VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANANA SANGAMA", Belgaum-60

Mini Project Report
On

Bus Reservation System

SUBMITTED IN PARTIAL FULFILMENT FOR 6TH SEMESTER

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

SUBMITTED BY

Pavan Y N (1JB20IS041)

Under the Guidance of

Vishruth B G
Assisstant Professor
Dept. of ISE, SJBIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY

BGS HEALTH AND EDUCATION CITY,
KENGERI, BENGALURU-560060, KARNATAKA, INDIA.

2022 - 2023

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust ®

SJB INSTITUTE OF TECHNOLOGY

BGS Health & Education City, Kengeri, Bengaluru – 560 060

Department of Information Science & Engineering

CERTIFICATE

Certified that the Mini project work entitled **Bus Reservation System**" carried out by Pavan Y N **bearing USN 1JB20IS041**.Bonafede students of **SJB Institute of Technology** in partial fulfilment for 6th semester in **INFORMATION SCIENCE AND ENGINEERING** of the **Visvesvaraya Technological University**, **Belagavi** during the academic year **2022-23.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini project report has been approved as it satisfies the academic requirements in respect of Mini Project prescribed by the institution.

Vishruth B G
Asst. Professor
Dept. of ISE, SJBIT.

Dr. Shashidhar H R
Professor & Head
Dept. of ISE, SJBIT

1 Internal Examiner:

2 External Examiner:

ACKNOWLEDGMENT

The satisfaction & euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible because "Success is the abstract of hard work & perseverance, but steadfast of all is encouragement guidance". So I would like to acknowledge all those whose guidance and encouragement served as a beacon light & crowned our efforts with success.

I am grateful to his divine soul **Bhairavaikya Padmabhushana Sri Sri Sri Dr.Balagangadharanatha MahaSwamiji** and to His Holiness **Sri Sri Sri Jagadguru Dr.Nirmalanandanatha MahaSwamiji,** for providing us an opportunity to complete our academics in this esteemed college.

I would like to express my profound gratitude to his holiness **Reverend Sri Sri Dr.Prakashnath Swamiji, Managing Director**, SJBIT for providing an opportunity to complete my academics and present this project.

I am grateful to **Dr. K V Mahendra Prashanth**, **Principal** for his kind co-operation and encouragement and extremely grateful to **Dr. Shashidhar Professor and Head of the Department** of Information Science and Engineering, for his co-operation and encouragement.

I wish to express heartfelt gratitude to our guide**, Vishruth B G, Professor, Dept. of ISE, SJBIT**, for his valuable guidance, suggestions and cheerful encouragement during the entire period of this work. I express our truthful thanks to **Vishruth B G, Mini Project Coordinator**, Dept. of ISE for her valuable support.

Finally, I take this opportunity to extend our earnest gratitude and respect to our parents, Teaching & Non-teaching staffs of the department, the library staff and all our friends, who have directly or indirectly supported us during the period of this mini project work.

Regards,

Pavan Y N (1JB20IS041)

TABLE OF CONTENTS

ABSTRACT

Travel industry is evolving day to day. As the industry evolves the need to digitalize all the transactions becomes need of the hour. This project which is implemented on C++ platform helps to manage bus scheduling and bookings. The existing system is not completely computerized. The customer has to visit any booking branch if he wants to book a ticket. Bus scheduling, ticket booking and many other operations are done manually. This may lead to incorrect entries and there is a lot of room for errors as the data is not completely synced.

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 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.

1.1.1 History

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 binary tree 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 binary tree, but a file block containing dozens, perhaps even hundreds, of records. Hashing is a good 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 at most 2 disk accesses, no matter how big the file became.

1.1.2 About the File

When we talk about a file on disk or tape, we refer to a particular collection of bytes stored there. A 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 from the physical files on the disk or tape.

1.1.3 Various Kinds of storage of Fields and Records

A field is the smallest, logically meaningful, unit of information in a file.

Field Structures:

• Force the fields into a predictable length.

• Begin each field with a length indicator.

• Place a delimiter at the end of each field to separate it from the next field.

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

Method 1: Fix the Length of Fields

In the above example, each field is a character array that can hold a string value of some maximum size. The size of the array is 1 larger than the longest string it can hold. Simple arithmetic is sufficient to recover data from the original fields. The disadvantage of this approach is adding all the padding required to bring the fields up to a fixed length, makes the file much larger. We encounter problems when data is too long to fit into the allocated amount of space. We can solve this by fixing all the fields at lengths that are large enough to cover all cases, but this makes the problem of wasted space in files even worse. Hence, this approach isn't used with data with large amount of variability in length of fields, but where every field is fixed in length if there is very little variation in field lengths.

Method 2: Begin Each Field with a Length Indicator

We can count to the end of a field by storing the field length just ahead of the field. If the fields are not too long (less than 256 bytes), it is possible to store the length in a single byte at the start of each field. We refer to these fields as length-based.

Method 3: Separate the Fields with Delimiters

We can preserve the identity of fields by separating them with delimiters. All we need to do is choose some special character or sequence of characters that will not appear within a field and then insert that delimiter into the file after writing each field. White-space characters (blank, new line, tab) or the vertical bar character, can be used as delimiters.

Method 4: Use a "Keyword=Value" Expression to Identify Fields

This has an advantage the others don't. It is the first structure in which a field provides information about itself. Such self-describing structures can be very useful tools for organizing files in many applications. It is easy to tell which fields are contained in a file. Even if we don't know ahead of time which fields the file is supposed to contain. It is also a good format for dealing with missing fields. If a field is missing, this format makes it obvious, because the keyword is simply not there. It is helpful to use this in combination with delimiters, to show division between each value and the keyword for the following field. But this also wastes a lot of space: 50% or more of the file's

space could be taken up by the keywords. A record can be defined as a set of fields that belong together when the file is viewed in terms of a higher level of organization.

Record Structures:

• Require the records to be predictable number of bytes in length.

• Require the records to be predictable number of fields in length.

• Begin each record with a length indicator consisting of a count of the number of bytes that the record contains.

• Use a second file to keep track of the beginning byte address for each record.

• Place a delimiter at the end of each record to separate it from the next record.

Method 1: Make the Records a Predictable Number of Bytes (Fixed-Length Record)

A fixed-length record file is one in which each record contains the same number of bytes. In the field and record structure shown, we have a fixed number of fields, each with a predetermined length, that combine to make a fixed-length record.

Fixing the number of bytes in a record does not imply that the size or number of fields in the record must be fixed. Fixed-length records are often used as containers to hold variable numbers of variable-length fields. It is also possible to mix fixed and variable-length fields within a record.

Method 2: Make Records a Predictable Number of Fields

Rather than specify that each record in a file contains some fixed number of bytes, we can specify that it will contain a fixed number of fields. In the figure below, we have 6 contiguous fields and we can recognize fields simply by counting the fields modulo 6.

Method 3: Begin Each Record with a Length Indicator

We can communicate the length of records by beginning each record with a filed containing an integer that indicates how many bytes there are in the rest of the record. This is commonly used to handle variable-length records.

Method 4: Use an Index to Keep Track of Addresses

We can use an index to keep a byte offset for each record in the original file. The byte offset allows us to find the beginning of each successive record and compute the length of each record. We look up the position of a record in the index, then seek to the record in the data file.

Method 5: Place a Delimiter at the End of Each Record

It is analogous to keeping the fields distinct. As with fields, the delimiter character must not get in the way of processing. A common choice of a record delimiter for files that contain readable text is the end-of-line character (carriage return/ new-line pair or, on Unix systems, just a new-line character: \n). Here, we use a | character as the record delimiter

1.1.4 Application of File Structure

Relative to other parts of a 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.

Chapter 2

Requirements Specification

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 Telecom Billing System are stated as follows:

2.1 Software Requirements

- **Software's used:**
  - Operating System – Windows OS
  - Front End – Sublime Text (or any other Text Editor)
  - Back End – Turbo C++ compiler
- **Technologies used:**
  - Front End – C++ programming
  - Controller – C++
  - Back End – C++ compiler

2.2 Hardware Requirements

- **Hardware Components used:**
  - CPU – Intel Core i3 and Above
  - RAM – 4GB and Above
  - Peripherals – Standard PS/2 or USB Keyboard, Standard PS/2 or USB Wheel/Optical Mouse

- ## Technology Used:

C++ is a multi-paradigm programming language that supports object-oriented programming (OOP), created by Bjarne Stroustrup in 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 writes language statements in a language such as Pascal or C one line at a time using an editor.

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 easily run 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. In C++ programming 'break and BREAK' both are different. If any language treats lower case latter separately and upper-case latter separately than they can be called as case sensitive programming language [Example 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.

Turbo C++ is a discontinued C++ compiler and integrated development environment and computer language originally from Borland. Most recently it was distributed by Embarcadero Technologies, which acquired all of Borland's compiler tools with the purchase of its Code Gear division in 2008. The original Turbo C++ product line was put on hold after 1994 and was revived in 2006 as an introductory-level IDE, essentially a stripped-down version of their flagship C++Builder. Turbo C++ 2006 was released on September 5, 2006 and was available in 'Explorer' and 'Professional' editions. The Explorer edition was free to download and distribute while the Professional edition was a commercial product. In October 2009 Embarcadero Technologies discontinued support of its 2006 C++ editions.

CHAPTER 3

SYSTEM DESIGN

The purpose of the design phase is to develop a clear understanding of what the developer wants people to gain from his/her project. As the developer works on the project, the test for every design decision should be

"Does this feature fulfil the ultimate purpose of the project?"

A purpose statement affects the design process by explaining what the developer wants the project to do, rather than describing the project itself. The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.

3.1 Operations Performed on a File

- **Insertion:**

The system is initially used to add customer records containing the Booking ID, Bus ID, source, destination into the file. Records with duplicate Booking id fields are not allowed to be inserted. The length of the Booking ID is checked to see whether it contains only 10 characters while the first 5 characters being letter and last 5 characters being digits.

- **Display**

The system can then be used to display existing records. The records are displayed based on the way we inserted. It contains Booking ID, Name, Bus No, Source, Destination, Date, Cost, Phone Number.

- **Search:**

The system can then be used to search for existing records. The user is prompted for a Booking ID. If Booking ID is matched then entire details of the existing record will be displayed. If Booking ID does not exist it displays the message saying record not found.

- **Delete:**

The system can then be used to delete existing records. The user is prompted for a Booking ID, which is needed to be deleted. The requested record, if found is cleared, a “record deleted” message

is displayed, and the record of the respective Booking ID will also be deleted in the file (The key of the record will be replaced by *). If absent record not found message will be displayed.

▪ **Modify:**

We need to prompt for Booking ID which is to be modified and we give the provision for the user to modify the entire entered details.

3.2 Data Flow Diagrams:

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. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

Data flow diagrams were popularized in the late 1970s, arising from the book Structured Design, by computing pioneers Ed Yourdon and Larry Constantine. They based it on the "data flow graph" computation models by David Martin and Gerald Estrin. The structured design concept took off in the software engineering field, and the DFD method took off with it. It became more popular in business circles, as it was applied to business analysis, than in academic circles.

Also contributing were two related concepts:

- Object Oriented Analysis and Design (OOAD), put forth by Yourdon and Peter Coad to analyse and design an application or system.
- Structured Systems Analysis and Design Method (SSADM), a waterfall method to analyse and design information systems. This rigorous documentation approach contrasts with modern agile approaches such as Scrum and Dynamic Systems Development Method (DSDM.)

Three other experts contributing to this rise in DFD methodology were Tom DeMarco, Chris Gane and Trish Sarson. They teamed up in different combinations to be the main definers of the symbols and notations used for a data flow diagram. A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

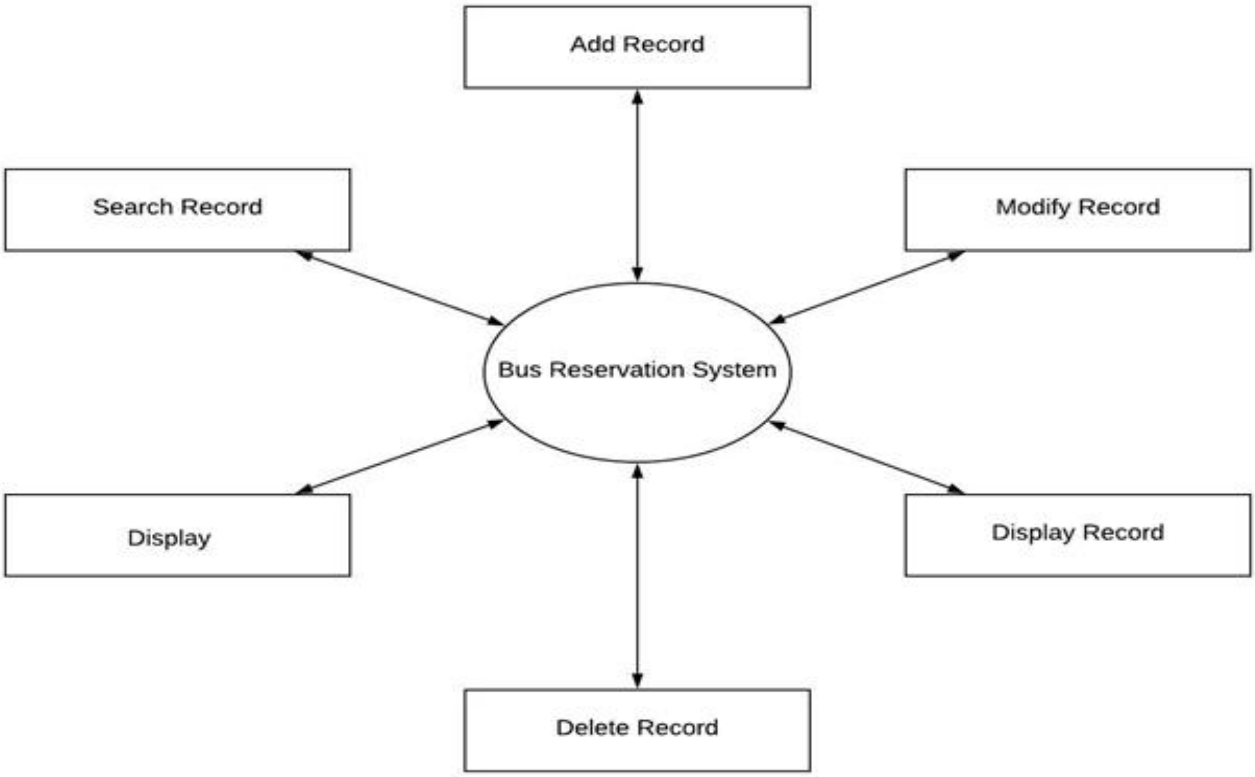

Figure.3.1: ZERO LEVEL DFD

The above figure shows zero level data flow diagram of telecom billing system. Zero level 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 a wide audience, including stakeholders, business analysts, data analysts and developers.

Chapter 4

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.

4.1 Console Window

The console window consists of 6 options which will direct to particular tasks.

The options are

- Display all records
- Add record into the file
- Search for records
- Delete record
- Update record
- Quit

Adding of Record:

- Prompts the user to enter the Booking ID.
- Prompts the user to enter the name.
- Prompts the user to enter the busid.
- Prompts the user to enter the date.
- Prompts the user to enter the source.
- Prompts the user to enter the destination.
- Prompts the user to enter the cost.
- Prompts the user to enter the phone number

Search for Record:

- Prompts the user to enter the Booking ID.
- Displa**ys** the details of the corresponding Booking ID if present.
- If the Booking ID is not present then it prompts the message "Booking ID not found".

Delete the Record:

- Prompts the user to enter the Booking ID.
- Deletes the record if Booking ID is present.
- If Booking ID is not present it displays message saying record not found.

Update the Record:

- Prompts the user to enter the Booking ID.
- We can enter the details and update the record.
- After entering all the information, the newly updated record will be stored in the file by replacing the old one.
- If Booking ID is not present it displays message saying record not found.
- If the user enters N[no] then he will be directed to Main menu from update record window, without updating any of the records in the file.

Display all Records:

- Prompts the user to enter option 1 when he/she is in the console window.
- This will display all the records which are saved in the file.
- Firstly, it shows primary indexing table which consists of key and offset value of all the records stored in the file.
- It displays all the information of each record like Booking ID, name, busid, source, destination, date, cost and phone number.
- Information of only one record will be displayed on to the screen at once and on pressing of enter key next record will be displayed and so on until all records which are saved are displayed, and if all records are displayed the user will be directed to console window.

Quit:

- Prompts the user to enter option 6 when he/she is in the console window.
- When user enter 6 then the program will be terminated.

Chapter 5

TESTING

System testing is the stage of implementation, which is aimed at ensuring that the system 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

The software units in a system are modules and routines that are assembled and 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. The various controls are tested to ensure that each performs its action as required.

5.2 Integration Testing

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

5.3 User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required.

Case Id	Description	Expected Output	Actual Output	Status
1	Testing whether the module *append()* working with valid inputs.	Should add record to the file.	Record is added to the file	Pass
2	Testing whether the module *display()* working with valid inputs.	Should list all records in the file.	Records are listed systematically.	Pass
3	Testing whether the module *update()* working with valid inputs.	Should modify record as per the user and update the file	Record is modified successfully in the file.	Pass
4	Testing whether the module *delete()* working with valid inputs.	Need delete the record prompted by the user.	Record is successfully deleted.	Pass
5	Testing whether the module *search()* working with valid inputs.	Need to search record based on primary key that is Booking ID.	Records are fetched successfully based on primary key.	Pass

6	Testing whether the module *pack()* working with valid inputs.	Used to pack the details of the user and store it in buffer	Packed successfully, and stored in buffer.	Pass
7	Testing whether the module *unpack()* working with valid inputs.	Used to unpack the details of the user from the buffer to display on console.	Unpacking is done successfully on to console.	Pass
8	Testing whether the module *write()* working with valid inputs.	Used to write the buffer contents to file	Successfully written to file.	Pass
9	Testing whether the module *read()* working with valid inputs.	Used to read to the buffer from file	Successfully read from file.	Pass
10	Testing whether the module *checkName()* working with valid inputs.	Used to validate name.	Module working successfully.	Pass
11	Testing whether the module *checkID()* working with valid inputs.	Used to validate Booking ID.	Module working successfully.	Pass
12	Testing whether the module *insert()* working with valid inputs.	Should add record to the tree structure.	Tree structure was successfully updated	Pass

13	Testing whether the module *create()* working with valid inputs.	Should create an empty file for storing records	File was successfully created	Pass

Chapter 6

SNAPSHOTS

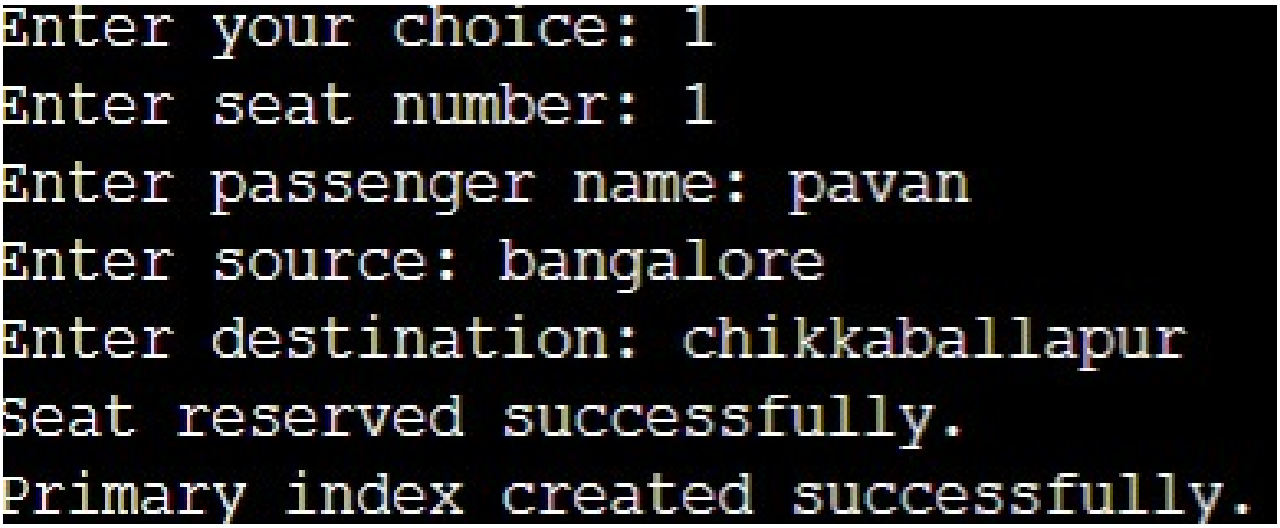

Figure 1:First screen

This is the first screen the user gets once he/she runs the program. Here user can addarecord,search booking details, displayrecords,display in treestructure, deleterecord, update record or
quit the program.

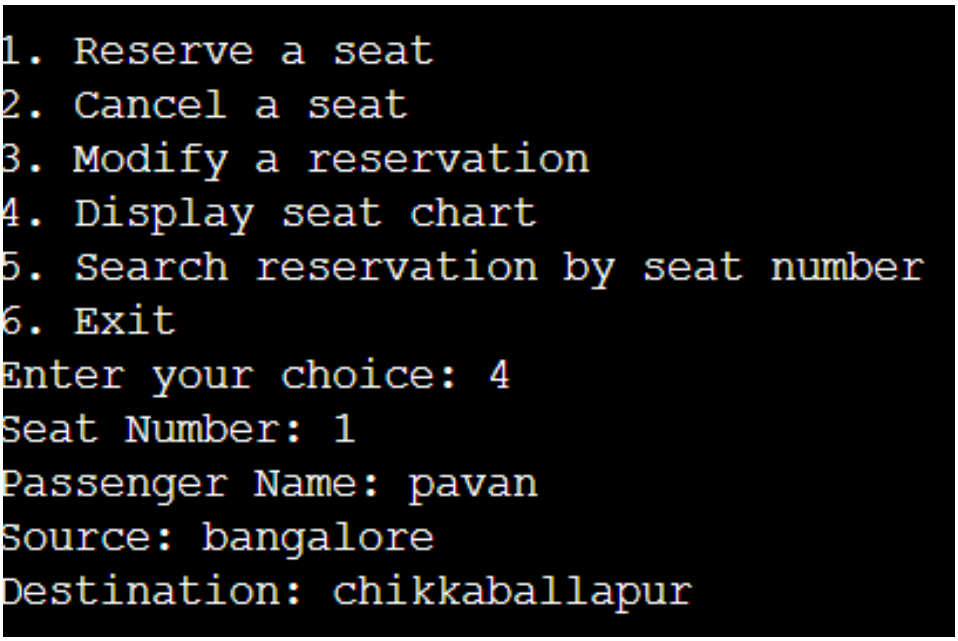

Figure 2: display record using secondary index

This window is used to display all the records which are saved in the file in the form of tree structure.

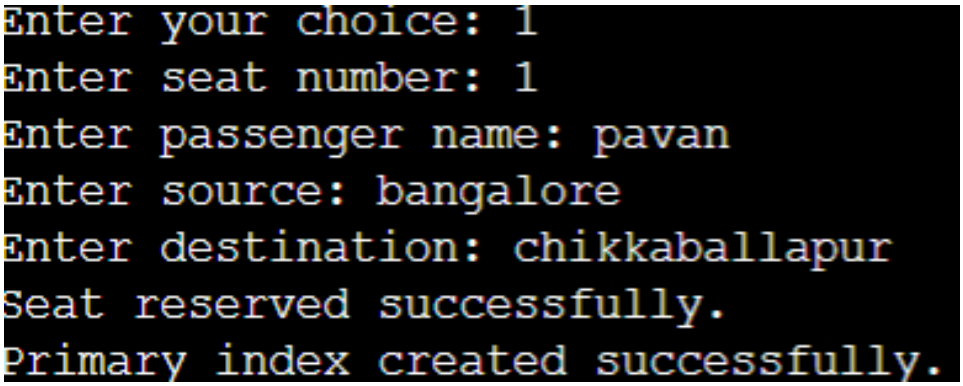

Figure 3: Record Addition

This window is used to add new record to the file, with its Booking ID, name, Busid, date, source, destination, cost and phone. After addition of each record user will be directed to main menu by clicking any key.

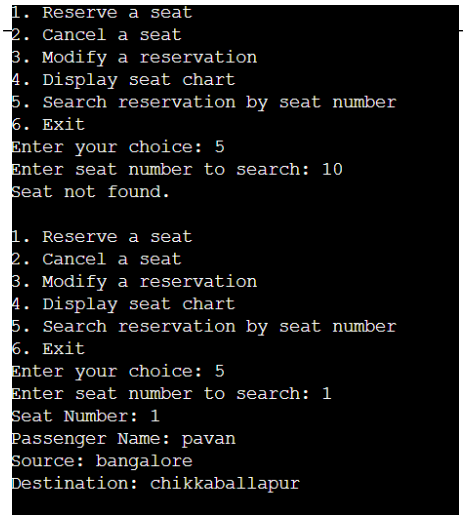

Figure 4: Search Record

This window is used to search the record by entering Booking ID. If the record with entered Booking ID is present in the file then it will be displayed on to the screen or else “no record found” message will be displayed.

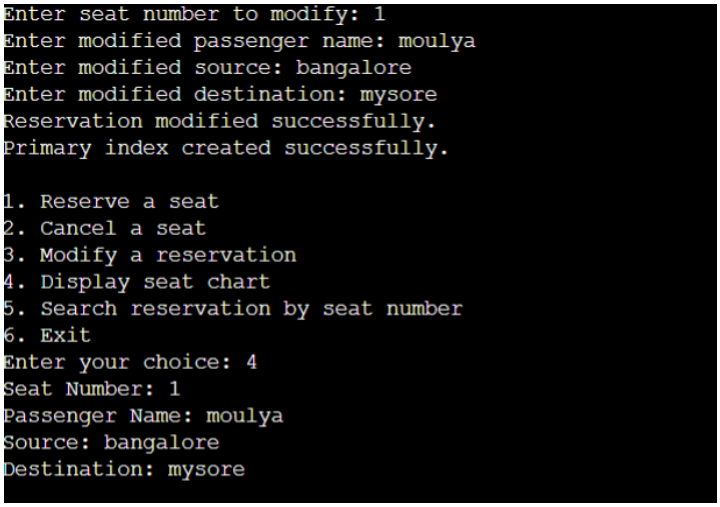

Figure 5: Update record

This window is used to update the record by entering Booking ID. Whenever we enter the Booking ID the relative record information will be displayed on to the screen if and only if that record was saved in the file or else “no record found” message will be displayed. If record is present after it’s display “confirm permanent updation: [y/n]” message will be printed, if we give y the it will execute adding record method and replaces the freshly updated record in the place of old one and saves into the file.

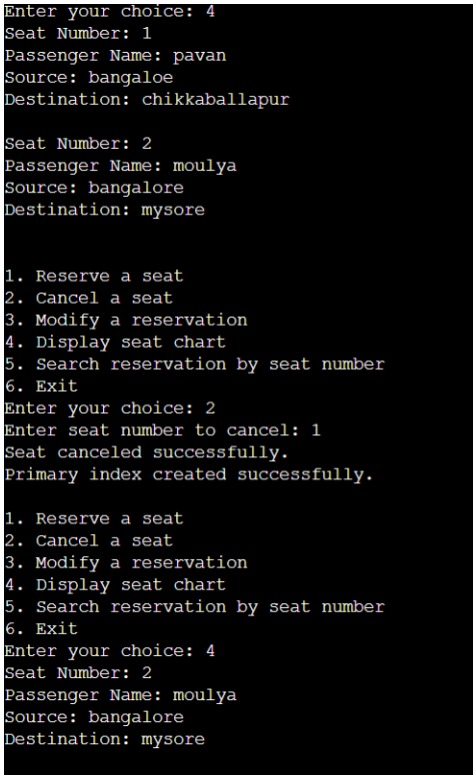

Figure 6: Delete record

This window is used to delete the record by entering Booking ID. when we enter the Booking ID ,if the record is present the it will be displayed on to the screen along with the message “confirm permanent deletion:[y/n]:”, if we give ‘y’ the record will be deleted permanently from the file. If record is not saved in the file then “no record found” message will be displayed on to the screen.

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The main aim of Bus **Reservation System** is to get details about the bus reserved and the details of the customer and even the updation and modification of one's booking details can be done.

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.

All this goal is achieved and now here we are with a Bus Reservation System Application.

Future Enhancements

- We can add number of seats for each bus and check there availabity.
- We can display the customers details with respect to their Bus ID
- We can add payment method for reservation.

REFERENCES

BOOKS

- File Structures: An Object-Oriented Approach with C++ 3rd Edition by Michael J. Folk (Author), Bill Zoellick (Author), Greg Riccardi (Author).
- The C++ Programming Language, 4th Edition 4th Edition by Bjarne Stroustrup (Author).
- The Waite Group's C Programming Using Turbo C+/Book and Disk Subsequent Edition by Robert Lafore (Author).

ONLINE SOLUTIONS

- For Data Flow Diagram and Flowchart, Lucid chart is a web-based commercial service to create flowcharts, organizational charts, website wireframes, and other things.
- https://www.lucidchart.com/documents/edit/6830d573-57de-4424-83b2- 660f3108bd9b- for Data Flow Diagram
- www.stackoverflow.com/files/
- Turbo C++ tutorial in YouTube: https://www.youtube.com/watch?vR15KpkibkR0 www.codebook.com/files/