

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELGAVI-590018



FILE STRUCTURE LABORATORY WITH
MINI PROJECT (18CSL67)

A File Structure Mini-Project Report On
Buffer Management System
“Bus Reservation System”

Submitted in partial fulfilment of the requirements for the 6th semester

Submitted by

RITVIZ RANJAN

1BI18IS067

Under the guidance of

Mrs Anupama KC

Assistant Professor

Dept. of ISE, BIT, Bangalore-04



2021

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
BANGALORE INSTITUTE OF TECHNOLOGY
KR Road, VV Puram, Bengaluru-560004



Bangalore Institute of Technology
KR Road, VV Puram, Bengaluru-560004
Department of Information Science and Engineering

CERTIFICATE

Certified that the Project Work (18ISL67) work entitled “**BUS RESERVATION SYSTEM**” carried out by **RITVIZ RANJAN (1BI18IS067)** in partial fulfilment of the requirements for the **VI Semester degree of Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belgavi during academic year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of Project work for the said degree.

Mrs. Anupama KC
Assistant Professor
Department of ISE

Dr J Prakash
Professor and HOD
Department of ISE

Examiners: 1)
2)

ABSTRACT

Bus Reservation System is basically used to automate the existing manual system by the help of a computerised equipment and fully fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Bus Reservation System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organisation can maintain computerized records without redundant entries, which means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerised equipment providing services such as login and account creation facilities for the reservation of tickets. User can also modify and cancel the ticket with the help of this proposed system.

ACKNOWLEDGEMENT

This knowledge and satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned any effort with success. I would like to thank and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my thanks to the Head of Department **Dr J Prakash** for being kind enough to provide the necessary support to carry out the Mini Project.

I am most humbled to mention the enthusiastic influence provided by the lab-in-charge **Professor Anupama KC** on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I extend my thanks to our Principal **Dr MU Aswath** for encouraging us in all aspects to complete the project.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleased to express my sincere gratitude to the friendly co-operation showed by all the staff members of Information Science and Engineering Department, BIT.

RITVIZ RANJAN (1BI18IS067)

LIST OF FIGURES

Fig no.	Title	Page no.
3.1	Context Flow Diagram	3
3.2	System Design for Bus Reservation System	4
3.3	Data Flow diagram for new account registration	5
3.4	Data Flow diagram for login	6
3.5	Data Flow Diagram for modify account details	7
3.6	Data flow diagram for deleting account details	8
3.7	Data Flow Diagram for new reservation	9
3.8	Data Flow Diagram for modify reservation	10
3.9	Data Flow Diagram for cancel reservation	11
3.10	Data Flow Diagram for displaying reserved tickets	12
4.1	Double I/O Buffer	18
4.2	System I/O Buffer	14
6.1	Bus Reservation System Home Page	18
6.2	Customer Menu page	18
6.3	Create new account page	19
6.4	Account deletion page	20
6.5	Reservation Menu Page	20
6.6	Login Page	21
6.7	New Reservation Page	21
6.8	Displaying tickets	22
6.9	Cancellation of tickets	22
6.10	Passenger.txt file	23
6.11	Payments.txt	23
6.12	Reservations.txt file	24

LIST OF TABLES

Table no.	Title	Page no.
5.1	Test Cases	17

TABLE OF CONTENTS

CHAPTER	CONTENTS	PAGE NO
	CERTIFICATE	
	ABSTRACT	
	ACKNOWLEDGEMENT	
	LIST OF FIGURES	
	LIST OF TABLES	
1	INTRODUCTION	
	1.1 General Overview	01
	1.2 Problem Statement	01
	1.3 Objectives	01
2	REQUIREMENT SPECIFICATION	
	2.1 Software Requirements	02
	2.2 Hardware Requirements	02
3	SYSTEM DESIGN	
	3.1 Context Flow Diagram	03
	3.2 System Design	03
	3.3 Data Flow Diagram	05
4	IMPLEMENTATION	
	4.1 Technique Used–Buffering Management	13
	4.2 Buffering Strategies	13
	4.3 How does it affect programming?	15
5	TESTING	
	5.1 Unit Testing	16
6	RESULTS	18
7	CONCLUSION	25
	REFERENCES	26
	APPENDIX	27

CHAPTER 1

INTRODUCTION

1.1 GENERAL OVERVIEW

Travelling is a large growing business across all countries. The bus reservation system that is used at the counter of bus stations currently is an internal system and is just used to sell the bus tickets at the counter only. The customer has to go to the counter to buy bus tickets or ask for a bus schedule. The existing system is totally on a book hence a significant amount of manual work increases exponentially with a rise in bus service.

Offline ticket booking reduced the scope of customers to choose different options based on their travel criteria. It also increased the franchising cost for the bus operators. At the same time, the bus operators were also finding it difficult to monitor their bus seat filling information. Many small and medium bus service organizations do not have their own online bus ticket booking system.

1.2 PROBLEM STATEMENT

Bus Reservation System is a C++ language-based reservation system.

It manages the reservation of seats in various buses having different routes, the login and account creation facility being provided for the reservation of seats. The customer can also modify their account details after finishing the process of registration and if they want to delete their account, it can also be over with to provide data privacy.

After booking tickets, customers are provided with the facility of modifying, cancelling tickets and they can also check the reservation made by them.

1.3 OBJECTIVES

The main objective of the project on Bus Reservation System is to manage the details of customer account such as passenger name, passenger city, contact number, username and password. It manages all the information about Bus Schedule, Bus Route, Departure and Arrival station. The purpose of the project is to build an application program to reduce the manual work for managing all the details.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

A major element in building a system is the section of compatible software since the software in the market is experiencing geometric progression. Selected software should be acceptable by the firm and one user as well as it should be feasible for the system.

This document gives a detailed description of the software requirement specification. The study of requirement specification is focused especially on the functioning of the system. It allows the developer or analyst to understand the system, function to be carried out, the performance level to be obtained and corresponding interfaces to be established.

1. Operating System : Windows 8,10
2. Text Editor : Visual Studio Code / Sublime Text
3. GUI Tool : Qt Creator 6.1.2 (MinGW 8.1.0 64-bit)

2.2 HARDWARE REQUIREMENTS

The section of hardware configuration is an important task related to the software development insufficient random-access memory may affect adversely on the speed and efficiency of the entire system. The process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and application.

1. Intel CORE i3, i5, i7 Processor
2. 8 GB RAM
3. 40 GB HDD
4. 1024*768 Resolution Colour Display

- The hardware requirements specified are the hardware components/capacity of the system in which the application is developed and deployed.
- The above software requirements are the necessary software required to develop the application and the run the application

CHAPTER 3

SYSTEM DESIGN

3.1 CONTEXT FLOW DIAGRAM

The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.

A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

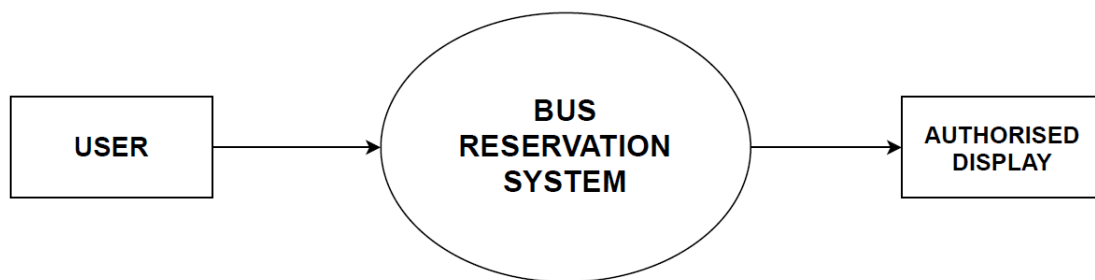


Fig 3.1 Context Flow Diagram

3.2 SYSTEM DESIGN

System design is a solution, a “HOW TO” approach to the creation of a new system. It translates system requirements into ways by which they can be made operational. It is a translational from a user-oriented document to a document oriented programmer. For that, it provides the understanding and procedural details necessary for the implementation. Systems design is the process of defining the architecture, modules, interfaces, and data for a system based on specified requirements.

A systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to take into account all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system

that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

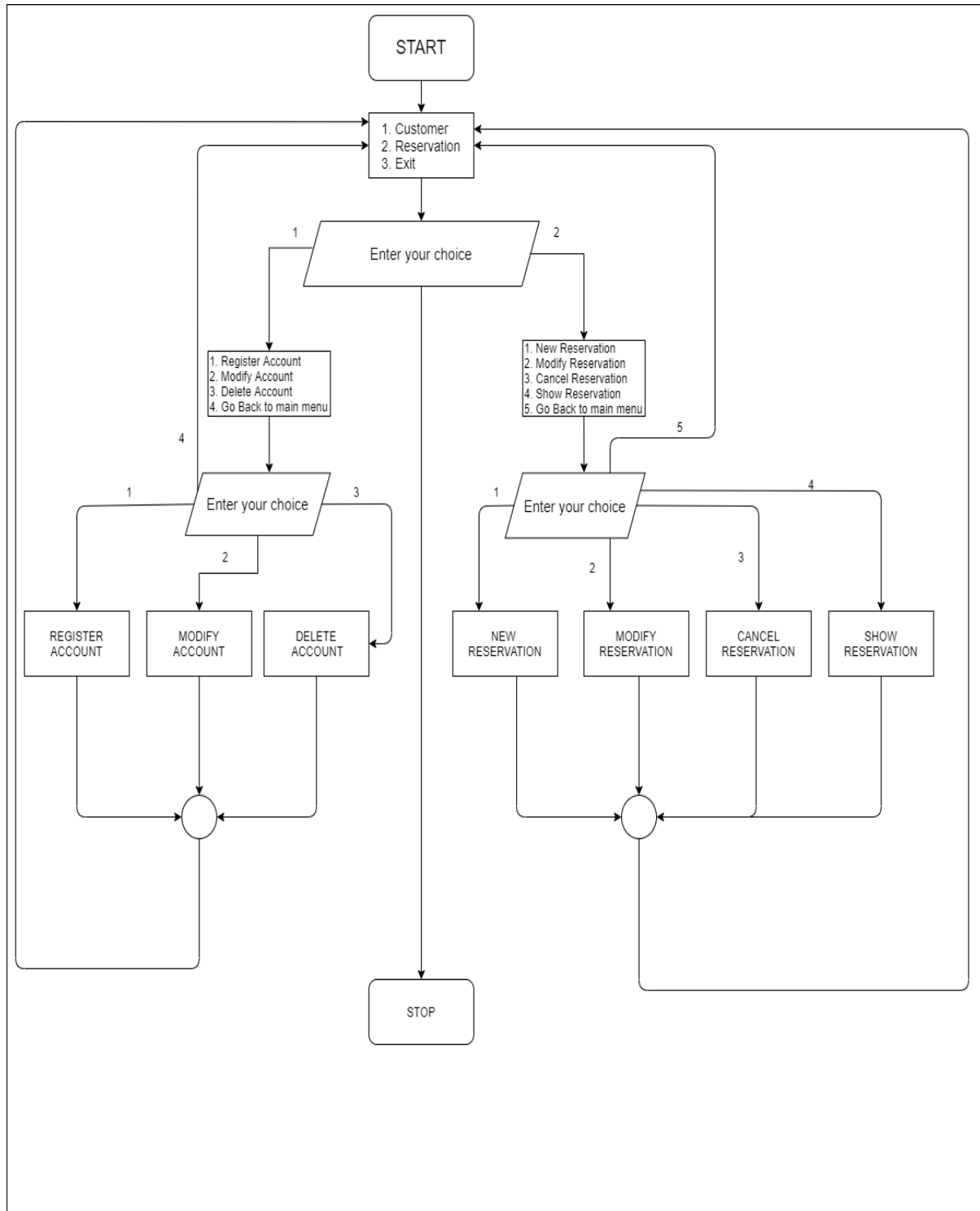


Fig 3.2 System Design for Bus Reservation System

3.3 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. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one.

3.3.1 DATA FLOW DIAGRAM FOR NEW ACCOUNT REGISTRATION

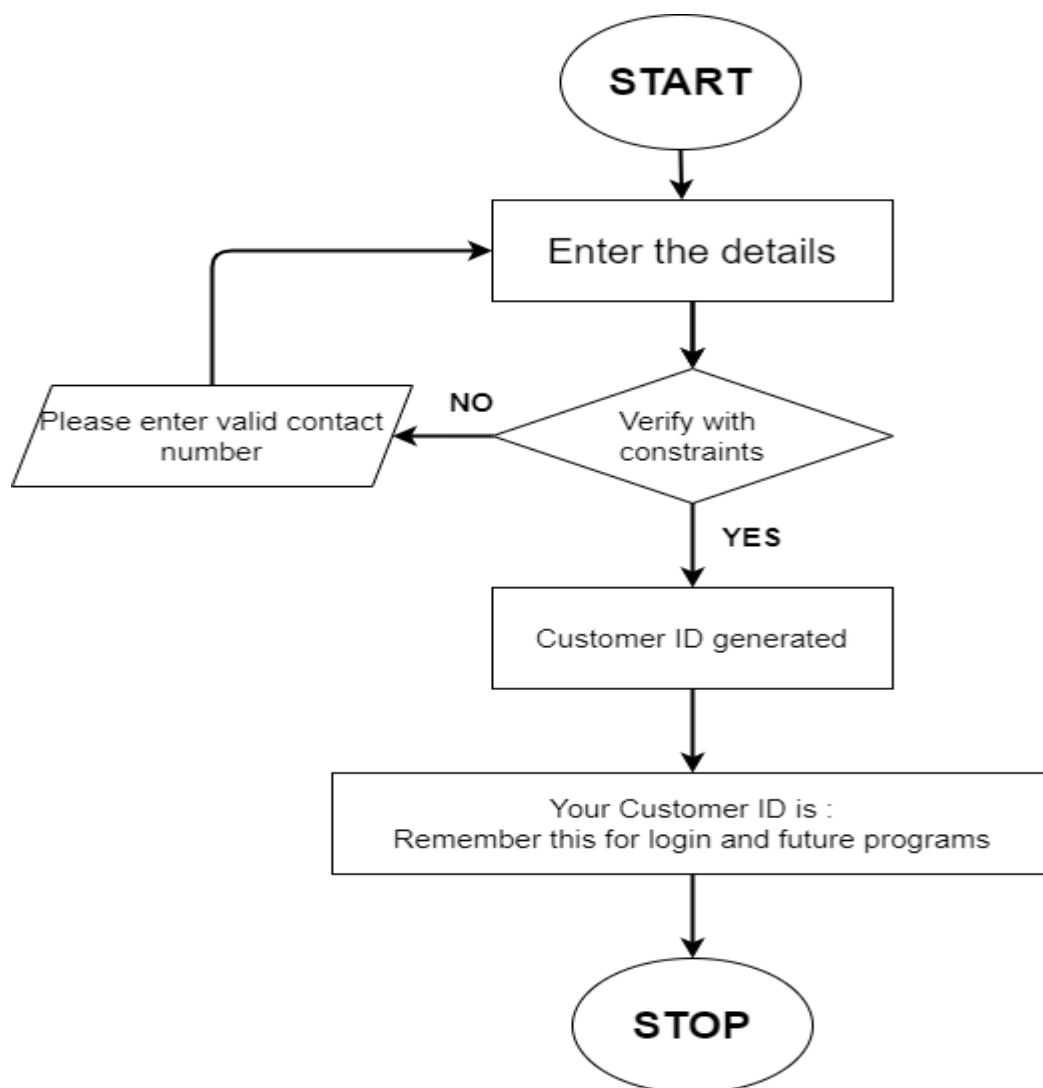


Fig 3.3 Data Flow diagram for new account registration

In the bus reservation system, the user should login to the application to get access to different functions. If the user is new, then a option of new account registration is given. The user will

enter all the necessary details required for account creation and the application will verify it with the constraints specified, if all details are valid, a customer ID will be generated and displayed.

3.3.2 DATA FLOW DIAGRAM FOR LOGIN

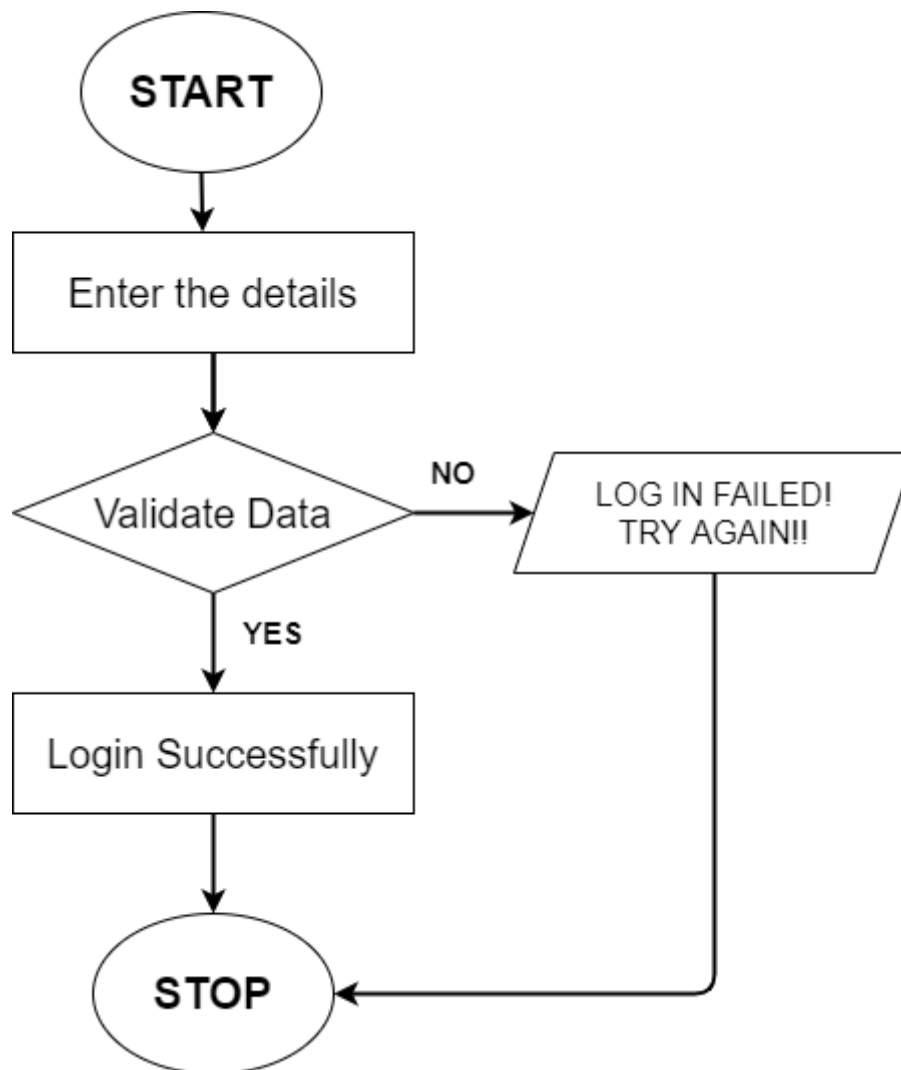


Fig 3.4 Data Flow diagram for login

In the Bus Reservation System, the user need to login to reserve, modify or cancel the seats. A user need to enter username and password. The application validates the data with the database. A user with valid login credentials can access the functionalities. If the credentials are not valid, error message will be displayed.

3.3.3 DATA FLOW DIAGRAM FOR MODIFY ACCOUNT DETAILS

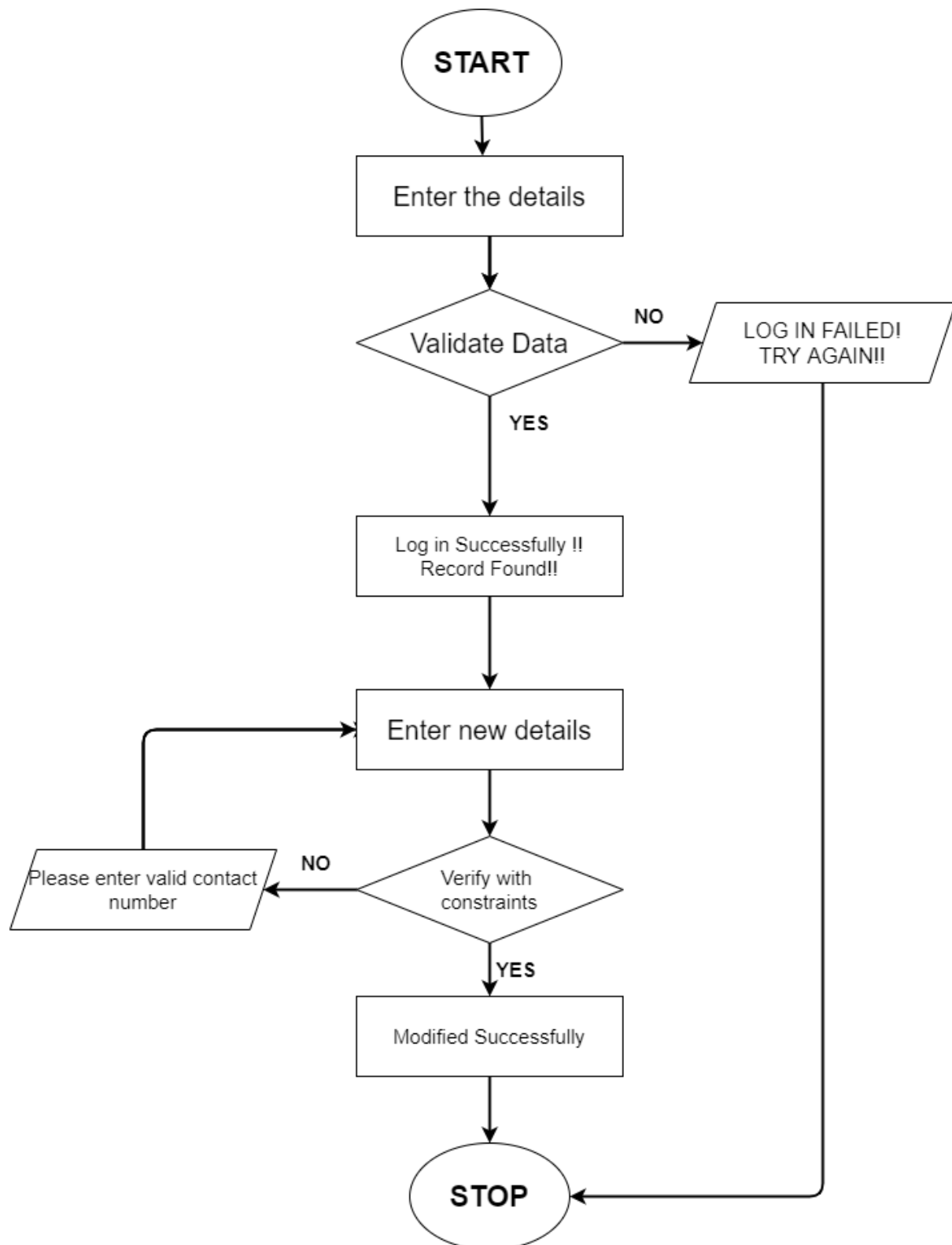


Fig 3.5 Data Flow Diagram for modify account details

In the Bus Reservation System, a user can modify the account details by login into the application and then providing the new details. The new details will be verified with the constraints, if it satisfies it, the details are modified successfully.

3.3.4 DATA FLOW DIAGRAM FOR DELETING ACCOUNT DETAILS

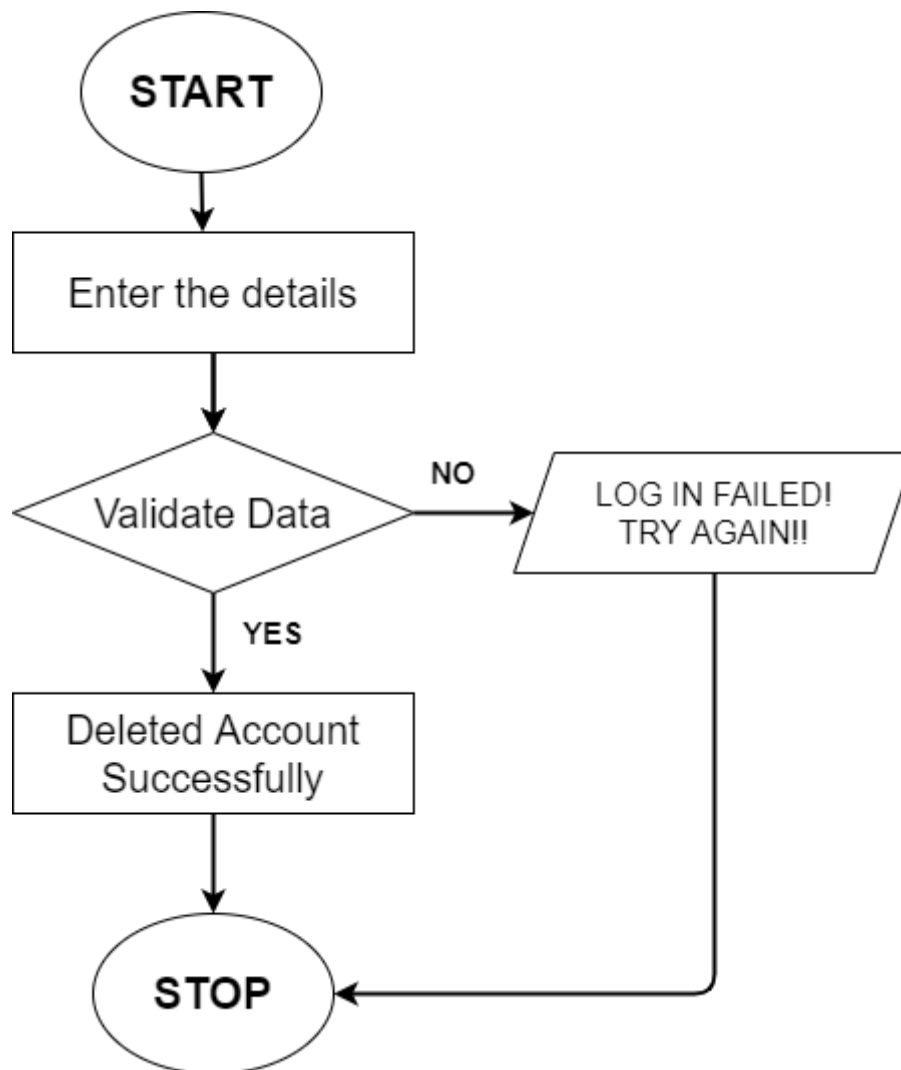


Fig 3.6 Data flow diagram for deleting account details

In the Bus Reservation System, a user can delete the account details by selecting the option of delete account details and providing the credentials. If a user has valid credentials, the account will be deleted from the file.

3.3.5 DATA FLOW DIAGRAM FOR NEW RESERVATION

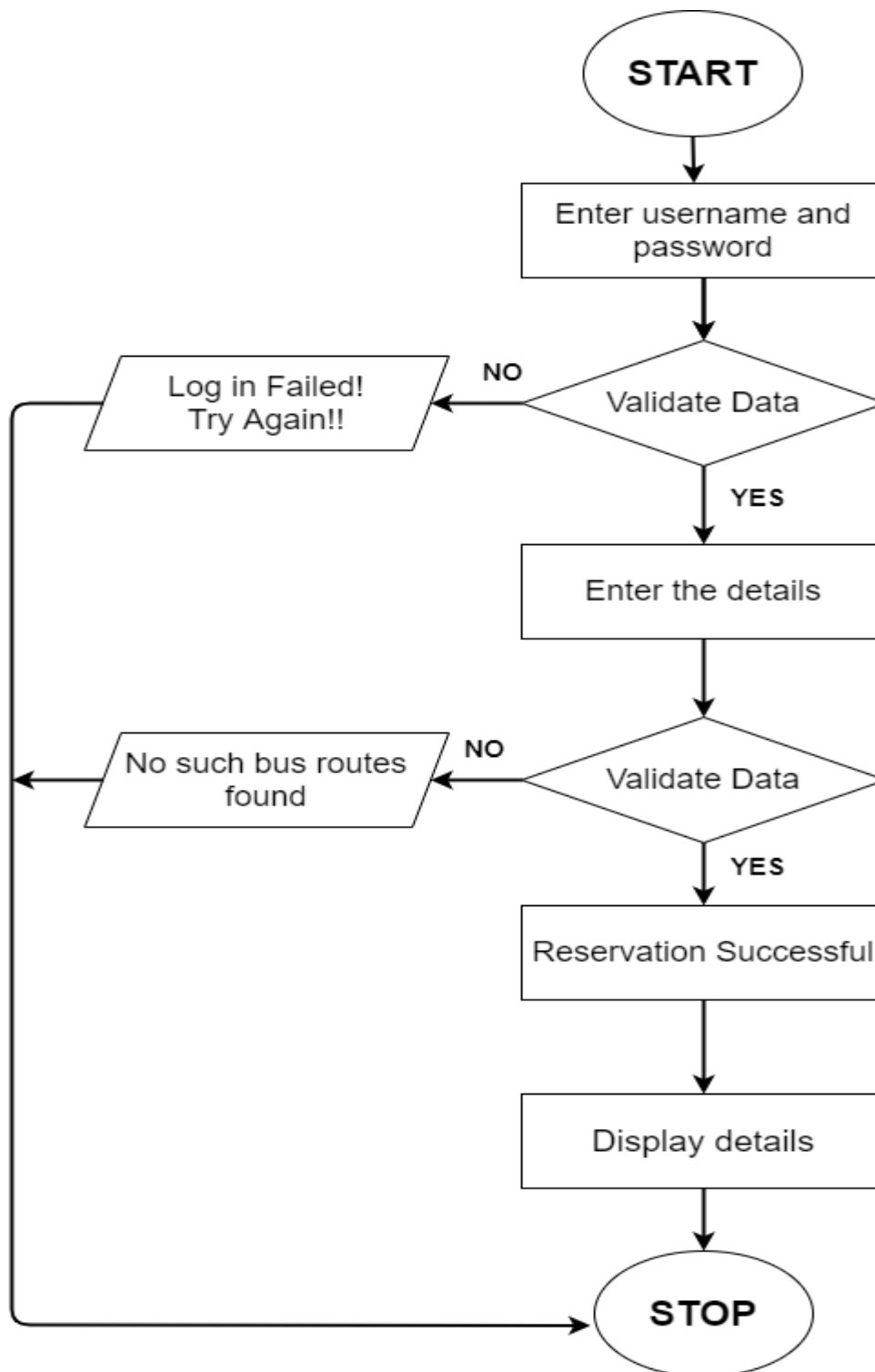


Fig 3.7 Data Flow Diagram for new reservation

In Bus Reservation System, a user can make a reservation for different bus routes, if the user has valid login credentials. After login, the user needs to provide details of departure and arrival station, the application will validate the data and then the ticket is displayed.

3.3.6 DATA FLOW DIAGRAM FOR MODIFY RESERVATION

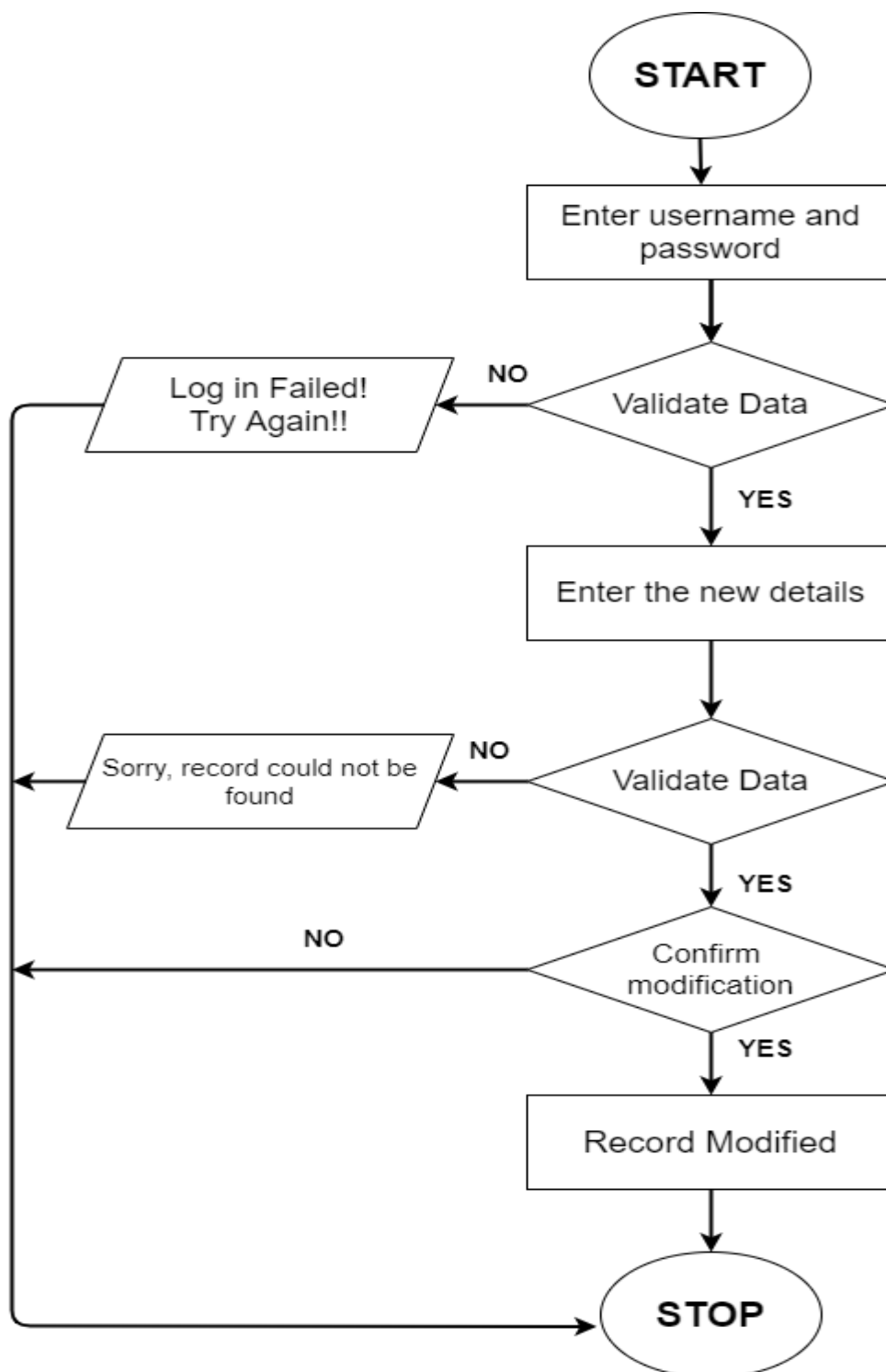


Fig 3.8 Data Flow Diagram for modify reservation

In the Bus Reservation System, a user can modify the reservation made by the credentials. A user needs to enter the valid credentials used for reservation and then need to enter new details. If the details are valid, reservation will be modified successfully.

3.3.7 DATA FLOW DIAGRAM FOR CANCEL RESERVATION

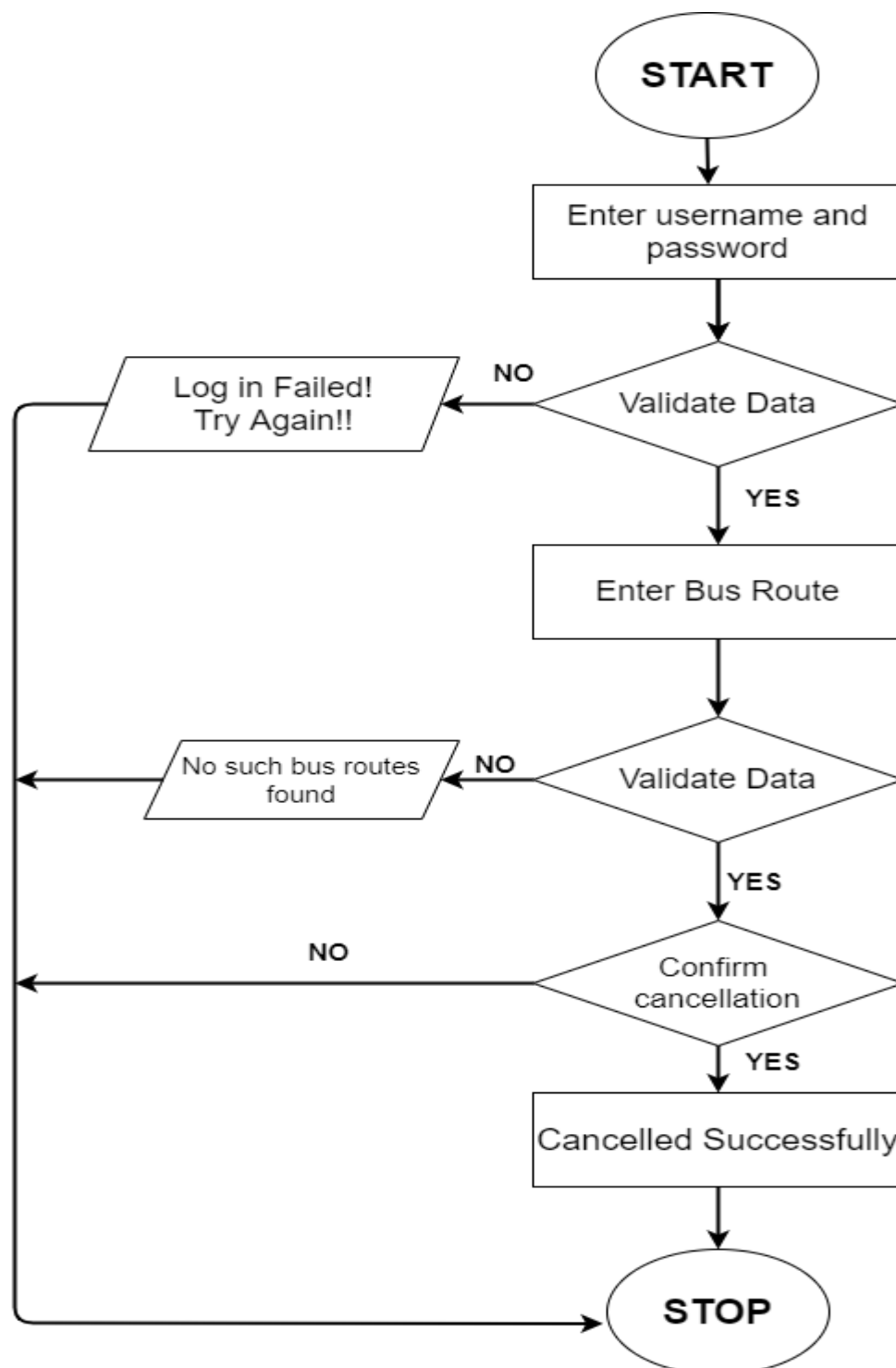


Fig 3.9 Data Flow Diagram for cancel reservation

In the Bus Reservation System, a user can cancel the reserved tickets by providing the login credentials used during reservation. After adding the credentials, user needs to confirm the cancellation of tickets. If confirmed, the reserved ticket is cancelled.

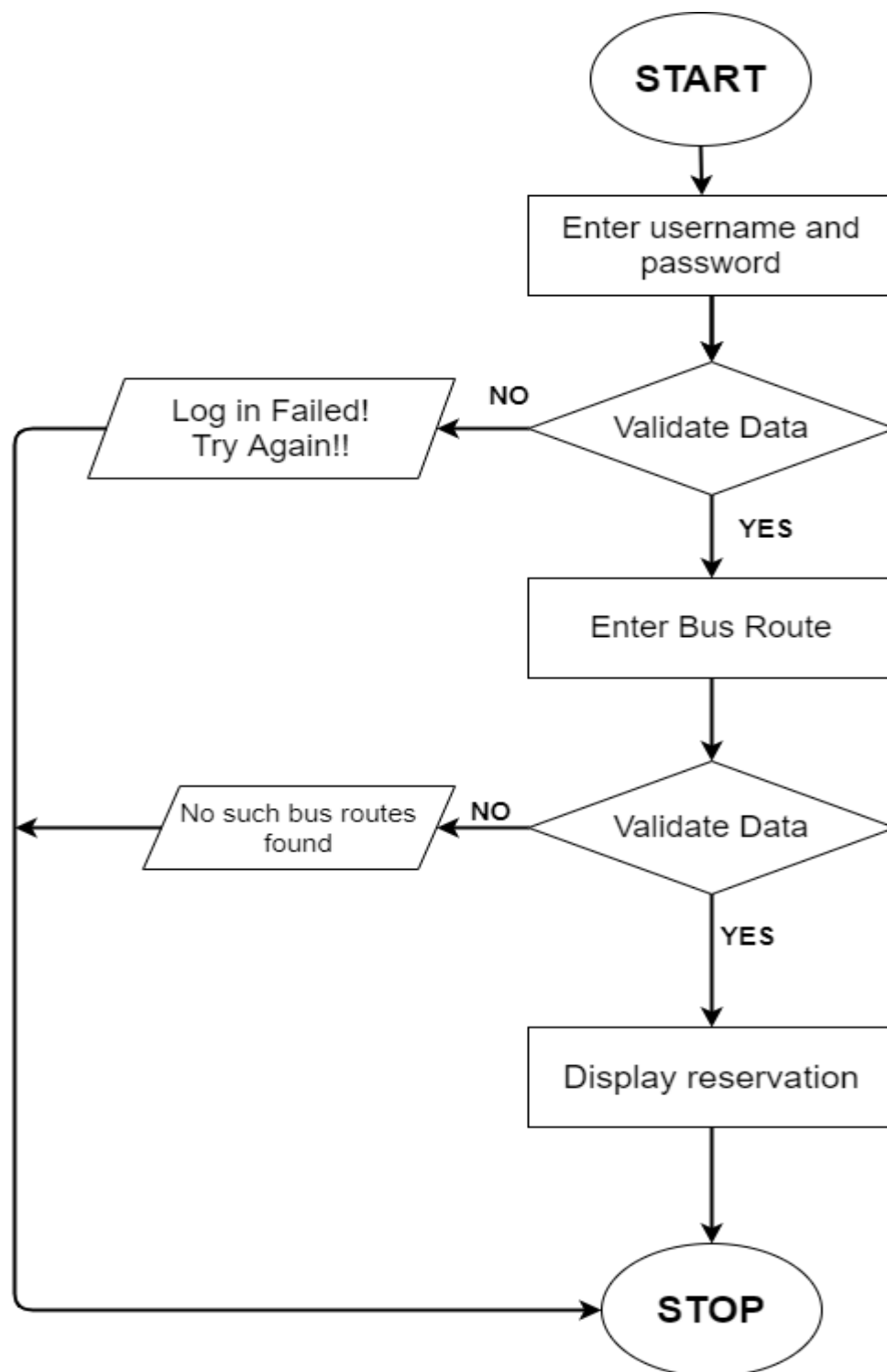
3.3.8 DATA FLOW DIAGRAM FOR DISPLAYING RESERVED TICKET

Fig 3.10 Data Flow Diagram for displaying reserved tickets

In the Bus Reservation System, a user can see the reserved tickets by providing valid login credentials and information of departure and arrival station. If the data entered are valid, the application will display the reserved tickets.

CHAPTER 4

IMPLEMENTATION

4.1 TECHNIQUE USED - BUFFER MANAGEMENT

"Buffer" is a generic term that refers to a block of computer memory that serves as a temporary placeholder.

A temporary storage area is called buffer. All standard input and output devices contain an input and output buffer. In standard C/C++, streams are buffered, for example in the case of standard input, when we press the key on keyboard, it isn't sent to your program, rather it is buffered by operating system till the time is allotted to that program. You might encounter the term in your computer, which uses RAM as a buffer, or in video streaming where a section of the movie you are streaming downloads to your device to stay ahead of your viewing.

In computer programming, data can be placed in a software buffer before it is processed. Because writing data to a buffer is much faster than a direct operation, using a buffer while programming in C and C++ makes a lot of sense and speeds up the calculation process. Buffers come in handy when a difference exists between the rate data is received and the rate it is processed.

4.2 BUFFERING STRATEGIES

Double Buffering:

- Two buffers can be used to allow processing and I/O to overlap.
- Suppose that a program is only writing to a disk.
- CPU wants to fill a buffer at the same time that I/O is being performed.
- If two buffers are used and I/O-CPU overlapping is permitted, CPU can be filling one buffer while the other buffer is being transmitted to disk.
- When both tasks are finished, the roles of the buffers can be exchanged.
- The actual management is done by operating system.

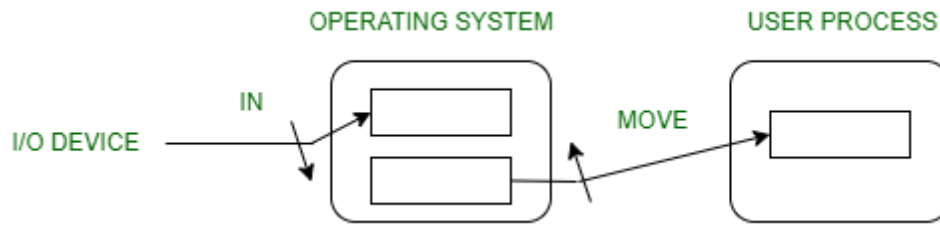


Fig 4.1 Double I/O Buffer

Multiple Buffering:

- In this, instead of two buffers any number of buffers can be used to allow processing and I/O to overlap

Buffer pooling:

- There is a pool of buffers.
- When a request for a sector is received, O.S. first looks to see that sector is in some buffer.
- If not there, it brings the sector to some free buffer. If no free buffer exists, it must choose an occupied buffer. (usually LRU strategy is used)

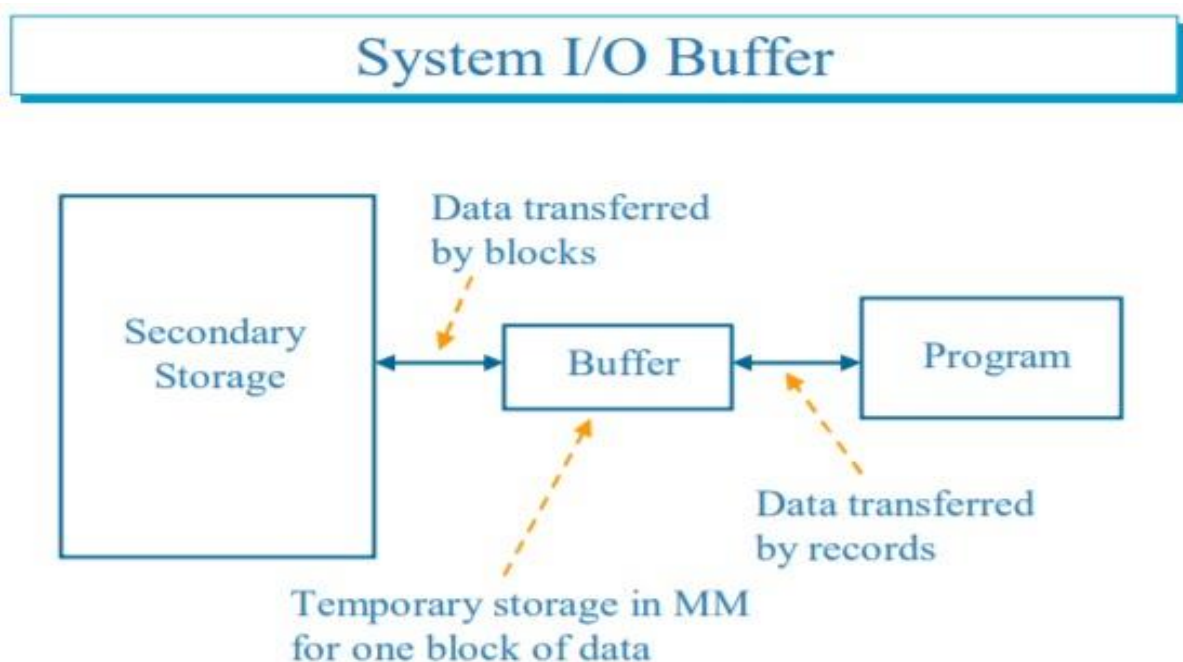


Fig 4.2 System I/O Buffer

4.3 HOW DOES IT EFFECT PROGRAMMING?

On various occasions you may need to clear the unwanted buffer so as to get the next input in the desired container and not in the buffer of previous variable. For example, in case of C after encountering “scanf ()”, if we need to input a character array or character, and in case of C++, after encountering “cin” statement, we require to input a character array or a string, we require to clear the input buffer or else the desired input is occupied by buffer of previous variable, not by the desired container. On pressing “Enter” (carriage return) on output screen after the first input, as the buffer of previous variable was the space for new container (as we didn’t clear it), the program skips the following input of container.

CHAPTER 5

TESTING

Software testing is the stage of implementation, which is aimed at ensuring that the software works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance. Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies, a test plan is carried out on each module. The various tests performed are unit testing, integration testing and user acceptance testing.

5.1 UNIT TESTING

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software perform as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module). Unit Testing frameworks, drivers, stubs, and routines that are assembled add integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by Java. The various controls are tested to ensure that each performs its action as required.

During development, a software developer may code criteria, or results that are known to be good, into the test to verify the unit's correctness. During test case execution, frameworks log tests that fail any criterion and report them in a summary. For this, the most commonly used approach is test - function - expected value.

Table 5.1 Test Cases

Sl. no.	TEST CASE DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	User with correct username and password	Login Successfully!	Login Successfully!	PASS
2	User with invalid username or password	Login Failed!! Try Again!	Login Failed!! Try Again!	PASS
3	User enters less or more than 10 digit in contact number	Please enter valid contact number	Please enter valid contact number	PASS
4	Wrong details for departure station while new reservation, modification or cancellation of tickets	No such Bus routes found	No such Bus routes found	PASS
5	Wrong details for arrival station while new reservation, modification or cancellation of tickets	No such Bus routes found	No such Bus routes found	PASS
6	All details entered correctly	Executed successfully	Executed successfully	PASS

CHAPTER 6

RESULTS

The project is compiled and executed on Visual Studio Code 2019. The few snapshots are shared here to show the working of the Application.

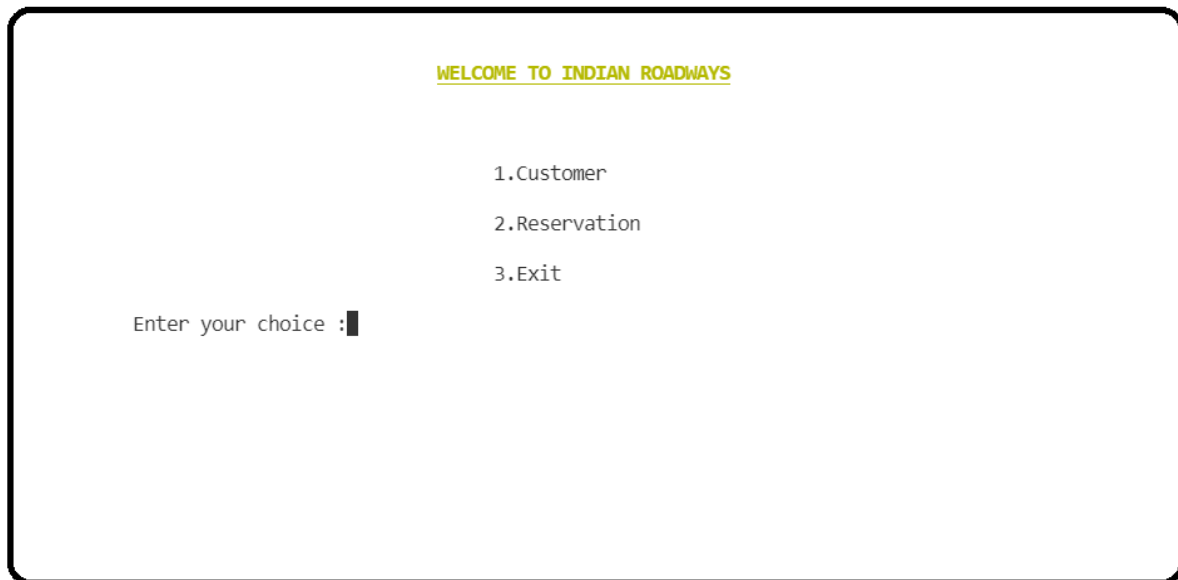


Fig 6.1 Bus Reservation System Home Page

When the Bus Reservation System application is executed, the above shown snap is the first thing that appears. From here, user can choose the desired option.

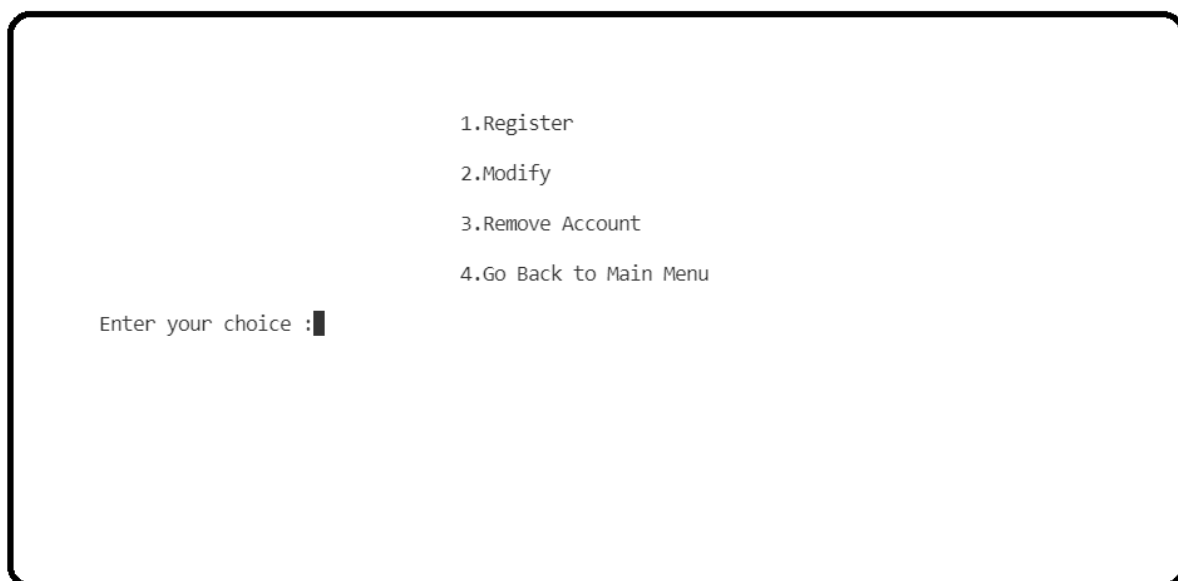


Fig 6.2 Customer Menu page

The customer menu page comes with the option of creating new account, modify the previous created account, delete the user account or go back to the home page.

REGISTER HERE!!

(Please fill in this information for the Registration)

Note: All entries in the form should be of one word and should not contain spaces.

Passenger Name:

ROSHAN

Passenger City:

THANE

Passenger : Contact No

7231949705

Passenger : UserName

ROSH001

Passenger : Password

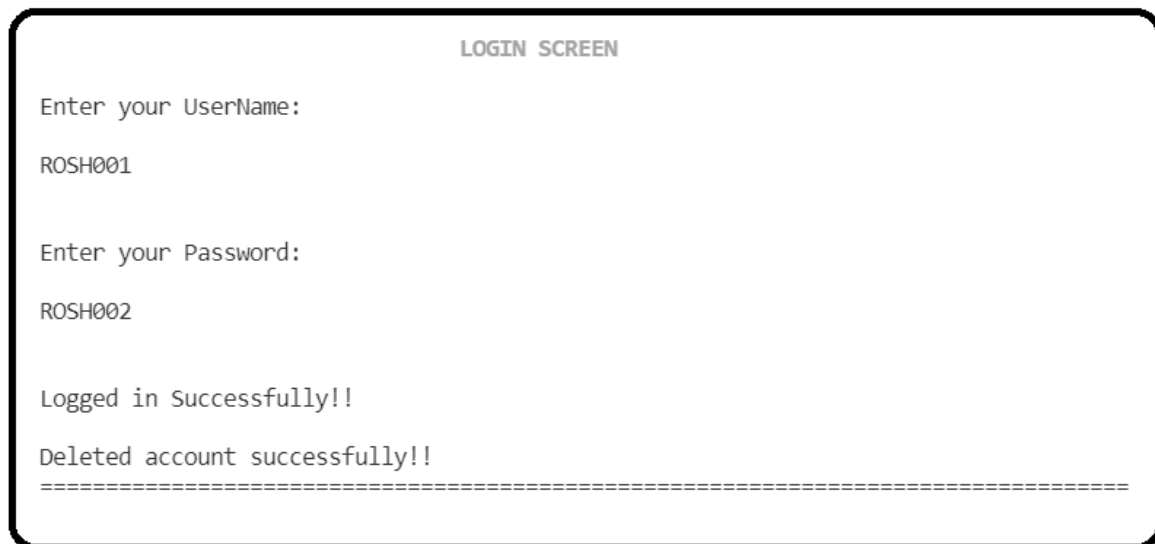
ROSH002

Your Customer ID is: **190009**

Remember this for login and other future purposes.

Fig 6.3 Create new account page

The above figure is of creation of new user account. It requires user to fill in all the details required and then generate a customer ID for the customer. This username and password will be used by the customer for their reservation, cancellation or the other functionalities in the application.



LOGIN SCREEN

Enter your UserName:
ROSH001

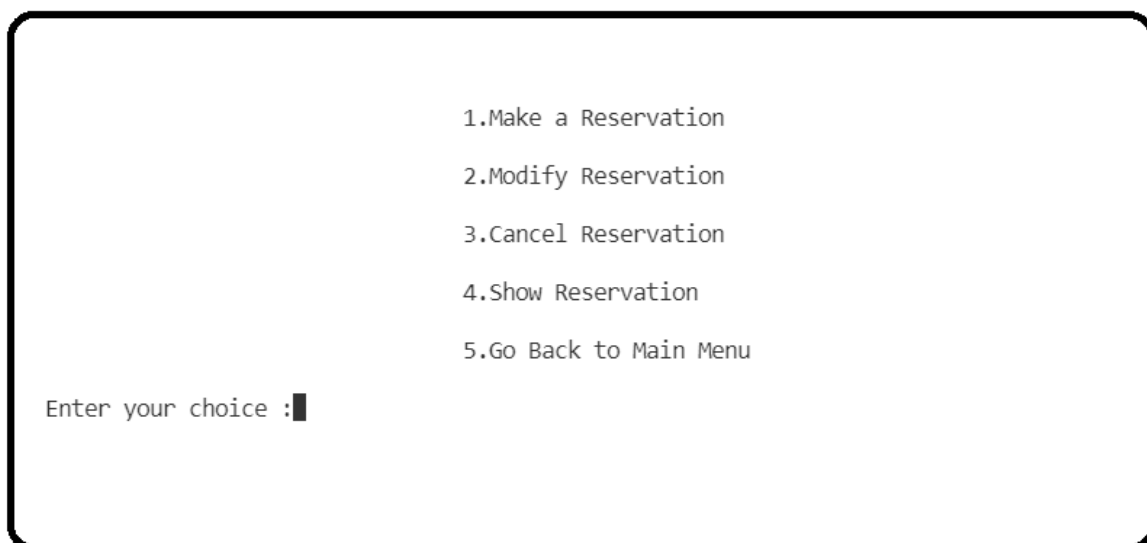
Enter your Password:
ROSH002

Logged in Successfully!!

Deleted account successfully!!
=====

Fig 6.4 Account deletion page

This figure is of account deletion to protect customer data privacy. With the valid username and password, the user can delete his data with the system.

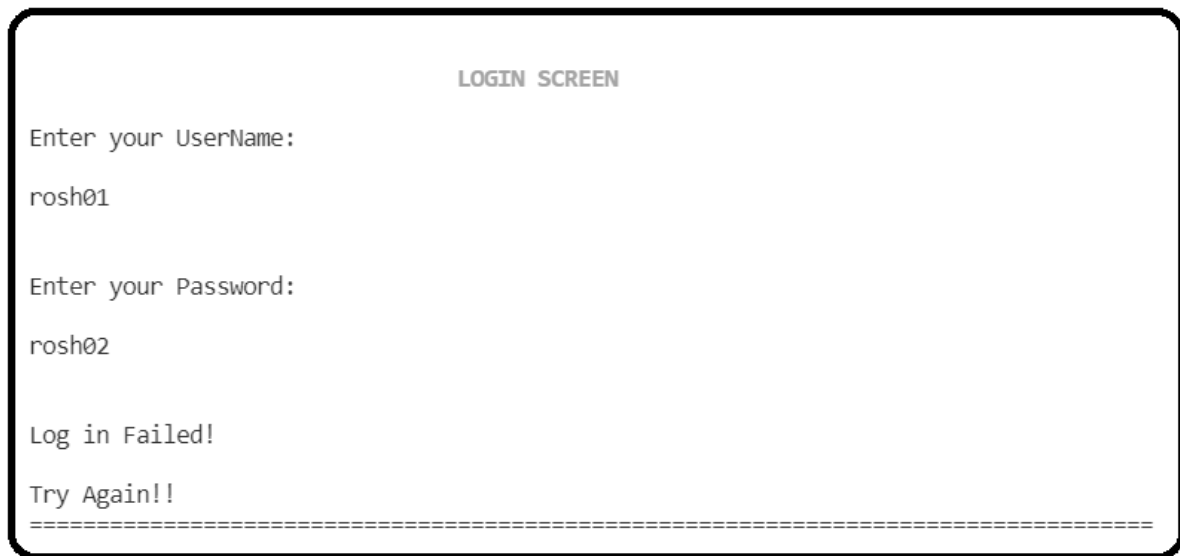


1.Make a Reservation
2.Modify Reservation
3.Cancel Reservation
4.Show Reservation
5.Go Back to Main Menu

Enter your choice :█

Fig 6.5 Reservation Menu Page

This page shows the Reservation Menu Page with the option of new reservation, modify reservation, cancel reservation, show reservation or go back to main menu. To access the mentioned functionalities, the user needs to login with the valid username and password.



A screenshot of a login screen titled "LOGIN SCREEN". It contains two input fields: "Enter your UserName:" with the value "rosh01" and "Enter your Password:" with the value "rosh02". Below these fields, the text "Log in Failed!" and "Try Again!!" is displayed. At the bottom, there is a dashed line.

```
LOGIN SCREEN

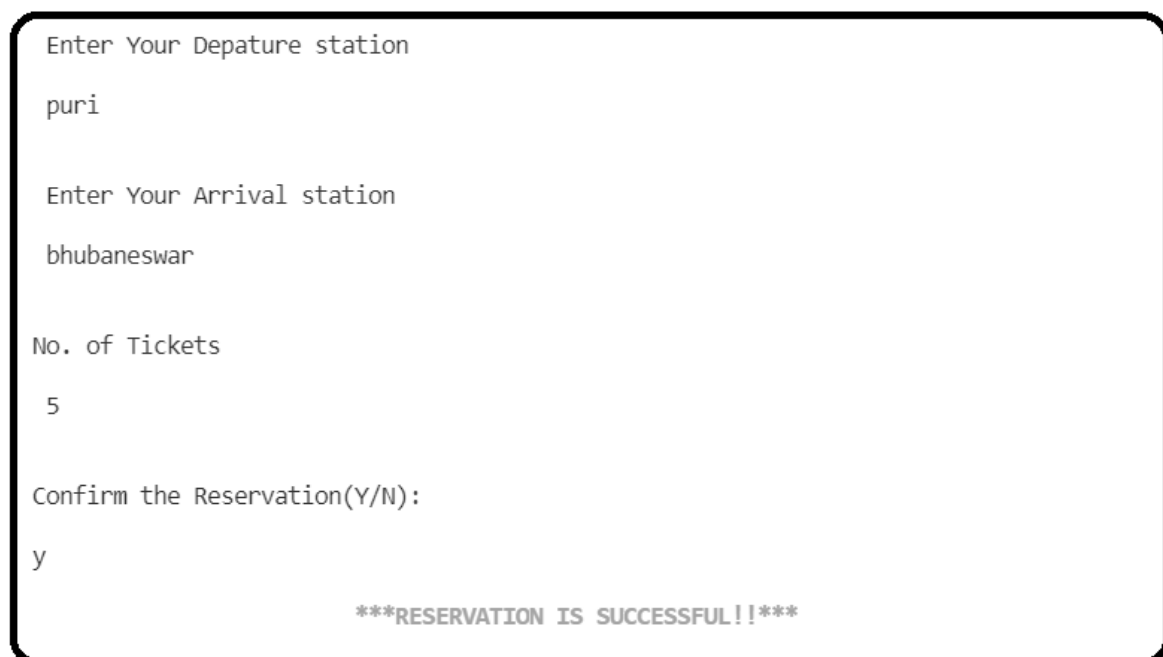
Enter your UserName:
rosh01

Enter your Password:
rosh02

Log in Failed!
Try Again!!
=====
```

Fig 6.6 Login page

The above image is of the login page, when the details provided are invalid. It shows the message that login failed, try again.



A screenshot of a new reservation page. It contains four input fields: "Enter Your Depature station" with the value "puri", "Enter Your Arrival station" with the value "bhubaneswar", "No. of Tickets" with the value "5", and "Confirm the Reservation(Y/N):" with the value "y". At the bottom, the text "***RESERVATION IS SUCCESSFUL!!***" is displayed.

```
Enter Your Depature station
puri

Enter Your Arrival station
bhubaneswar

No. of Tickets
5

Confirm the Reservation(Y/N):
y

***RESERVATION IS SUCCESSFUL!!***
```

Fig 6.7 New Reservation Page

The above image is of the reservation page, the user needs to login with valid credentials for this page and provide all the details required. If the reservation is successful, the application will display the ticket.

```
*****
                        INDIAN ROADWAYS
Customer ID :190009
Dep. St. : puri           Departure Time: 7:15PM
Arrival St. : bhubaneswar   Arrival Time: 8:30PM
No. of Tickets :5
-----
                Total Fare = 375
*****
```

Fig 6.8 Displaying tickets

If all the details filled by the user is valid and the required number of seats are available in the bus , it will display the ticket as in figure above.

```
Depature station
puri

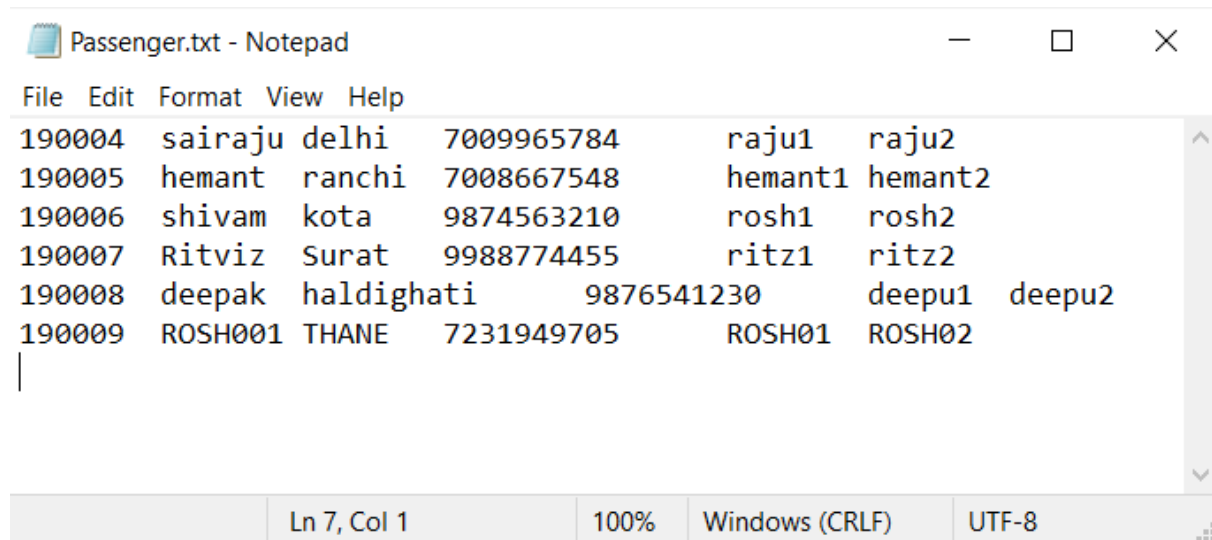
Arrival station
bhubaneswar

-----
***Press 1 for final confirmation or else press 0!!***
1

***Your Reservation Cancelled Successfully!!***
```

Fig 6.9 Cancellation of tickets

The user needs to login in the user page with valid credentials, and provide the necessary details. If the details are valid the application will ask for user confirmation.



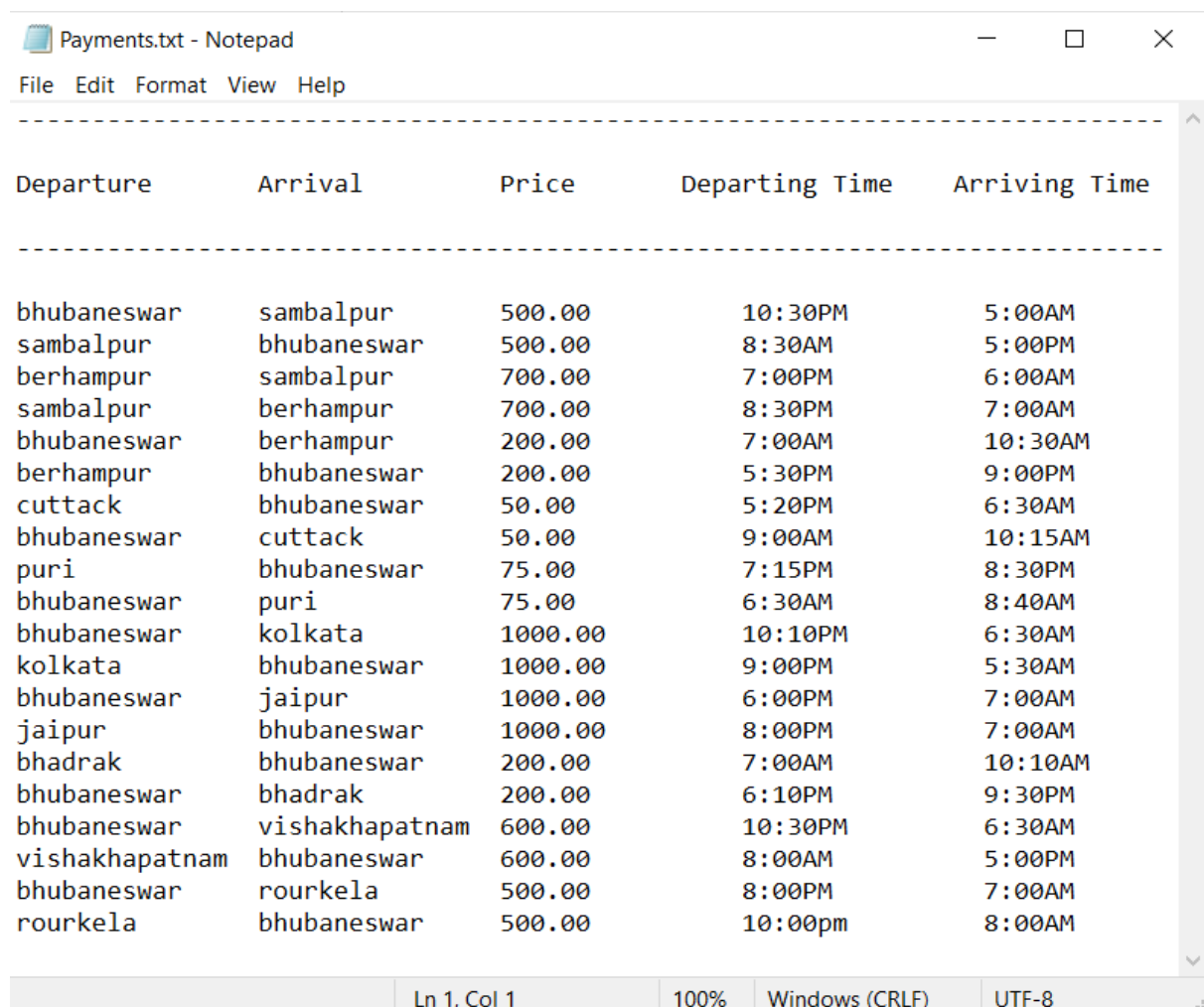
```

File Edit Format View Help
190004 sairaju delhi 7009965784 raju1 raju2
190005 hemant ranchi 7008667548 hemant1 hemant2
190006 shivam kota 9874563210 rosh1 rosh2
190007 Ritviz Surat 9988774455 ritz1 ritz2
190008 deepak haldighati 9876541230 deepu1 deepu2
190009 ROSH001 THANE 7231949705 ROSH01 ROSH02

```

Fig 6.10 Passenger.txt file

This file contains all the details of the user such as name, contact number, username and password. The application crosschecks with this file for every login function.



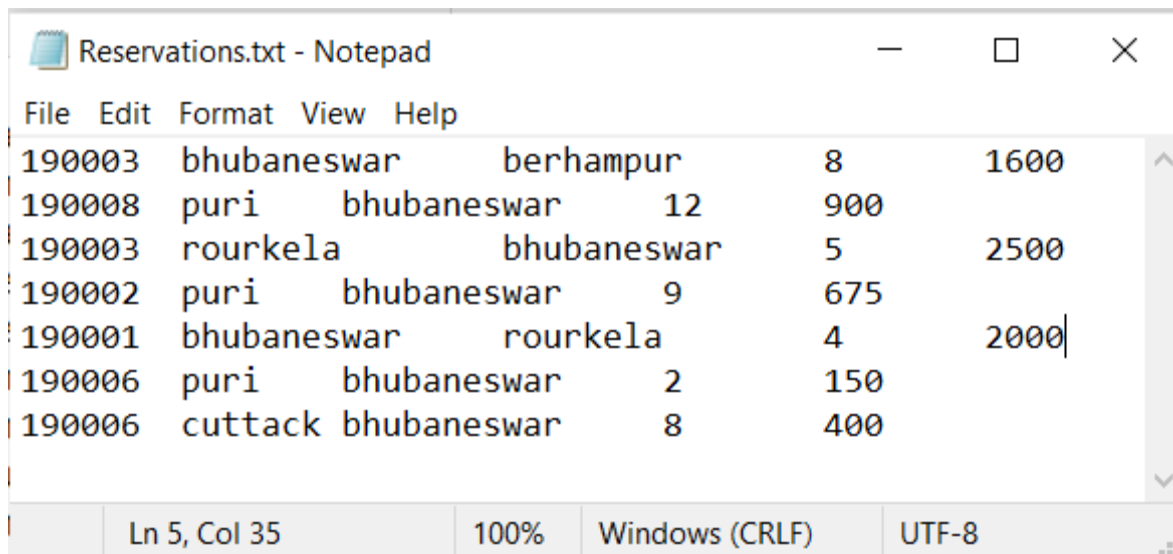
```

File Edit Format View Help
-----
Departure      Arrival      Price      Departing Time      Arriving Time
-----
bhubaneswar    sambalpur    500.00      10:30PM             5:00AM
sambalpur      bhubaneswar  500.00      8:30AM              5:00PM
berhampur      sambalpur    700.00      7:00PM              6:00AM
sambalpur      berhampur    700.00      8:30PM              7:00AM
bhubaneswar    berhampur    200.00      7:00AM              10:30AM
berhampur      bhubaneswar  200.00      5:30PM              9:00PM
cuttack        bhubaneswar  50.00       5:20PM              6:30AM
bhubaneswar    cuttack      50.00       9:00AM              10:15AM
puri           bhubaneswar  75.00       7:15PM              8:30PM
bhubaneswar    puri         75.00       6:30AM              8:40AM
bhubaneswar    kolkata     1000.00     10:10PM             6:30AM
kolkata        bhubaneswar  1000.00     9:00PM              5:30AM
bhubaneswar    jaipur      1000.00     6:00PM              7:00AM
jaipur         bhubaneswar  1000.00     8:00PM              7:00AM
bhadrak        bhubaneswar  200.00      7:00AM              10:10AM
bhubaneswar    bhadrak     200.00      6:10PM              9:30PM
bhubaneswar    vishakhapatnam 600.00     10:30PM             6:30AM
vishakhapatnam bhubaneswar  600.00      8:00AM              5:00PM
bhubaneswar    rourkela    500.00      8:00PM              7:00AM
rourkela       bhubaneswar  500.00      10:00pm             8:00AM

```

Fig 6.11 Payments.txt

This file contains details of bus routes, their fares and timings through which user can book their ticket and the fare is calculated.



190003	bhubaneswar	berhampur	8	1600
190008	puri	bhubaneswar	12	900
190003	rourkela	bhubaneswar	5	2500
190002	puri	bhubaneswar	9	675
190001	bhubaneswar	rourkela	4	2000
190006	puri	bhubaneswar	2	150
190006	cuttack	bhubaneswar	8	400

Fig 6.12 Reservations.txt file

This file contains all the data of the reserved tickets, fares, their customer Id, No. of seats booked.

CHAPTER 7

CONCLUSION

This project is developed successfully and the performance is found to be satisfactory. This project is designed to meet the requirements of the user. It has been developed in C++ and the database has been built using the Buffer Management Technique. The user will be able to create new account, modify or delete account. User can also do new reservations, modify or cancel the earlier reservations. We have designed the project to provide the user with easy retrieval of data, details of bus routes, etc. In this project, the user is provided with username and password to get access to all the functionalities.

REFERENCES

- File Structures: An Object Oriented approach with C++, 3rd Edition, Pearson Education
- <https://stackoverflow.com>
- <https://www2.cs.sfu.ca>
- <https://www.geeksforgeeks.com>

APPENDIX

main.cpp

```
#include <iostream>
#include <stdio.h>
#include <map>
#include <math.h>
#include <vector>
#include <set>
#include <queue>
#include <numeric>
#include <fstream>
#include <bits/stdc++.h>
using namespace std;
#include "reservations.h"

#define UNDERLINE "\033[33;4m"
#define CLOSEUNDERLINE "\033[0m"

string gen_id();

int main()
{
    int mainchoice, subChoiceOne, nt;
    string name, customer_id, City, Password, UserName, ContactNumber, DepSt, Arr
St;
    try
    {
        mainMenu:
        cout << "\n\n\t\t\t " << UNDERLINE << bold_on << "WELCOME TO INDIA
N ROADWAYS" << bold_off << CLOSEUNDERLINE << "\n\n";
        cout << "\n\n\t\t\t1.Customer\n\n"
        << "\t\t\t2.Reservation\n\n"
        << "\t\t\t3.Exit\n\n"
        << "Enter your choice :";
        try
        {
            cin >> mainchoice;
            if ((mainchoice != 1) & (mainchoice != 2) & (mainchoice != 3))
            {
                throw 11;
            }
        }
        catch (int x)
        {
            cout << "\n\n Please select a Relevant Number from the menu \n\n";
            return main();
        }
    }
}
```

```

switch (mainchoice)
{
case 1:
{
subMenu:

cout << "=====
=====\\n"
    << "\\n\\n\\t\\t\\t1.Register\\n\\n"
    << "\\t\\t\\t2.Modify\\n\\n"
    << "\\t\\t\\t3.Remove Account\\n\\n"
    << "\\t\\t\\t4.Go Back to Main Menu\\n\\n"
    << "Enter your choice :";

try
{
    cin >> subChoiceOne;
    if ((subChoiceOne != 1) & (subChoiceOne != 2) & (subChoiceOne != 3) &
(subChoiceOne != 4))
    {
        throw 12;
    }
}
catch (int y)
{
    cout << "\\n\\nError - Please select a Relevant Number from the menu \\n\\n";
    goto subMenu;
}
switch (subChoiceOne)
{
case 1:
{

cout << "=====
=====\\n";
    cout << "\\t\\t\\t" << UNDERLINE << bold_on << "REGISTER HERE!!" <<
bold_off << CLOSEUNDERLINE << "\\n\\n";
    cout << "\\t(Please fill in this information for the Registration)\\n";
    cout << "Note: All entries in the form should be of one word and should not
contain spaces.";
    cout << "\\n\\nPassenger Name:\\n\\n";
    cin >> name;
    cin.ignore(numeric_limits<streamsize>::max(), '\\n');
    cout << "\\n\\nPassenger City:\\n\\n ";
    cin >> City;
    cin.ignore(numeric_limits<streamsize>::max(), '\\n');
    contact:
    cout << "\\n\\nPassenger : Contact No\\n\\n ";
    cin >> ContactNumber;
    if (ContactNumber.size() > 10 || ContactNumber.size() < 10)
    {

```

```

        cout << "\n\t\t!!!!Please Enter Valid Contact Number:!!!!" << endl;
        goto contact;
    }
    cout << "\n\nPassenger : UserName \n\n ";
    cin >> UserName;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "\n\nPassenger : Password\n\n ";
    cin >> Password;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    //make the password protective
    customer_id = (gen_id());
    cout << "\n\t\tYour Customer ID is: " << bold_on << customer_id << bold_
off << endl;
    cout << "\tRemember this for login and other future purposes.\n"
        << endl;
    Passenger P1(name, customer_id, City, ContactNumber, UserName, Passw
ord);
        goto mainMenu;
    }; // Sub choice first case
    break;
    case 2:
    {
        Passenger P2;
        int n = P2.Login();
        if (n)
        {
            P2.SearchFile_and_Update(n);
        }
        goto subMenu;
    };
    break;
    case 3:
    {
        Passenger P3;
        int n = P3.Login();
        if (!n)
        {
            goto subMenu;
        }
        P3.Deletes(n);
        goto subMenu;
    };
    break;
    case 4:
        goto mainMenu;
        break;
    } //sub switch one
}; //main choice first case
break;
case 2:

```

```

{
subMenu2:
int subChoice2;
cout << "=====
=====
<< "\n\n\t\t\t1.Make a Reservation\n\n"
<< "\t\t\t2.Modify Reservation\n\n"
<< "\t\t\t3.Cancel Reservation\n\n"
<< "\t\t\t4.Show Reservation\n\n"
<< "\t\t\t5.Go Back to Main Menu"
<< "\n\nEnter your choice :";
try
{
cin >> subChoice2;
if ((subChoice2 != 1) & (subChoice2 != 2) & (subChoice2 != 3) & (subChoice2 != 4) & (subChoice2 != 5))
{
throw 13;
cin.ignore(numeric_limits<streamsize>::max(), '\n');
}
}
catch (int y)
{
cout << "\n\nError - Please select a Relevant Number from the menu \n\n";
goto subMenu2;
}
switch (subChoice2)
{
case 1:
{
Reservations R2;
int n = R2.Login();
if (!n)
{
cout << "Try Again!!" << endl;
goto subMenu2;
}
char check = 'y';
if (check == 'y')
{
Payments Pa1;
Pa1.Show();
string search1, search2;
cout << "\n\n Enter Your Depature station\n\n ";
cin >> DepSt;
cin.ignore(numeric_limits<streamsize>::max(), '\n');
cout << "\n\n Enter Your Arrival station\n\n ";
cin >> ArrSt;
cin.ignore(numeric_limits<streamsize>::max(), '\n');
cout << "\n\nNo. of Tickets\n\n ";

```

```
        cin >> nt;
        Reservations R2(n, DepSt, ArrSt, nt);
        goto mainMenu;
    }
};
break;
case 2:
{
    Reservations R3;
    int n = R3.Login();
    if (!n)
    {
        goto subMenu2;
    }
    R3.SearchFile_and_Update(n);
    cout << "Moving to Main Menu...\n\n";
    goto subMenu2;
};
break;
case 3:
{
    Reservations R4;
    int n = R4.Login();
    if (!n)
    {
        goto subMenu2;
    }
    R4.Deletes(n);
    cout << "Moving to the Main Menu...\n\n";
    goto mainMenu;
};
break;
case 4:
{
    Reservations myr;
    int n = myr.Login();

    myr.Show(n);
    goto mainMenu;
}
break;
case 5:
    goto mainMenu;
    break;
} // second sub switch end
};
break;
case 3:
{
    cout << "\n\nThanks for Using Our Service! Hope to see you again!\n\n";
```

```
        exit(0);
    };
    break;
} // main switch end
}
catch (...)
{
    cout << "\n\nUnexpected Error occurred, Program is terminating!!\n\n";
    exit(0);
}
return 0;
}

//generates a unique customer_id
string gen_id()
{
    long long x = 190000;
    string line;
    /* Creating input filestream */
    ifstream file("Passenger.txt");
    string s1, s2, s3, s4, s5, s6, id;
    while (file >> s1 >> s2 >> s3 >> s4 >> s5 >> s6)
    {
        x = stoll(s1);
    }
    id = to_string(x + 1);
    return id;
}
```

passenger.h

```
#include <iostream>
#include <map>
#include <math.h>
#include <vector>
#include <set>
#include <queue>
#include <numeric>
#include <fstream>
using namespace std;
int count01;
//text bold
ostream &bold_on(ostream &os)
{
    return os << "\e[1m";
}

ostream &bold_off(ostream &os)
{
    return os << "\e[0m";
}

// class Passenger
class Passenger
{
    string str_City, str_ContactNumber, str_UserName, str_Password, str_name;

protected:
    string str_customer_id;

public:
    Passenger() {}
    Passenger(string name, string customer_id, string City, string ContactNumber, string Username, string Password);
    int Login();
    void SearchFile_and_Update(int);
    void Deletes(int);
    ~Passenger();
};
Passenger :: ~Passenger(){}

// Passenger constructor
Passenger :: Passenger(string name, string customer_id, string City, string ContactNumber, string Username, string Password)
{
    str_name = name;
    str_customer_id = customer_id;
    str_City = City;
    str_ContactNumber = ContactNumber;
    str_UserName = Username;
```



```
    str_Password = Password;
    fstream registration;
    registration.open("Passenger.txt", ios::app | ios::out | ios::ate);
    registration << str_customer_id << "\t" << str_name << "\t" << str_City << "\t" <<
    str_ContactNumber << "\t" << str_UserName
        << "\t" << str_Password << "\n";
    registration.close();
}

// Delete record function
void Passenger::Deletes(int log_customerid)
{
    int customer_id;
    string name, city, username, password, cn;
    // cout << "\n\nEnter your Customer ID to delete the record :\n\n";
    // cin >> str_customer_id;
    fstream deleteF;
    ofstream df;
    deleteF.open("Passenger.txt", ios::in);
    while (deleteF >> customer_id >> name >> city >> cn >> username >> password)
    {
        df.open("temp2.txt", ios::out | ios::app);
        if (customer_id != log_customerid)
        {
            df << customer_id << "\t" << name << "\t" << city << "\t"
                << cn << "\t" << username
                << "\t" << password << "\n";
        }
        else
        {
            count01++;
        }
        df.close();
    } // end of while
    deleteF.close();
    if (count01 == 0)
    {
        remove("temp2.txt");
        cout << "\n\nRecord not found\n\n";
    }
    if (count01 > 0)
    {
        remove("Passenger.txt");
        rename("temp2.txt", "Passenger.txt");
        cout<<"Deleted account successfully!!\n";
    }
} //end of function DeleteRecord

// Search file and update
void Passenger ::SearchFile_and_Update(int log_customerid)
```

```

{
    count01 = 0;
    string name, city, username, password, cn;
    int customer_id;
    cout << "\n\n-----
\n\n";
    // cout << "\n\nEnter your Customer ID :\n\n";
    // cin >> str_customer_id;
    ifstream PassengerIn;
    ofstream writeFile;
    PassengerIn.open("Passenger.txt");
    while (PassengerIn >> customer_id >> name >> city >> cn >> username >> password)
    {
        writeFile.open("temp1.txt", ios::app);
        if (customer_id == log_customerid)
        {
            cout << "\n\nRecord found!\n\n";
            cout << "\n\n-----
--\n\n";
            cout << "\n\nNew Passenger Name:\n\n ";
            cin >> str_name;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\n\nNew Passenger City:\n\n "; //address change to city
            cin >> str_City;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            contact:
            cout << "\n\nNew Passenger : Contact No\n\n ";
            getline(cin, str_ContactNumber);
            if (str_ContactNumber.size() > 10 || str_ContactNumber.size() < 10)
            {
                cout << "\n\t\t!!!!Please Enter Valid Contact Number:!!!!" << endl;
                goto contact;
            }
            cout << "\n\nNew Passenger : UserName \n\n ";
            cin >> str_UserName;
            cout << "\n\nNew Passenger : Password\n\n ";
            cin >> str_Password;
            writeFile << customer_id << "\t" << str_name << "\t" << str_City << "\t"
                << str_ContactNumber << "\t" << str_UserName
                << "\t" << str_Password << "\n";
            count01++;
        }
        else
        {
            writeFile << customer_id << "\t" << name << "\t" << city << "\t"
                << cn << "\t" << username
                << "\t" << password << "\n";
        }
    }
    writeFile.close();
}

```

```

    } // end of while
    PassengerIn.close();
    if (count01 == 0)
    {
        cout << "\n\nRecord  could not be found!\n\n";
        remove("temp1.txt");
    }
    if (count01 > 0)
    {
        remove("Passenger.txt");
        rename("temp1.txt", "Passenger.txt");
        cout<<bold_on<<"MODIFIED SUCCESSFULLY!!\n"<<bold_off;
    }
}
// Login
int Passenger ::Login()
{
    long long x=0; //to detect correct username and password for login
    string username01, password01;
    cout << "=====
=====
    << "\n\n\t\t\t"<<bold_on << "LOGIN SCREEN"<< bold_off<<"\n\n"
    << "Enter your UserName:\n\n";
    cin >> username01;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "\n\nEnter your Password:\n\n";
    cin >> password01;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    ifstream matchup;
    matchup.open("Passenger.txt", ios::in);
    int s1, retype;
    string s2, s3, s4, s5, s6;
    while (matchup >> s1 >> s2 >> s3 >> s4 >> s5 >> s6)
    {
        if (username01==s5 && password01==s6)
        {
            x++;
            retype = s1;
        }
    }
    matchup.close(); //end of while
    if(x>0){
        cout << "\n\nLogged in Successfully!!\n\n";
        return retype;
    }
    else{
        cout<< "\n\nLog in Failed!\n\n";
        return 0;
    }
}

```

reservations.h

```
#include <iostream>
#include <map>
#include <math.h>
#include <vector>
#include <stdio.h>
#include <set>
#include <queue>
#include <numeric>
#include <fstream>
#include <bits/stdc++.h>
using namespace std;

#include "seats.h"
#include "passenger.h"
#define UNDERLINE "\033[33;4m"
#define CLOSEUNDERLINE "\033[0m"

// class Reservations
class Reservations : public Passenger, public Seats
{
protected:
    string str_Arrival_St, str_Dep_St;
    int i_No_Tickets, i_Charge;
    double i_total;

public:
    Reservations();
    // to make a reservation
    Reservations(int customer_id, string DeptSt, string ArrivalSt, int NoTickets);
    void SearchFile_and_Update(int);
    // ModifyReservation Function
    void Deletes(int);
    // cancel reservation function
    void Show(int);
    void Show(int, string, string, int, int, int);
    double CalculateFee(double, int);
    ~Reservations();
};
Reservations ::Reservations() {}

// class Payments
class Payments : public Reservations
{
public:
    void Show() const;
    ~Payments();
};
```

```
// payments - show
void Payments ::Show() const
{
    fstream payments;
    payments.open("Payments.txt", ios::in);
    string line;
    cout << "\n\n";
    if (!payments.is_open())
    {
        cout << "Unable to open payment records!!" << endl;
    }
    while (!payments.eof())
    {
        getline(payments, line);
        cout << line << endl;
    }
    payments.close();
}
Payments::~Payments() {}
// Reservations constructor
Reservations ::Reservations(int customer_id, string DeptSt, string ArrivalSt, int NoTi
ckets)
{
    int n;
    Reservations r2;
    str_Dep_St = DeptSt;
    str_Arrival_St = ArrivalSt;
    i_No_Tickets = NoTickets;
    ifstream PaymentsRead;
    PaymentsRead.open("Timetables.txt");
    double ch, ticket;
    string s1, s2, s3, s4;
    int ot = 0;
    while (PaymentsRead >> s1 >> s2 >> ch >> s3 >> s4)
    {
        if ((str_Dep_St == s1) & (str_Arrival_St == s2))
        {
            i_total = CalculateFee(ch, i_No_Tickets);
            ticket = ch;
            ot = 1;
        }
    }
    if (ot != 1)
    {
        cout << "\t\tSorry!! No bus found on your searched route\n"
            << endl;
    }
    else if (ot >= 0)
    {
        char confirmation;
```

```

        cout << "\n\nConfirm the Reservation(Y/N): \n\n";
        cin >> confirmation;
        confirmation = tolower(confirmation);
        if (confirmation == 'y')
        {
            fstream Reservations;
            string line;

            n = r2.CheckSeatAvailabilty(DeptSt, ArrivalSt, NoTickets);
            if (n == 1)
            {
                Reservations.open("Reservations.txt", ios::app | ios::out | ios::ate | ios::in)
;
                Reservations << customer_id << "\t" << str_Dep_St << "\t" << str_Arrival_
St << "\t" << i_No_Tickets
                << "\t" << i_total << "\n";
                Reservations.close();
                cout << bold_on << "\n\t\t\t***RESERVATION IS SUCCESSFUL!!***\n"
                << bold_off << endl;
                r2.Show(customer_id, str_Dep_St, str_Arrival_St, i_No_Tickets, ticket, i_to
tal);
            }
        }
        else
            cout << "\n\n\t\t\t***Confirmation Denied***\n\n";
    }
}
// modify Reservation function
void Reservations::SearchFile_and_Update(int log_customerid)
{
    int new_ticket_no;
    count01 = 0;
    int flag00=0;
    Reservations r1;
    double ch;
    char confirmation;
    ifstream reservationsIn;
    fstream reservationsOut;
    cout << "\n\nConfirm that you want to modify reservation(Y/N): \n\n";
    cin >> confirmation;
    confirmation = tolower(confirmation);
    if (confirmation == 'y')
    {
        Payments P1;
        P1.Show();
        // cout << "\n\nEnter your Customer ID:\n\n";
        // cin >> str_customer_id;
        cout << "\n\n-----
\n\n";
        cout << "\n\nDeparture station\n\n ";
    }
}

```

```

cin >> str_Dep_St;
cin.ignore(numeric_limits<streamsize>::max(), '\n');
cout << "\n\n Arrival station\n\n ";
cin >> str_Arrival_St;
cin.ignore(numeric_limits<streamsize>::max(), '\n');
cout << "\n\n-----\n\n";
cout << "\n\nEnter New No. of Tickets\n\n ";
cin >> i_No_Tickets; //new no of tickets
fstream PaymentsRead;

PaymentsRead.open("Timetables.txt", ios ::in);
string s1, s2, s3, s4, s5, s6, s7, s8;
while (PaymentsRead >> s1 >> s2 >> ch >> s7 >> s8)
{
    if ((str_Dep_St == s1) & (str_Arrival_St == s2))
    {
        i_total = CalculateFee(ch, i_No_Tickets);
    }
}
reservationsIn.open("Reservations.txt");
int s10;
while (reservationsIn >> s10 >> s2 >> s3 >> s4 >> s5)
{
    reservationsOut.open("temp7.txt", ios ::app);
    if ((s10 == log_customerid) & (s2 == str_Dep_St) & (s3 == str_Arrival_St))
    {
        new_ticket_no = i_No_Tickets - stoi(s4);
        int n = CheckSeatAvailabilty(str_Dep_St, str_Arrival_St, new_ticket_no);
        if (n == 1)
        {
            reservationsOut << log_customerid << "\t" << str_Dep_St << "\t" << str_
Arrival_St << "\t" << i_No_Tickets
                << "\t" << i_total << "\n";
            cout << "\n\t\tRecord found & updated!\n\n";
            cout << "\n\n-----\n\n";
            count01++;
        }
    }
    else
    {
        reservationsOut << s10 << "\t" << s2 << "\t" << s3 << "\t" << s4 << "\t" <<
s5 << "\n";
    }
    reservationsOut.close();
} // end of while
reservationsIn.close();
if (count01 == 0)
{
    cout << "\n\n\t\tSorry, Record could not be found!\n\n";
    remove("temp7.txt");
}

```

```

    }
    else if (count01 != 0)
    {
        cout<<"***Press 1 for make the changes or else press 1***";
        cin>>flag00;
        if(flag00==1)
        {
            remove("Reservations.txt");
            rename("temp7.txt", "Reservations.txt");
            r1.Show(log_customerid, str_Dep_St, str_Arrival_St, i_No_Tickets, ch, i_to
tal);
        }
        else
        {
            remove("temp7.txt");
            cout<<"***No modification is done as per your choice***\n";
        }
    }
    reservationsOut.close();
}
else
{
    cout << "\n\t\t***Confirmation Denied!!***\n"
    << endl;
}
}
//cancel Reservations0
void Reservations ::Deletes(int log_customerid)
{
    int cntr5 = 0;
    int flag =0;
    char confirmation;
    int customer_id;
    string dept, arr;
    int nt, tot;
    ifstream ResIn;
    ofstream ResOut;
    cout << "\n\nConfirm to "<<bold_on<< "Cancel"<<bold_off<<" the Reservation of
your Customer ID (y/n): \n\n";
    cin >> confirmation;
    confirmation = tolower(confirmation);
    if (confirmation == 'y')
    {
        cout << "\n\n-----
\n\n";
        // cout << "\n\nEnter your Customer ID :\n\n";
        // cin >> str_customer_id;
        cout << "\n\nDepature station\n\n ";
        cin >> str_Dep_St;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
}

```



```

    cout << "\n\n Arrival station\n\n ";
    cin >> str_Arrival_St;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "\n\n-----\n\n";
    ResIn.open("Reservations.txt");
    while (ResIn >> customer_id >> dept >> arr >> nt >> tot)
    {
        ResOut.open("temp5.txt", ios ::app);
        if ((customer_id == log_customerid) & (dept == str_Dep_St) & (arr == str_Arrival_St))
        {
            int n = CheckSeatAvailabilty(str_Dep_St, str_Arrival_St, -nt);
        }
        else
        {
            ResOut << customer_id << "\t" << dept << "\t" << arr << "\t" << nt << "\t"
            << tot << "\n";
            cntr5++;
        }
        ResOut.close();
    }
    ResIn.close();
    if (cntr5 == 0)
    {
        cout << "\n\nRecord  could not be found!\n\n";
        remove("temp5.txt");
    }
    else if (cntr5 > 0)
    {
        cout<<"***Press 1 for final confirmation or else press 0!***\n";
        cin>>flag;
        if (flag==1)
        {
            remove("Reservations.txt");
            rename("temp5.txt", "Reservations.txt");
            cout << "\n\n\t\t***Your Reservation Cancelled Successfully!***\n\n\n";
        }
        else
        {
            remove("temp5.txt");
            cout<<"***No cancellation of ticket is done as per your choice***\n";
        }
    }
    }
    else
    {
        cout << "\n\n\t\t***Confirmation Denied!***" << endl;
    }
}
// Reservations Show

```

```
void Reservations ::Show(int customer_id, string Dept_St, string Arrival_St, int No_
Tickets, int Charge, int total)
{
    string s6, s7, s8, s9, s10;
    string deptime, arrtime;
    fstream time;
    time.open("Timetables.txt");
    while (time >> s6 >> s7 >> s8 >> s9 >> s10)
    {
        if ((s6 == Dept_St) & (s7 == Arrival_St))
        {
            deptime = s9;
            arrtime = s10;
        }
    }
    cout << "\n\n\t*****\n\n";
    << bold_on << "\t\t\t\t\t " << UNDERLINE << "INDIAN ROADWAYS" <<
CLOSEUNDERLINE << bold_off << "\n\n"
<< "\t\tCustomer ID :" << customer_id << "\n\n"
<< "\t\tDep. St. : "
<< left<< setw(20)<< Dept_St<< "\t" << "Departure Time: " << deptime << "\n\
n"
<< "\t\tArrival St. : "
<< left<< setw(17)<< Arrival_St << "\t" << "Arrival Time: " << arrtime << "\n\
n"
<< "\t\tNo. of Tickets :" << No_Tickets << "\n\n"
//<< "\t\tCharge for one ticket : " << Charge << "\n\n"
<< "\t\t\t-----\n"
<< bold_on<< "\n\t\t\tTotal Fare = " << total <<bold_off<< "\n"
<< "\n\t*****\n\n";
}
void Reservations ::Show(int log_customerid)
{
    int cntr20 = 0;
    int s1;
    string s2, s3, s4, s5;
    string s6, s7, s8, s9, s10;
    string deptime, arrtime;
    // cout << "\n\nEnter your Customer ID:\n\n";
    // cin >> str_customer_id;
    cout << "\n\n-----
\n\n";
    cout << "\n\n Depature station\n\n ";
    cin >> str_Dep_St;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "\n\n Arrival station\n\n ";
    cin >> str_Arrival_St;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

```

fstream read;
read.open("Reservations.txt");
fstream time;
time.open("Timetables.txt");
while (time >> s6 >> s7 >> s8 >> s9 >> s10)
{
    if ((s6 == str_Dep_St) & (s7 == str_Arrival_St))
    {
        deptime = s9;
        arrtime = s10;
    }
}
while (read >> s1 >> s2 >> s3 >> s4 >> s5)
{
    if ((s1 == log_customerid) & (s2 == str_Dep_St) & (s3 == str_Arrival_St))
    {
        cout << "\n\n\t*****\n\n"
        << bold_on << "\t\t\t " << UNDERLINE << "INDIAN ROADWAYS"
<< CLOSEUNDERLINE << bold_off << "\n\n"
        << "\t\tCustomer ID :" << s1 << "\n\n"
        << "\t\tDep. St. : "
        << left<< setw(20)<< s2 << "\t" << "Departure Time: " << deptime << "\n\n"
        << "\t\tArrival St. : "
        << left<< setw(17)<<s3 << "\t" << "Arrival Time: " << arrtime << "\n\n"
        << "\t\tNo. of Tickets :" << s4 << "\n\n"
        << "\t\t\t-----\n"
        << bold_on<<"\n\t\t\tTotal Fare = " << s5 <<bold_off<< "\n"
        << "\n\t*****\n\n";
        cntr20++;
    }
}
read.close();
if (cntr20 == 0)
{
    cout << "\n\n\t***Record Not Found!!***" << endl;
}
// Reservations Destructor
Reservations::~Reservations() {}
// Payments Calculate
double Reservations::CalculateFee(double fee, int nt)
{
    i_total = nt * fee;
    return i_total;
}

```

seats.h

```
#include <iostream>
#include <map>
#include <math.h>
#include <vector>
#include <set>
#include <queue>
#include <numeric>
#include <fstream>
using namespace std;

// class Seats
class Seats
{
protected:
    int seats;

public:
    int CheckSeatAvailability(string DepSt, string ArrSt, int nt);
};

// Seats - CheckSeatAvailability
inline int Seats ::CheckSeatAvailability(string DepSt, string ArrSt, int nt)
{
    int seats, cnt=0;
    ifstream SeatAvailability;
    ofstream seat_update;
    string s1, s2;
    SeatAvailability.open("seat_available.txt");
    while ( SeatAvailability >> s1 >> s2 >> seats)
    {
        seat_update.open("temp11.txt", ios::app);
        if ((s1 == DepSt) & (s2 == ArrSt))
        {
            seats = seats - nt;
            if(seats < 0)
            {
                cout << "\n\n*****SORRY! THIS RESERVATION IS NOT POSSIBLE*****\n\n\n" << endl;
                return(0);
            }
            seat_update << DepSt << "\t" << ArrSt << "\t" << seats << "\n";
            cnt++;
        }
        else
        {
            seat_update << s1 << "\t" << s2 << "\t" << seats << "\n";
        }
        seat_update.close();
    }
}
```

```
SeatAvailability.close();
if (cntr == 0)
{
    remove("temp11.txt");
}
if (cntr > 0)
{
    remove("seat_available.txt");
    rename("temp11.txt", "seat_available.txt");
    return(1);
}
return 0;
}
```