ticketFile.seekg(0);

if(!ticketFile){

cout<<RED\_TXT<<"\nTicket with ticket number "<<old\_T->ticket\_no<<" does not exist!\n"<<RESET\_CLR<<endl;

}

else{

// read data from file in old\_T BGT::deserialize\_ticket(old\_T,ticketFile); old\_T->ticket\_no = new\_T->ticket\_no;

if(ticketFile.fail()){

cout<<YELLOW\_TXT<<"\nError : I/O Error has occurred !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

ticketFile.close();

char temp[Ticket\_old.length() + 1]; strcpy(temp, Ticket\_old.c\_str());

// create new file with new name ticketFile.open(Ticket\_new, ios::in);

if(ticketFile){

cout<<YELLOW\_TXT<<"\nTicket already exists !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

else{

if(remove(temp) != 0){

cout<<RED\_TXT<<"/nError in deleting old file !"<<RESET\_CLR<<endl; return 1;

}

ticketFile.close();

ticketFile.open(Ticket\_new, ios::out|ios::binary); ticketFile.seekg(0); BGT::serialize\_ticket(old\_T,ticketFile);

}

}

if(ticketFile.fail()){

cout<<YELLOW\_TXT<<"\nError : I/O Error has occurred !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

if(ticketFile.bad()){

cout<<RED\_TXT<<"\nFatal Error : Invalid operation attempted !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

if(ticketFile.good()){

cout<<GREEN\_TXT<<"\nTicket updated successfully\n"<<RESET\_CLR;

}

ticketFile.close(); return 0;

}

void ticket :: option3()

{

ticket T;

cout<<"\nEnter Ticket number : "; cin>>T.ticket\_no;

cin.ignore();

cout<<"\nEnter ticket price: "; cin>>T.price;

if(ticket :: add\_price\_tofile(&T) == 0) BGT::display\_details(&T);

getchar();

}

int ticket :: add\_price\_tofile(ticket \*ptr)

{

string ticketNo = ptr->ticket\_no + ".txt"; fstream ticketFile;

ticket temp;

ticketFile.open(ticketNo, ios::in|ios::binary);

//checks if file exists

if(ticketFile){

//copys contents from file to temp ticketFile.seekg(0); BGT::deserialize\_ticket(&temp,ticketFile); if(ticketFile.fail()){

cout<<YELLOW\_TXT<<"\nError : I/O Error has occurred !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

ticketFile.close();

//update price in temp temp.price = ptr->price;

//update contents of file ticketFile.open(ticketNo, ios::out|ios::binary); ticketFile.seekg(0); BGT::serialize\_ticket(&temp, ticketFile); if(ticketFile.fail()){

cout<<YELLOW\_TXT<<"\nError : I/O Error has occurred !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

}

else{

//gives error if file does not exists

cout<<RED\_TXT<<"\nFile does not exists!"<<RESET\_CLR<<endl; ticketFile.close();

return 1;

}

if(ticketFile.bad()){

cout<<RED\_TXT<<"\nFatal Error : Invalid operation attempted !\n"<<RESET\_CLR; ticketFile.close();

return 1;

}

ticketFile.close(); return 0;

}

int main()

{

int choice;

while (1)

{

system("CLS"); cout<<intro;

cout <<BLUE\_TXT<<"\n-----------------------MENU " << endl;

cout << "[1] Enter Ticket Details " << endl; cout << "[2] Edit Ticket Number " << endl;

cout << "[3] Append Price to a Ticket " << endl; cout << "[4] Search and Display Ticket"<<endl; cout << "[5] Exit" << endl;

cout << "Enter your choice: "<<RESET\_CLR; cin >> choice;

switch (choice)

{

case 1:

ticket::option1(); break;

case 2:

ticket::option2(); break;

case 3:

ticket::option3(); break;

case 4:

BGT::serach\_ticket(); break;

case 5:

return 0;

default:

cout <<YELLOW\_TXT<< "Invalid choice. Please try again."<<RESET\_CLR<< endl; break;

}

cout<<GREEN\_TXT<<"\ncontinue ? (y/n) :"<<RESET\_CLR; char c = getchar();

if(c =='n' || c == 'N') return 0;

}

return 0;

}

# Code Documentation:

BGT Class:

1. search\_ticket() Purpose:

Searches for a ticket based on the provided

ticket number and displays its details.

Input Parameters:

None

Output Values:

None

Exceptions:

None

Example Usage: BGT::search\_ticket();

1. display\_details(ticket \*T) Purpose:

Displays details of a ticket.

Input Parameters:

- T: Pointer to a ticket object.

Output Values:

None

Exceptions:

None

Example Usage:

ticket T;

// Populate T with ticket details BGT::display\_details(&T);

1. generate\_ticket\_no() Purpose:

Generates a random 6-digit ticket number.

Input Parameters:

None

Output Values:

- Returns a string representing the generated ticket number.

Exceptions:

None

Example Usage:

string ticket\_no = BGT::generate\_ticket\_no();

1. generate\_train\_no() Purpose:

Generates a random 6-digit train number.

Input Parameters:

None

Output Values:

* Returns a string representing the generated train number.

Exceptions:

None

Example Usage:

string train\_no = BGT::generate\_train\_no();

1. get\_names(int pas\_count) Purpose:

Gets names of passengers from user input.

Input Parameters:

* pas\_count: Number of passengers.

Output Values:

* Returns a string containing names separated by '$' delimiter.

Exceptions:

None

Example Usage:

int passenger\_count = 3; string all\_names =

BGT::get\_names(passenger\_count);

1. serialize\_ticket(ticket \*T, fstream &ticketFile)

Purpose:

Serializes ticket details to a file. Input Parameters:

* T: Pointer to a ticket object.
* ticketFile: Reference to a file stream.

Output Values:

None

Exceptions:

None

Example Usage:

ticket T;

fstream ticketFile;

// Open ticket file stream BGT::serialize\_ticket(&T, ticketFile);

1. deserialize\_ticket(ticket \*T, fstream &ticketFile)

Purpose:

Deserializes ticket details from a file.

Input Parameters:

* T: Pointer to a ticket object.
* ticketFile: Reference to a file stream.

Output Values:

None

Exceptions:

None

Example Usage:

ticket T;

fstream ticketFile;

// Open ticket file stream BGT::deserialize\_ticket(&T, ticketFile);

1. serialize\_string(string \*str, fstream &ticketFile)

Purpose:

Serializes a string to a file.

Input Parameters:

* str: Pointer to a string object.
* ticketFile: Reference to a file stream.

Output Values: None

Exceptions: None

Example Usage:

string myString = "Hello, World!"; fstream ticketFile;

// Open ticket file stream BGT::serialize\_string(&myString, ticketFile);

1. deserialize\_string(string \*str, fstream &ticketFile)

Purpose:

Deserializes a string from a file.

Input Parameters:

* str: Pointer to a string object.
* ticketFile: Reference to a file stream.

Output Values:

None

Exceptions:

None

Example Usage:

string myString; fstream ticketFile;

// Open ticket file stream BGT::deserialize\_string(&myString, ticketFile);

ticket Class:

1. option1() Purpose:

Allows the user to enter details for a new

ticket.

Input Parameters:

None

Output Values:

None

Exceptions:

None

Example Usage: ticket::option1();

1. put\_details\_tofile(ticket \*ptr) Purpose:

Puts ticket details into a file.

Input Parameters:

* ptr: Pointer to a ticket object.

Output Values:

* Returns 0 if successful.

Exceptions:

* Displays an error message if the ticket already exists.

Example Usage:

ticket T;

// Populate T with ticket details

int result = ticket::put\_details\_tofile(&T);

1. option2() Purpose:

Allows the user to edit a ticket number in the file.

Input Parameters:

None

Output Values:

None

Exceptions:

None

Example Usage: ticket::option2();

1. edit\_tno\_infile(ticket \*old\_T, ticket

\*new\_T)

Purpose:

Edits the ticket number in the file.

Input Parameters:

* old\_T: Pointer to a ticket object with the old ticket number.
* new\_T: Pointer to a ticket object with the new ticket number.

Output Values:

* Returns 0 if successful. Exceptions:
* Displays an error message if the old ticket number does not exist.

Example Usage:

ticket oldTicket, newTicket;

// Populate oldTicket and newTicket with details

int result = ticket::edit\_tno\_infile(&oldTicket, &newTicket);

1. option3() Purpose:

Allows the user to add a price to an existing

ticket.

Input Parameters:

None

Output Values:

None

Exceptions:

None

Example Usage: ticket::option3();

1. add\_price\_tofile(ticket \*ptr) Purpose:

Adds a price to an existing ticket in the file.

Input Parameters:

* ptr: Pointer to a ticket object with the ticket number and price.

Output Values:

* Returns 0 if successful.

Exceptions:

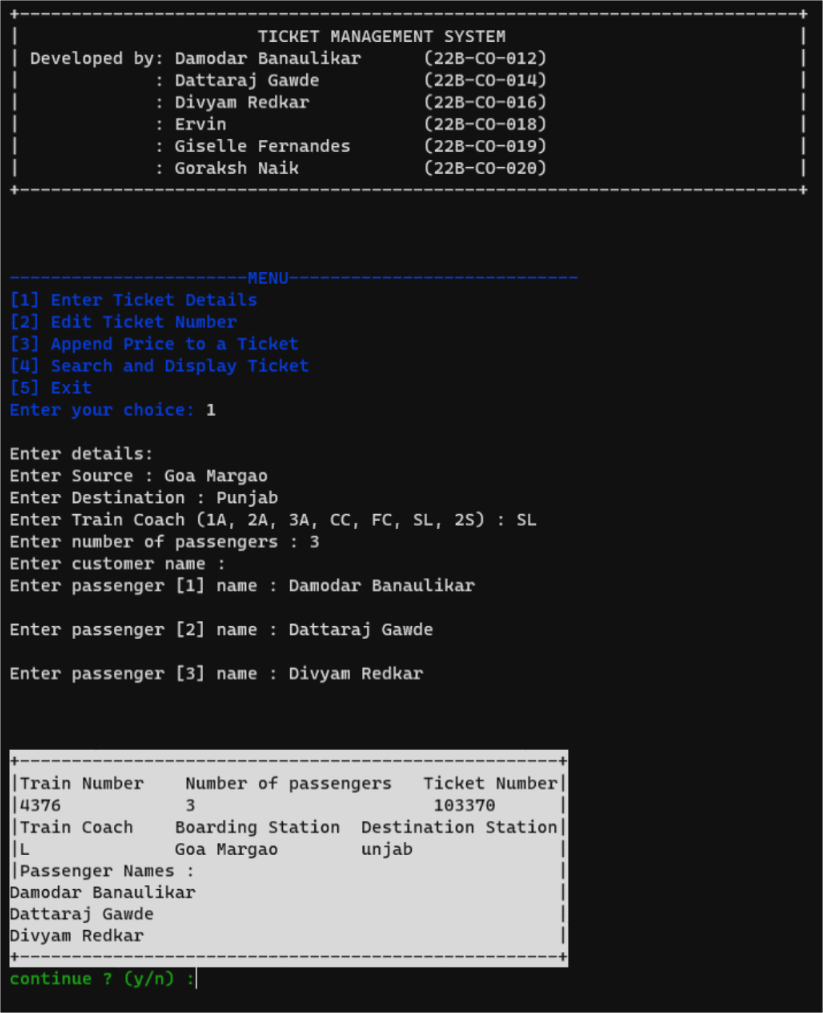
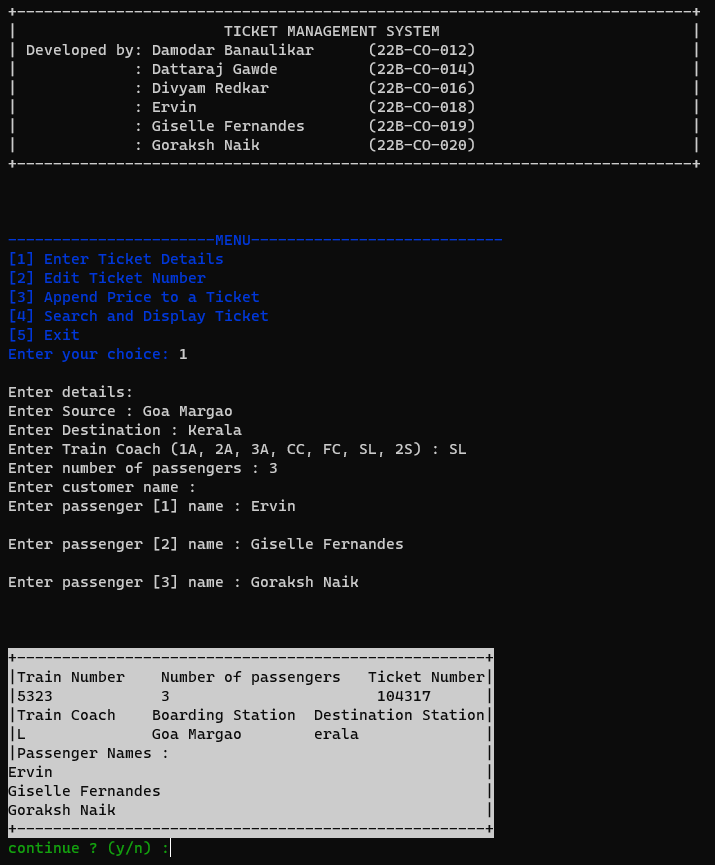
* Displays an error message if the ticket does not exist.

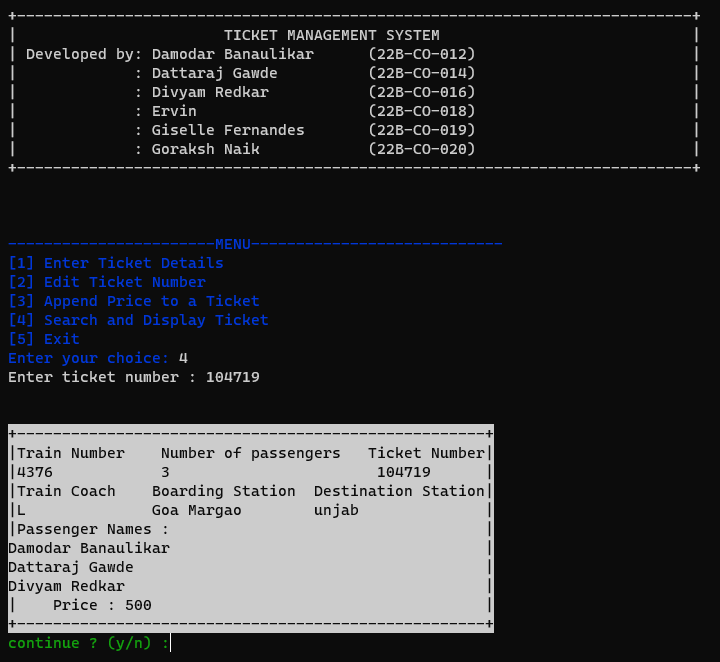
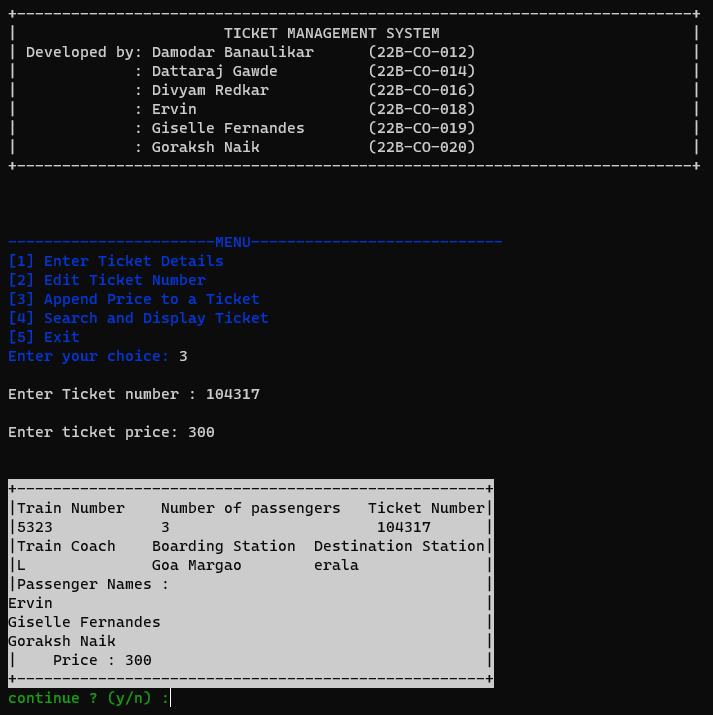
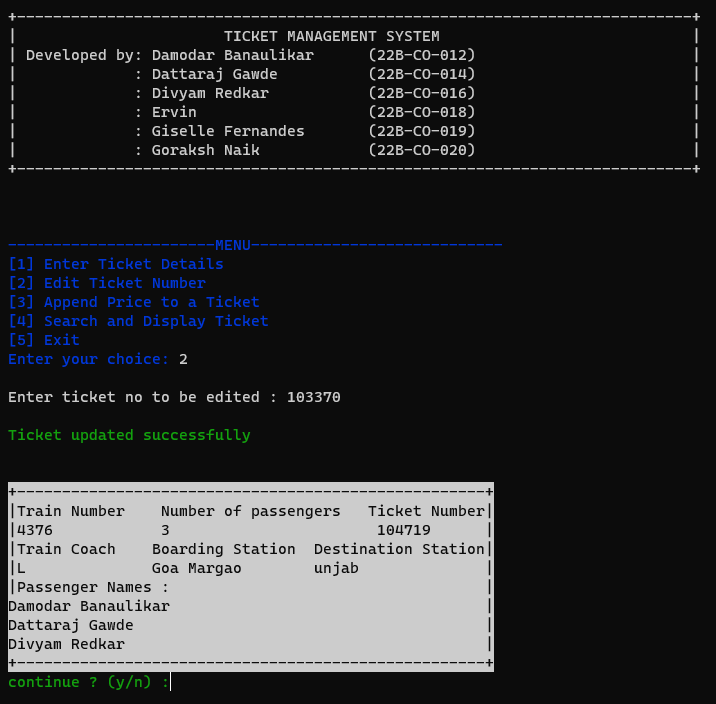
Example Usage:

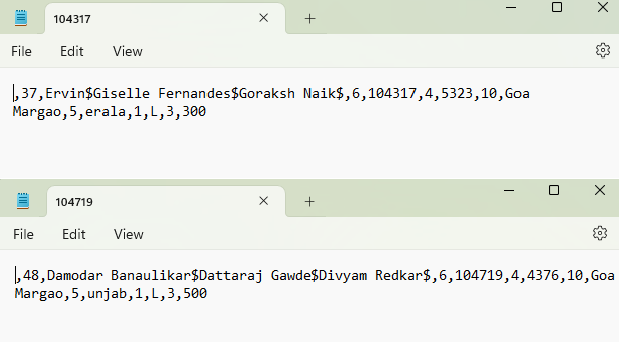
ticket T;

// Populate T with ticket details

int result = ticket::add\_price\_tofile(&T);







Developed by

: Damodar Banaulikar (22B-CO-012)

: Dattaraj Gawde (22B-CO-014)

: Divyam Redkar (22B-CO-016)

: Ervin (22B-CO-018)

: Giselle Fernandes (22B-CO-019)

: Goraksh Naik (22B-CO-020)

**FACULTY: - Prof. AMIT PATIL CLASS: - S.E COMPUTER Prof. NITESH NAIK**