

## Chapter 1

# INTRODUCTION

### 1.1 Overview

The purpose of the project entitled as “TRAVEL AGENCY MANAGEMENT SYSTEM” is to develop software using C++, a menu driven program which allows us to add, update, delete and search records of a user who has booked a package in our organization. The program travel management system stores the user’s username, password, place, and price, number of tickets and date of departure. Initially, it had no data. Thus, user’s have to add user if he is not an existing user.

### 1.2 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 also support 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.

## Chapter 2

# SYSTEM REQUIREMENT

### 2.1 Software Requirements:

- Operating system - Windows 7, Windows 10 or above
- Language - C++
- Text Editor – Notepad

### 2.2 Hardware Requirements:

- Hardware - Pentium
- RAM - 1GB
- Hard Disk - 2GB+
- Keyboard - standard Windows Keyboard

### 2.3 Software Description

Code::Blocks is a free, open-source cross-platform IDE that supports the incorporation of the multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using Widgets as the GUI toolkit. Using a plug-in architecture, its capabilities and features are defined by the provided plug-ins. currently; Code::Blocks is oriented towards C, C++, and Fortran. It has a custom build system and optional Make support.

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications including desktop applications.

## Chapter 3

# PROJECT DESIGN

### 3.1 Design

Once the requirements document for the software to be developed is available, the software design phase begins. While the requirement specification activity deals entirely with the problem domain, design is the first phase of transforming the problem into a solution. In the design phase, the customer and business requirements and technical considerations all come together to formulate a product or a system.

The design process comprises a set of principles, concepts and practices, which allow a software engineer to model the system or product that is to be built. This model, known as design model, is assessed for quality and reviewed before a code is generated and tests are conducted. The design model provides details about software data structures, architecture, interfaces and components which are required to implement the system. This chapter discusses the design elements that are required to develop a software design model. It also discusses the design patterns and various software design notations used to represent a software design.

#### 3.1.1 Design Model

To develop a complete specification of design (design model), four design models are needed. These models are listed below.

- **Data design:** This specifies the data structures for implementing the software by converting data objects and their relationships identified during the analysis phase. Various studies suggest that design engineering should begin with data design, since this design lays the foundation for all other design models.
- **Architectural design:** This specifies the relationship between the structural elements of the software, design patterns, architectural styles, and the factors affecting the ways in which architecture can be implemented.
- **Component-level design:** This provides the detailed description of how structural elements of software will actually be implemented.
- **Interface design:** This depicts how the software communicates with the system that interoperates with it and with the end-users.

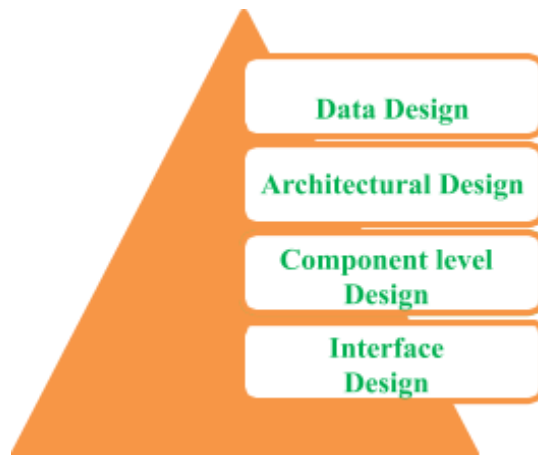


Fig1. Design models and its elements

### 3.1.2 Block Diagram

A block diagram is a diagram of a system in which the principal parts of a system are represented by blocks connected by lines that show the relationships of the block. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.

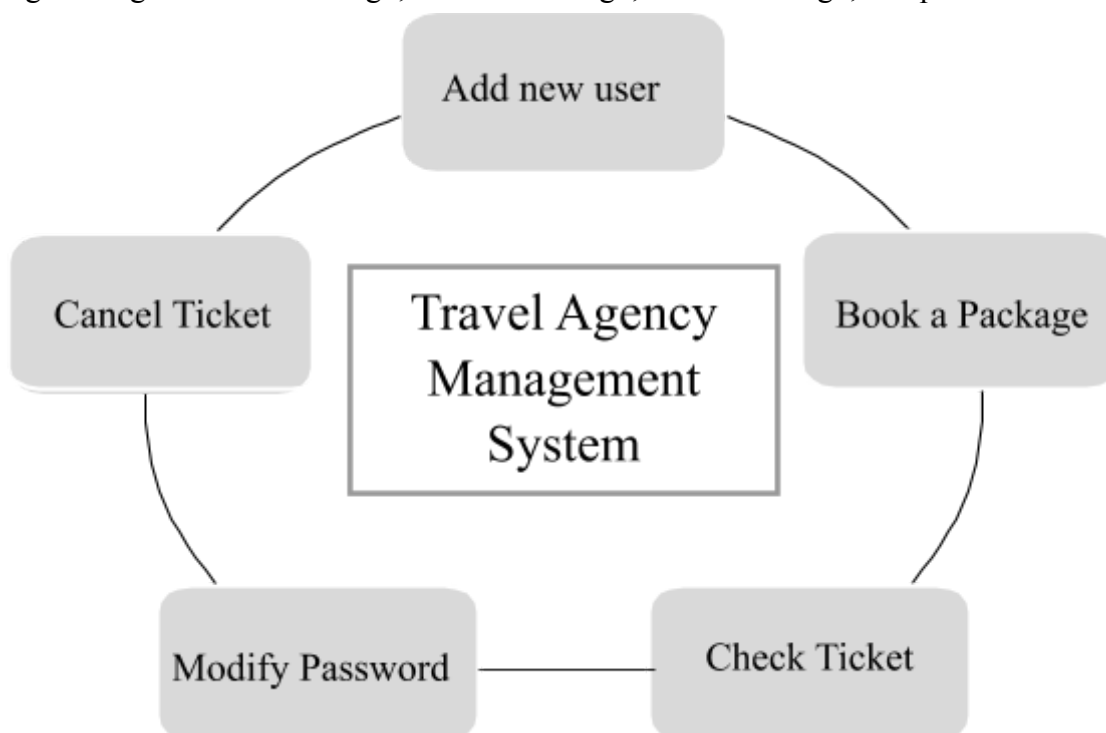


Fig 2. Block diagram of travel agency management system

## Chapter 4

# IMPLEMENTATION

### 4.1 Introduction

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

### 4.2 Algorithms

In mathematics and computer science, an algorithm is an unambiguous specification of how to solve a class of problems. Algorithms can perform data processing, calculation and automated reasoning tasks.

It is simply a series of instructions that are followed step by step to solve some problem or do something useful. Algorithms are a formal way of describing very precisely how to carry out a certain task.

### 4.3 SOURCE CODE

```
#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fstream>

#include <limits>

using namespace std;

enum state {menu, loggedin};

enum state currentstate=menu;

typedef struct user

{

    char username[100];

    char password[100];

    char place[100];

    float price;

    int numtick;

    int day;

    int month;

    int year;

    int valid = 1;
```

```
    struct user *next;

}user;

void ShowBrochure();

user* Initializelist(user*);

user* AddUser(user*);

void LoginUser(user*);

void BookTicket(user*);

void PrintTicket(user*);

void CancelTicket(user*);

void ChangePassword(user*);

void LogoutUser();

void CheckTicket(user*);

void DisplayAll(user*);

void WriteToFile(user*);

void Help();

int datevalidate(int dd, int mm, int yy);

void ExitProgram();

char currentuser[100];

int main()
```

Page 8



```
int ch1,ch2;

h=Initializelist(h);

while(1)

{

    if(currentstate==menu)

    {

        printf("\n\t\t\t\tAdd User - 1 \n\t\t\t\tLogin User - 2\n\t\t\t\tBrochure - 3\n\t\t\t\tExit - 4\n\n\t\t\t\tPlease
Enter your choice:");

        scanf("%d",&ch1);

        switch(ch1)

        {

            case 1:

                h=AddUser(h);

                break;

            case 2:

                LoginUser(h);

                break;

            case 3:

                ShowBrochure();

                break;

            case 4:

                ExitProgram();
```

```
        exit(0);

        break;

default:

    printf("Not a valid input at this stage\n");

}

}

else if(currentstate==loggedin)

{

    system("CLS");

    printf("\n+++++\n");

    printf("\n+ TICKET BOOKING SYSTEM +");

    printf("\n+++++\n");

    printf("\n\t\t\t\tBook Package - 1\n\t\t\t\tCheck Ticket - 2\n\t\t\t\tCancel Ticket - 3\n\t\t\t\tChange
Password - 4\n\t\t\t\tLogout user - 5\n\t\t\t\tBrochure - 6\n\t\t\t\tExit - 7\n\t\t\t\tenter the choice:");

    scanf("%d",&ch2);

    switch(ch2)

    {

    case 1:

        BookTicket(h);

        system("PAUSE");

        system("CLS");

        break;
```

case 2:

CheckTicket(h);

system("PAUSE");

system("CLS");

break;

case 3:

CancelTicket(h);

system("PAUSE");

system("CLS");

break;

case 4:

ChangePassword(h);

system("PAUSE");

system("CLS");

break;

case 5:

LogoutUser();

system("PAUSE");

system("CLS");

break;

case 6:

```
        ShowBrochure();

        system("PAUSE");

        system("CLS");

        break;

    case 7:

        ExitProgram();

        exit(0);

        break;

    default:

        printf("not a valid input\n");

    }

}

else

    break;

}

return 0;

}

user* Initializelist(user *h)

{

    user* t,*ptr,temp;

    FILE *fp;
```

```
int cc=0,x;

float ff;

fp=fopen("useRM.txt","r");

if(fp==NULL)

    return NULL;

if(fgetc(fp)==EOF)

    return NULL;

rewind(fp);

while(fscanf(fp,"%s          %s          %s          %f          %d          %d          %d
%d",temp.username,temp.password,temp.place,&temp.price,&temp.numtick,&temp.day,&temp.month,&te
mp.year)!=EOF)

{

    ptr=(user*)malloc(sizeof(user));

    strcpy(ptr->username,temp.username);

    strcpy(ptr->password,temp.password);

    strcpy(ptr->place,temp.place);

    ptr->price=temp.price;

    ptr->numtick=temp.numtick;

    ptr->day=temp.day;

    ptr->month=temp.month;

    ptr->year=temp.year;

    ptr->next=NULL;
```

```
    if(h==NULL)

        h=t=ptr;

    else

    {

        h->next=ptr;

        h=ptr;

    }

}

fclose(fp);

return t;

}

void WriteToFile(user *h)

{

    FILE *fp;

    fp=fopen("useRM.txt","w");

    while(h!=NULL)

    {

        fprintf(fp,"%s    %s    %s    %f    %d    %d    %d    %d\n",h->username,h->password,h->place,h->price,h->numtick,h->day,h->month,h->year);

        h=h->next;

    }

    fclose(fp);
```

```
}

void ShowBrochure()

{

    system("CLS");

    printf("\nPRICE LIST\n=====\n1.KER : Kerala Tour Package (6D/5N) Rs - 17999\n
2.AND : Andaman Tour Package (6D/5N) Rs - 17999\n 3.KAS : Kashmir Tour Package (7D/6N) Rs -
31999 \n 4.HIM : Himachal Tour Package (7D/6N) Rs - 25999\n 5.COR : Explore Coorg Packages
(3D/2N) Rs - 16200\n 6.MAN : Manali Trip Package (6D/5N) Rs - 25000\n 7.SIK : Visit Gangtok and
Darjeeling Package (5D/4N) Rs - 19233\n 8.SHIL : Excursion to Shillong (4D/3N) Rs - 20500\n 9.GOA :
Weekend in Goa (3D/3N) Rs - 17999\n 10.LAD : Ladakh Package (5D/4N) Rs - 20400\n 11.RAJ :
Colourful Rajasthan Package (9D/8N) Rs - 68000\n ");

}

void CheckTicket(user *h)

{

    while(h!=NULL)

    {

        if(!strcmp(h->username,currentuser))

            break;

        h=h->next;

    }

    if(!strcmp(h->place,"\0") || h->price==0.0 || h->numtick==0 || h->day==0 || h->month==0 || h->year==0)

    {

        printf("You do not have a ticket booked yet\n");

        return;

    }

}
```

```
}

float total=0.0;

total=(h->price)*(h->numtick);

printf("You have booked %d tickets for a sum total of Rs %f for tour code %s on following date %d -
%d - %d \n",h->numtick,total,h->place,h->day,h->month,h->year);

}

user* AddUser(user* h)

{

    user *t;

    t=h;

    user *nw;

    nw=(user*)malloc(sizeof(user));

    fflush(stdin);

    printf("Enter username or email\n");

    scanf("%s",nw->username);

    while(h!=NULL)

    {

        if(!strcmp(h->username,nw->username))

        {

            printf("The email already exists\n");

            return t;

        }

    }

}
```



```
    h=h->next;

}

h=t;

fflush(stdin);

printf("Enter password\n");

scanf("%s",&nw->password);

nw->next=NULL;

strcpy(nw->place,"N/A");

nw->price=0.0;

nw->numtick=0;

nw->day=0;

nw->month=0;

nw->year=0;

if(h==NULL)

{

    h=t=nw;

}

else

{

    while(h->next!=NULL)

    {
```

```
        h=h->next;

    }

    h->next=nw;

}

WriteToFile(t);

return t;

}

void LoginUser(user* h)

{

    char username[100];

    char password[100];

    fflush(stdin);

    printf("\n\n");

    printf("\t\tEnter email/username\n\t\t");

    scanf("%s",username);

    fflush(stdin);

    printf("\n\t\tEnter Password:\n\t\t");

    scanf("%s",password);

    while(h!=NULL)

    {

        if((!strcmp(h->username,username)) && (!strcmp(h->password,password)))
```

```
{

    currentstate=loggedin;

    strcpy(currentuser,username);

    printf("\n\t\tLogin Successful!!\n");

    system("PAUSE");

    return;

}

else if((!strcmp(h->username,username)) && (strcmp(h->password,password)))

{

    printf("Password Mismatch\n");

    return;

}

h=h->next;

}

printf("Sorry, no such user record was found\n");

}

void BookTicket(user *h)

{

    user *t=h;

    char place[100];

    int day;
```

```
int month;

int year;

while(h!=NULL)

{

    if(!strcmp(h->username,currentuser))

        break;

    h=h->next;

}

if(h==NULL)

    return;

if(h->price!=0.0)

{

    printf("You must cancel your previous ticket before buying a new one\n");

    return;

}

ShowBrochure();

float
pricelist[]={17999.0,27999.0,31999.0,25999.0,16200.0,25000.0,19233.0,20500.0,17999.0,20400.0,68000.0};

fflush(stdin);

printf("\nEnter place code(eg: INDO,MAS)\n");

scanf("%s",place);
```

```
char choice;

fflush(stdin);

printf("\n Wold you like to confirm Booking?\n[1] -Yes\n[2] -No\n");

scanf("%c",&choice);

float price;

if(choice!='1')

    return;

if(strcmp(place,"KER")==0)

    price=pricelist[0];

else if(strcmp(place,"AND")==0)

    price=pricelist[1];

else if(strcmp(place,"KAS")==0)

    price=pricelist[2];

else if(strcmp(place,"HIM")==0)

    price=pricelist[3];

else if(strcmp(place,"COR")==0)

    price=pricelist[4];

else if(strcmp(place,"MAN")==0)

    price=pricelist[5];

else if(strcmp(place,"SIK")==0)

    price=pricelist[6];
```

```
else if(strcmp(place,"SHIL")==0)

    price=pricelist[7];

else if(strcmp(place,"GOA")==0)

    price=pricelist[8];

else if(strcmp(place,"LAD")==0)

    price=pricelist[9];

else if(strcmp(place,"RAJ")==0)

    price=pricelist[10];

else

{

    printf("That tour code doesn't exist\n");

    return;

}

printf("Enter the number of tickets you want to book?\n");

scanf("%d",&h->numtick);

if(h->numtick==0)

    return;

strcpy(h->place,place);

h->price=price;

printf("\n\n\t*****Date of Departure*****");

printf("\n\n *NOTE:Format for entering:DAY(press enter) MONTH(press enter) YEAR *");
```

```
printf("\n\n\t Enter your preferred date of departure:");

cout<<"Enter Date "<<" : ";

scanf("%d",&h->day);

scanf("%d",&h->month);

scanf("%d",&h->year);

h->day,day;

h->month,month;

h->year,year;

WriteToFile(t);

printf("Booking Done!!\n");

system("PAUSE");

}

void CancelTicket(user *h)

{

    user *t=h;

    while(h!=NULL)

    {

        if(!strcmp(h->username,currentuser))

            break;

        h=h->next;

    }
```

```
int flag=-1;

if(h==NULL)

    printf("No such user\n");

if(strcmp(h->place,"KER")==0)

    flag++;

else if(strcmp(h->place,"AND")==0)

    flag++;

else if(strcmp(h->place,"KAS")==0)

    flag++;

else if(strcmp(h->place,"HIM")==0)

    flag++;

else if(strcmp(h->place,"COR")==0)

    flag++;

else if(strcmp(h->place,"MAN")==0)

    flag++;

else if(strcmp(h->place,"SIK")==0)

    flag++;

else if(strcmp(h->place,"SHIL")==0)

    flag++;

else if(strcmp(h->place,"GOA")==0)

    flag++;
```



```
else if(strcmp(h->place,"LAD")==0)

    flag++;

else if(strcmp(h->place,"RAJ")==0)

    flag++;

else

{

    printf("You haven't booked a package yet\n");

    return;

}

if(flag==0)

{

    printf("Your ticket has successfully cancelled A refund of Rs %f for Tour Code %s for %d ticket on
%d-%d-%d will be made to your original
source",h->price,h->place,h->numtick,h->day,h->month,h->year);

    strcpy(h->place,"N/A");

    h->price=0.0;

    h->numtick=0;

    h->day=0;

    h->month=0;

    h->year=0;

    WriteToFile(t);

}
```

```
}

void ChangePassword(user *h)

{
    user *t=h;

    char passcurr[100];

    fflush(stdin);

    printf("Enter your current password to continue:\n");

    scanf("%s",passcurr);

    while(h!=NULL)

    {

        if(!strcmp(h->username,currentuser))

            break;

        h=h->next;

    }

    if(h==NULL)

        return;

    if(!strcmp(passcurr,h->password))

    {

        printf("Enter new password:\n");

        scanf("%s",h->password);

    }

}
```

```
    WriteToFile(t);  
  
}  
  
void LogoutUser()  
  
{  
  
    if((currentstate==menu) || (strcmp(currentuser,"\0")==0))  
  
    {  
  
        printf("you must be logged in to logout\n");  
  
        return;  
  
    }  
  
    strcpy(currentuser,"\0");  
  
    currentstate = menu;  
  
    printf("You have successfully logged out\n");  
  
}  
  
void ExitProgram()  
  
{  
  
    printf("Exiting...\n\n");  
  
    char exitprog;  
  
    fflush(stdin);  
  
    scanf("%c",&exitprog); }
```

## Chapter 5

# TESTING

### 5.1 Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or applications with the intent of finding software bugs(errors or other defects).

It involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- Meets the requirements that guided its design and development.
- Responds correctly to all kinds of inputs.
- Performs its functions within an acceptable time.
- Is sufficiently usable.
- Can be installed and run in its intended environments.
- Achieves the general results its stakeholders desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically(but not exclusively)attempts to execute a program or application with the intent of finding software bugs(errors or other defects).

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

## **5.2 TESTING PROCESS**

### **5.2.1 LEVELS OF TESTING**

1. **UNIT TESTING:** Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level.
2. **INTEGRATION TESTING:** Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.
3. **COMPONENT INTERFACE TESTING:** The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full testing between those units.
4. **TESTING SYSTEM:** System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements.

### **5.2.2 TESTING METHODS**

#### **5.2.2.1 UNITS TESTING**

Units testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific functions are working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs.

#### **5.2.2.2 INTEGRATION TESTING**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together (“big bang”). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

## **5.3 TEST ENVIRONMENT**

### BLACK BOX TESTING

Black box testing treats the software as a “black box”, examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

### WHITE BOX TESTING

White-box testing tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Through this method of test design can uncover many errors or problems; it might not detect unimplemented parts of the specification or missing requirements.

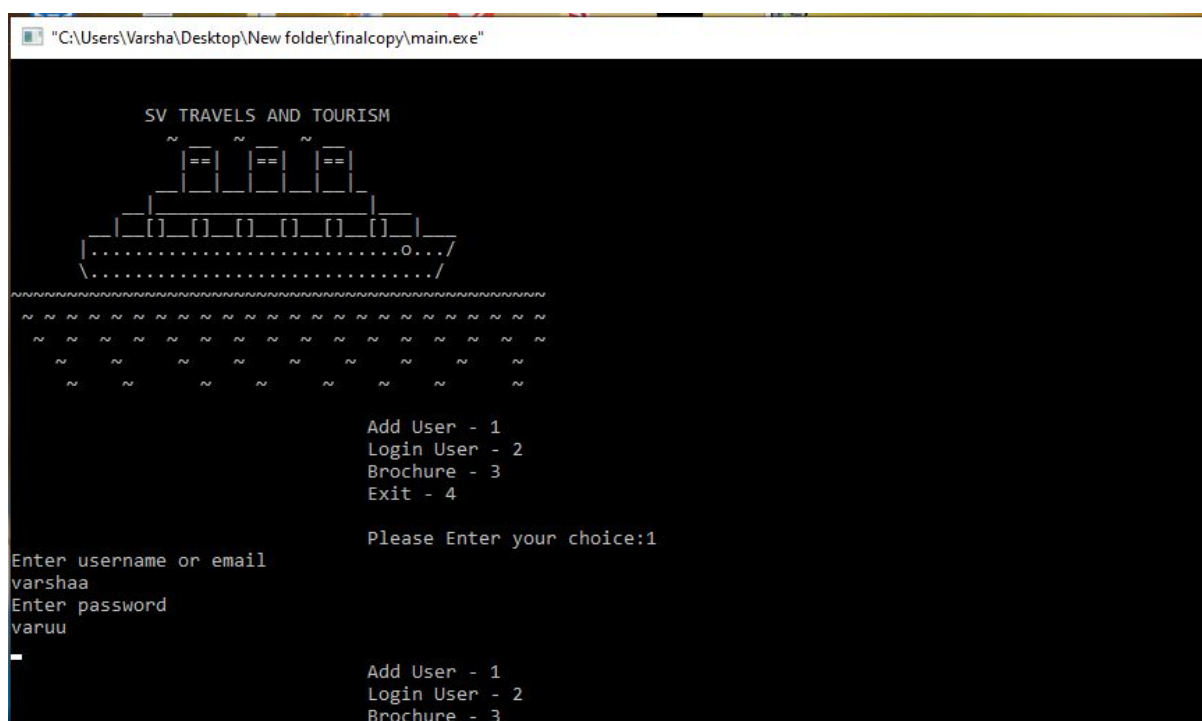
## Chapter 6

### USER INTERFACE

Fig 6.1 Front page



Fig 6.2 Add user



[illegible]

```
"C:\Users\Varsha\Desktop\New folder\finalcopy\main.exe"

+++++
+ TICKET BOOKING SYSTEM +
+++++

Book Package - 1
Check Ticket - 2
Cancel Ticket - 3
Change Password - 4
Logout user - 5
Brochure - 6
Exit - 7
enter the choice:
```



Fig 6.5 Ticket Booking page

```

C:\Users\Varsha\Desktop\New folder\finalcopy\main.exe

PRICE LIST
=====
1.KER : Kerala Tour Package (6D/5N) Rs - 17999
2.AND : Andaman Tour Package (6D/5N) Rs - 17999
3.KAS : Kashmir Tour Package (7D/6N) Rs - 31999
4.HIM : Himachal Tour Package (7D/6N) Rs - 25999
5.COR : Explore Coorg Packages (3D/2N) Rs - 16200
6.MAN : Manali Trip Package (6D/5N) Rs - 25000
7.SIK : Visit Gangtok and Darjeeling Package (5D/4N) Rs - 19233
8.SHIL : Excursion to Shillong (4D/3N) Rs - 20500
9.GOA : Weekend in Goa (3D/3N) Rs - 17999
10.LAD : Ladakh Package (5D/4N) Rs - 20400
11.RAJ : Colourful Rajasthan Package (9D/8N) Rs - 68000

Enter place code(eg: INDO,MAS)
GOA

Wold you like to confirm Booking?
[1] -Yes
[2] -No
1
Enter the number of tickets you want to book?
2

*****Date of Departure*****

*NOTE:Format for entering:DAY(press enter) MONTH(press enter) YEAR *
```

Fig 6.6 Check Ticket

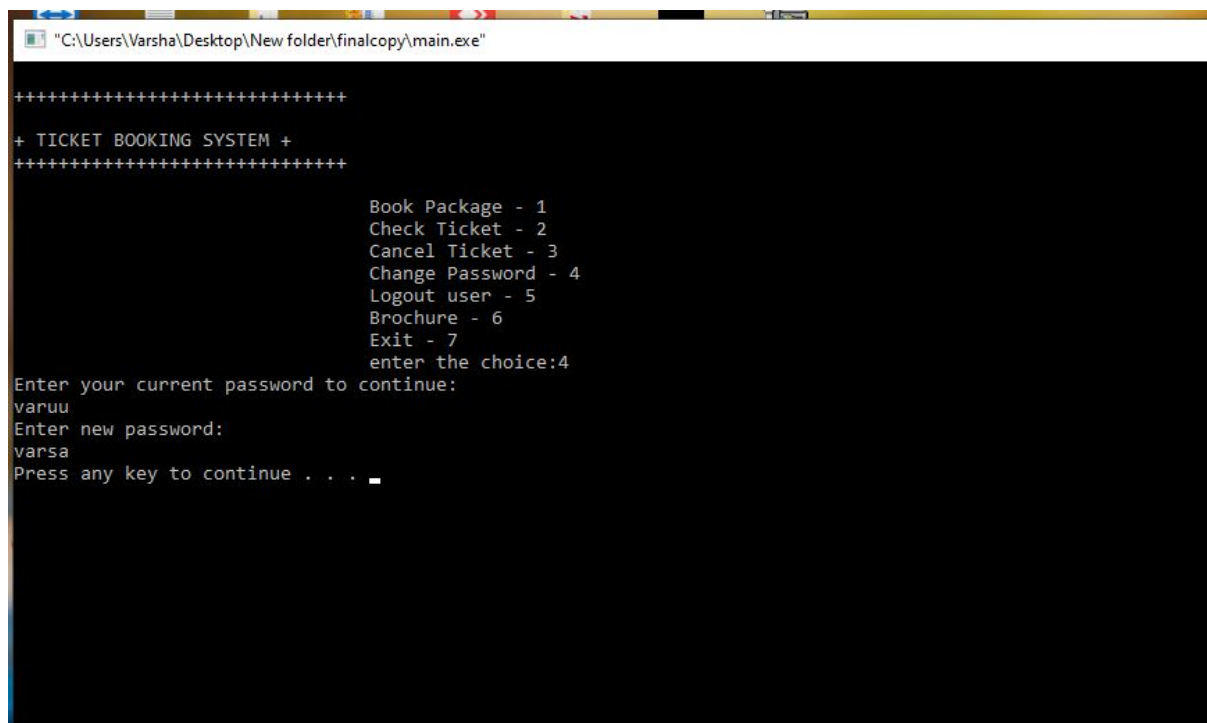
```

C:\Users\Varsha\Desktop\New folder\finalcopy\main.exe

+++++
+ TICKET BOOKING SYSTEM +
+++++

Book Package - 1
Check Ticket - 2
Cancel Ticket - 3
Change Password - 4
Logout user - 5
Brochure - 6
Exit - 7
enter the choice:2
You have booked 2 tickets for a sum total of Rs 35998.000000 for tour code GOA on following date 2 - -10 - -
Press any key to continue . . .
```

Fig 6.7 Change Password



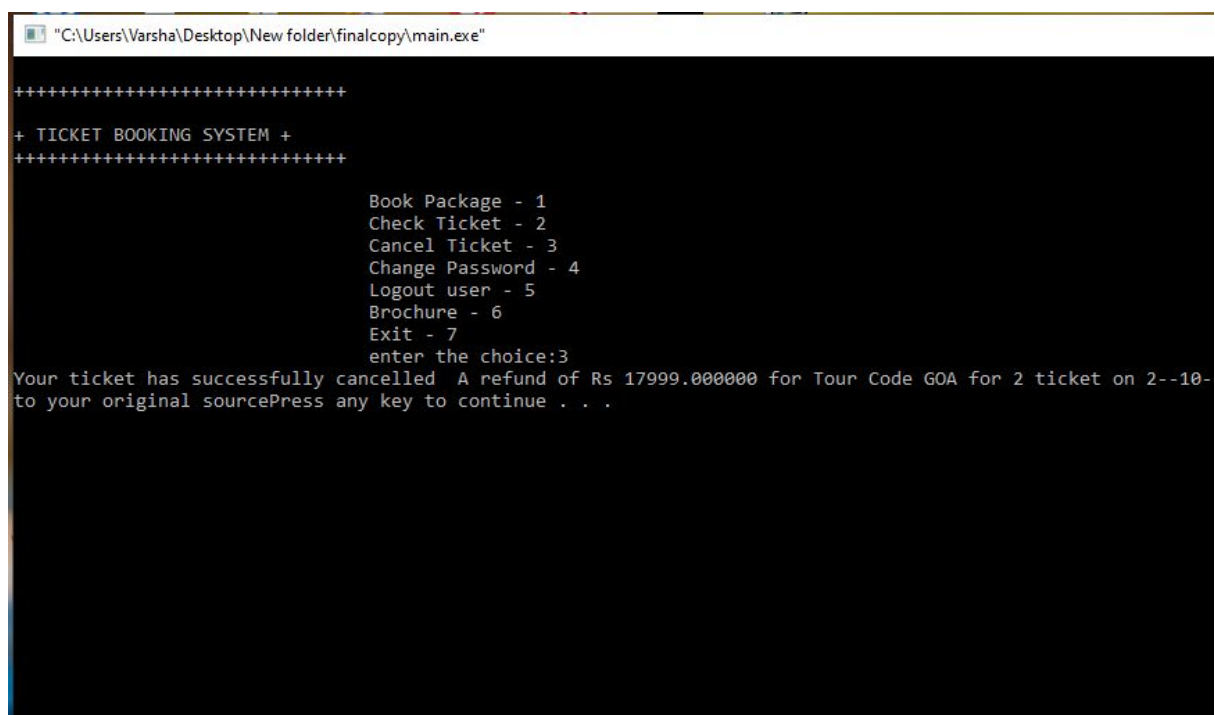
```
"C:\Users\Varsha\Desktop\New folder\finalcopy\main.exe"

+++++
+ TICKET BOOKING SYSTEM +
+++++

Book Package - 1
Check Ticket - 2
Cancel Ticket - 3
Change Password - 4
Logout user - 5
Brochure - 6
Exit - 7
enter the choice:4

Enter your current password to continue:
varuu
Enter new password:
varsa
Press any key to continue . . .
```

Fig 6.8 Cancel Ticket



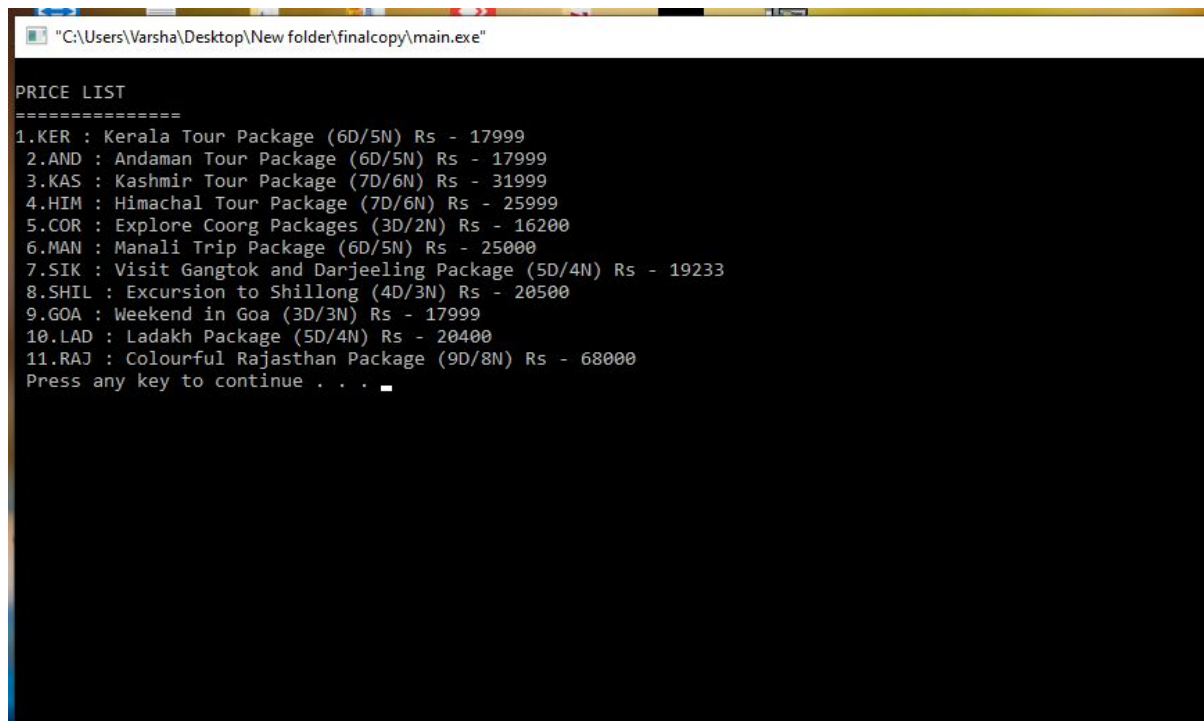
```
"C:\Users\Varsha\Desktop\New folder\finalcopy\main.exe"

+++++
+ TICKET BOOKING SYSTEM +
+++++

Book Package - 1
Check Ticket - 2
Cancel Ticket - 3
Change Password - 4
Logout user - 5
Brochure - 6
Exit - 7
enter the choice:3

Your ticket has successfully cancelled A refund of Rs 17999.000000 for Tour Code GOA for 2 ticket on 2--10-
to your original sourcePress any key to continue . . .
```

Fig 6.9 Brochure



The screenshot shows a Windows command prompt window with the title bar "C:\Users\Varsha\Desktop\New folder\finalcopy\main.exe". The window displays a "PRICE LIST" with 11 tour packages and their prices. The packages are listed as follows:

Package Code	Package Name	Duration	Price (Rs)
1.KER	Kerala Tour Package	6D/5N	17999
2.AND	Andaman Tour Package	6D/5N	17999
3.KAS	Kashmir Tour Package	7D/6N	31999
4.HIM	Himachal Tour Package	7D/6N	25999
5.COR	Explore Coorg Packages	3D/2N	16200
6.MAN	Manali Trip Package	6D/5N	25000
7.SIK	Visit Gangtok and Darjeeling Package	5D/4N	19233
8.SHIL	Excursion to Shillong	4D/3N	20500
9.GOA	Weekend in Goa	3D/3N	17999
10.LAD	Ladakh Package	5D/4N	20400
11.RAJ	Colourful Rajasthan Package	9D/8N	68000

Press any key to continue . . .

Fig 6.10 Data Stored in Files Before Cancellation Tickets



The screenshot shows a Windows file explorer window with the title bar "finalcopy.layout" and the date "03-06-2020 13:47". The file is a "LAYOUT File" and is 1 KB in size. Below the file explorer, a Notepad window is open with the title "useRM - Notepad". The Notepad window displays the following text:

```
File Edit Format View Help
varshaa varsa SIK 19233.000000 2 2 -10 -2020
```

Fig 6.11 Data Stored in Files After Cancellation of Tickets

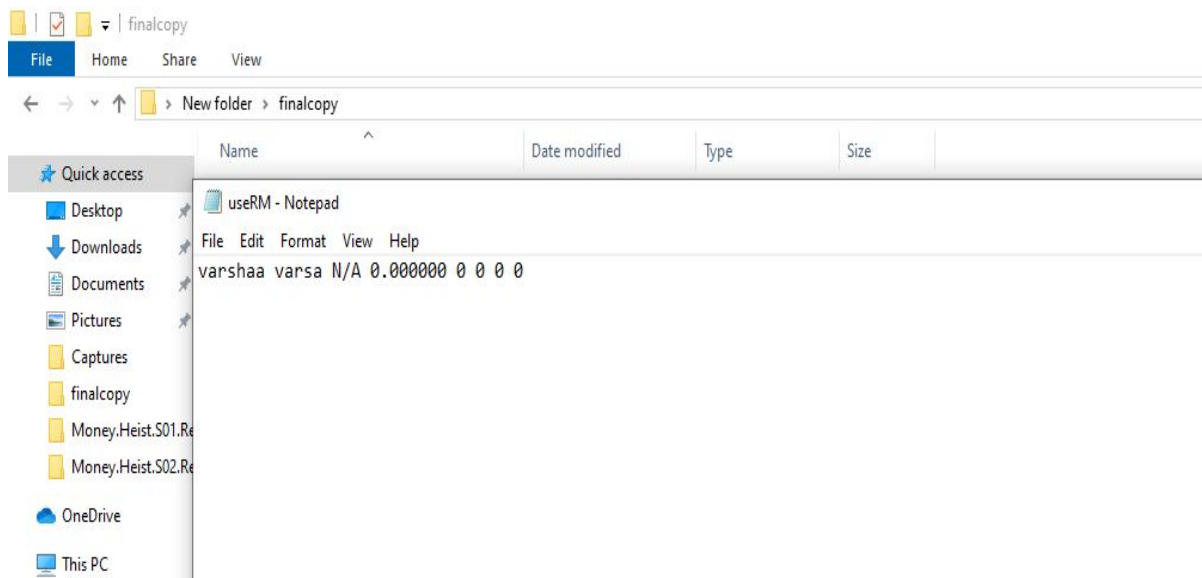
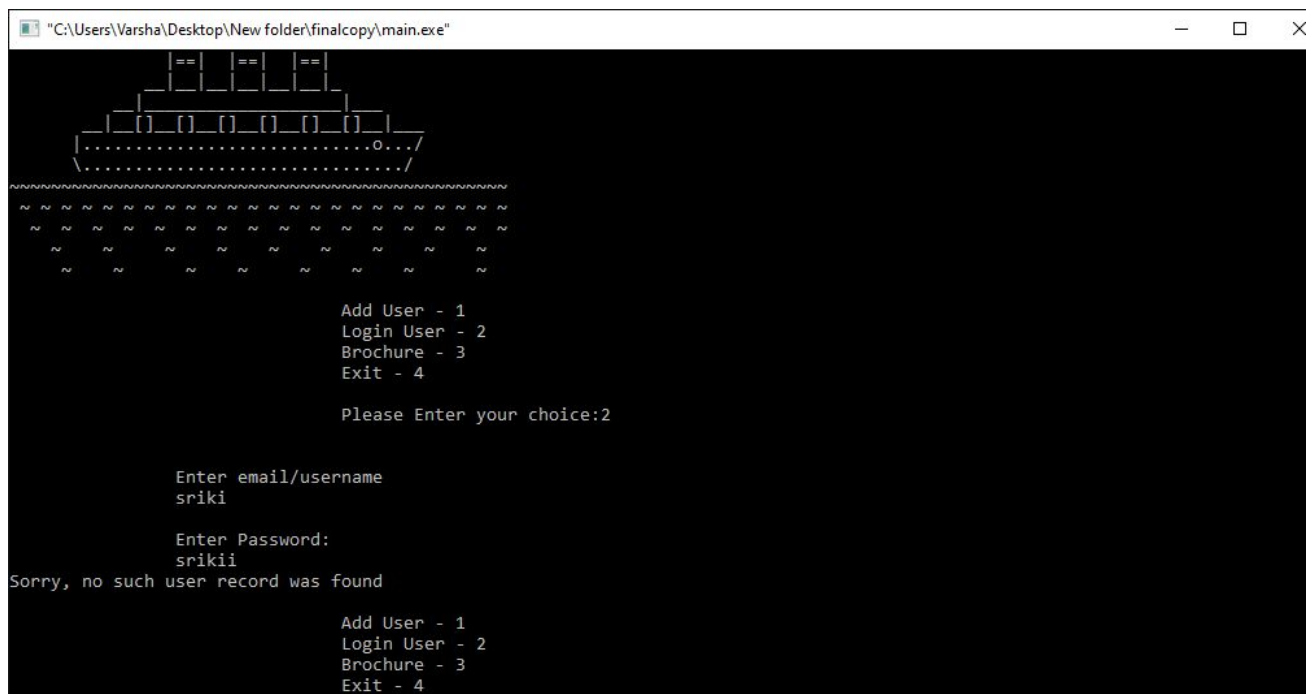


Fig 6.12 When Data is not Registered



## Chapter 7

### CONCLUSION

This was an effort to develop a travel agency management system which is helpful in the market sector.

Since this project has been designed exclusively as a project, certain complexities that are faced by any real life manual problem. But enhancement to the project can easily be made without changing the current design and programming structure.

This software is efficient in maintaining customer's or user's details and can easily perform operations on customer's or user's records. This software also reduces the workload of the travel agency manager.

In future, this system can launch web sites for easy online registration. In this system there are limitations for places and vehicles. In future, it can be extended to add more places and vehicles.

## Chapter 8

### FUTURE ENHANCEMENT

- Modify this system to perform additional operations such as modification of the Registered date of travels or Departure date of travels etc.
- This system will be extended in future to handle a number of places and also provide facilities for more vehicles.
- In future the system can be done online.

## REFERENCES

1. Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3rd Edition, Pearson Education, 1998.
2. K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj: File Structures Using C++, Tata McGraw-Hill, 2008.