VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgavi – 590018, Karnataka, India

A MINI PROJECT REPORT ON

"TOURISM RESERVATION SYSTEM"

Submitted on partial fulfillment of the academic requirement of the 6th semester

"FILE STRUCTURE Laboratory with Mini Project(18ISL67)"

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

Submitted by

Chethan A (1SJ20IS127)

Under the guidance of :

Mrs. NANDINI S

Assistant Professor

Dept. of ISE, SJCIT

S J C INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CHICKBALLAPUR-562101

2022-23

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust

S.J.C INSTITUTE OF TECHNOLOGY

Department of Information Science & Engineering

Chickballapur-562101

CERTIFICATE

This is to certify that the mini-project work entitled **"TOURISM RESERVATION SYSTEM"**is a work being carried by **CHETHAN A (1SJ20IS127)** in partial fulfillment for the award of **Bachelor of Engineering inInformation Science and Engineering in sixth semester of the Visvesvaraya Technological University, Belgaum** during the year **2023**. It is certified that all corrections/suggestions indicated for Internal Assessment have been in corporated in the report deposited in the department library. The file structure laboratory with mini project report has been approved as it satisfies the academic requirements in respect to sixth semester mini-project work.

Signature of the Guide	**Signature of the HOD**
prof. Nandini S	**prof.Satheesh Chandra Reddy**
Assistant Professor	**Professor and HOD**
Department of ISE, SJCIT	**Department of ISE, SJCIT**

External Viva:

Name of the Examiners: **Signature with Date:**

1. _____
2. _____

ACKNOWLEDGEMENT

With great pride, we would like to convey our gratitude and appreciation to our Institution **"S.J.C Instituteof Technology"** for giving us the required platform for the fulfillment of the Mini project on FS as per the
V.T.U requirements, for the sixth-semester lab on the same subject.

We express our sincere thanks to **Dr. G T RAJU**, Principal of SJCIT**,** Chikkballapur for providing us withexcellent infrastructure to complete the Mini project.

We express whole-hearted gratitude to **Prof. SATHEESH CHANDRA REDDY,** who is the respectable HOD of the Information Science Department**.** We wish to acknowledge his help in making our task easy by providing us with his valuable help and encouragement.

It is our pleasure to thank our guide **Nandini S, Associate professor, Department of Information Science and Engineering, SJCIT** for her guidance, encouragement, and valuable suggestion from the beginning of the mini-project work till its completion without which this mini-project work would not have been accomplished.

And last but not the least, we would be very pleased to express our hearts full thanks to the teaching and non-teaching staff of the Department of Information Science and Engineering, SJCIT for their motivationand support.

We also thank all those who extended their support and co-operation for the successful completion the of Mini-project.

CHETHAN A (1SJ20IS127)

ABSTRACT

The tourism industry has experienced significant growth over the years, with an increasing number of individuals and groups seeking seamless and efficient ways to make reservations for travel accommodations, transportation, and tourist attractions. In order to address the challenges associated with manual booking processes, this project proposes the development of a Tourism Reservation System using file structures.

The objectives of this system is to provide a user-friendly platform that enables tourists to make reservations for various travel-related services in a a convenient and efficient manner. The system will utilize file structures to organize and store data, ensuring fast and reliable access to reservation information.

By implementing the Tourism Reservation System, tourism business can streamline their booking processes, improve customer satisfaction, and optimize their operations. The system will provide a centralized platform that simplifies reservation management for both tourists and services providers, contributing to the growth and development of the tourism industry as a whole.

TABLE OF CONTENTS

CHAPTER - 1

INTRODUCTION

1.1 Introduction to File Structure

A file structure is a combination of representations for data in files and of operations for accessing the data. A file structure allows applications to read, write, and modify data. It might alsosupport finding the data that matches some search criteria or reading through the data in some particular order. An improvement in file structure design may make an application hundreds of times faster. The details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications. Disks can pack thousands of megabytes. Disks are very slow in accessing data compared with memory. Hence file structures came to light where we can access data quickly. A File Structure is a combination of representation of data in files and the operations for accessing the data.

1.2 History of File Structure Design

Early work with files presumed that files were on tape, since most files were. Access was sequential, and the cost of access grew in direct proportion, to the size of the file. As files grew intolerably large for unaided sequential access and as storage devices such as hard disks became available, indexes were added to files. The indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly. With key and pointer, the user had direct access to the large, primary file. But simple indexes had some of the same sequential flaws as the data file, and as the indexes grew, they too became difficult to manage, especially for dynamic files in which the set of keys changes.

In the early 1960's, the idea of applying tree structures emerged. But trees can grow very unevenly as records are added and deleted, resulting in long searches requiring many disk accesses to find a record.

In 1963, researchers developed an elegant, self-adjusting binarytree structure, called AVL tree, for

data in memory. The problem was that, even with a balanced binary tree, dozens of accesses were required to find a record in even moderate-sized files. A method was needed to keep a tree balanced when each node of thee tree was not a single record , as in a binarytree , but a file block containing dozens, perhaps even hundreds, ofrecords. Hashing is agood way to get what we want with a single request, with files that do not change size greatly over time. Hashed indexes were used to provide fast access to files. But until recently, hashing did not work well with volatile, dynamic files. Extendible dynamic hashing can retrieve information with 1 or atmost 2 disk accesses, no matter how big the file became.

1.3 About the file

When we talk about a file on disk or tape, we refer to a particular collection of bytes stored there . The file , when the word is used in this sense, physically exists . A disk drive may contain hundreds, even thousands of these physical files. From the standpoint of an application program, a file is somewhat like a telephone line connection to a telephone network. The program can receive bytes through this phone line or send bytes down it, but it knows nothing about where these bytes come from or where they go. The program knows only about its end of the line. Even though there may be thousands of physical files on a disk,a single program is usually limited to the use of only about 20 files.

The application program relies on the OS to take care of the details of the telephone switching system. It could be that bytes coming down the line into the program originate from a physical file they come from the keyboard or some other input device. Similarly, bytes the program sends down the line might end up in a file, or they could appear on the terminal screen or some other output device. Although the program doesn't know where the bytes are coming from or where they are going, it does know which line it is using. This line is usually referred to as the logical file, to distinguish it fromthe physical files on the disk or tape.

1.4 Application of File Structure

Relative to other parts ofa computer, disks are slow. 1 can pack thousands of megabytes on a

disk that fits into a notebook computer.

The time it takes to get information from even relatively slow electronic random-access memory (RAM) is about 120 nanoseconds. Getting the same information from a typical disk takes 30 milliseconds. So, the disk access is a quarter of a million times longer than a memory access. Hence, disks are very slow compared to memory. On the other hand, disks provide enormous capacity at much less cost than memory. They also keep the information stored on them when they are turned off. Tension between a disk's relatively slow access time and its enormous, non-volatile capacity, is the driving force behind file structure design. Good file structure design will give us access to all the capacity without making our applications spend a lot of time waiting for the disk.

1.5 Introduction to Tourism Reservation System

This application is based on railway system where the application provides access to differentkind of user like admin, public, station manager, employee. The main aim of the project is to developa c++ application which is used to maintain a railway details. In the nation like Indiawhere daily thousands of trains run through hundreds of stations this project can make the task easy by converting all the records which were traditionally maintained in some ledger into well organized files with faster access using the primary indexing. The admin need to login through hislogin credentials where he will be given access to Admin section . the admin section has access to Train section where he can add new Train, he can view all details of the Trains, he can remove aTrain detail. the admin section has the access to Station section where he can perform the operationsof adding , removing of new Railway Station. He has the access to add removethe employeedetailsof the Railway Department . He can add and remove the platform details of the Railway Station. theapplication also has the user section where the public can view the particular Train detail by usingits id or he can view the details of all the train with its timings . the Station manager can view Thedetailsrelatedtothe trains including its timings and the route. The railway employee can add all thedetails related to the train which is running on a particular at particular time

CHAPTER - 2

FUNDAMENTALS

This chapter contains methodology, file operations for the implementationofclient serverapplication in the resort management system.

2.1 Physical files and logical files

Physical files contain the actual data that is stored, and a description of how data is to be presented to or received from a program. They contain only one record format, and one or more members. A physical file can have a keyed sequence access path. This means that data is presented to an ILE C/C++ program in a sequence that is based on one or more key fields in the file. Logical files do not contain data. They contain a description of records that are found in one or more physicalfiles. A logical file is a view or representation of one or more physical files. Logical files that containmore than one format are referred to as multi-format logical files.

2.2 File Operations

2.2.1 Opening files:

opening a file makes it ready for use by the program. We are positioned at the beginning of the file and are ready to start reading or writing. The Open function takes two required arguments and a third argument is optional. fd = open (filename, flags[pmode]); fd: The file descriptor or the logical file identifier; filename: A character string containing the physical file name; flags: controls the operation of the open function ,determining opens an determining whether existing file for reading or writing; pmode: specifies protection mode for file.

2.2.2 Closing files:

when a file is closed, the logical file name is available for use with another file. It ensures that everything has been written to the file. The close statement is needed to protect against data loss and to free up logical filenames for reuse

2.2.3 Reading :

It makes file processing an input/output.

read(source_file,dest_addr,size);

source_file: logical file name from where data is read;

dest_addr: the first address of the memory block where the data is stored;

dest_addr: the first address of the memory block where the data is stored;

size: Byte count specifying how much information to read from the file.

write(dest_file, source_addr, size);

dest_addr: the first address of the memory block where the data is stored;

source_file: logical file name that is used for sending data; size: the number of bytes to be written.

2.2.4 Seeking :

An object of type fstream has two file pointers: a get pointer for input and a put pointer for output. Two functions are supplied for seeking: seekg which moves the get pointer and seekp whichmoves the put pointer.

file.seekg(byte_offset,origin) file.seekp(byte_offset,origin) value of origin comes from class ios and the values are :

ios::beg(beginning of file) ios::cur(current position) ios::end(end of file)

2.2.3 Fields :

The basic unit of data is field which contains a single data value. A field is the smallest logically meaningful unit of information in a file.

2.2.3 Field Structure :

ways of adding structure to files to maintain the identity of fields.

1. Force the fields into predictable length.

2. Begin each field with a length indicator.

3. Place a delimiter at the end of each field to separate it fromthe next field.

4. Use a "keyword=value" expression to identify each field and its contents.

2.2.3 Record Structures :

A record can be defined as a set of fields that belong together when the file is viewedin terms of a higher level of organization. A record in a file represents a structured data object.

8. Record Access:

1. Sequential access: It involves reading through the file, record by record, looking for a record with particular key. 2. Direct access: It allows to seek directly to the beginning of the record and read it. No matter how large the file is, we can access any record with a single seek.

2. 2.3 Record Access :

- Sequential access: It involves reading throughthe file, record byrecord, looking for a record with particular key.

- Direct access: It allows to seek directly to the beginning of the record and read it. No matter how large the file is, we can access any record with a single seek.

CHAPTER - 3

REQUIREMENT SPECIFICATION

A high-level requirements specification is required. The purpose of the requirements analysis is to identify requirements for the proposed system. The emphasis is on the discovery of user requirements. A computerized way of handling information about property and users details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a menu driven console-based application whose requirements are mentioned in this section.

The specific requirements of Tourism Reservation System are stated as follows:

3.1 Software requirements

- **Software's used:**
    - Operating System – Windows OS
    - Software – Code::Blocks

3.2 Hardware Requirements

- **Hardware Components used:**
    - CPU – Intel core i3 and Above with a processor above 500 MHz
    - RAM – 4GB and Above (8GB preferred)
    - Hard Disk – 2 GB free space
    - Systemtype – 32-bit or 64-bit operating system
    -

3.2 Technology Used:

3.2.1 C++ programming language:

C++ is a multi-paradigm programming language that supports object-oriented programming (OOP), created by Bjarne Stroustrupin 1983 at Bell Labs, C++ is an

extension(superset) of C programming and the programs are written in C language can run in C++ compilers. An interpreter is a computer program that directly executes, i.e . performs, instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program. A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writeslanguage statements in a language such as Pascal or C one line at a time using an editor.

3.2.2 C++ Features:

C++ is object-oriented programming language and it is a very simple and easy language; it is the enhanced form of C programming language. this language has following features and here we discuss some important features of C++.

• Simple: Every C++ program can be written in simple English language so that it is very easy to understand and developed by programmer.

• Platform dependent: A language is said to be platform dependent whenever the program is executing in the same operating system where that was developed and compiled but not run and execute on another operating system. C++ is platform dependent language.

• Portability: It is the concept of carrying the instruction from one system to another system. In C++ Language. Cpp file contain source code, we can edit also this code. .exe file contain application, only we can execute this file. When we write and compile any C++ program on window operating system that program easilyrun on other window-based system.

• Powerful: C++ is a very powerful programming language, it has a wide verity of data types, functions, control statements, decision making statements, etc.

• Object oriented Programming language: This main advantage of C++ is; it is object-oriented programming language. It follows concept of oops like polymorphism, inheritance, encapsulation, abstraction.

• Case sensitive: C++ is a case sensitive programming language.

C, C++, java, .net are sensitive programming languages.] otherwise it is called as case insensitive programming language [Example HTML, SQL is case insensitive programming languages].

• Compiler based: C++ is a compiler-based programming language that means without compilation no C++ program can be executed. First, we need compiler to compile our program and then execute.

• Syntax based language: C++ is a strongly tight syntax-based programming language. If any language, follow rules and regulation very strictly known as strongly tight syntax-based language. Example C, C++, Java, .net etc. If any language not follow rules and regulation very strictly known as loosely tight syntax-based language. Example HTML.

• Efficient use of pointers: Pointers is a variable which hold the address of another variable, pointer directly direct access to memory address of any variable due to this performance of application is improve. In C++ language also concept of pointers are available.

3.2.2 Code::Blocks

Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided byplugins. The first stable release was on February 28, 2008, with the version number changed to 8.02. The versioning scheme was changed to that of Ubuntu, with the major and minor number representing the year and month of the release. Code::Blocks supports multiple compilers , including GCC, Watcom , LCC and the Intel C++ Compiler. Although the IDE was designed for the C++ language, there is some support for other languages, including Fortran

and D. A plug-in system is included to support other programming languages. Code::Blocks uses a custom build system, which stores its information in XML-based project files. It can optionally use external make files, which simplifies interfacing with projects using the GNU.

CHAPTER - 4

SYSTEM DESIGN

Systems design could be seen as the application of systems theory to System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements development.

4.1 Data flow diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyze an existing system or model a new one.

The symbols depict the four components of data flow diagrams

- **External entity**: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

- **Process**: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process.

- **Data store**: files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label.

- **Data flow**: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labelled with a short data name.

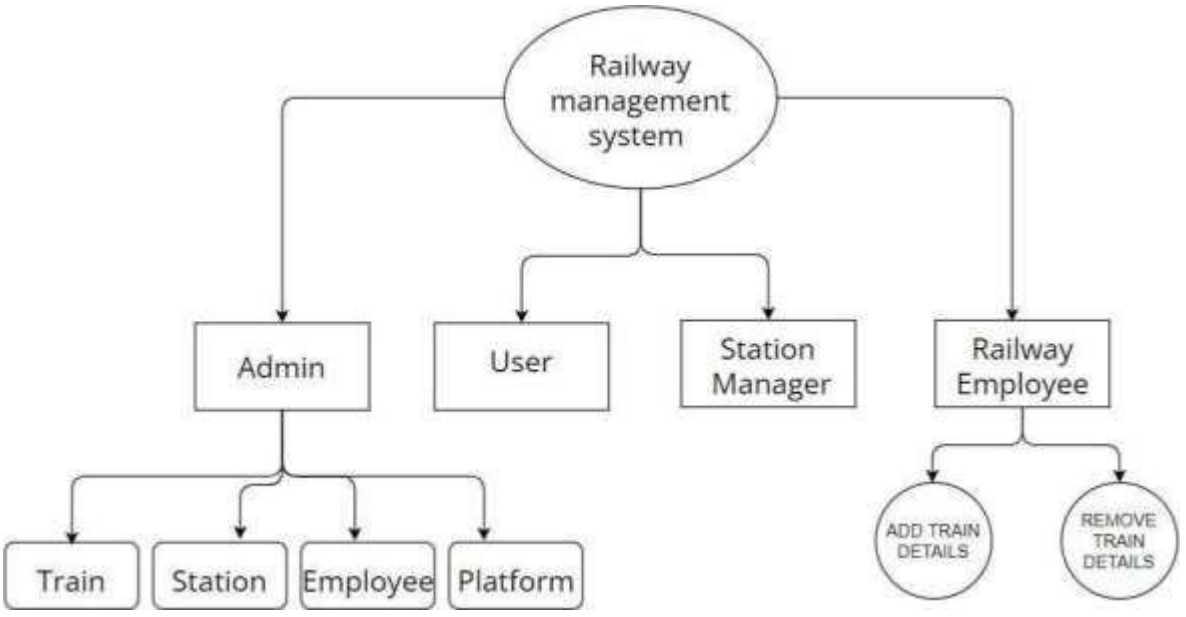

Fig.4.1: Tourism Reservation System DFD

The above figure shows data flow diagram of Tourism Reservation System. The data flow diagrams concentrate mainly on overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by awide audience, including stakeholders, business analysts, data analysts and developers.

CHAPTER - 5

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods. Implementation is the most important phase. The most critical stage in achieving a successful new system is giving the users confidence that the new system will work and be effective. Any system developed should be secured and protected against possible hazards.

Component testing is a method where testing of each component in an application is done separately. Component testing is also known as module, unit or program testing. It finds the defects in the module and verifies the functioning of software. Our project is console-based, so all the implementation of our project is in console

5.1 Console Window:

The console window consists of 4 options with different kinds of users

The options in main screen

- Admin
- User

The main console has 2 options each having 5 more operations on file

- Admin
  - New Trips
  - Displaya Trip
  - Displaya Reservations
  - Update Trip
  - Exit
- User
  - New Reservation
  - Search Reservation
  - Cancel Reservation

The New trips has takes 5 inputs

- Destination code
- Destination Place
- Country of Destination
- Number of days of the Trip
- Cost of the Trip

The New Reservation takes 5 inputs

- Name of the user
- Contact Number
- No. of Tourists
- Destination code
- Date Of Journey

5.2 Code Snippets:

5.2.1 Main.cpp

```
#include<iostream>
#include<fstream>
#include<string.h>
        using namespace std;

    char adminf[20]="Trip.txt";
    char userf[20]="Reservation.txt";

    struct tour{
        char ad_place[20],ad_code[20],ad_country[20],ad_days[20],ad_prices[20];
        char
    us_phone[20],us_name[20],us_people[20],us_date[20],us_status[20],us_dcode[20];
    };

    class Tourist{
         public:
           void ad_pack(tour r);
                tour ad_unpack(char a[]);
                void  ad_writedata();
                void ad_display();
                void ad_modify();
                void us_pack(tour r);
                tour us_unpack(char a[]);
                void  us_writedata();
                void us_display();
                void us_search();
                void us_modify();
    };

    // ADMIN PANEL
    void Tourist ::ad_pack(tour r){
        fstream fp;
```

```
        fp.open(adminf,ios::out|ios::app);
        if(!fp){
                cout<<"\nCannot open file";
                exit(0);
        }
        char buff[45];
        strcpy(buff,r.ad_code);
        strcat(buff,"|");
        strcat(buff,r.ad_place);
        strcat(buff,"|");
        strcat(buff,r.ad_country);
        strcat(buff,"|");
        strcat(buff,r.ad_days);
        strcat(buff,"|");
        strcat(buff,r.ad_prices);
        strcat(buff,"|");
        for(int i=0;i<45-strlen(buff);i++)
            strcat(buff,"|");
        fp<<buff<<endl;
        fp.close();
    }

    tour Tourist::ad_unpack(char buff[]){
            tour r;
            int i=0,j=0;
            while(buff[j]!='|')
                r.ad_code[i++]=buff[j++];
            r.ad_code[i]='\0';
            i=0;
            j++;
            while(buff[j]!='|')
                r.ad_place[i++]=buff[j++];
            r.ad_place[i]='\0';
            i=0;
```

```
        j++;
        while(buff[j]!='|')
            r.ad_country[i++]=buff[j++];
        r.ad_country[i]='\0';
        i=0;
        j++;
        while(buff[j]!='|')
            r.ad_days[i++]=buff[j++];
        r.ad_days[i]='\0';
        i=0;
        j++;
        while(buff[j]!='|')
            r.ad_prices[i++]=buff[j++];
        r.ad_prices[i]='\0';
        return(r);
}

void Tourist::ad_writedata(){
    tour s;
    cout<<"Enter the Destination Code\nD-";
    cin>>s.ad_code;
    cout<<"Enter the Destination Place\n";
    cin>>s.ad_place;
    cout<<"Enter the Country of the Destination\n";
    cin>>s.ad_country;
    cout<<"Enter the Number of days of the Trip\n";
    cin>>s.ad_days;
    cout<<"Enter the Cost of the trip\n";
    cin>>s.ad_prices;
    ad_pack(s);
}

void Tourist::ad_display(){
```

```
        fstream fp;
        char buff[45];
        tour r;
           fp.open(adminf,ios::in);
   if(!fp){
                cout<<"\nCannot open file";
                exit(0);
   }
   cout<<"----------------------------------------------------------------------------------------
\n";
   cout<<"Sr No.\t\tDEST. CODE\tPLACE\t\tCOUNTRY\t\tDAYS\t\tPRICES\n";
   cout<<"----------------------------------------------------------------------------------------
\n";
   int c=1;
   while(1){
      fp.getline(buff,45);
      if(fp.eof())break;
                r=ad_unpack(buff);
      cout<<c++<<"\t\t"<<r.ad_code<<"\t\t"<<r.ad_place<<"\t\t"<<r.ad_country<<"\t\t"
<<r.ad_days<<"\t\t"<<r.ad_prices<<endl;
   }
   fp.close();
   return;
}


void Tourist ::ad_modify(){
   fstream fp;
   char ad_code[15],buff[45];
   int i,j,ch;
   tour s[100];
   fp.open(adminf,ios::in);
   if(!fp){
                cout<<"\nCannot open file";
```

```
                exit(0);
}
cout<<"\nENTER THE DESTINATION CODE TO BE MODIFIED: ";
cin>>ad_code;
i=0;
    while(1){
  fp.getline(buff,45);
  if(fp.eof())break;
  s[i]=ad_unpack(buff);
  i++;
}
    for(j=0;j<i;j++){
            if(strcmp(s[j].ad_code,ad_code)==0){
          cout<<"VALUES OF THE TRIP\n";
          cout<<"\nDestination Code: "<<s[j].ad_code;
          cout<<"\nDestination Place: "<<s[j].ad_place;
          cout<<"\nDestination Country: "<<s[j].ad_country;
          cout<<"\nNo of days of Trip: "<<s[j].ad_days;
          cout<<"\nTotal Cost of the Trip: "<<s[j].ad_prices;
          cout<<"\nWhat you want to Update: ";
                  cout<<"\n\n\t\tEnter 1 for changing DESTINATION PLACE\n";
                  cout<<"\t\tEnter 2 for changing DAYS OF THE TRIP\n";
                  cout<<"\t\tEnter 3 for changing COST OF THE TRIP\n";
                  cout<<"\t\t\t";
                  cin>>ch;
                  switch(ch){
        case 1:
          cout<<"Destination Place: ";
          cin>>s[j].ad_place;
          cout<<"Destination Country: ";
          cin>>s[j].ad_country;
          break;
        case 2:
          cout<<"No of days of Trip: ";
```

```
                cin>>s[j].ad_days;
                break;
            case 3:
                cout<<"Total Cost of the Trip: ";
                cin>>s[j].ad_prices;
                break;
                        }
                break;
            }
    }
        if(j==i){
      cout<<"\n TRIP NOT FOUND";
      return;
    }
    fp.close();
    fstream fd;
    fd.open(adminf,ios::out|ios::trunc);
    if(!fd){
      cout<<"\nFile Not Found";
      exit(0);
    }
        for(j=0;j<i;j++)
     ad_pack(s[j]);
        fd.close();
}


// USER PANEL


void Tourist ::us_pack(tour r){
    fstream fp;
    fp.open(userf,ios::out|ios::app);
    if(!fp){
```

```
        cout<<"\nFile Not Found";
                exit(0);
    }
    char            buff[45];
    strcpy(buff,r.us_name);
    strcat(buff,"|");
    strcat(buff,r.us_phone);
    strcat(buff,"|");
    strcat(buff,r.us_people);
    strcat(buff,"|");
    strcat(buff,r.us_dcode);
    strcat(buff,"|");
    strcat(buff,r.us_date);
    strcat(buff,"|");
    strcat(buff,r.us_status);
    strcat(buff,"|");
    for(int i=0;i<45-strlen(buff);i++)
        strcat(buff,"|");
    fp<<buff<<endl;
    fp.close();
}


tour Tourist::us_unpack(char buff[]){
        tour r;
        int i=0,j=0;
        while(buff[j]!='|')
            r.us_name[i++]=buff[j++];
        r.us_name[i]='\0';
        i=0;
        j++;
        while(buff[j]!='|')
            r.us_phone[i++]=buff[j++];
        r.us_phone[i]='\0';
```

```
            i=0;
            j++;
            while(buff[j]!='|')
        r.us_people[i++]=buff[j++];
    r.us_people[i]='\0';
    i=0;
    j++;
    while(buff[j]!='|')
        r.us_dcode[i++]=buff[j++];
    r.us_dcode[i]='\0';
    i=0;
    j++;
    while(buff[j]!='|')
        r.us_date[i++]=buff[j++];
    r.us_date[i]='\0';
    i=0;
    j++;
    while(buff[j]!='|')
        r.us_status[i++]=buff[j++];
    r.us_status[i]='\0';
    return(r);
    }
    void Tourist::us_writedata(){
       tour r;
       cout<<"ENTER YOUR NAME: ";
       cin>>r.us_name;
       cout<<"ENTER YOUR CONTACT NO.: ";
       cin>>r.us_phone;
       cout<<"ENTER NO. OF TOURIST: ";
       cin>>r.us_people;
       cout<<"ENTER THE DESTINATION CODE: D-";
       cin>>r.us_dcode;
       cout<<"ENTER DATE OF JOURNEY (DD/MM/YY): ";
       cin>>r.us_date;
```

```
    cout<<"\nENTER 'Confirm'TO CONFIRM YOUR RESERVATION: ";
    cin>>r.us_status;
    us_pack(r);
}

void Tourist::us_display(){
        fstream fp;
        char buff[45];
        tour r;
    fp.open(userf,ios::in);
    if(!fp){
                cout<<"\nCannot open file";
                exit(0);
    }
    cout<<"----------\n";
    cout<<"Sr No.\tNAME\t\tCONTACT NO\t\tTOTAL PEOPLE\t\tDATE\t\tDEST.
CODE\t\tSTATUS\n";
    cout<<"----------\n";
    int c=1;
    while(1){
       fp.getline(buff,45);
       if(!fp.eof()){
                r=us_unpack(buff);
       }else{
       break;
       }

cout<<c++<<"\t"<<r.us_name<<"\t\t"<<r.us_phone<<"\t\t"<<r.us_people<<"\t\t\t"<<r.us_da
te<<"\t\t"<<r.us_dcode<<"\t\t"<<r.us_status<<endl;
    }
    fp.close();
    return;
```

```
}

void Tourist ::us_modify(){
    fstream fp;
    char us_name[15],buff[45];
    int i,j;
    tour s[100];
    fp.open(userf,ios::in);
    if(!fp){
                cout<<"\nFile not Found";
                exit(0);
    }
    cout<<"\nEnter Your name to cancel your reservation: ";
    cin>>us_name;
    i=0;
        while(1){
       fp.getline(buff,45);
       if(fp.eof())break;
       s[i]=us_unpack(buff);
       i++;
    }
        for(j=0;j<i;j++){
       if(strcmp(s[j].us_name,us_name)==0){
                        cout<<"\nYour Reservation Details are:\n";
                        cout<<"\nName: "<<s[j].us_name;
                        cout<<"\nContact No.: "<<s[j].us_phone;
                        cout<<"\nNo.of People: "<<s[j].us_people;
                        cout<<"\nDestination: "<<s[j].us_dcode;
                        cout<<"\nDate of Journey: "<<s[j].us_date;
                        cout<<"\nStatus: "<<s[j].us_status;
                        cout<<"\n\nENTER 'Cancel' TO CANCEL YOUR TICKET: \n";
                        cin>>s[j].us_status;
                        cout<<"\nYour Reservation has been Cancelled\n";
```

```
                        cout<<"\n\nName: "<<s[j].us_name;
                        cout<<"\nContact No.: "<<s[j].us_phone;
                        cout<<"\nNo.of People: "<<s[j].us_people;
                        cout<<"\nDestination: "<<s[j].us_dcode;
                        cout<<"\nDate of Journey: "<<s[j].us_date;
                        cout<<"\nStatus: "<<s[j].us_status;
            break;
          }
        }
            if(j==i){
          cout<<"\nRECORD NOT FOUND";
          return;
        }
        fp.close();
        fstream fd;
        fd.open(userf,ios::out|ios::trunc);
        if(!fd){
          cout<<"\nCannot open file";
          exit(0);
        }
            for(j=0;j<i;j++)
          us_pack(s[j]);
            fd.close();
    }

    void Tourist::us_search(){
            fstream fp;
            char us_name[15],buff[45];
            int i,j;
            tour s;
        fp.open(userf,ios::in);
        if(!fp){
                        cout<<"\nCannot open file";
                        exit(0);
```

```
            }
            cout<<"\nENTER THE NAME TO BE SEARCHED: ";
            cin>>us_name;
        i=0;
            while(1){
          fp.getline(buff,45);
          if(fp.eof())break;
          s=us_unpack(buff);
          if(strcmp(s.us_name,us_name)==0){
                        cout<<"\n RESERVATION FOUND\n";
                        cout<<"\nNAME: "<<s.us_name;
                        cout<<"\nCONTACT NO.: "<<s.us_phone;
                        cout<<"\nNO. OF PEOPLE: "<<s.us_people;
                        cout<<"\nDATE OF JOURNEY: "<<s.us_date;
                        cout<<"\nSTATUS:  "<<s.us_status;
                        return;
                    }
          }
          cout<<"\nRESERVATION NOT FOUND";
          return;
      }

      int main(){
          cout<<"                    \t\t WELCOME TO TOURIST RESERVATION
      SYSTEM\n\n\n";
          cout<<"                        \t   TOURIST RESERVATION SOFTWARE\n\n\n";
          cout<<"                         \t     PRESS ENTER TO CONTINUE...\n\n";
          if(cin.get()=='\n'){
            system("cls");
            int choice,ch1,ch2;
            Tourist obj;
            cout<<"\n Who are You?\n";
            cout<<"1. ADMIN\n";
    cout<<"2. USER\n\n";
```

```
cout<<"Enter any other key for exit\n\n\n";
cout<<"Enter Your Option\n";
cin>>choice;
switch(choice){
  case 1:
    system("cls");
    cout<<"WELCOME ADMIN\n";
    cout<<"Enter your option\n";
    cout<<"1. New trips\n";
    cout<<"2. Display a Trip\n";
    cout<<"3. Display a Reservation\n";
    cout<<"4. Update Trip\n";
    cout<<"Enter any other key for exit\n\n";
    while(1){
      cout<<"\nENTER UR CHOICE: ";
      cin>>ch1;
      switch(ch1){
        case 1:
          obj.ad_writedata();
          break;
        case 2:
          obj.ad_display();
          break;
        case 3:
          obj.us_display();
          break;
        case 4:
          obj.ad_modify();
          break;
        default:
          exit(0);
  }
}
  case 2:
```

```
        system("cls");
        cout<<"WELCOME USER\n";
        cout<<"Enter your option\n";
                cout<<"1. New Reservation\n";
                cout<<"2. Search Reservation\n";
                cout<<"3. Cancel Reservation\n";
                cout<<"Enter any other key for exit\n\n\n";
                while(1){
                   cout<<"\nENTER UR CHOICE: ";
                   cin>>ch2;
                   switch(ch2){
                      case 1:
                         cout<<"..........";
                         cout<<"\n Choose Your destination\n";
                         cout<<"..........\n";
                         obj.ad_display();
                         cout<<"\n";
                         obj.us_writedata();
                         break;
                      case 2:
                         obj.us_search();
                         break;
                      case 3:
                         obj.us_modify();
                         break;
                      default:
                         exit(0);
                   }
                }
             default:
                exit(0);
          }
       }
    }
```

Chapter - 6

TESTING

6.1 Testing

The implementation phase of software development is concerned with translating design specification into source code. The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straight forward as possible. Simplicity, clarityand elegance are the hallmark of good programs, obscurity, cleverness, and complexity are indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages.

The implementation team should be provided with a well-defined set of software requirement, an architectural design specification, and a detailed design description. Each team member must understand the objectives of implementation.

CHAPTER - 7

SNAPSHOTS

This section describes the screens of the "TOURISM RESERVATION SYSTEM". The snapshotsare shown below for each module.

7.1 Snapshot

7.1.1 Welcome page for Tourism Reservation System

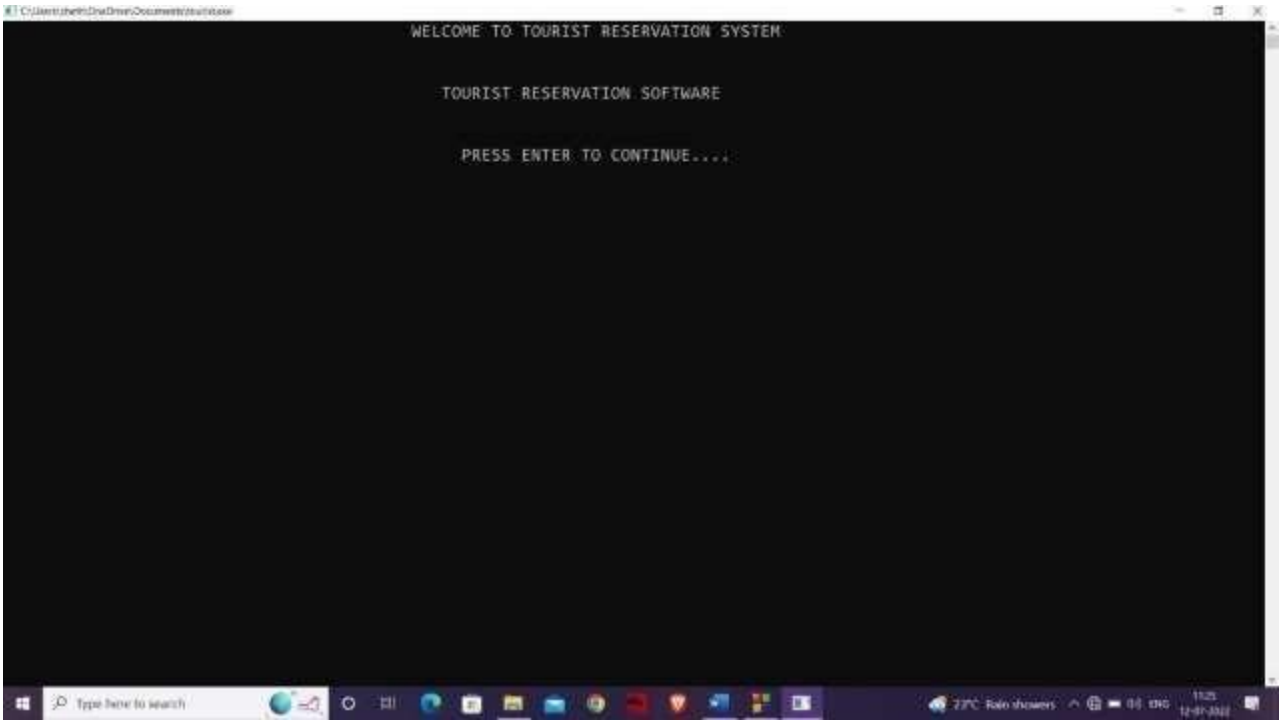

Figure 7.1.1 Welcome page for Tourism Reservation System

This is the first page of the project. This is the welcome page of our project, Tourism Reservation System. When the button enter is clicked, it moves to the next page.

7.1.2 Admin section

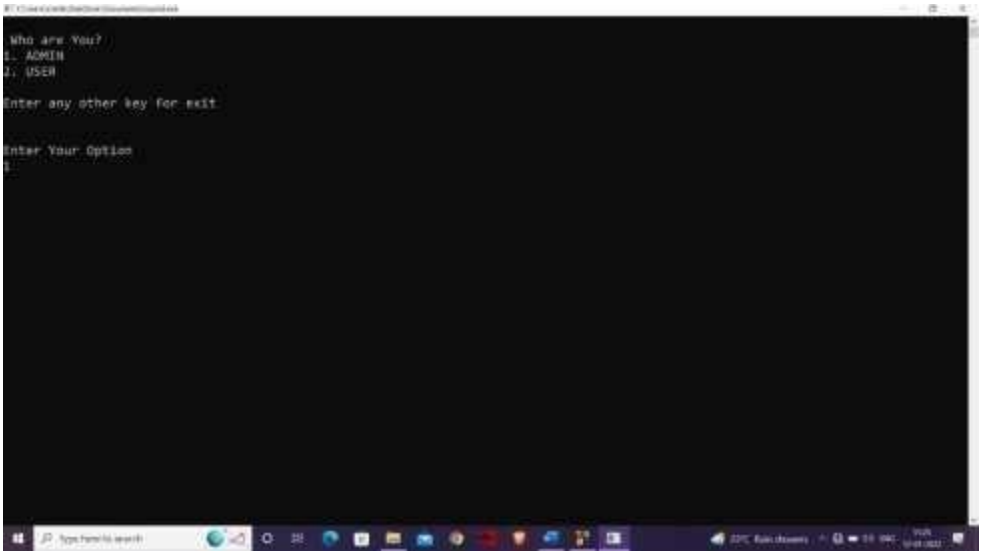

Figure 7.1.2 Admin Section

This page asks whether you are an admin or user. You choose 1 if you are an admin. Option 2 is chosen if you are not an admin.

7.1.3 Add new Trips

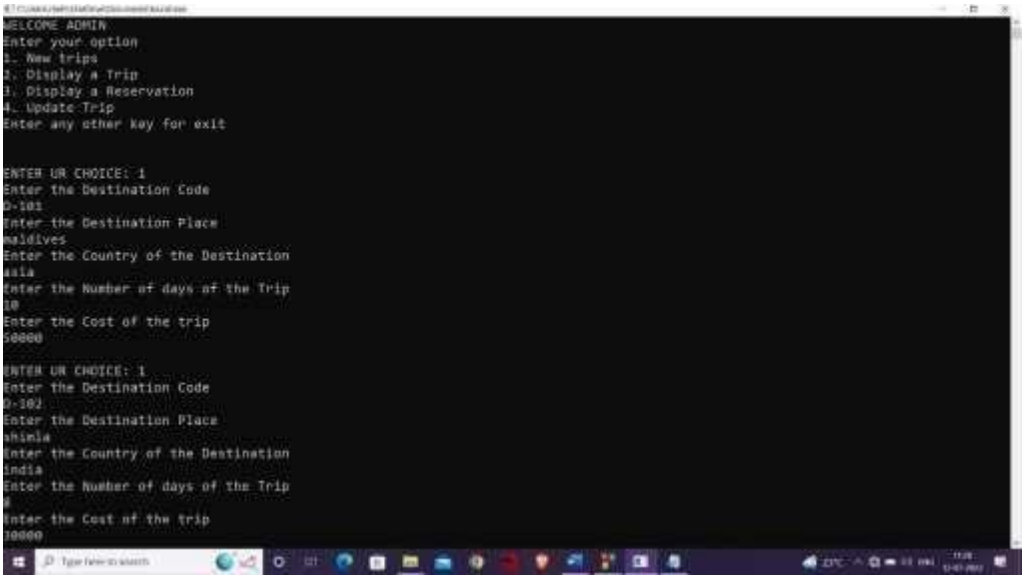

Figure 7.1.3 Add New Trips

Once the admin is logged in, admin plans a new trip by giving information about the trip such as destination code, destination place, country of the destination, number of days of the trip, and cost for the trip.

7.1.4 Display Trips

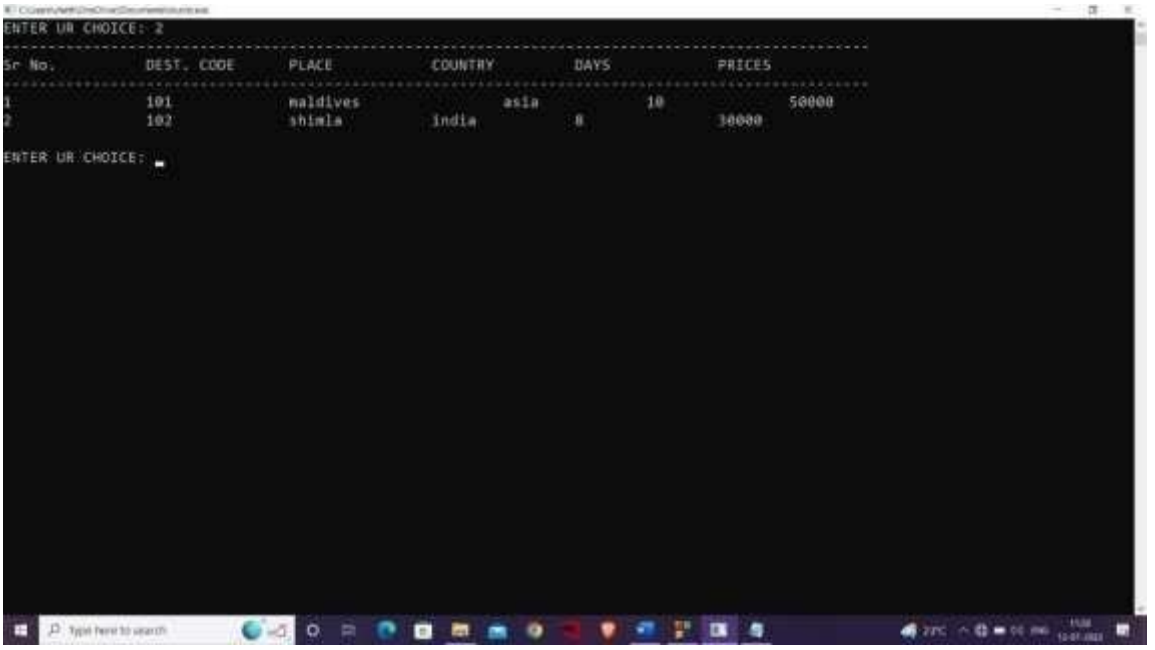

Figure 7.1.4 Display Trips

Once the new trips are added, and if choice 2 is chosen the trips are displayed on the screen.

7.1.5 Update Trip

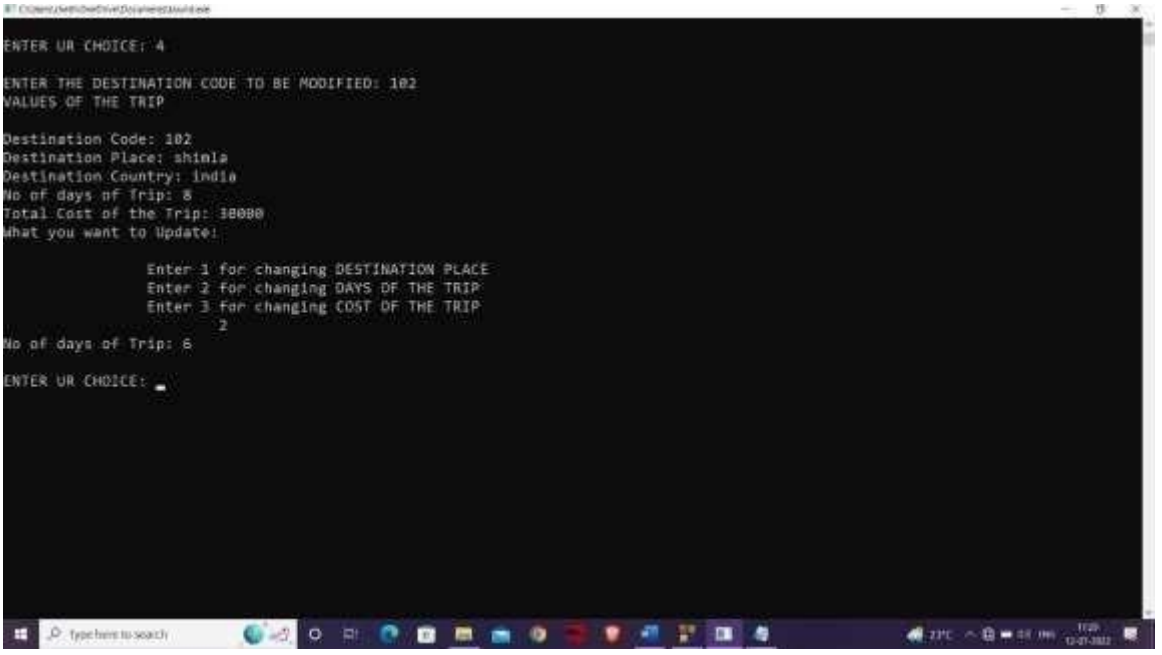

Figure 7.1.5 Update Trip

If any updation is needed for existing trips, the admin chooses choice 4 where he can change the destination place, days of the trip, cost of the trip.

7.1.6 User section

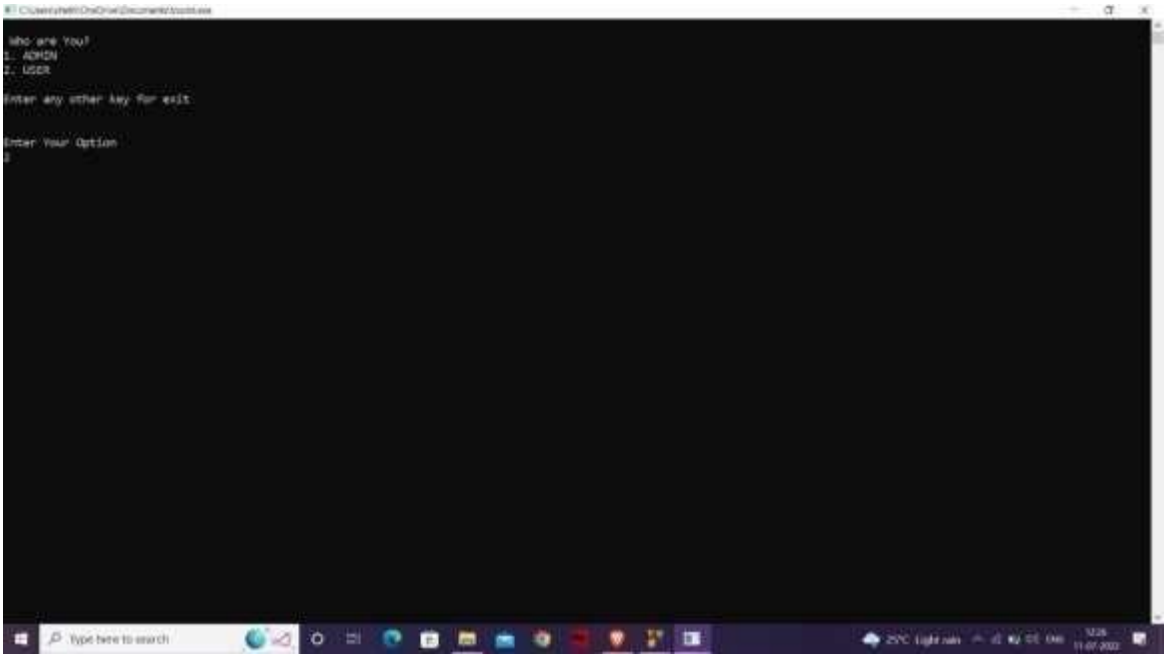

Figure 7.1.6 User Section

This page asks whether you are an admin or user. You choose 2 if you are a user. Option 1 is chosen if you are not a user.

7.1.7 New Reservation

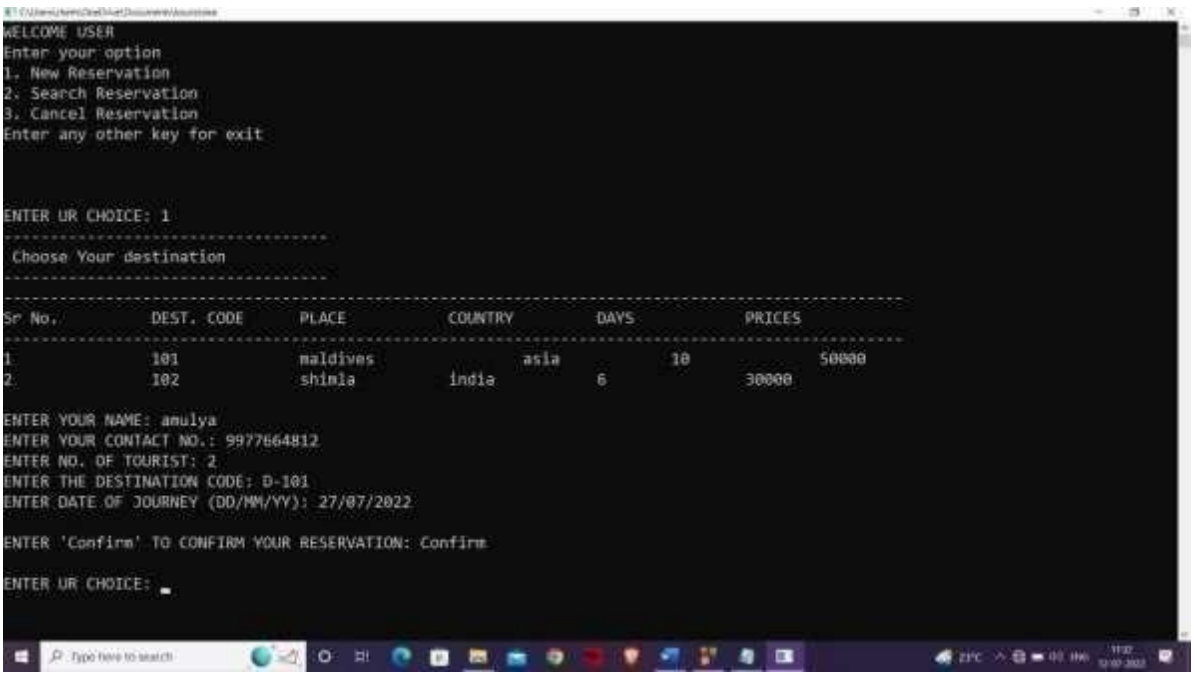

Figure 7.1.7 New Reservation

If the user chooses for new reservation, the user has to provide his personal information like name, contact No., No. of tourists, destination code, date of journey.

7.1.8 Display Reservation

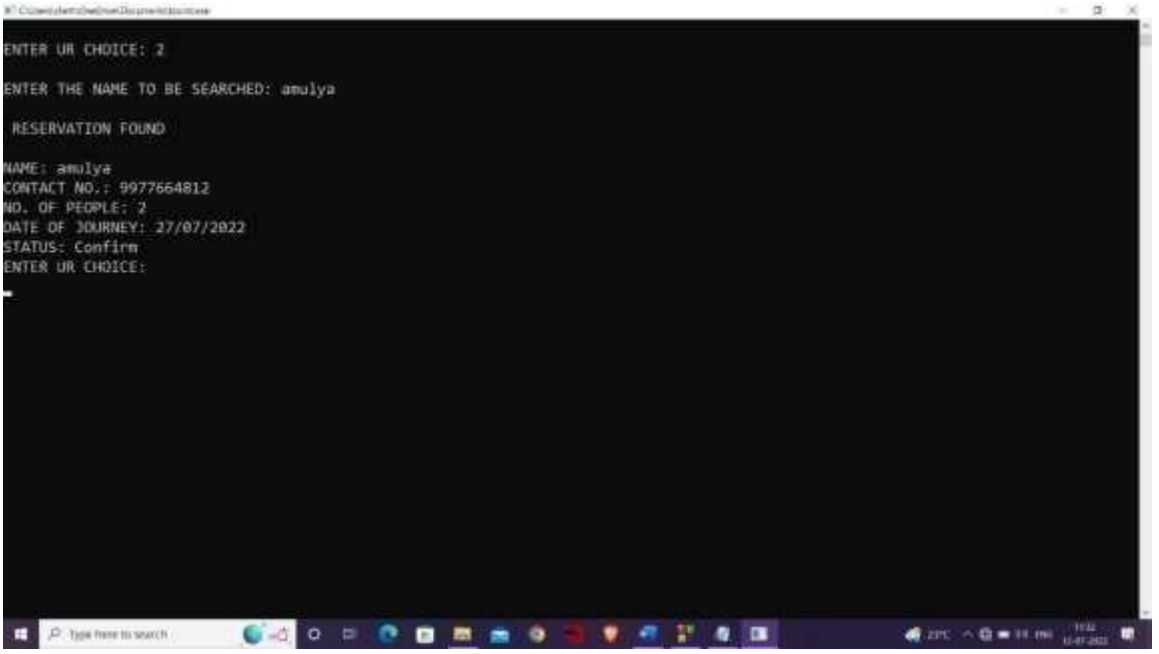

Figure 7.1.8 Display Reservation

The reservation is searched using the name of the user. The reservation is displayed, if it is found.

7.1.9 Delete Reservation

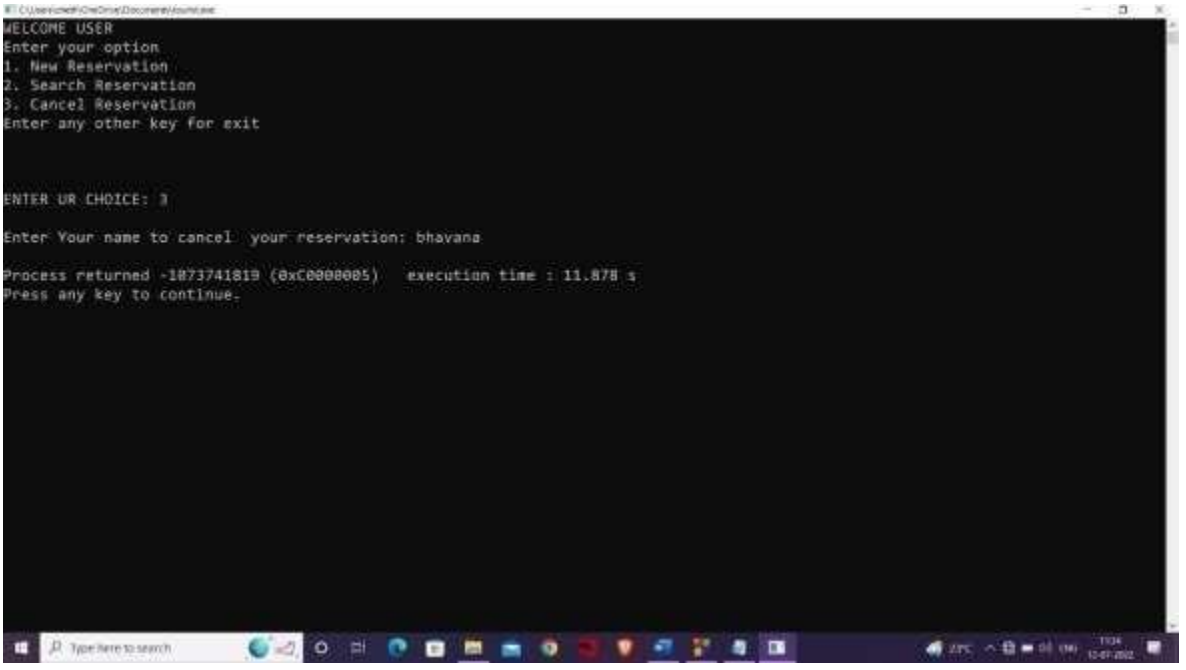

Figure 7.1.9 Delete Reservation

For cancellation of user's reservation , choice 3 is entered and once the name of the reserved person is chosen, that particular reservation gets cancelled.

CONCLUSION

.

Tourism Reservation System project has been successfully developed using Dev C++ under Windows platform. This Project is minimizing the task of reservation by advance booking the tours and saying some details about tours to save data. In this the tours are planned as a safe and secure. This project is done as Efficient as possible

Advantages:

- Fast retrieval of information.
- Easy access.
- The user can keep track of the reservation details easily. All these goals are achieved and now here we are with a Tourism Reservation Application.

Future Enhancement:

- Adding Feedback and suggestion option.
- It can be implemented through web pages. New effectives modules can be added time to time. The main goals of this mini project are:
- To learn accessing data from file system and displaying, fetching, retrieving, inserting, deleting to it using different techniques available.
- To write Dataflow Diagram and Flow-Chart for the same file system.

REFERENCES

[1] File Structures-An Object Oriented Approach with C++ by Micheal J. Folk, Bill Zoellick, GregRiccardi.

[2] The c++ programming language by Bjarne Stroustrup

[3] C++ online learning https://www.youtube.com/watch?v=GQp1zzTwrIg

[4] https://online.visualparadigm.com/app/diagrams/#diagram:proj=0&type=DataFlowDiagram&width=11& height=8.5&unit=inch Data Flow Diagram

[5] https://stackoverflow.com/

[6] https://cplusplus.com/info/

[7] http://www.codeblocks.org/

[8] https://www.tutorialspoint.com/cprogramming/index.htm

[9] https://www.udemy.com/