TELECOME BILLING SYSTEM

*by*

D.DANUSRI (2403713820522010)

G.HARIPRIYA (2403713820522017)

First Year B.Tech Information Technology



Sri Ramakrishna Institute of Technology

Coimbatore 641010

DEC 2024-2025

APPROVAL AND DECLARATION

This project report titled TELECOME BILLING SYSTEM prepared and submitted by D.DANUSRI (Register Number: 2403713820522010), G.HARIPRIYA (Register Number: 2403713820522017) and has been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the requirement for the Bachelor of Technology (Information Technology) in Sri Ramakrishna Institute of Technology, Coimbatore (SRIT).

Checked and Approved by

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ms. J.SANGEETHA SARAL M.E.,

Project Supervisor

Assistant Professor

Department of Information Technology

Department of Science and Humanities

Sri Ramakrishna Institute of Technology, Coimbatore10

DEC2024

TABLE OF CONTENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | |  |  |
| S.NO | CONTENT | PAGE\_NO |
| 1 | ABSTRACT | 4 |
| 2 | INTRODUCTION | 5 |
| 3 | ALGORITHM | 6 |
| 4 | PSEUDO CODE | 9 |
| 5 | FLOW CHART | 10 |
| 6 | SOURCE CODE | 11 |
| 7 | OUTPUT | 13 |
| 8 | RESULT | 18 |
| 9 | CONCLUSION | 19 |

TELECOME BILLING SYSTEM

ABSTRACT

The telecommunication billing system is a software solution developed in C programming language to manage customer billing efficiently. It provides core functionalities such as adding new customer records, viewing all stored records, modifying customer details, retrieving individual customer payment information, and deleting outdated records.

The system calculates the total bill dynamically based on the customer's usage, using a predefined rate per minute of $0.10. Customer data, including name, phone number, usage, and total bill, is stored using a structured approach with the struct data type, allowing for clear organization and easy manipulation.

To handle multiple customers, the program utilizes an array of structures with a capacity of up to 100 records. A menudriven interface facilitates user interaction, enabling seamless navigation between options like adding, updating, viewing, and deleting records. The modular design of the program ensures that each functionality is implemented as a separate function, promoting reusability and ease of maintenance.

This system is designed to cater to small and mediumsized telecommunication businesses, providing an effective and userfriendly platform for managing billing operations. By automating calculations and simplifying record management, it reduces the manual effort required for customer billing, ensuring accuracy and efficiency.

INTODUCTION

The telecommunication industry relies heavily on accurate and efficient billing systems to manage customer data and generate bills. The Telecommunication Billing System implemented in C is a software application designed to streamline the billing process by automating tasks such as tracking customer usage, calculating charges, and maintaining a database of customer records.

This system provides a comprehensive solution for small and mediumsized telecommunication providers to manage customer billing efficiently. It allows users to add, modify, view, and delete customer records with ease. The program uses a fixed rate of $0.10 per minute to calculate the total bill based on the customer's usage. The data, including the customer's name, phone number, usage in minutes, and total bill, is stored in a structured format using C's struct data type.

The billing system features a menudriven interface, making it userfriendly and easy to navigate. Users can interact with the system through options such as adding new customer details, viewing a complete list of records, retrieving individual payment details, and deleting outdated records. The program supports up to 100 customer records, ensuring scalability for smallscale telecommunication services.

By automating the billing process, this system reduces human error, saves time, and ensures accuracy in customer payments. It demonstrates key programming concepts such as modular design, arrays, functions, and the use of structured data, making it a valuable learning tool for students and a practical solution for businesses. The Telecommunication Billing System is a reliable, efficient, and costeffective tool for managing customer billing in the telecommunication sector.

ALGORITHM

1. Initialization

1. Define the Customer structure with fields: name, phoneNumber, usage, and totalBill.

2. Declare an array of Customer structures (customers[100]) to store up to 100 records.

3. Initialize customerCount to 0 to track the number of stored records.

1. Display Menu

1. Print the following menu options:

1. Add New Record

2. View List of Records

3. Modify Record

4. View Payment

5. Delete Record

6. Exit

2. Prompt the user for their choice.

3. Process User Choice

Repeat until the user selects the "Exit" option:

1. Read the user's choice.

2. Depending on the choice, perform the corresponding action:

Choice 1: Add New Record

Call addRecord() function.

Choice 2: View List of Records

Call viewRecords() function.

Choice 3: Modify Record

Prompt the user for the phone number.

Call modifyRecord(phoneNumber) function.

Choice 4: View Payment

Prompt the user for the phone number.

Call viewPayment(phoneNumber) function.

Choice 5: Delete Record

Prompt the user for the phone number.

Call deleteRecord(phoneNumber) function.

Choice 6: Exit

Exit the program.

Invalid Choice

Print an error message and redisplay the menu.

1. Functions

1. Add Record (addRecord()):

If customerCount < 100:

1. Prompt the user for name, phoneNumber, and usage.

2. Calculate totalBill as usage 0.1.

3. Store the data in customers[customerCount].

4. Increment customerCount.

5. Print "Record added successfully!"

Else, print "Maximum number of records reached!"

2. View Records (viewRecords()):

Print the headers: "Name", "Phone Number", "Usage (min)", "Total Bill ($)".

For each record in the customers array (from index 0 to customerCount1):

Print the customer's details.

3. Modify Record (modifyRecord(phoneNumber))

Search for a record with the given phoneNumber:

1. Iterate through the customers array.

2. If a match is found:

Prompt the user for the new usage.

Recalculate totalBill as usage 0.1.

Print "Record modified successfully!"

Return.

If no match is found, print "Record not found!"

4. View Payment (viewPayment(phoneNumber))

Search for a record with the given phoneNumber:

1. Iterate through the customers array.

2. If a match is found, display the customer's details (name, phoneNumber, usage, and totalBill).

Return.

If no match is found, print "Record not found!"

5. Delete Record (deleteRecord(phoneNumber))

Search for a record with the given phoneNumber:

1. Iterate through the customers array.

2. If a match is found:

Shift all subsequent records in the array one position left to overwrite the record.

Decrement customerCount.

Print "Record deleted successfully!"

Return.

If no match is found, print "Record not found!"

1. Exit Program

When the user selects "Exit," terminate the program.

Algorithm Flow

1. Start

2. Initialize data structures and variables.

3. Display the menu.

4. Process the user's choice.

Perform operations based on the choice (add, view, modify, delete, view payment).

5. If the user chooses "Exit," terminate the program.

6. End

PSEUDO CODE

BEGIN

Initialize customerCount to 0

WHILE true

Display menu options

Input user choice

SWITCH choice

CASE2005 1:

Call addRecord function

CASE 2:

Call viewRecords function

CASE 3:

Input phone number

Call modifyRecord with phone number

CASE 4:

Input phone number

Call viewPayment with phone number

CASE 5:

Input phone number

Call deleteRecord with phone number

CASE 6:

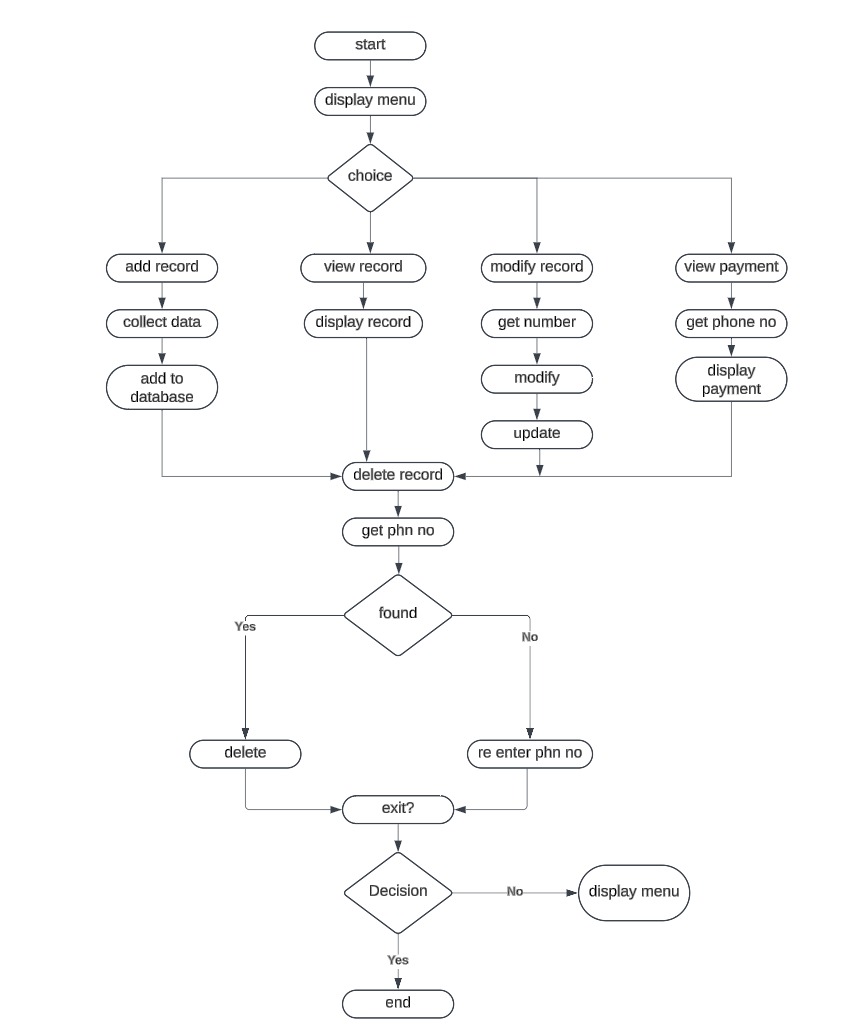
EXIT program

DEFAULT:

Display invalid choice message

END

FLOW CHART



SOURCE CODE

#include <stdio.h>

#include <string.h>

// Structure to hold customer information

struct Customer {

char name[50];

char phoneNumber[15];

float usage;

float totalBill;

};

struct Customer

customers[100]; // Array to store customer data

int customerCount = 0; // Variable to keep track of the

// number of customers

// Function to add a new customer record

void addRecord()

{

if (customerCount < 100) {

printf("\nEnter name: ");

scanf(" %[^\n]s", customers[customerCount].name);

printf("Enter phone number: ");

scanf("%s", customers[customerCount].phoneNumber);

printf("Enter usage (in minutes): ");

scanf("%f", &customers[customerCount].usage);

customers[customerCount].totalBill

= customers[customerCount].usage \* 0.1;

customerCount++;

printf("\nRecord added successfully!\n");

}

else {

printf("\nMaximum number of records reached!\n");

}

}

// Function to view the list of customer records

void viewRecords()

{

printf("\nName\tPhone Number\tUsage(min)\tTotal "

"Bill($)\n");

for (int i = 0; i < customerCount; i++) {

printf("%s\t%s\t%.2f\t\t%.2f\n", customers[i].name,

customers[i].phoneNumber, customers[i].usage,

customers[i].totalBill);

}

}

// Function to modify a customer record

void modifyRecord(char phoneNumber[])

{

for (int i = 0; i < customerCount; i++) {

if (strcmp(customers[i].phoneNumber, phoneNumber)

== 0) {

printf(

"\nEnter new usage (in minutes) for %s: ",

customers[i].name);

scanf("%f", &customers[i].usage);

customers[i].totalBill

= customers[i].usage \* 0.1;

printf("\nRecord modified successfully!\n");

return;

}

}

printf("\nRecord not found!\n");

}

// Function to view payment for a customer

void viewPayment(char phoneNumber[])

{

for (int i = 0; i < customerCount; i++) {

if (strcmp(customers[i].phoneNumber, phoneNumber)

== 0) {

printf(

"%s\t%s\t%.2f\t\t%.2f\n", customers[i].name,

customers[i].phoneNumber,

customers[i].usage, customers[i].totalBill);

return;

}

}

printf("\nRecord not found!\n");

}

// Function to delete a customer record

void deleteRecord(char phoneNumber[])

{

for (int i = 0; i < customerCount; i++) {

if (strcmp(customers[i].phoneNumber, phoneNumber)

== 0) {

for (int j = i; j < customerCount - 1; j++) {

customers[j] = customers[j + 1];

}

customerCount--;

printf("\nRecord deleted successfully!\n");

return;

}

}

printf("\nRecord not found!\n");

}

// Function to display menu options

void displayMenu()

{

printf("\n1. Add New Record\n");

printf("2. View List of Records\n");

printf("3. Modify Record\n");

printf("4. View Payment\n");

printf("5. Delete Record\n");

printf("6. Exit\n");

}

int main()

{

int choice;

char phoneNumber[15];

while (1) {

displayMenu();

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addRecord();

break;

case 2:

viewRecords();

break;

case 3:

printf(

"\nEnter phone number to modify record: ");

scanf("%s", phoneNumber);

modifyRecord(phoneNumber);

break;

case 4:

printf(

"\nEnter phone number to view payment: ");

scanf("%s", phoneNumber);

viewPayment(phoneNumber);

break;

case 5:

printf(

"\nEnter phone number to delete record: ");

scanf("%s", phoneNumber);

deleteRecord(phoneNumber);

break;

case 6:

return 0;

default:

printf("\nInvalid choice! Please try again.\n");

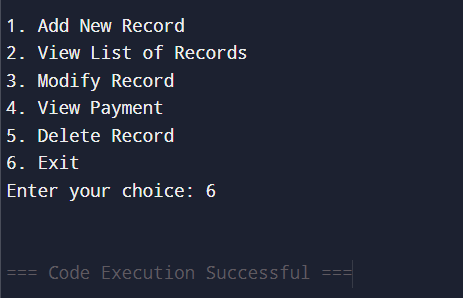
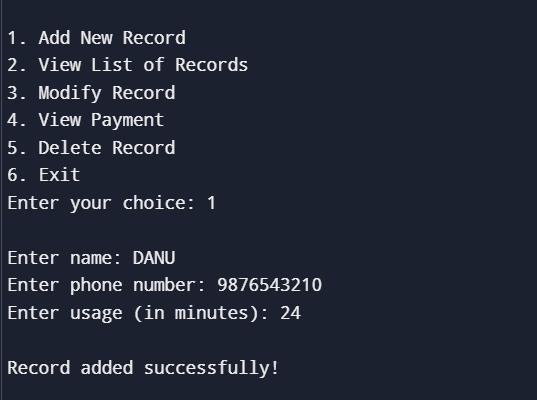
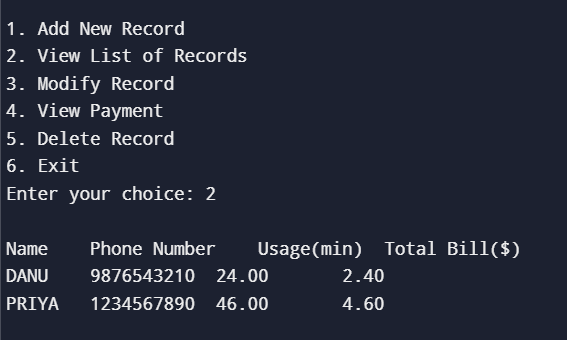
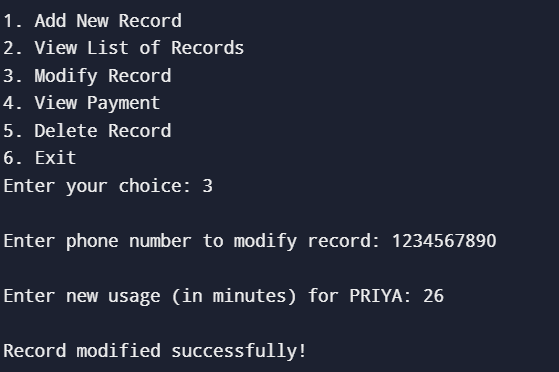
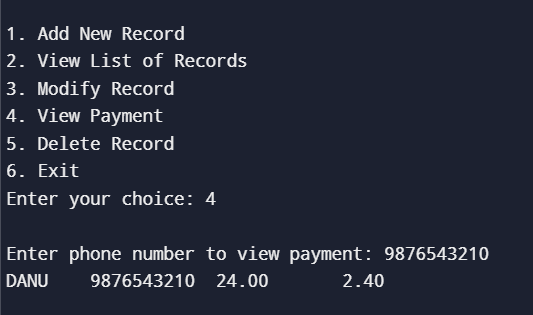
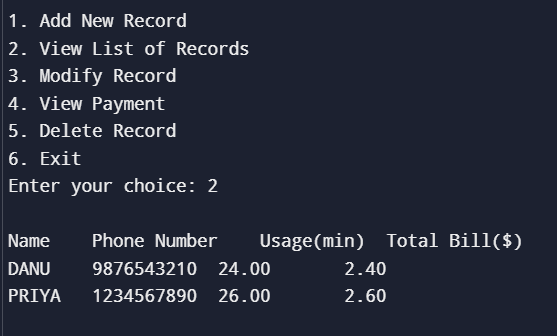
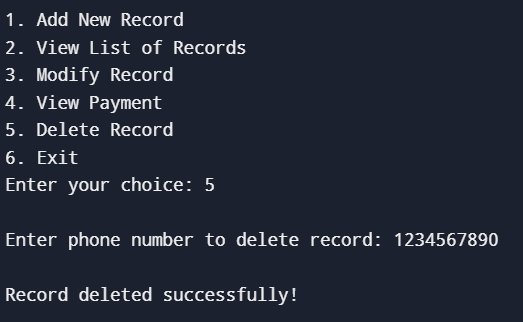
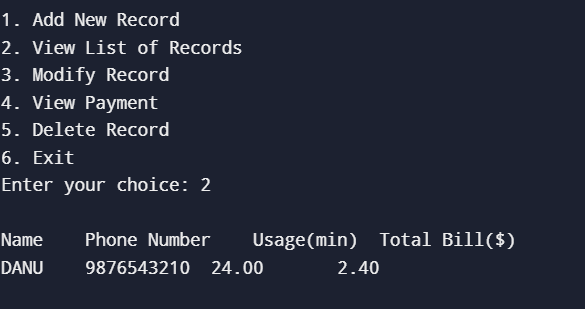
}

}

return 0;

}

OUTPUT



RESULT

The Telecom Billing System in C effectively manages customer billing by allowing users to add, view, modify, and delete records with accurate bill calculation based on usage. It provides a user-friendly, menu-driven interface for seamless interaction and ensures organized data storage. The system meets its objectives, demonstrating efficiency and reliability in handling telecom billing tasks.

CONCLUSION

The Telecom Billing System in C is a practical and efficient solution for managing customer billing records in a telecommunications service. It leverages the power of structured programming, arrays, and functions to handle essential operations like adding, viewing, modifying, and deleting customer records. The system ensures accuracy by automatically calculating the total bill based on usage, streamlining the billing process and minimizing manual errors. Its menu-driven interface makes it user-friendly and accessible, while the modular design allows for easy maintenance and scalability. Though currently limited to handling up to 100 records, the system can be enhanced with features like file storage for data persistence, dynamic memory allocation for handling larger datasets, and advanced functionalities such as detailed reporting and secure user authentication. Overall, this project serves as an excellent example of applying basic programming concepts to solve real-world problems, providing a strong foundation for developing more sophisticated telecom management systems.