

Siddaganga Institute of Technology, Tumakuru

(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi, Approved by AICTE, New Delhi, Accredited by NAAC and ISO 9001:2015 certified)

OPEN ENDED PROJECT

on

“SCHOOL BILLING SYSTEM”

submitted

in the partial fulfillment of the requirements III semester

Bachelor of Engineering

In

Computer Science and Engineering

by

ADITYA RANJAN

1SI20CS005



Department of Computer Science & Engineering

(Program Accredited by NBA)

Siddaganga Institute of Technology

B.H Road, Tumakuru-572103, Karnataka, India.

Web: www.sit.ac.in

Abstract

School Billing Systems are one of the most popular and recognized tools to assist in the management of fee and dues. Billing System is one of the most widely needed tool in the schools and colleges, and it is helpful for the institutes to update the student fee details easily and efficiently. This application will ease the process of this job than earlier pen-paper based system.

Now a days this kind of application is very essential for any small or medium sized organization. This School Billing System enables the user to maintain the billing system in schools and helps in adding, deleting, updating, displaying and resetting a student detail. It is a software with menu driven programs with user-friendly interface which efficiently takes care of the billing activities of the organization. An accountant regardless of the number of staffs and students can maintain fee details digitally. The software provides benefits like quick find out of information and faster access of information. It also removes the monotonous access to the information.

Main aim of this software is to provide easy and efficient functionalities for billing activities and also to provide full functional reports to management of institute with necessary details.

Table of Contents

Sl. No	Contents	Page No.
1	Introduction	3
2	Literature Survey	4
3	Purpose and Benefits	5
4	Problem Statement	6
5	User Defined Functions	7
6	Source Code	8
7	Methodology	15
8	Working	16
9	Conclusion	25
10	Reference	26

Introduction

The School Billing System project in C keeps record of all the students studying in the institution. The application is run by the administrator who can add record, modify, delete and find records according to the need. The basic feature of this project is that it shows fees that the students need to pay and fine of the students. It also records the information of the students studying in the institution.

The data within the program can be recorded and saved in files. The administrator will need the admission ID to perform billing operations as per requirement.

The application keeps record of student name, admission ID and fee status.

The School Billing System project in C utilizes data structures to store the records and provides access to the user whenever required.

Together with data structure, the project utilizes functions to perform various operations. The functions in the program are used for checking data, adding details of student, modifying details of a student, searching and deleting records, calculating dues, total fee and displaying details of students.

In this application user can add record, search, modify and delete records and also reset the file. In addition to that, this application in C allows user to display fees along with dues.

For a specific operation user has to choose from specified options and then user can perform that operation. In the add student info, the name, admission ID and fees amount is asked. The admission ID need to be unique otherwise student with same admission ID will not be added.

Background to the study

Billing management is very common task for any school or institution which has a number of students. Though the method differs from organization to organization. The billing management system of earlier times had a manual system using ledger books to keep track of every single student fee history, calculate. This pen-paper based system is much time consuming and there is a great chance to make mistake as there are very good number of students in the organization and keeping patience is a tough job to manipulate so many things. So, such a system is time consuming, prone to errors of entry and analysis resulting from the fatigue of the users.

Now if we view the system from other student (who are end user of the system) point of view, then the system is also monotonous. Because if one student wants to check his/her statistics of the fees record then it is very difficult to get it without any help of automated system.

So, it is obvious to migrate the whole process in an automated way so that it helps the authority and user to maintain all the things with ease.

Literature Survey

Conceptual Literature

According to **Mostafa (2005)** stated, “Billing process includes receiving billing records determining the billing rates associated with the billing records, calculating the cost each billing record. Aggregating these records periodically to generate invoice, sending invoice to the customer the customer and collecting payments received from the customer. “This article shows the different process that the billing system must associate moreover it is also clear here that an invoice is essential between the customer and the company. It means that an invoice must be including to the billing system to be develop.

Conceptual Literature

According to **Sinsong (2009)**. “Sales invoice which is more popularly known as a “bill” is simply an itemized list of goods or services sold for which payment is due”. This statement about an invoice shows the relevance of it to the study. An invoice must be enforcing to the billing system because it serves as a proof between the concerned parties that a payment was due, benefits of Billing System.

Conceptual Literature

Patel (2000) stated that “small business billing system is there to help reduce errors that most generally annoy the customers “It is very obvious that the new system being develop for the PGA will surely give a lot of benefits.

LAN-Based Assessment and Billing System for Camiling

According to **Camiling College**, “LAN-based Assessment and Billing System for Camiling Colleges”, presents the possibilities of helping the cashier or person-in-charge by developing a system that enhances the existing manual system to satisfy their clients, the students.

Purpose

Main aim of this software is to provide easy and efficient functionalities for billing activities and also to provide full functional reports to management of institute with necessary details.

Nowadays large-scale organizations (school in case) are committed to bring the best way of management in the various forms. It is a tool to manage the inner operation of the organization related to student fees.

Benefits

- Interact with the software with menu-driven programs with user friendly interface.
- Manage student information efficiently.
- To provide easy and faster access to information.
- Prepare the detailed fees record of all the students in an institute.

Problem Statement

Nowadays schools and universities are committed to bring the best way of management in the various forms. In every part of the institution like library, accounts etc. there are need of software which must provide efficient and faster completion of work. Among the various methods of maintaining records of students most used once are maintaining ledgers or excel files which are commonly used in the organizations. These methods are time consuming and inefficient. Hence an alternative is needed. The alternative must need to have few objectives according to the requirements of the organization.

Objectives

- Develop a system that will improve the school process in the timekeeping and maintaining record of students.
- Develop a system that will monitor student data that is efficient to use.
- Secure the record of students and to have more manageable files, managed by administrator of the department.
- Calculate and update fees amount easily.
- Summarize all the details of students.
- Develop a system to provide easy and faster access to information.

User Defined Functions

- 1> `add_student` : The `add_student` function adds a student record on file. Student record consists of name of student, admission ID and monthly fee. The admission ID must be unique for each student.
- 2> `search_student` : The `search_student` function searches for a particular student record in the file. The function takes admission ID as an input and searches for the student with their unique admission ID.
- 3> `delete_student` : The `delete_student` function deletes a student record from the file. The function takes admission ID as an input and deletes the information of the particular student.
- 4> `update_student` : The `update_record` function check for the monthly fee status. It updates the fee with the late fee charges if there is any due.
- 5> `fee_payment` : The `fee_payment` function searches for a particular student record in the file with their admission ID and confirms their fee status and update it to zero if fee is paid.
- 6> `display_all_student` : The `display_all_student` function displays all records of the file in a proper format.
- 7> `read_from_file` : The `read_from_file` function opens the student file in read mode and add the student records to it using the user defined `add_student` function.
- 8> `write_to_file` : The `write_to_file` function opens the student file in write mode and prints the student record to the student file.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

void clrscr()
{
    system("@cls||clear");
    printf("\n      SCHOOL BILLING SYSTEM \n\n");
}

typedef struct node
{
    char name[20];
    int admn_no;
    int monthly_fee;
    long to_be_paid;
    struct node *next;
} NODE;

NODE *student_list = NULL;
NODE *search_student(NODE *, int, bool);

NODE *add_student(NODE *first, NODE data, bool print)
{
    clrscr();
    NODE *new_node, *temp = first;
    new_node = (NODE *)malloc(sizeof(NODE));
    strcpy(new_node->name, data.name);
    new_node->adm_n_no = data.admn_no;
    new_node->monthly_fee = data.monthly_fee;
    new_node->to_be_paid = data.to_be_paid;

    if (search_student(student_list, new_node->adm_n_no, false) == NULL)
    {
        if (first == NULL)
        {
            new_node->next = NULL;
            first = new_node;
        }

        else
        {
            NODE *prev_Node;
            while (temp != NULL)
            {
                prev_Node = temp;
            }
        }
    }
}
```

```

        temp = temp->next;
    }
    prev_Node->next = new_node;
    new_node->next = NULL;
}

if (print)
{
    clrscr();
    printf("\n Student added successfully!");
}

else
{
    clrscr();
    printf("\nStudent could not be added : The unique admission no. already exists in system
!");
}

return first;
}

NODE *search_student(NODE *first, int Temp_adminNo, bool print)
{
    NODE *temp = first;
    clrscr();

    while (temp != NULL)
    {
        if (temp->admn_no == Temp_adminNo)
        {
            if (print)
            {
                printf("\n\n !!! Student Found !!! \n  Name : %s\n  Admission no. :
%d\n  Monthly Fee : Rs. %d\n  Fee Dues : Rs. %ld\n", temp->name, temp->admn_no, temp-
>monthly_fee, temp->to_be_paid);
            }

            return temp;
        }
        temp = temp->next;
    }

    if (print == true)
        printf("\n\n !!! Student couldn't be found !!!");

    return NULL;
}

```

```

void fee_payment(NODE *first)
{
    clrscr();
    int TempAdminNo;
    char c;
    NODE *temp;
    printf("\n Fee Payment : \n  Enter the Admission no. : ");
    scanf("%d", &TempAdminNo);

    temp = search_student(student_list, TempAdminNo, true);
    while (temp != NULL)
    {
        if (temp->to_be_paid != 0)
        {
            getchar();
            printf("\n  Confirm the payment (Y/N) : ");
            scanf("%c", &c);

            if (c == 'Y' || c == 'y')
            {
                temp->to_be_paid = 0;
                printf("\n  !! Payment successful for %s :) !!", temp->name);
                break;
            }

            else if (c == 'N' || c == 'n')
            {
                printf("\n  !! Payment cancelled :( !!");
                break;
            }
        }

        else
        {
            printf("\n  No Fees due :)");
            break;
        }
    }
}

void update_record(NODE *first)
{
    clrscr();
    NODE *temp = first;
    while (temp != NULL)
    {
        if (temp->to_be_paid != 0)
        {

```

```

        temp->to_be_paid = temp->to_be_paid + 0.05 * temp->monthly_fee; // Adding Fine for
each month
    }

```

```

    else
        temp->to_be_paid = temp->monthly_fee;
        temp = temp->next;
    }
    printf("\n All records updated successfully :");
}

```

```

void display_all_student(NODE *first)
{

```

```

    clrscr();
    printf("\n All Records : ");
    NODE *temp = first;
    bool isEmpty = true;

    while (temp != NULL)
    {
        isEmpty = false;
        printf("\n\n   Name : %s\n   Admission no. : %d\n   Monthly Fee : Rs. %d\n   Fee Dues :
Rs. %ld\n", temp->name, temp->admn_no, temp->monthly_fee, temp->to_be_paid);

        temp = temp->next;
    }

    if (isEmpty == true)
        printf("   The list is Empty :");
}

```

```

NODE *delete_student(NODE *first, int Temp_adminNo)
{

```

```

    clrscr();
    NODE *prev_Node, *temp;
    bool deleted = false;
    temp = first;
    prev_Node = NULL;

    while (temp != NULL)
    {
        if (temp->admn_no == Temp_adminNo)
        {
            if (prev_Node == NULL)
            {
                printf("\n   !! Data of %s deleted successfully !!", temp->name);
                first = temp->next;
                free(temp);
            }

```

```

        else
        {
            printf("\n  !! Data of %s deleted successfully !!", temp->name);
            prev_Node->next = temp->next;
            free(temp);
        }
        deleted = true;
        break;
    }
    prev_Node = temp;
    temp = temp->next;
}

if (!deleted)
    printf("\n  !!! Data could not be deleted !!!");
return first;
}

void read_from_file(NODE *first)
{
    FILE *fp;
    NODE data;
    int length;
    fp = fopen("student_details", "r");
    printf("\n Loading Data from File..... ");

    if (fp != NULL)
    {
        while (!feof(fp))
        {
            fscanf(fp, "%d\t", &length);
            fgets(data.name, length + 1, fp);
            fscanf(fp, "\t%d\t%d\t%d\n", &data.admn_no, &data.monthly_fee, &data.to_be_paid);
            student_list = add_student(student_list, data, false);
        }
    }
    fclose(fp);
    clrscr();
}

void write_to_file(NODE *first)
{
    printf("\n Writing to file .... ");
    FILE *fp;
    NODE *temp;
    temp = first;
    fp = fopen("student_details", "w");

```

```

while (temp != NULL)
{
    fprintf(fp, "%d\t%s\t%d\t%d\t%d\n", (int)strlen(temp->name), temp->name, temp-
>admn_no, temp->monthly_fee, temp->to_be_paid);
    temp = temp->next;
}
fclose(fp);
clrscr();
}

int main()
{
    clrscr();
    read_from_file(student_list);
    NODE data;
    int c;
    int Temp_adminNo;

    while (1)
    {
        printf("\n MENU : \n 1 > Add Student Info\n 2 > Search Record\n 3 > Delete Student
Info\n 4 > Update All Records Payment Amount\n 5 > Fee Payment\n 6 > Display All
Records\n 7 > Reset the File \n 8 > Save and Exit\n\n Choice : ");
        scanf("%d", &c);

        switch (c)
        {
            case 1:
                printf("\nEnter Student Info : \n   Name : ");
                getchar();
                gets(data.name);
                printf("   Admission No. : ");
                scanf("%d", &data.admn_no);
                printf("   Monthly Fee : ");
                scanf("%d", &data.monthly_fee);
                data.to_be_paid = 0;
                data.next = NULL;
                student_list = add_student(student_list, data, true);
                break;

            case 2:
                printf("\nStudent Search : \n   Enter Admission No. of student to search : ");
                scanf("%d", &Temp_adminNo);
                search_student(student_list, Temp_adminNo, true);
                break;

            case 3:
                printf("\nStudent Search : \n   Enter Admission No. of student to delete : ");
                scanf("%d", &Temp_adminNo);

```

```

    student_list = delete_student(student_list, Temp_adminNo);
    break;

case 4:
    update_record(student_list);
    break;

case 5:
    fee_payment(student_list);
    break;

case 6:
    display_all_student(student_list);
    break;

case 7:
    printf("\n Existing file deleted <-_->");
    NODE *temp = student_list;
    while (temp != NULL)
    {
        NODE *prev_node = temp;
        temp = temp->next;
        free(prev_node);
    }
    student_list = NULL;
    remove("student_details");
    break;

case 8:
    write_to_file(student_list);
    exit(0);
    break;

default:
    printf(" !!! Invalid Input !!!");
}
}
}

```


Methodology

User will have 8 different choices as shown below:

MENU:

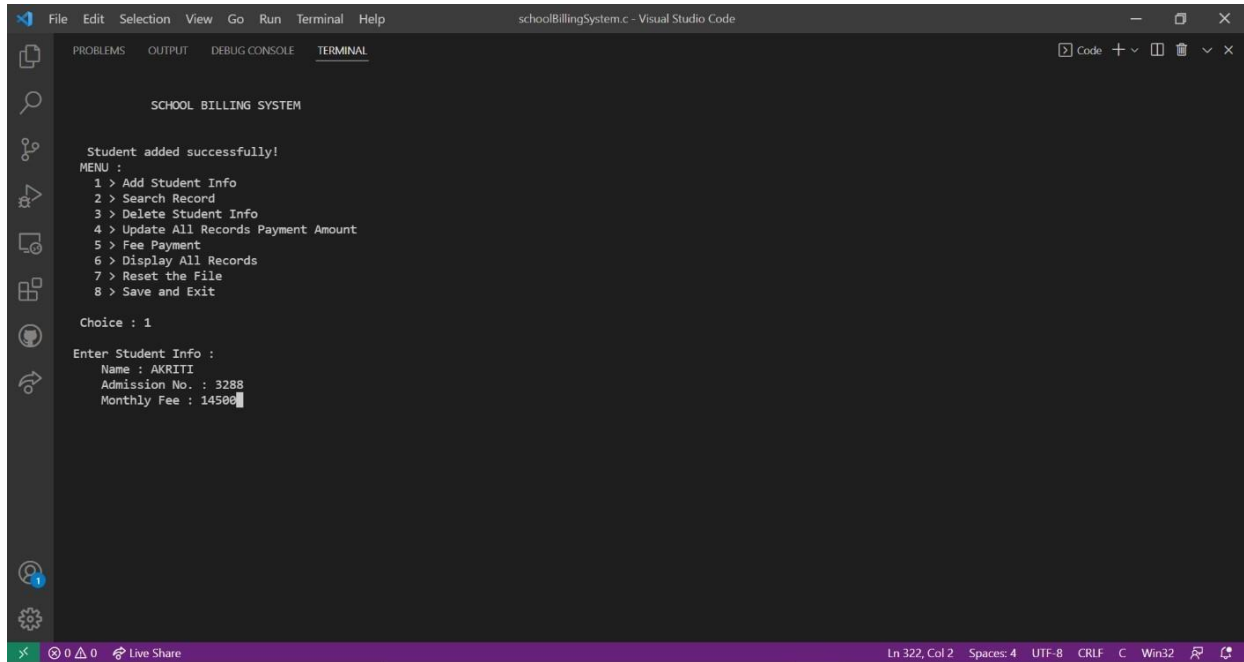
- 1 > Add Student Info
- 2 > Search Record
- 3 > Delete Student Info
- 4 > Update All Records Payment Amount
- 5 > Fee Payment
- 6 > Display All Records
- 7 > Reset the File
- 8 > Save and Exit

Choice:

User will have to choose one of the choices provided and application will provide the service accordingly. User can add, search, remove, update, display, reset and save a student fee information at any time easily.

Working

Adding a new student



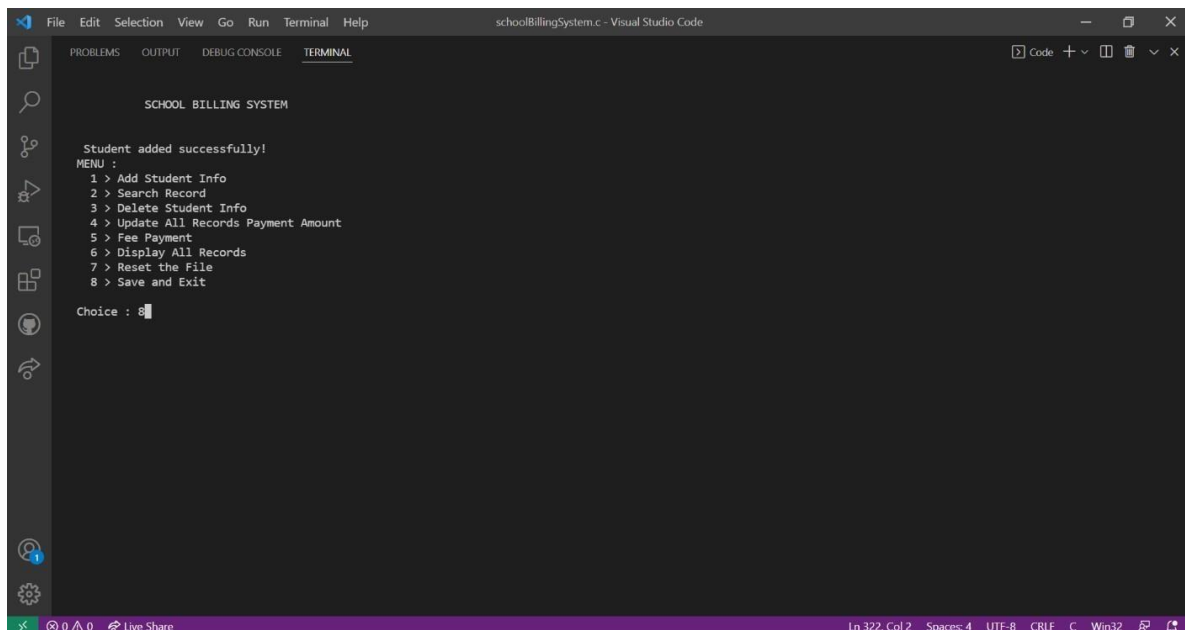
```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM

Student added successfully!
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 1

Enter Student Info :
Name : AKRITI
Admission No. : 3288
Monthly Fee : 14580
```

Saving student record to the file “student_details”

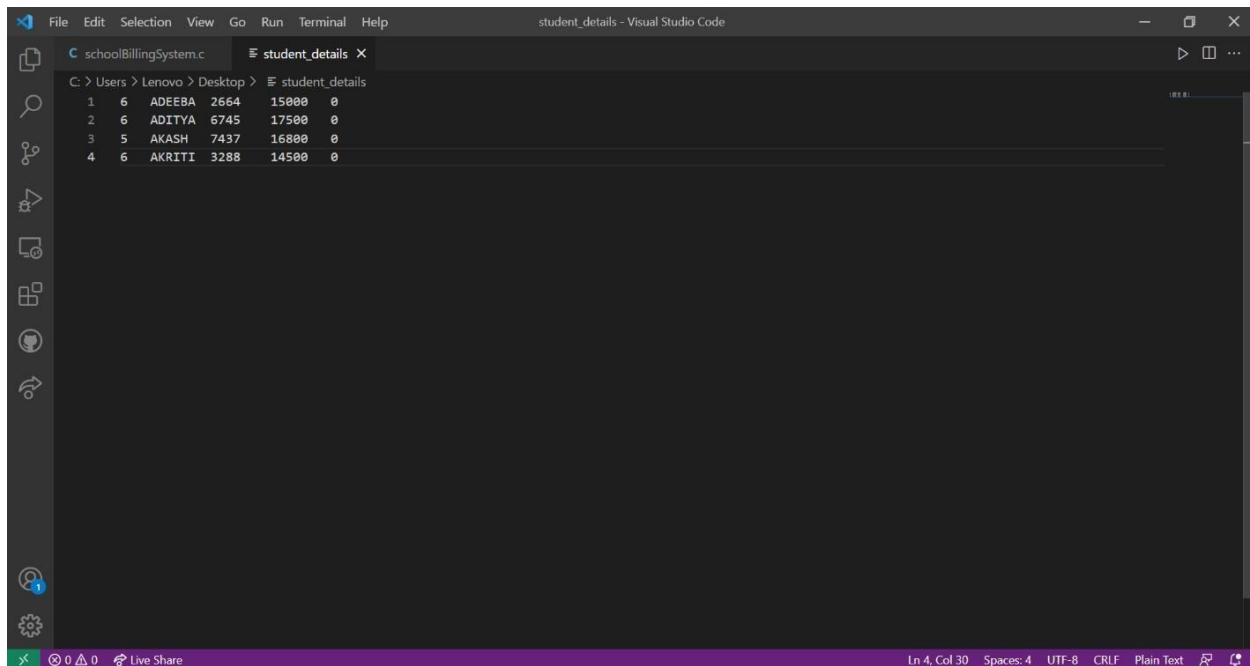


```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM

Student added successfully!
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 8
```

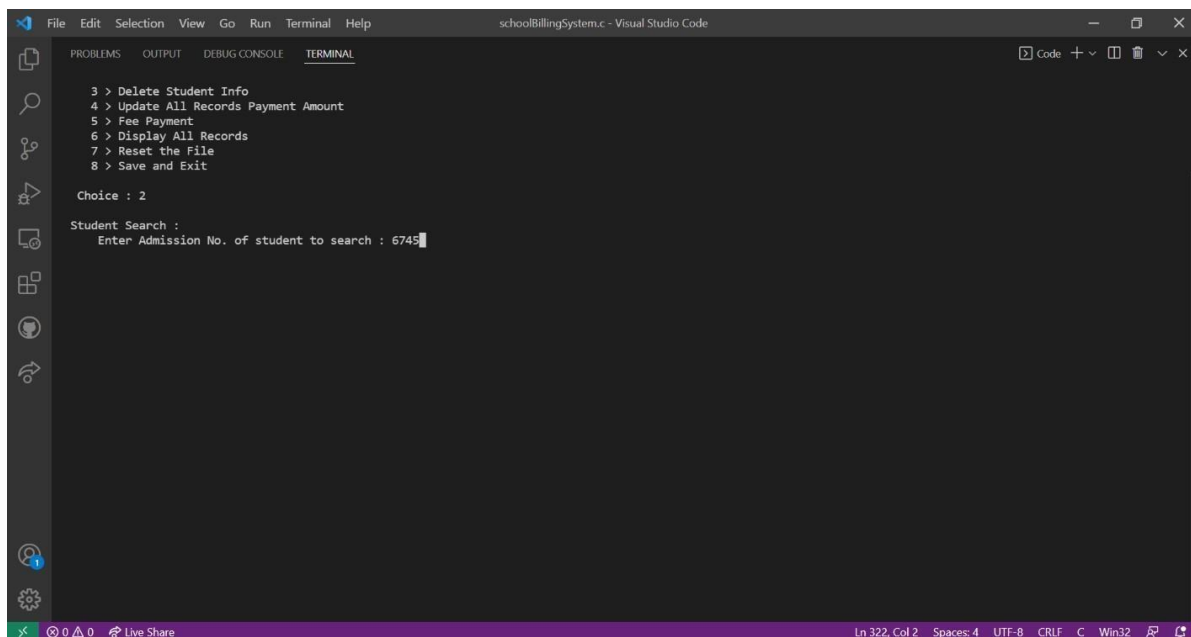
“student_details” File



```
File Edit Selection View Go Run Terminal Help
student_details - Visual Studio Code
C: > Users > Lenovo > Desktop > student_details
1 6 ADEEBA 2664 15000 0
2 6 ADITYA 6745 17500 0
3 5 AKASH 7437 16800 0
4 6 AKRITI 3288 14500 0
```

Ln 4, Col 30 Spaces: 4 UTF-8 CRLF Plain Text

Searching a student record with specific admission ID



```
File Edit Selection View Go Run Terminal Help
schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 2

Student Search :
Enter Admission No. of student to search : 6745
```

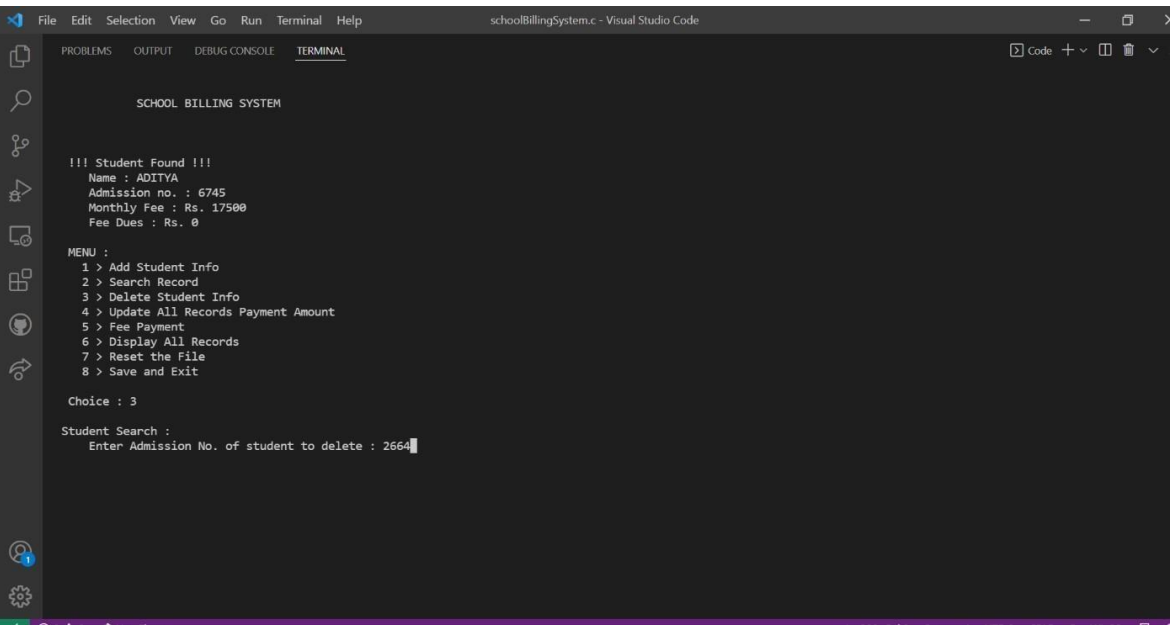
Ln 322, Col 2 Spaces: 4 UTF-8 CRLF C Win32

The image shows a Visual Studio Code editor window with a terminal running a C program. The title bar indicates the file is 'schoolBillingSystem.c - Visual Studio Code'. The terminal output is as follows:

```
SCHOOL BILLING SYSTEM  
  
!!! Student Found !!!  
Name : ADITYA  
Admission no. : 6745  
Monthly Fee : Rs. 17500  
Fee Dues : Rs. 0  
  
MENU :  
1 > Add Student Info  
2 > Search Record  
3 > Delete Student Info  
4 > Update All Records Payment Amount  
5 > Fee Payment  
6 > Display All Records  
7 > Reset the File  
8 > Save and Exit  
  
Choice : 
```


The Visual Studio Code interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help), a sidebar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions, and a bottom status bar showing 'Ln 322, Col 2', 'Spaces: 4', 'UTF-8', 'CRLF', 'C', 'Win32', and a 'Live Share' icon.

Deleting a student information from “student_details” File



```
File Edit Selection View Go Run Terminal Help
schoolBillingSystem.c - Visual Studio Code

SCHOOL BILLING SYSTEM

!!! Student Found !!!
Name : ADITYA
Admission no. : 6745
Monthly Fee : Rs. 17500
Fee Dues : Rs. 0

MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 3

Student Search :
Enter Admission No. of student to delete : 2664
```

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal output displays the 'SCHOOL BILLING SYSTEM' menu. A message indicates that data for 'ADEEBA' has been deleted successfully. The menu options are listed, and the user has entered '4' as their choice, which corresponds to 'Update All Records Payment Amount'. The status bar at the bottom shows the cursor is at line 322, column 2.

```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

SCHOOL BILLING SYSTEM

!! Data of ADEEBA deleted successfully !!
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 4
```

Ln 322, Col 2 Spaces: 4 UTF-8 CRLF C Win32

Updating all records payment amount

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal output displays the 'SCHOOL BILLING SYSTEM' menu. A message indicates that all records have been updated successfully. The menu options are listed, and the user has entered '6' as their choice, which corresponds to 'Display All Records'. The status bar at the bottom shows the cursor is at line 322, column 2.

```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

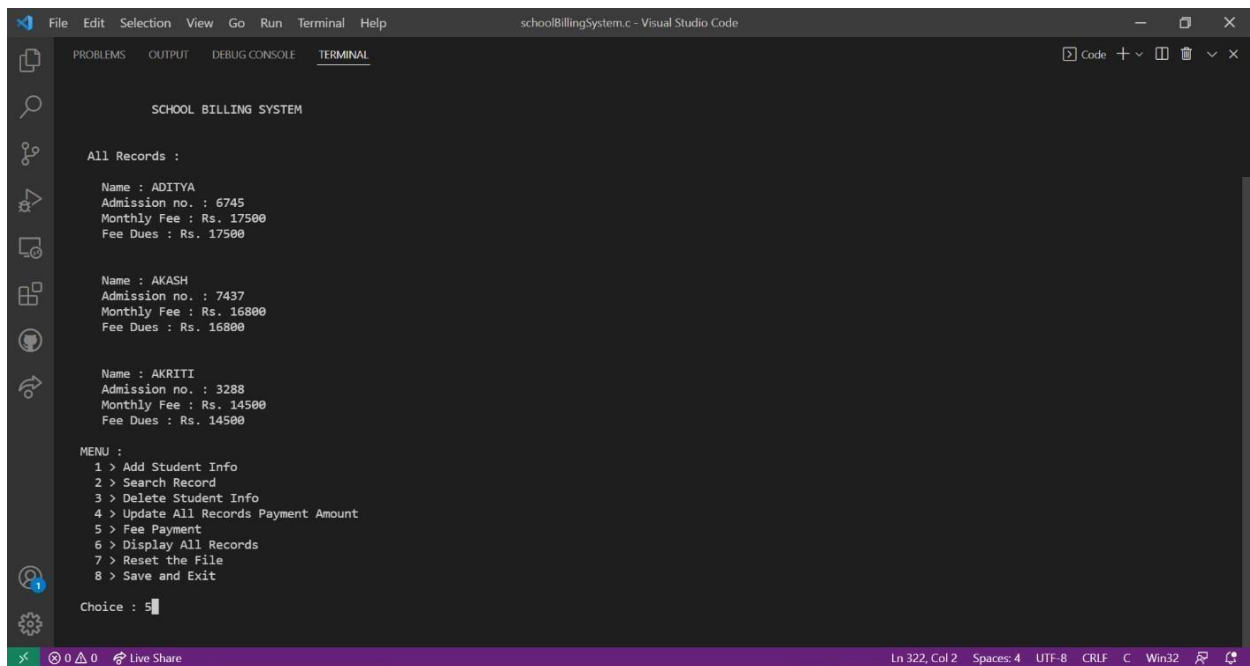
SCHOOL BILLING SYSTEM

All records updated successfully :)
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 6
```

Ln 322, Col 2 Spaces: 4 UTF-8 CRLF C Win32

Displaying all records



The screenshot shows a Visual Studio Code window with a terminal running a C program. The program is titled 'SCHOOL BILLING SYSTEM'. It displays 'All Records :' and lists three students: ADITYA (Admission no. 6745, Monthly Fee Rs. 17500, Fee Dues Rs. 17500), AKASH (Admission no. 7437, Monthly Fee Rs. 16800, Fee Dues Rs. 16800), and AKRITI (Admission no. 3288, Monthly Fee Rs. 14500, Fee Dues Rs. 14500). Below the records, a 'MENU :' is shown with options 1 to 8. Option 5 is 'Fee Payment', and option 6 is 'Display All Records'. The prompt 'Choice : 5' is visible at the bottom.

```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM

All Records :

Name : ADITYA
Admission no. : 6745
Monthly Fee : Rs. 17500
Fee Dues : Rs. 17500

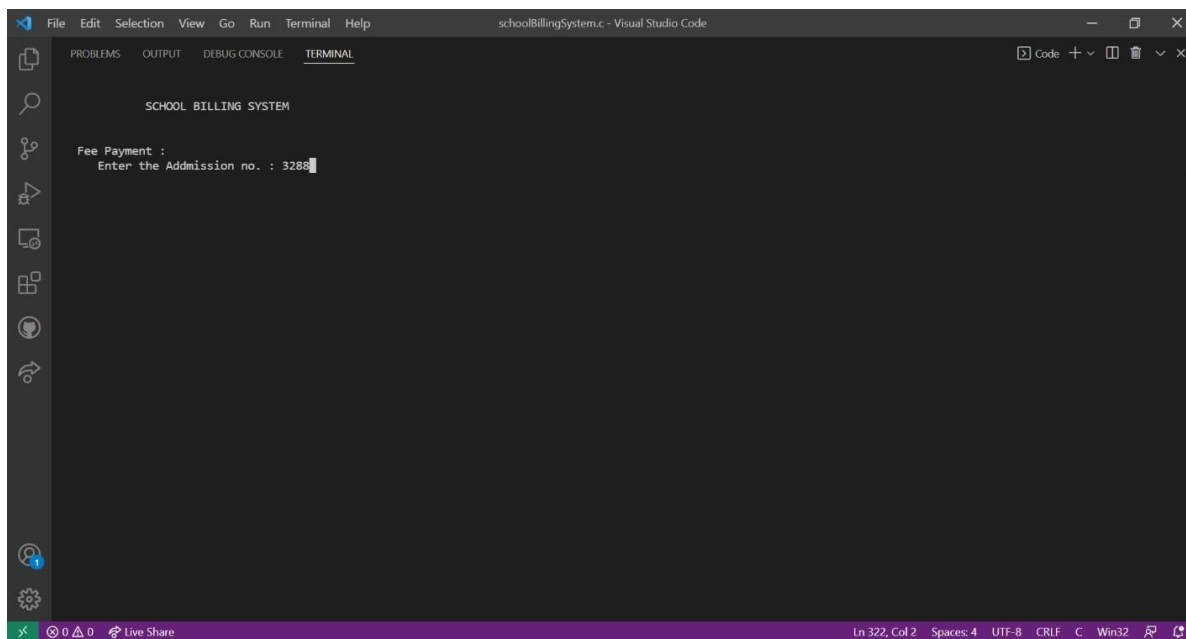
Name : AKASH
Admission no. : 7437
Monthly Fee : Rs. 16800
Fee Dues : Rs. 16800

Name : AKRITI
Admission no. : 3288
Monthly Fee : Rs. 14500
Fee Dues : Rs. 14500

MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 5
```

Fee payment



The screenshot shows the same Visual Studio Code window. The terminal now displays 'Fee Payment :' and prompts the user to 'Enter the Admission no. : 3288'. The cursor is positioned after the number 3288.

```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM

Fee Payment :
Enter the Admission no. : 3288
```

```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

SCHOOL BILLING SYSTEM

!!! Student Found !!!
Name : AKRITI
Admission no. : 3288
Monthly Fee : Rs. 14500
Fee Dues : Rs. 14500

Confirm the payment (Y/N) : Y

!! Payment successful for AKRITI :) !!

MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 8
```

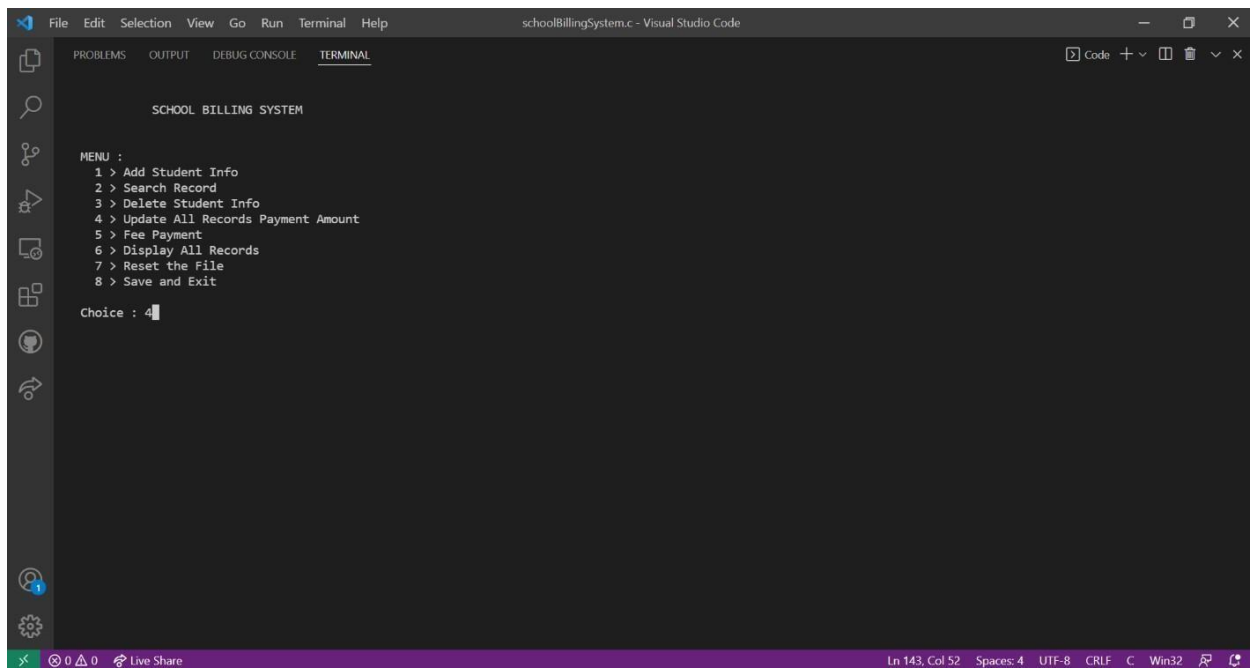
Updated student_details File

```
File Edit Selection View Go Run Terminal Help student_details - Visual Studio Code

C schoolBillingSystem.c student_details x

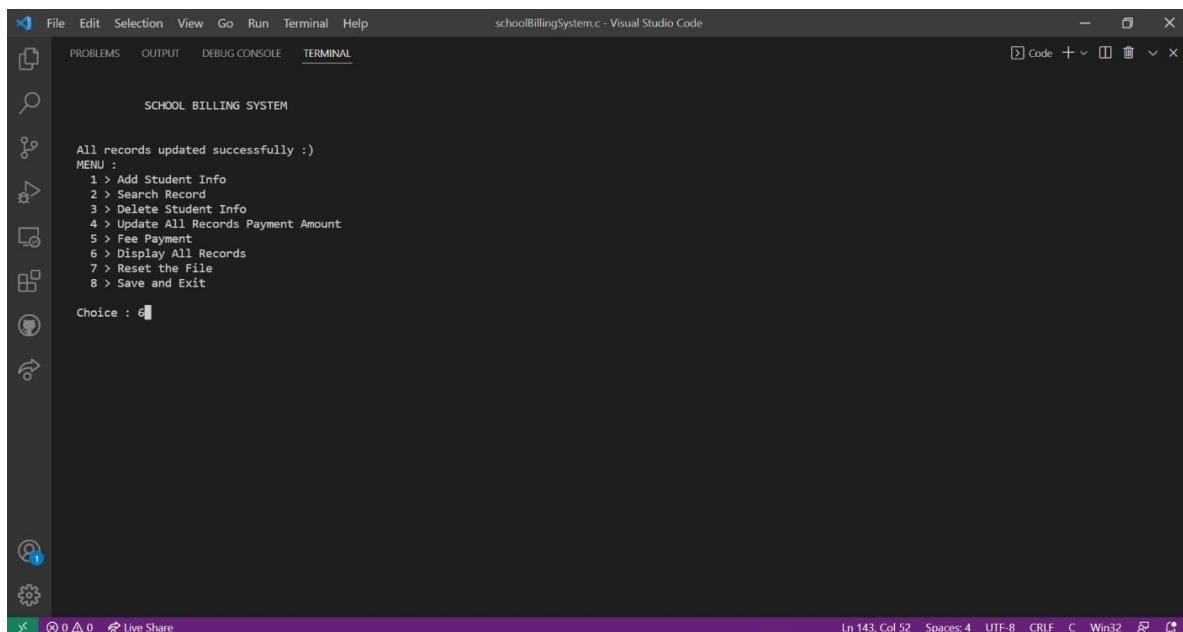
C:\Users\Lenovo\Desktop> student_details
1 6 ADITYA 6745 17500 17500
2 5 AKASH 7437 16800 16800
3 6 AKRITI 3288 14500 0
```

Updating the students fee with late fee charges if there is any due



```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit
Choice : 4
```

Updated record



```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM
All records updated successfully :)
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit
Choice : 6
```



```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM

All Records :

Name : ADITYA
Admission no. : 6745
Monthly Fee : Rs. 17500
Fee Dues : Rs. 35875

Name : AKASH
Admission no. : 7437
Monthly Fee : Rs. 16800
Fee Dues : Rs. 34440

Name : AKRITI
Admission no. : 3288
Monthly Fee : Rs. 14500
Fee Dues : Rs. 14500

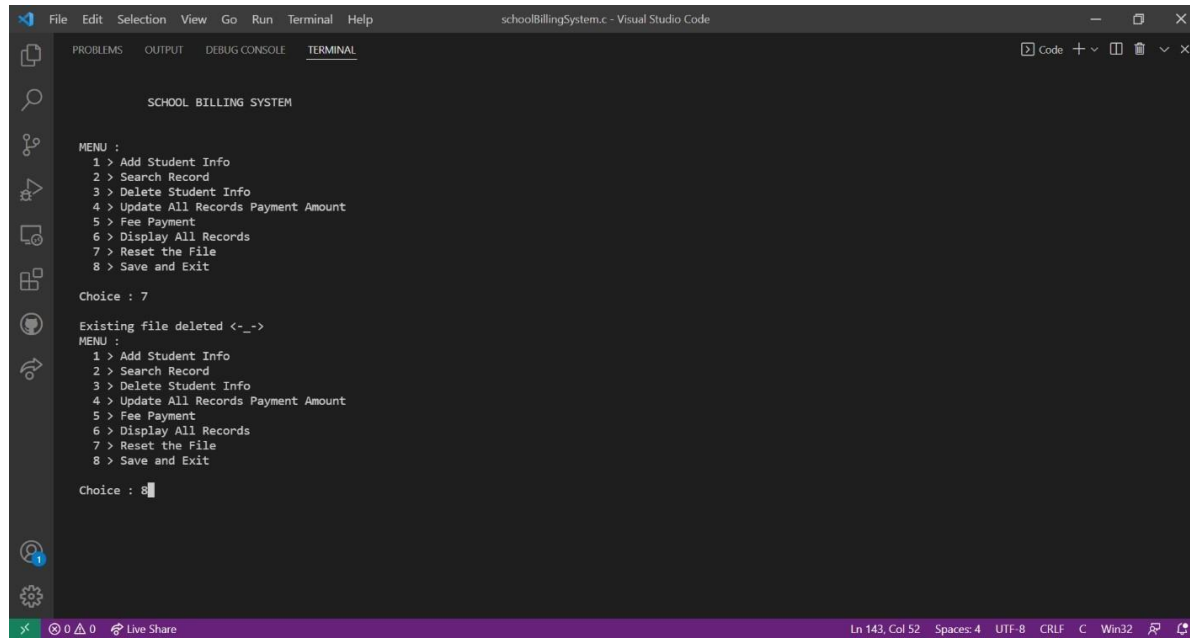
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : █
```

“student_details” File

```
File Edit Selection View Go Run Terminal Help student_details - Visual Studio Code
C schoolBillingSystem.c student_details x
C:\Users\Lenovo\Desktop> student_details
1 6 ADITYA 6745 17500 35875
2 5 AKASH 7437 16800 34440
3 6 AKRITI 3288 14500 14500
```

Reset student_details File



```
File Edit Selection View Go Run Terminal Help schoolBillingSystem.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SCHOOL BILLING SYSTEM

MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 7

Existing file deleted <-_->
MENU :
1 > Add Student Info
2 > Search Record
3 > Delete Student Info
4 > Update All Records Payment Amount
5 > Fee Payment
6 > Display All Records
7 > Reset the File
8 > Save and Exit

Choice : 8
```

Conclusion

The School Billing System is a novel and interesting application. Unlike those commonly used methods of maintaining hardcopy of details and maintaining those records. User can interact with the software with menu-driven programs with user friendly interface. These properties allow our model to be accepted widely in schools and universities. It makes the work easy and efficient. The proposed software will cover many aspects of time keeping and fee process. The system has a file management where it covers the records of students and a good amount of data can be stored in the files. The system also covers the tracking of fees of students. A record can be added or deleted easily due to linked list data structure implementation. Such advantages of this software have been verified empirically in various tasks including fee dues detection and fine imposition. The system is reliable and has a quick response time. The application can be used everywhere and it will reduce doing the same work over and over. The software has an effective and efficient way of monitoring their students to give a higher quality of service.

Reference

[Narasimha Karumanchi] "Data Structures and Algorithms Made Easy".

[S.K.Srivastava, Deepali Srivastava] "C in Depth", 3rd Edition.

[geeksforgeeks.org]

“ <https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/> “

“ <https://www.geeksforgeeks.org/basics-file-handling-c/> “

“ <https://www.geeksforgeeks.org/structures-c/> “.

“ <https://www.geeksforgeeks.org/data-structures/linked-list/> “