VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI, KARNATAKA - 590018

FILE STRUCTURE LABORATARY MINI PROJECT REPORT ON

"BUS RESERVATION MANAGEMENT SYSTEM"

Submitted by

PRANEETH	**4DM20IS034**
PRANEETH JAIN	**4DM20IS035**
S HARSHAVARDHAN	**4DM20IS039**
THOUFEEQ M I	**4DM20IS054**

UNDER THE GUIDANCE OF

Mrs.Deeksha K R

Assistant Professor, Dept. Of ISE

In the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE AND ENGINEERING

YENEPOYA INSTITUTE OF TECHNOLOGY

N.H. 13, Thodar, Vidyanagar, Moodbidri, Mangalore,
Karnataka - 574225

2022-2023

YENEPOYA INSTITUTE OF TECHNOLOGY

N.H.13, Thodar, Vidyanagar, Moodbidri, Mangalore, Karnataka 574225

(Affiliated of Visvesvaraya Technological University)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

Certified that the file structure laboratory Mini Project Work titled **“BUS RESERVATION MANAGEMENT SYSTEM”** carried out by **Mr.Praneeth(USN 4DM20IS034), Mr.Praneeth Jain(USN 4DM20IS035), Mr.S Harshavardhan(USN 4DM20IS039)** and **Mr.Thoufeeq M I(USN 4DM20IS054)** are bonafide students of **Yenepoya Institute Of Technology** in partial fulfillment for the award of **Bachelor of Engineering in Information Science & Engineering** of the **Visvesvaraya Technological University**,Belagavi during the 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 project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

………………………..	……………………..
Prof. Deeksha K R	**Prof. Rashmi P C**
Internal guide	**HOD**
Dept. of ISE	**Dept. of ISE**

External Viva

Name of the examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGEMENT

The successful completion of any work would be incomplete without a mention of the people who made it possible, whose constant guidance and encouragement served as a beacon light and crowned our efforts with success. We owe our gratitude to many people who helped and supported us during our File Structure laboratory Mini Project "BUS RESERVATION MANAGEMENT SYSTEM".

Our deepest thanks to our guide Mrs.Deeksha K.R, Assistance Professor, Dept.of ISE, Yenepoya Institute Of Technology for her constant support and encouragement and providing with the necessary advice and help.We are highly indebted to her for taking keen interest in our work, monitoring and providing guidance throughout the course.

I sincerely express our gratitude to Prof.Rashmi P C, H.O.D, Dept. of ISE for her constant support and guidance for the successful completion of File Structure laboratory Mini Project.

I take immense pleasure in thanking my beloved Principal Dr. R. G D'Souza for his constant support.

I also thank all laboratory administrates and assistants who have helped us in making this project a successful one.

At last, but not the least I want to thank our classmates and friends who appreciated our work and motivated us.

PRANEETH	**4DM20IS034**
PRANEETH JAIN	**4DM20IS035**
S HARSHAVARDHAN	**4DM20IS039**
THOUFEEQ M I	**4DM20IS054**

ABSTRACT

The Bus Reservation Management System is a software solution that provides an efficient and user-friendly platform for managing bus ticket bookings. The system caters to both administrators and users, offering distinct functionalities for each user type.For administrators, the system allows them to perform various tasks such as adding new buses, deleting existing buses, editing bus details, and viewing a list of all available buses. The system ensures data integrity by maintaining separate index files for buses and tickets, which are updated as buses are added or deleted.Users can book tickets by specifying the bus number, passenger details, and desired seat number. The system verifies the availability of seats and updates the seat count accordingly. Users can also cancel their booked tickets, which updates the seat count and marks the ticket as canceled.The system employs file handling techniques to store and retrieve data efficiently. The bus details, including availability and seat counts, are stored in a file named "bus.txt," while the ticket details are stored in a file named "ticket.txt." Separate index files, "busindex.txt" and "ticketindex.txt," maintain the index data for quick access and updates.The bus reservation management system enhances the overall bus ticket booking process by automating administrative tasks and providing a seamless experience for users. The system's features contribute to better efficiency, accuracy, and convenience in managing bus reservations, making it a valuable tool for bus operators and passengers alike.

TABLE OF CONTENTS

CHAPTER NO.		TITLE	PAGE NO.
		ACKNOWLEDGEMENT	**i**
		ABSTRACT	**ii**
		LIST OF FIGURE	**iii**
1		**INTRODUCTION**	**7**
	1.1	Introduction to File Structures	7
	1.2	History	7
	1.3	About the File	8
	1.4	Various Kinds of Fields and Records	9
	1.5	Applications of File Structure	13
	1.6	Objectives	14
	1.7	Existing System	14
	1.8	Proposed System	15
2		**PROBLEM DEFINITION**	**16**
3		**REQUIREMENTS SPECIFICATION**	**17**
	3.1	Hardware Requirement	17
	3.2	Software Requirement	17
4		**SYSTEM ANALYSIS**	**18**
	4.1	Analysis of the Project	18
	4.2	Structure used to Store the Fields and Records	18
5		**FILE STRUCTURE IN PYTHON**	**20**
	5.1	Python	20

	5.2	Data files and it's fields	23
	5.3	Data files	25
6		**FRONT-END DESIGN**	**26**
	6.1	PyCharm IDE	26
	6.2	Python	26
	6.3	Pseudo Code	27
7		**RESULTS AND ANALYSIS**	**28**
8		**CONCLUSIONS**	**33**
9		**REFERENCE**	**34**

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	Four methods for field structure	11
1.2	Making Record predictable number of Bytes	12
1.3	Length indicator Files	13
5.3.1	Ticket File	25
5.3.2	Bus File	25
7.1	Home Page	28
7.2	Admin Page	28
7.3	Add Bus Page	29
7.4	Update Bus Page	29
7.5	View Available Bus Page	30
7.6	Delete Bus Page	30
7.7	Book Ticket Page	31
7.8	View Booked Ticket Page	31
7.9	Cancel Ticket Page	32
7.10	User Login Page	32

CHAPTER 1

INTRODUCTION

In this chapter, Introduction to File Structures and history, About the File, various kinds of storage of Fields and records have been explored. Application of File structures is discussed in section 1.6 and objectives, Existing System and Proposed Systems are discussed in section 1.7, 1.8 & 1.9 respectively.

1.1 Introduction to File Structures

A file structure is a combination of representations for data in files and of operations for accessing the data. A file structure allows 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.

1.2 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 key changes. In 1963, researches developed and elegant selfadjusting binary tree structure, called AVL tree, for data in memory, 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 the tree was not a single record, as in a binary tree, but a file block containing dozens, perhaps even hundreds, of records took 10 years until a solution emerged in the form of a B-Tree.

Whereas AVL trees grow from the top down as records were added, B-Trees grew from the bottom up. B-Trees provided excellent access performance, but there was a cost: no longer could a file be accessed sequentially with efficiency. The problem was solved by adding a linked list structure at the bottom level of the B-Tree. The combination of a B- Tree and a sequential linked list is called a B+ tree.

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.

1.3 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.

The application program relies on the OS to take care of the details of the telephone switching system. It could be that byte 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

1.4 Various Kinds of storage of Fields and Records

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

Field Structures

The four most common methods as shown in Fig. 1.1 of adding structure to files to maintain the identity of fields are:

- 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.

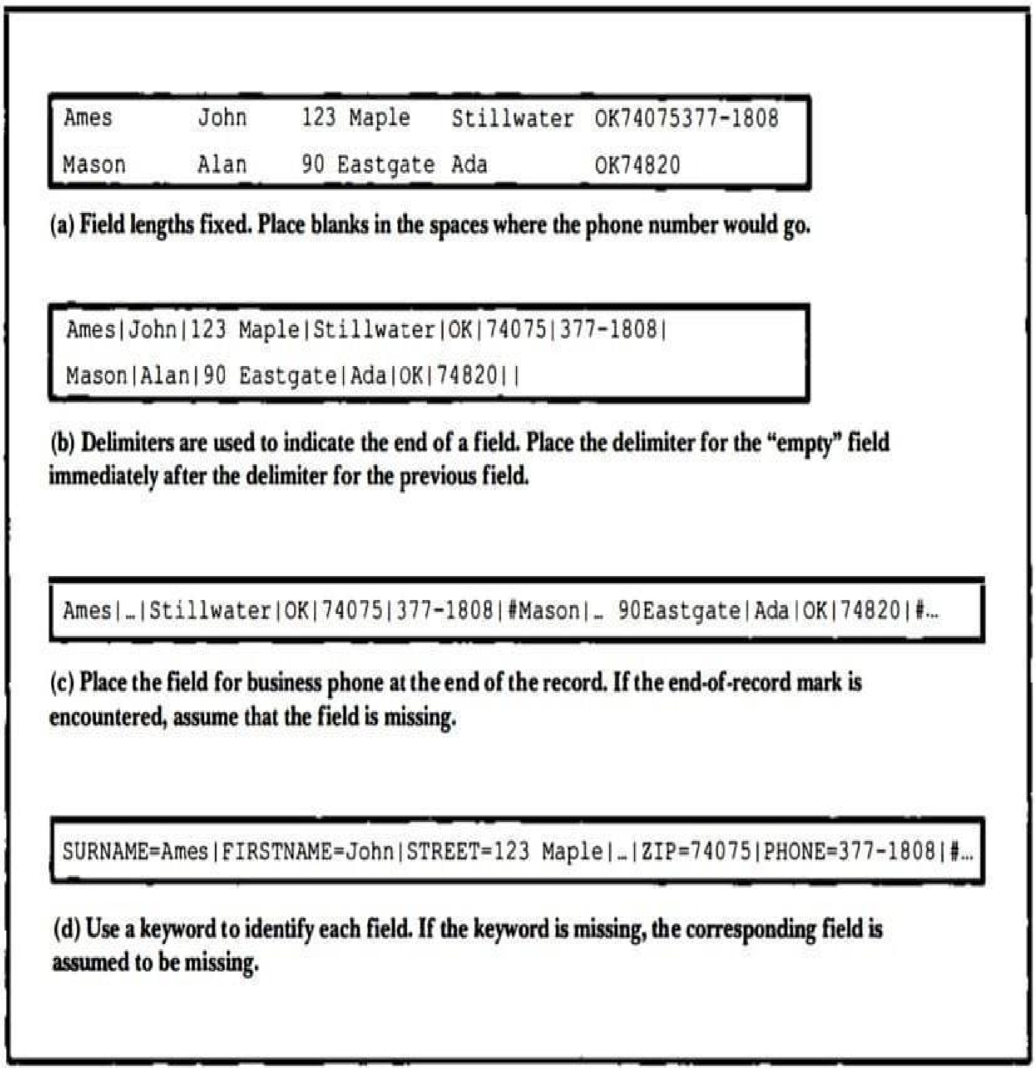
Method 3: Separate the Fields with Delimiters

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.



```
Ames      John     123 Maple   Stillwater  OK74075377-1808
Mason     Alan     90 Eastgate Ada         OK74820
```

(a) Field lengths fixed. Place blanks in the spaces where the phone number would go.

```
Ames|John|123 Maple|Stillwater|OK|74075|377-1808|
Mason|Alan|90 Eastgate|Ada|OK|74820||
```

(b) Delimiters are used to indicate the end of a field. Place the delimiter for the "empty" field immediately after the delimiter for the previous field.

```
Ames|…|Stillwater|OK|74075|377-1808|#Mason|… 90Eastgate|Ada|OK|74820|#…
```

(c) Place the field for business phone at the end of the record. If the end-of-record mark is encountered, assume that the field is missing.

```
SURNAME=Ames|FIRSTNAME=John|STREET=123 Maple|…|ZIP=74075|PHONE=377-1808|#…
```

(d) Use a keyword to identify each field. If the keyword is missing, the corresponding field is assumed to be missing.

Figure 1.1 Four methods for field structures

Record Structures

The five most often used methods for organizing records of a file are

- 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.

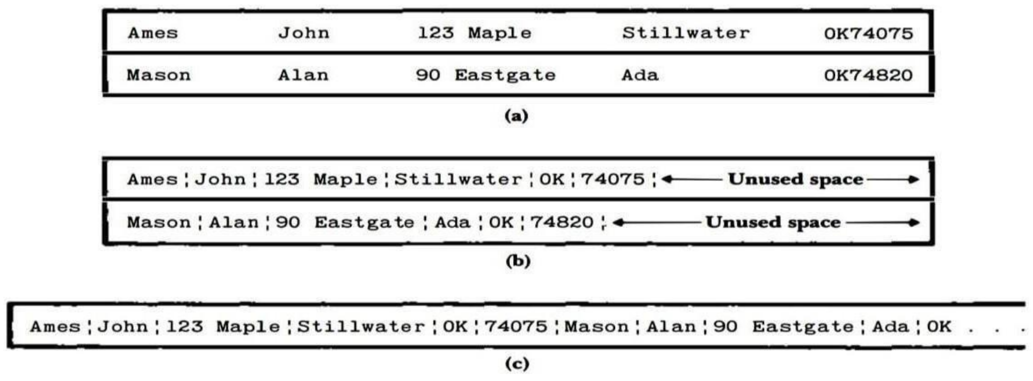

Ames John 123 Maple Stillwater OK74075
Mason Alan 90 Eastgate Ada OK74820
(a)
Ames ¦ John ¦ 123 Maple ¦ Stillwater ¦ OK ¦ 74075 ¦ ← Unused space →
Mason ¦ Alan ¦ 90 Eastgate ¦ Ada ¦ OK ¦ 74820 ¦ ← Unused space →
(b)
Ames ¦ John ¦ 123 Maple ¦ Stillwater ¦ OK ¦ 74075 ¦ Mason ¦ Alan ¦ 90 Eastgate ¦ Ada ¦ OK . . .
(c)

Figure 1.2 Making Records Predictable number of Bytes and Fields

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 ofeach 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".

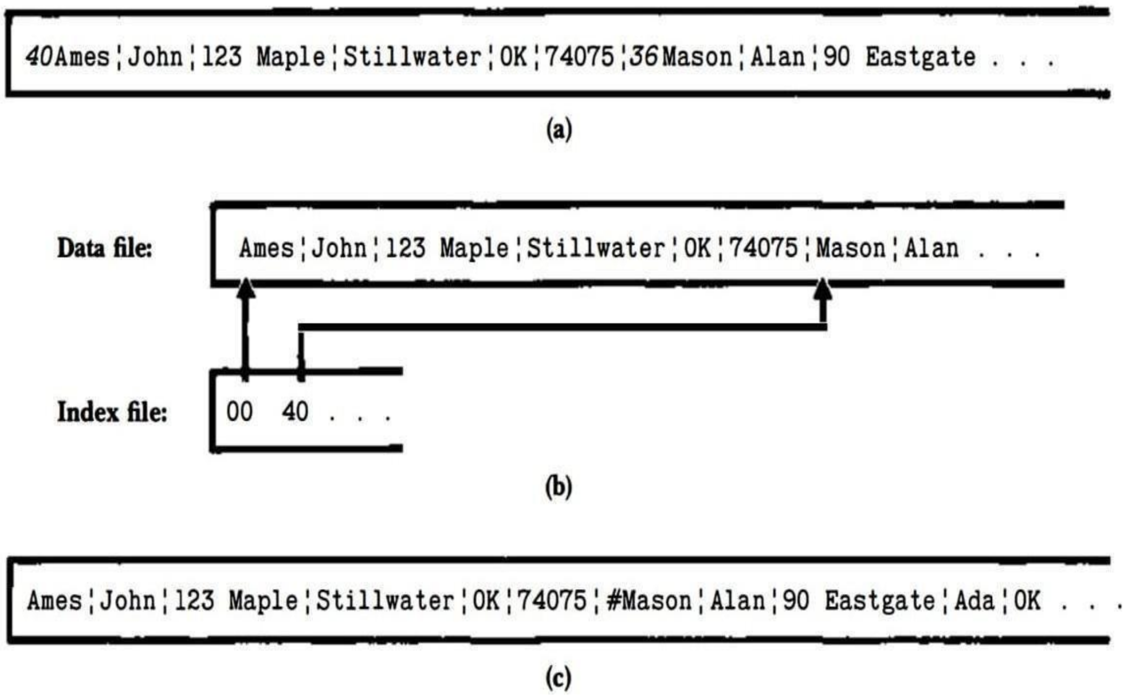

40Ames¦John¦123 Maple¦Stillwater¦OK¦74075¦36Mason¦Alan¦90 Eastgate . . .
(a)
Data file:
Ames¦John¦123 Maple¦Stillwater¦OK¦74075¦Mason¦Alan . . .
Index file:
00 40 . . .
(b)
Ames¦John¦123 Maple¦Stillwater¦OK¦74075¦#Mason¦Alan¦90 Eastgate¦Ada¦OK . . .
(c)

Figure 1.3 Using Length Indicator, Index and Record Delimiters

1.5 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.

1.6 Objectives

The proposed system is set for the following objectives.

- ✓ Automated system development for managing details of patients for covid-19 pandemic.
- ✓ With this it is very easy to add the patient's information, who are infected and who are not infected.

1.7 Existing System

Manual process of reserving bus and managing bus ticket data are very difficult. There are various problems also faced by the transportation management and their customer.

The problems occurred in Existing System are

1. **Expensive and Time consuming**: The process of collecting data and entering the data into the database takes much time and is expensive to conduct.
2. **Errors during work:** Errors are part of all human being; It is very unlikely for humans to be 100 percent efficient in data entry

1.8 Proposed System

- ✓ Automation and Efficiency: The system automates various administrative tasks such as adding buses, managing seat availability, and updating ticket information. This automation reduces the manual effort required and improves overall efficiency in managing bus reservations.
- ✓ User-Friendly Interface: The system provides a user-friendly interface for both administrators and users. Administrators can easily add, delete, and edit bus details, while users can conveniently book tickets, select seats, and cancel reservations. The intuitive design enhances the user experience and reduces the learning curve.
- ✓ Real-time Availability: The system maintains real-time seat availability information. Users can quickly check the availability of seats for a specific bus, allowing them to make informed decisions and choose their preferred seats. This real-time information reduces the chances of overbooking and improves customer satisfaction.
- ✓ Accurate Ticket Management: The system ensures accurate ticket management by maintaining a centralized database of ticket details. This eliminates the risk of manual errors, such as duplicate bookings or incorrect passenger information. Administrators can easily track and manage ticket data, simplifying the process of ticket validation and reconciliation.
- ✓ Data Integrity and Security: The system maintains separate index files for buses and tickets, ensuring data integrity and security. By organizing data in a structured manner, the system minimizes the chances of data corruption or loss. Additionally, access to the system can be restricted through user authentication, safeguarding sensitive information.
- ✓ Improved Efficiency: With automated processes for ticket booking, cancellation, and seat management, the system significantly improves overall efficiency. Users can quickly search for buses, book tickets, and receive instant confirmations. Administrators can easily track bus details, manage bookings, and generate reports, streamlining their operations.

CHAPTER 2

PROBLEM DEFINITION

The bus ticket booking system aims to provide a convenient and efficient way for users to book bus tickets and for administrators to manage the bus schedules and bookings. The system allows users to view available buses, book tickets, cancel tickets, and view their booked tickets. Administrators have additional functionalities such as adding and deleting buses, editing bus details, and viewing booked tickets. The system maintains separate files for storing bus and ticket data, and it includes index files to optimize data retrieval. However, the system could benefit from improvements in terms of error handling and input validation to ensure smooth and reliable operations.

Overall, the bus ticket booking system addresses the need for a streamlined process of reserving bus tickets, eliminating the need for manual booking and reducing human errors. It provides users with the convenience of viewing bus details, selecting seats, and making bookings, while administrators can efficiently manage the bus schedules and bookings. By implementing a user-friendly interface and organizing data in separate files, the system aims to simplify the ticket booking process for users and provide administrators with the necessary tools to maintain and update the bus schedules.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

In this chapter, hardware requirements have been discussed in section 2.1 and software requirements has been discussed in 2.2

3.1 Hardware Requirements

- Computer with a 1.1 GHz or faster processor
- Minimum 2GB of RAM or more
- 40 GB hard disk or above
- VGA COLOR Monitor
- 1366 × 768 or higher-resolution display
- Keyboard, Mouse

3.2 Software Requirements

- Operating System: Windows 10 or Windows 11
- Google Chrome/Internet Explorer
- Python main editor (user interface): PyCharm Community
- External tools – Notepad, Pyside6

CHAPTER 4

SYSTEM ANALYSIS

In this chapter, we discussed about the complete analysis of System.

4.1 Analysis of the Project

The system analysis for the bus reservation system involves assessing user requirements, analyzing existing processes, identifying system functionalities, and evaluating the feasibility of implementation. This analysis aims to ensure that the system meets user needs, improves efficiency, provides real-time updates on seat availability, enables easy ticket cancellation, and offers a secure and scalable platform for bus reservation management.

4.2 Structure used to Store the Fields and Records

a) Fixed number of Fields:

In Restaurant Management System, there are five files which contains different fields in it. All files contain fixed number of fields for each record for storing data.

b) Variable length record:

Variable-length records are the records that vary in size. Our project uses variable length record system in all files. And each record is separated by a '\n'.

c) Separate the fields with delimiters:

The fields within a record are followed by a delimiting byte or series of bytes. Fields within a record can have different sizes. Different records can have different length fields. Programs which access the fields must know the delimiter.

The delimiter cannot occur within the data. If used with delimited records, the field delimiter must be different from the record delimiter. Here the external overhead for field separation is equal to the size of the delimiter per field.

d) Deleation record with special character:

Suppose if any record or any field in the file if deleted means then that record should show the special character because of that we can identify the deleted record easily.

e) Index sorting on the records on files:

Index sorting is a technique used in the bus reservation system to optimize the search and retrieval of bus schedules and availability. It involves creating an index that maps key attributes, such as bus routes or departure times, to their corresponding data records. This index allows for efficient sorting and searching, reducing the time complexity of operations. By utilizing index sorting, the system can quickly locate and display relevant information, enhancing the user experience and improving overall system performance.

CHAPTER 5
FILE STRUCTURE IN PYTHON

5.1 PYTHON

In our project we use python as our language. All file handling and data retrieval and updates are done using python. Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Open() function

We use open () function in Python to open a file in read or write mode. As explained above, open() will return a file object. To return a file object we use open() function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: open(filename, mode). There are three kinds of mode, that Python provides and how files can be opened:
"r", for reading.

"w", for writing.

"a", for appending.

"r+", for both reading and writing

One must keep in mind that the mode argument is not mandatory. If not passed, then Python will assume it to be " r " by default. Let's look at this program and try to analyze how the read mode works:

```
# a file named "geek", will be opened with the reading mode.
file = open ('geek.txt', 'r')
# This will print every line one by one in the file for each in file:
print (each)
```

read() function

There is more than one way to read a file in Python. If you need to extract a string that contains all characters in the file then we can use file.read(). The full code would work like this:

```
# Python code to illustrate read() mode.
File = open("file.text", "r")
print (file.read())
```

Another way to read a file is to call a certain number of characters like in the following code

the interpreter will read the first five characters of stored data and return it as a string:

```
# Python code to illustrate read() mode character wise.
File = open("file.txt", "r")
print (file.read(5))
```

Creating a file using write() mode

Let’s see how to create a file and how write mode works:

To manipulate the file, write the following in your Python environment:

```
# Python code to create a file.
 file = open('geek.txt','w')
file.write("This is the write command")
file.write("It allows us to write in a particular file") file.close()
```

The close() command terminates all the resources in use and frees the system of this particular program.

append() function

Let’s see how the append mode works:

```
# Python code to illustrate append() mode.
File = open('geek.txt','a')
file.write("This will add this line") file.close()
```

There are also various other commands in file handling that is used to handle various tasks like:

rstrip(): This function strips each line of a file off spaces from the right-hand side.

lstrip(): This function strips each line of a file off spaces from the left-hand side. It is designed to provide much cleaner syntax and exceptions handling when you are working with code. That explains why it’s good practice to use them with a statement where applicable. This is helpful because using this method any files opened will be closed automatically after one is done, so auto-cleanup.

Example:

```
# Python code to illustrate with() with open("file.txt") as file:
data = file.read()
```

# do something with data Using write along with with() function

We can also use write function along with with() function:

# Python code to illustrate with() along with write() with open("file.txt", "w") as f:

f.write("Hello World!!!")

You can also split using any characters as we wish. Here is the code:

# Python code to illustrate split() function with open("file.text", "r") as file:

data = file.readlines() for line in data:

word = line.split()

print (word)

5.2 Index sorting in python

✓ **update_bus_index() function:**

This function reads the data from the bus.txt file, which contains the bus details, and creates a list of bus records.The bus records are then sorted using the sort() method, with a custom sorting key specified using a lambda function. The lambda function extracts the bus number from each bus record and uses it for comparison during sorting.

After sorting the bus records, the function writes the sorted data to the busindex.txt file, which serves as the index file for buses.

✓ **update_ticket_index() function:**

This function reads the data from the ticket.txt file, which contains the ticket details, and creates a list of ticket records.The ticket records are then sorted using the sort() method, with a custom sorting key specified using a lambda function. The lambda function extracts the ticket ID from each ticket record and uses it for comparison during sorting.

After sorting the ticket records, the function writes the sorted data to the ticketindex.txt file, which serves as the index file for tickets.

The sorting process in both functions follows a similar pattern. The sort() method provided by Python's built-in list type is used to sort the records. The key parameter of the sort() method accepts a function that is used to extract the value to be used for sorting from each record.By providing a lambda function as the sorting key, we specify that the sorting should be based on a specific field of the record. In this case, the bus number is used as the sorting key for the bus records, and the ticket ID is used as the sorting key for the ticket records.

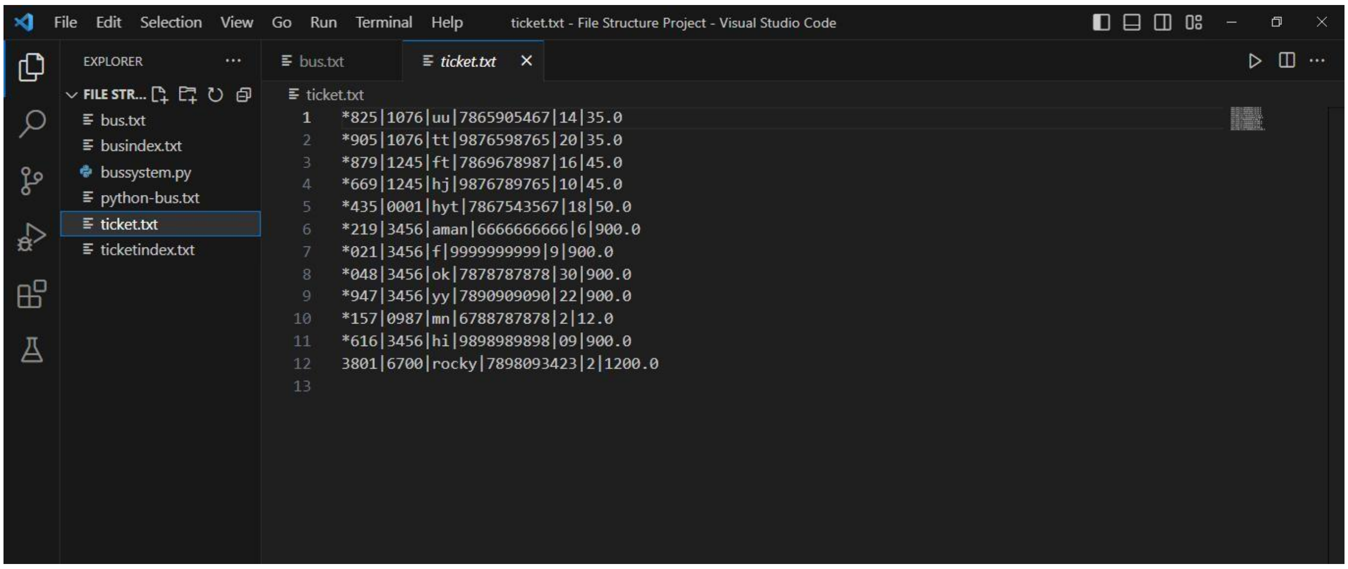
The lambda function takes each record as input and returns the value to be used for comparison during sorting. This allows the sorting algorithm to compare the records based on the specified field (bus number or ticket ID) and arrange them in the desired order.

Overall, index sorting in this bus reservation management system involves reading the data, creating a list of records, sorting the records based on a specific field using the sort() method, and then writing the sorted data to the index file. This ensures that the index files maintain a sorted order of buses and tickets for efficient searching and retrieval

5.3 Data Files

Ticket Details

This file shows the sorted details on the basis of ticket number.

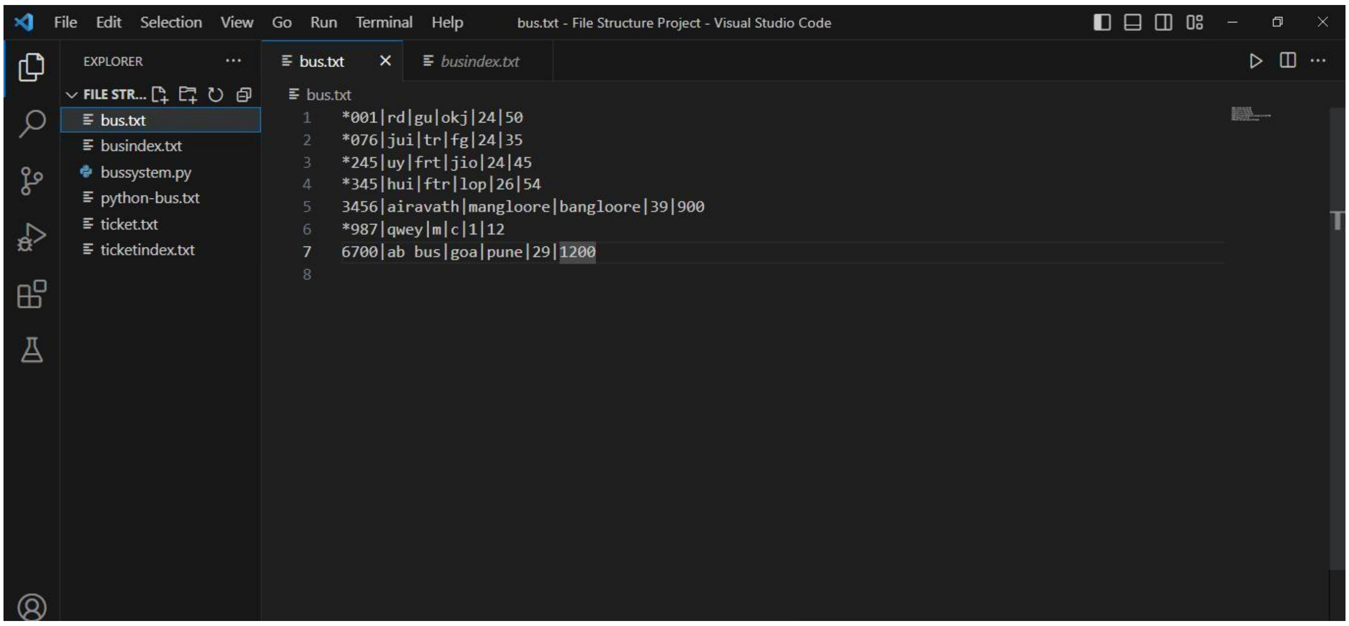


```
*825|1076|uu|7865905467|14|35.0
*905|1076|tt|9876598765|20|35.0
*879|1245|ft|7869678987|16|45.0
*669|1245|hj|9876789765|10|45.0
*435|0001|hyt|7867543567|18|50.0
*219|3456|aman|6666666666|6|900.0
*021|3456|f|9999999999|9|900.0
*048|3456|ok|7878787878|30|900.0
*947|3456|yy|7890909090|22|900.0
*157|0987|mn|6788787878|2|12.0
*616|3456|hi|9898989898|09|900.0
3801|6700|rocky|7898093423|2|1200.0
```

Fig 5.3.1 Ticket.txt File

Bus Details

This file shows the sorted details on the basis of bus number.



```
*001|rd|gu|okj|24|50
*076|jui|tr|fg|24|35
*245|uy|frt|jio|24|45
*345|hui|ftr|lop|26|54
3456|airavath|mangloore|bangloore|39|900
*987|qwey|m|c|1|12
6700|ab bus|goa|pune|29|1200
```

Fig 5.3.2 Bus.txt File

CHAPTER 6

FRONT-END AND BACK-END DESIGN

In this chapter we have explained about the front-end design tools such as PyCharm along with front end language Python.

6.1 PyCharm IDE:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

6.2 Python:

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

6.3 Pseudo code

Pseudo code is an informal high-level description of the operative principles of the algorithm and computer programs. Code for generation of timetable is given in the following sub sections.

6.3.1 Pseudo code for Add Bus

```
Step1: Prompt admin to enter bus details.
Step2: Append bus details to BUS_FILE and update bus index.
step3: Exit.
```

6.3.2 Pseudo code for Delete Bus

```
Step1: Prompt admin to enter bus number to delete.
Step2: Mark bus details with asterisk (*) in BUS_FILE.
Step3: Mark corresponding ticket details with asterisk (*) in TICKET_FILE.Step4:
Update bus index in BUS_INDEX_FILE and update ticket index.
Step5: Exit.
```

6.3.3 Pseudo code for Book Ticket

```
Step1: Prompt user and admin for bus number and details.
Step2: Check if bus exists and has available seats.
Step3: Reduce available seats count in BUS_FILE.
Step4: Write ticket details to TICKET_FILE and Update bus index.
Step4: Exit
```

6.3.4 Pseudo code for Cancel Ticket

```
Step1: Prompt user and admin for ticket ID to cancel.
Step2: Mark ticket details with asterisk (*) in TICKET_FILE.
Step3: Update available seats BUS_FILE.
Step4: Exit.
```

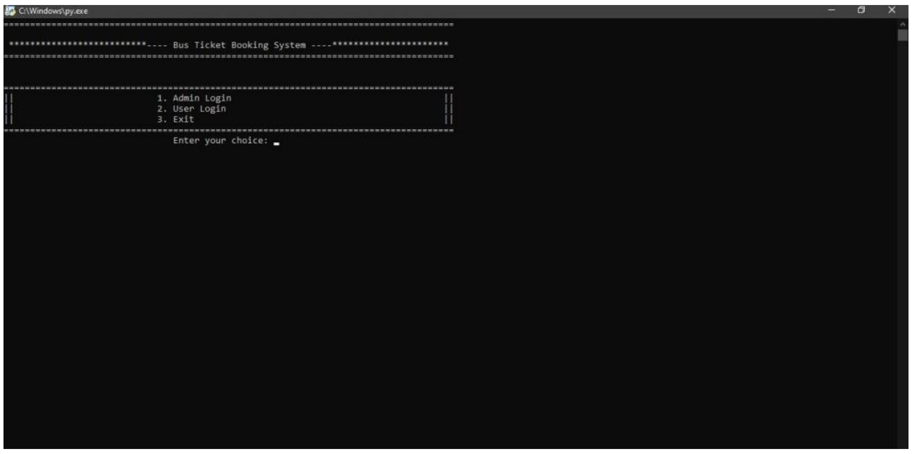
CHAPTER 7

SNAPSHOT AND RESULTS

In this chapter, snapshot of bus reservation system and results have been discussed.

7.1 HOME PAGE

The below figure 7.1 shows the Home Page.

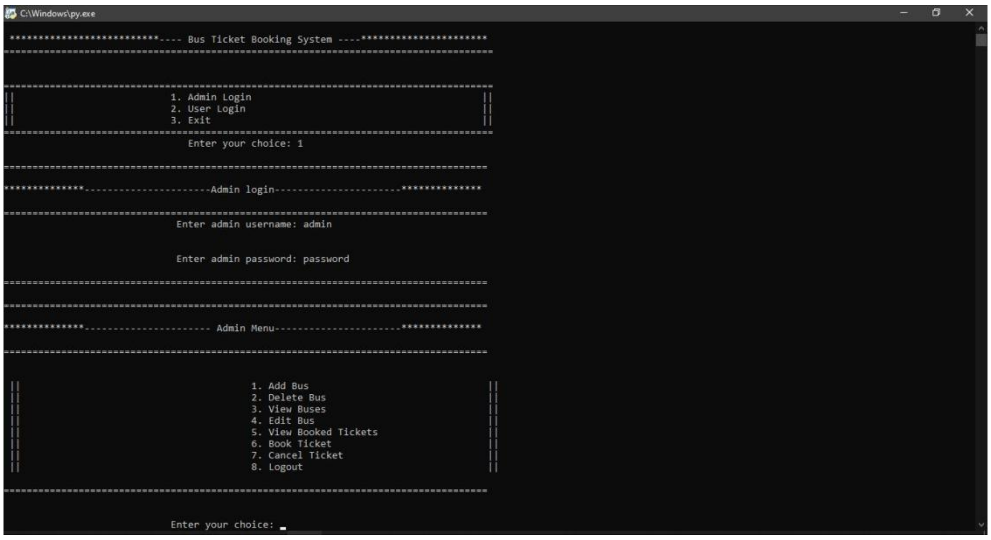


```
C:\Windows\py.exe
================================================================================

 *************************---- Bus Ticket Booking System ----*********************
================================================================================


================================================================================
||                          1. Admin Login                                    ||
||                          2. User Login                                     ||
||                          3. Exit                                           ||
================================================================================
                             Enter your choice: _
```

Fig 7.1 Home Page

7.2 ADMIN PAGE

The below figure 7.2 shows Admin Page.

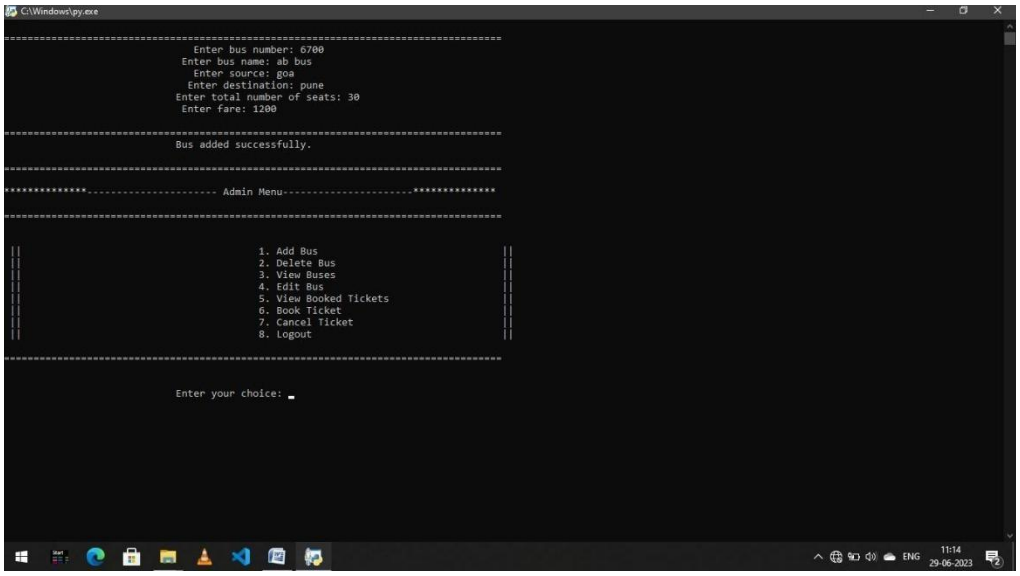


```
C:\Windows\py.exe
 *************************---- Bus Ticket Booking System ----*********************
================================================================================


================================================================================
||                          1. Admin Login                                    ||
||                          2. User Login                                     ||
||                          3. Exit                                           ||
================================================================================
                             Enter your choice: 1

================================================================================

*************.--------------------Admin login--------------------*************

================================================================================
                           Enter admin username: admin


                           Enter admin password: password

================================================================================

================================================================================

*************.-------------------- Admin Menu--------------------*************

================================================================================


 ||                                        1. Add Bus                              ||
 ||                                        2. Delete Bus                           ||
 ||                                        3. View Buses                           ||
 ||                                        4. Edit Bus                             ||
 ||                                        5. View Booked Tickets                  ||
 ||                                        6. Book Ticket                          ||
 ||                                        7. Cancel Ticket                        ||
 ||                                        8. Logout                               ||

================================================================================


                            Enter your choice: _
```

Fig 7.2 Admin Page

7.3 ADD BUS PAGE

The below figure 7.3 shows Add Bus Page.

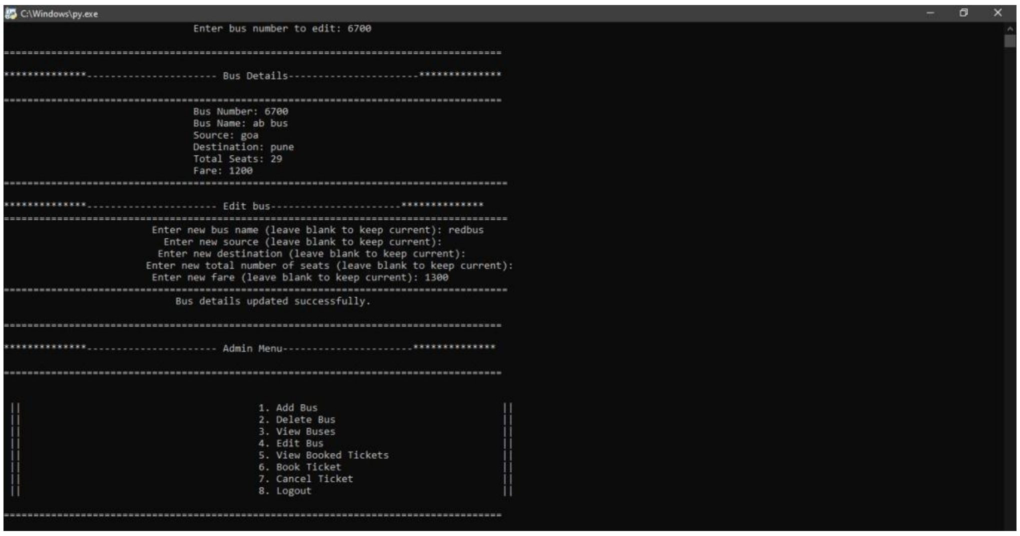


```
C:\Windows\py.exe
====================================================================================
                                Enter bus number: 6700
                              Enter bus name: ab bus
                                Enter source: goa
                               Enter destination: pune
                             Enter total number of seats: 30
                              Enter fare: 1200

====================================================================================
                             Bus added successfully.

====================================================================================

*************.--------------------- Admin Menu---------------------.*************

====================================================================================


 ||                                         1. Add Bus                                ||
 ||                                         2. Delete Bus                             ||
 ||                                         3. View Buses                             ||
 ||                                         4. Edit Bus                               ||
 ||                                         5. View Booked Tickets                    ||
 ||                                         6. Book Ticket                            ||
 ||                                         7. Cancel Ticket                          ||
 ||                                         8. Logout                                 ||

====================================================================================


                             Enter your choice: _
```

Fig 7.3 Add Bus Page

7.4 UPDATE BUS PAGE

The below figure 7.4 shows Update Bus Page.

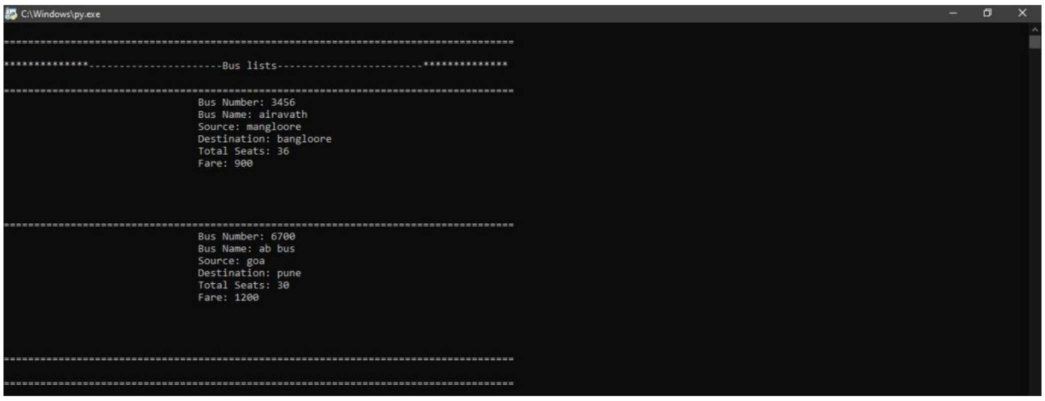


```
C:\Windows\py.exe
                                Enter bus number to edit: 6700

====================================================================================

*************.--------------------- Bus Details---------------------.*************

====================================================================================
                                Bus Number: 6700
                                Bus Name: ab bus
                                Source: goa
                                Destination: pune
                                Total Seats: 29
                                Fare: 1200
====================================================================================

*************.--------------------- Edit bus---------------------.*************
====================================================================================
                         Enter new bus name (leave blank to keep current): redbus
                           Enter new source (leave blank to keep current):
                          Enter new destination (leave blank to keep current):
                        Enter new total number of seats (leave blank to keep current):
                         Enter new fare (leave blank to keep current): 1300
====================================================================================
                             Bus details updated successfully.

====================================================================================

*************.--------------------- Admin Menu---------------------.*************

====================================================================================


 ||                                         1. Add Bus                                ||
 ||                                         2. Delete Bus                             ||
 ||                                         3. View Buses                             ||
 ||                                         4. Edit Bus                               ||
 ||                                         5. View Booked Tickets                    ||
 ||                                         6. Book Ticket                            ||
 ||                                         7. Cancel Ticket                          ||
 ||                                         8. Logout                                 ||

====================================================================================
```

Fig 7.4 Update Bus Page

7.5 VIEW AVAILABLE BUS PAGE

The below figure 7.5 shows Available Bus Page.

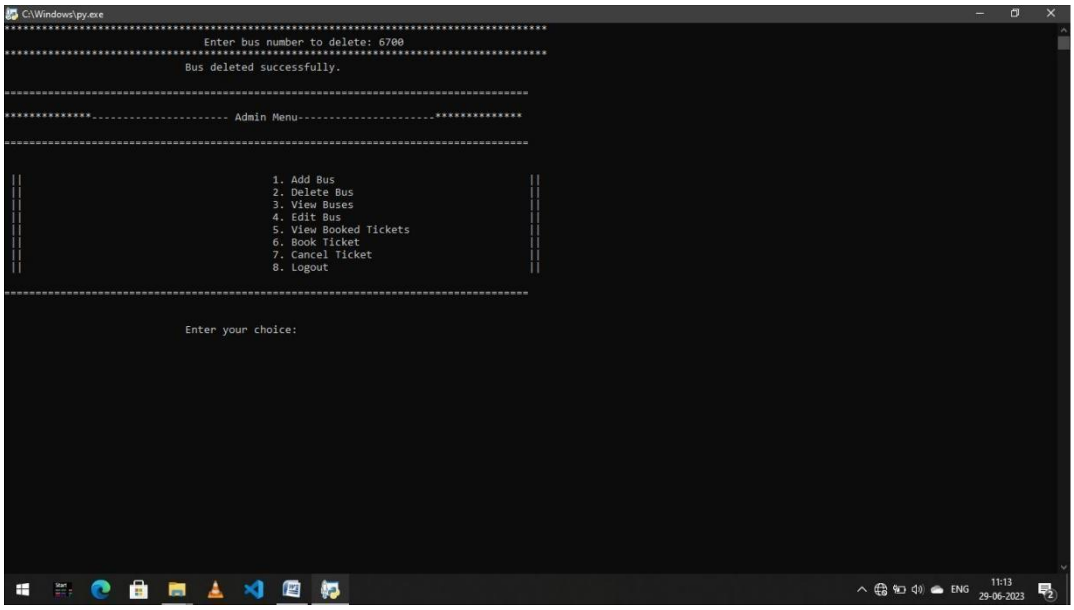


```
C:\Windows\py.exe
==================================================================================

**************.---------------------Bus lists-----------------------.**************

==================================================================================
                              Bus Number: 3456
                              Bus Name: airavath
                              Source: mangloore
                              Destination: bangloore
                              Total Seats: 36
                              Fare: 900



==================================================================================
                              Bus Number: 6700
                              Bus Name: ab bus
                              Source: goa
                              Destination: pune
                              Total Seats: 30
                              Fare: 1200



==================================================================================

==================================================================================
```

Fig 7.5 View Available Bus Page

7.6 DELETE BUS PAGE

The below figure 7.6 shows Delete Bus Page.

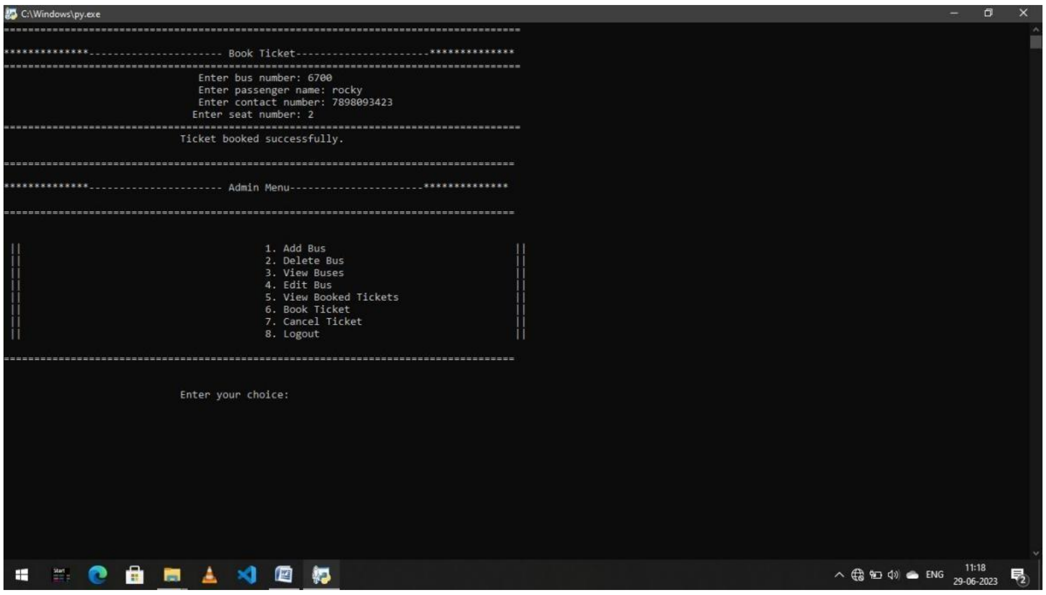


```
C:\Windows\py.exe
************************************************************************************
                                Enter bus number to delete: 6700
************************************************************************************
                             Bus deleted successfully.

==================================================================================

**************.--------------------- Admin Menu---------------------.**************

==================================================================================


 ||                                        1. Add Bus                             ||
 ||                                        2. Delete Bus                          ||
 ||                                        3. View Buses                          ||
 ||                                        4. Edit Bus                            ||
 ||                                        5. View Booked Tickets                 ||
 ||                                        6. Book Ticket                         ||
 ||                                        7. Cancel Ticket                       ||
 ||                                        8. Logout                              ||

==================================================================================


                             Enter your choice:
```

Fig 7.6 Delete Bus Page

7.7 BOOK TICKET PAGE

The below figure 7.7 shows Book Ticket Page.

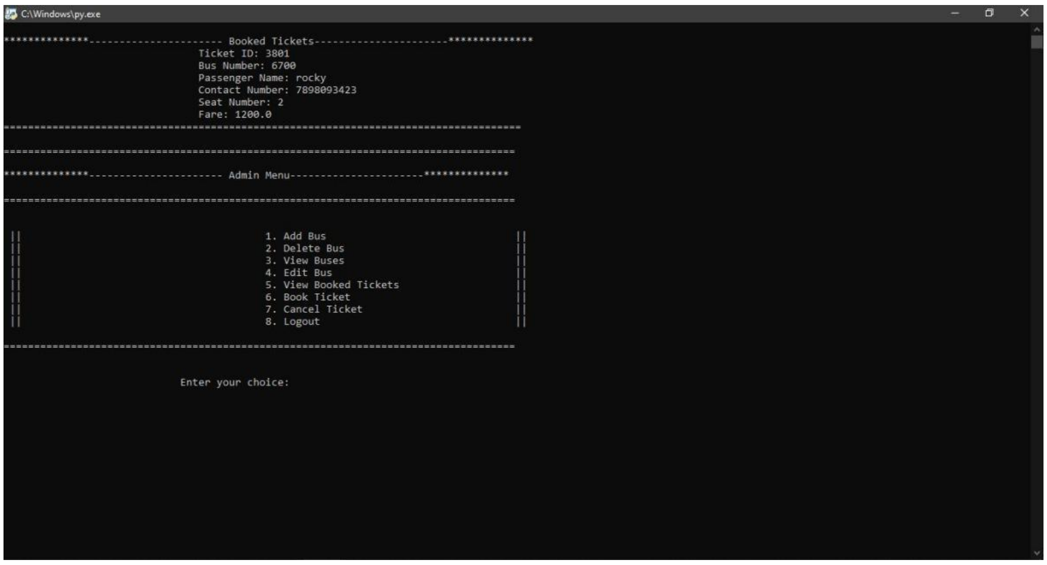


```
C:\Windows\py.exe
==========================================================================================

**************---------------------- Book Ticket----------------------**************
==========================================================================================
                                 Enter bus number: 6700
                                 Enter passenger name: rocky
                                 Enter contact number: 7898093423
                                Enter seat number: 2
==========================================================================================
                              Ticket booked successfully.

=========================================================================================

**************---------------------- Admin Menu----------------------**************

=========================================================================================


 ||                                         1. Add Bus                                   ||
 ||                                         2. Delete Bus                                ||
 ||                                         3. View Buses                                ||
 ||                                         4. Edit Bus                                  ||
 ||                                         5. View Booked Tickets                       ||
 ||                                         6. Book Ticket                               ||
 ||                                         7. Cancel Ticket                             ||
 ||                                         8. Logout                                    ||

=========================================================================================


                              Enter your choice:
```

Fig 7.7 Book Ticket Page

7.8 VIEW BOOKED TICKETS PAGE

The below figure 7.8 shows View Booked Tickets Page.

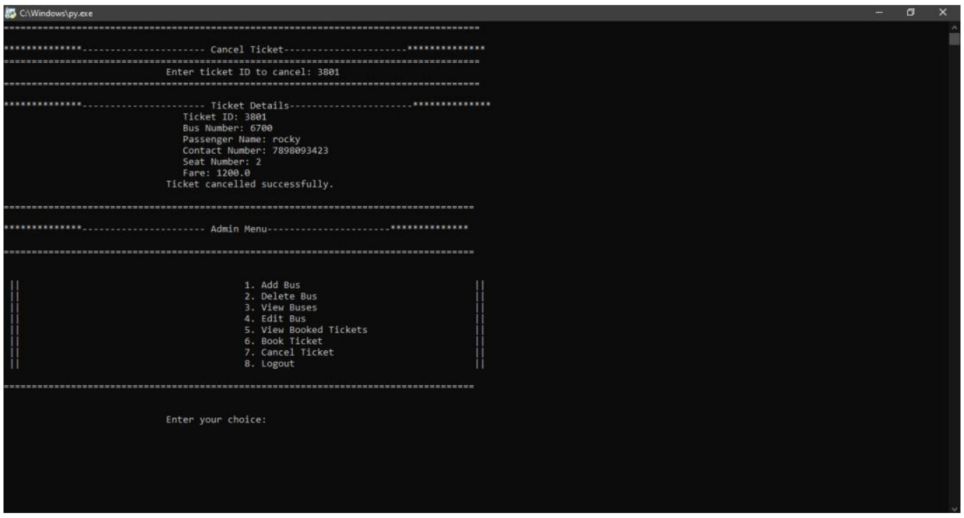


```
C:\Windows\py.exe

**************---------------------- Booked Tickets---------------------**************
                                Ticket ID: 3801
                                Bus Number: 6700
                                Passenger Name: rocky
                                Contact Number: 7898093423
                                Seat Number: 2
                                Fare: 1200.0
=========================================================================================

=========================================================================================

**************---------------------- Admin Menu----------------------**************

=========================================================================================


 ||                                         1. Add Bus                                   ||
 ||                                         2. Delete Bus                                ||
 ||                                         3. View Buses                                ||
 ||                                         4. Edit Bus                                  ||
 ||                                         5. View Booked Tickets                       ||
 ||                                         6. Book Ticket                               ||
 ||                                         7. Cancel Ticket                             ||
 ||                                         8. Logout                                    ||

=========================================================================================


                              Enter your choice:
```

Fig 7.8 View Booked Tickets Page

7.9 CANCEL TICKET PAGE

The below figure 7.9 shows Cancel Ticket Page.

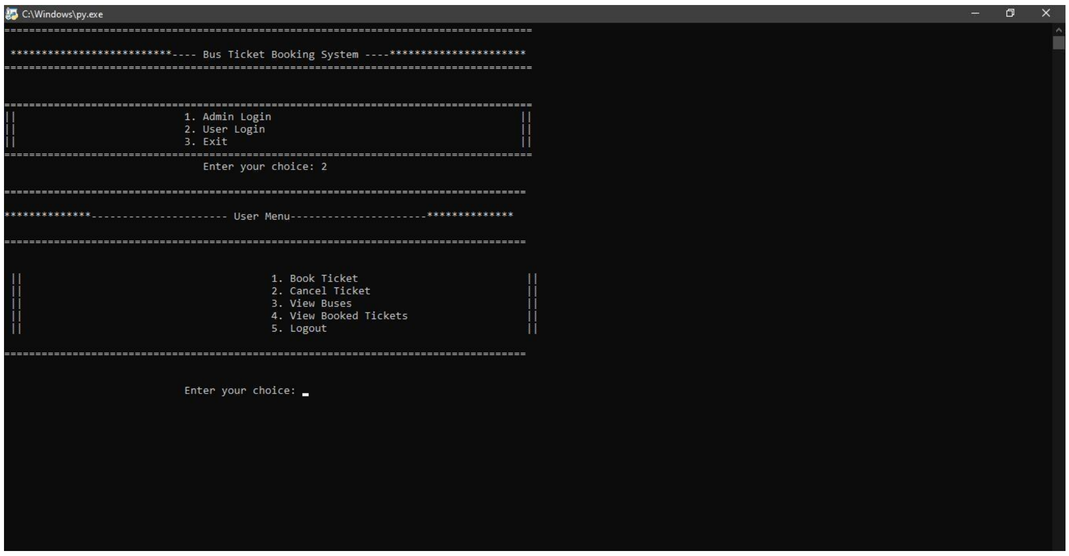


```
C:\Windows\py.exe
===================================================================================

*************---------------------- Cancel Ticket---------------------*************
===================================================================================
                             Enter ticket ID to cancel: 3801
===================================================================================

*************---------------------- Ticket Details---------------------*************
                                Ticket ID: 3801
                                Bus Number: 6700
                                Passenger Name: rocky
                                Contact Number: 7898093423
                                Seat Number: 2
                                Fare: 1200.0
                             Ticket cancelled successfully.

==================================================================================

*************---------------------- Admin Menu---------------------*************

==================================================================================


 ||                                         1. Add Bus                            ||
 ||                                         2. Delete Bus                         ||
 ||                                         3. View Buses                         ||
 ||                                         4. Edit Bus                           ||
 ||                                         5. View Booked Tickets                ||
 ||                                         6. Book Ticket                        ||
 ||                                         7. Cancel Ticket                      ||
 ||                                         8. Logout                             ||

==================================================================================


                             Enter your choice:
```

Fig 7.9 Cancel Ticket Page

7.10 USER LOGIN PAGE

The below figure 7.10 shows User Login Page.



```
C:\Windows\py.exe
===================================================================================

 *************************---- Bus Ticket Booking System ----*********************
===================================================================================


===================================================================================
||                            1. Admin Login                                     ||
||                            2. User Login                                      ||
||                            3. Exit                                            ||
===================================================================================
                               Enter your choice: 2

==================================================================================

*************---------------------- User Menu---------------------*************

==================================================================================


 ||                                          1. Book Ticket                       ||
 ||                                          2. Cancel Ticket                     ||
 ||                                          3. View Buses                        ||
 ||                                          4. View Booked Tickets               ||
 ||                                          5. Logout                            ||

==================================================================================


                            Enter your choice: _
```

Fig 7.10 User Login Page

CHAPTER 8

CONCLUSION

The mini project entitled as “BUS RESERVATION MANAGEMENT SYSTEM” is essential for you to run your transportation management efficiently and smoothly. you may not be at the booking office at all times but you must have access to this data at times and a robust Bus Reservation Management System helps you do just that. Data entering of users and admin are also included in this system along with the reserving the ticket and seat for the traveller, bus and ticket records are interconnected in-order to maintain the accuracy of this system .

This system can also be further improved adding many other features and including the other systems as well. Finally we believe that we were able to launch an effective computerized system to the Transportation causing the Bus Reservation System to perform well in the future regarding the billing and restaurant records.

CHAPTER 9

REFERENCES

- Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners
- File Structure with C++ by Michael J Folk

- www.stackoverflow.com
- www.tutorialpoint.com
- www.wikepedia.com
- www.softwaretestingfundamentals.com