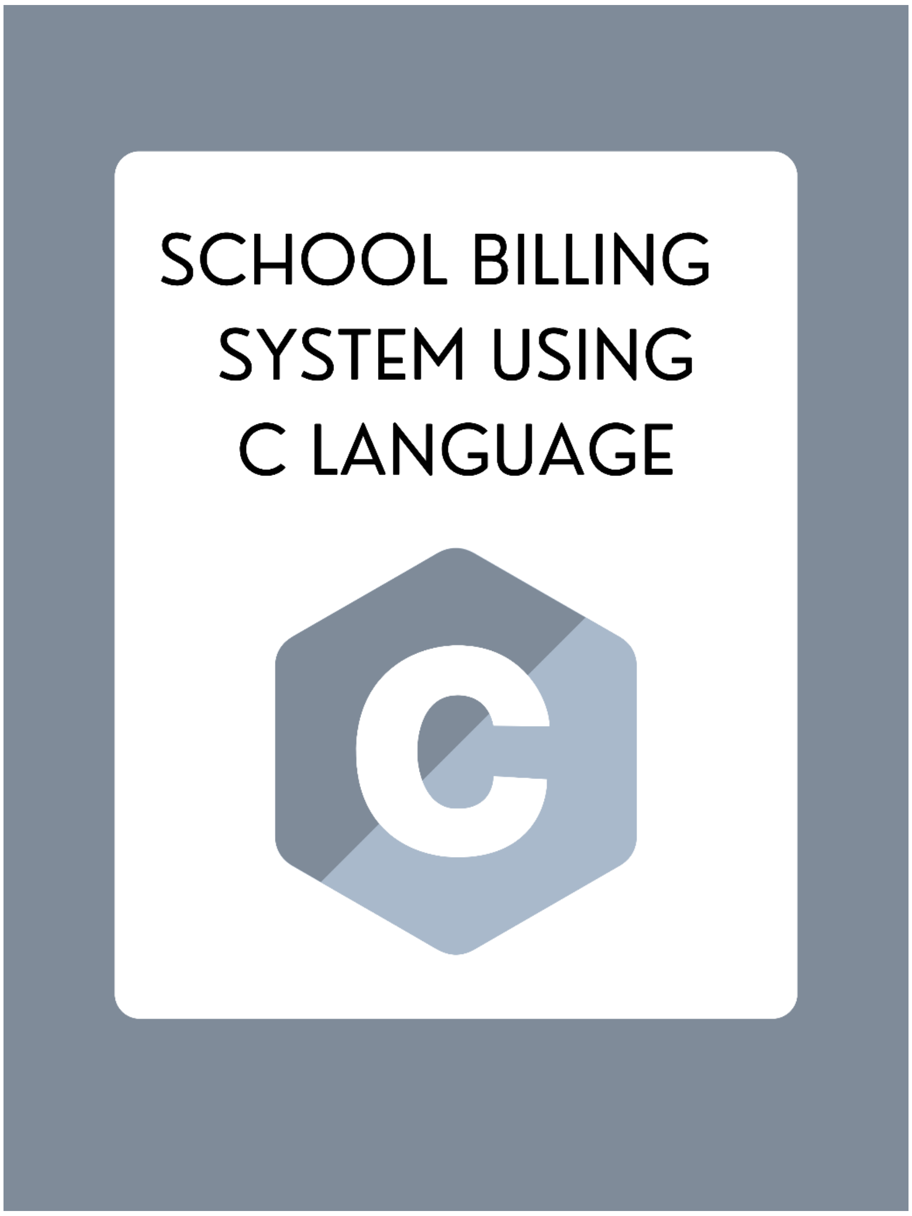
# <u>Technical Training Project</u>

**Team Members:**

1. *Mihir Jataniya*
2. *Jathin Reddy*
3. *Charith Shetty*
4. *Aditya Kamath*
5. *Achal M Jain*



# School Billing And Record System

## Table of Contents

## 1. Abstract

This document provides documentation for the School Billing System, a C program designed to manage student and employee accounts for a school. It includes an introduction, details about the technologies used, system architecture, and code documentation. Beyond traditional record-keeping, the system offers an efficient, error-reducing, and user-friendly platform that seeks to simplify the financial and personnel management tasks faced by schools. This software project, implemented in the C programming language, addresses these challenges by offering a reliable and automated solution.

## 2. Introduction

### 2.1 Background

The School Billing System is a software project created to simplify the management of student and employee accounts within a school. The project aims to automate and improve the efficiency of tasks related to fee tracking and personnel management. By transitioning from cumbersome manual processes to an automated platform, schools can enhance financial accuracy, reduce administrative burden, and ultimately focus on their core mission of providing quality education.

### 2.2 Objectives

The primary objectives of the School Billing System project are:

- Provide a user-friendly interface for administrators to manage student and employee accounts.

- Enhance data accuracy and reduce errors in fee tracking and employee salary management.

- Enable efficient retrieval of student and employee account information.

- Implement error handling and data validation for input.

## 3. Technologies Used

The School Billing System is developed using the following technologies:

- Programming Language: C

- Data Storage: Arrays

- User Interface: Command-line interface (CLI)

## 4. System Architecture

The system architecture of the School Billing System is structured as follows:

- **Main Functionality**: The core of the system containing menu-driven operations for managing student and employee accounts.

- **Data Storage:** Arrays are used to efficiently store student and employee account data.

- **Functions:** A collection of functions to perform various operations on accounts, including insertion, modification, searching, and deletion.

- **User Interface:** A command-line interface (CLI) provides an interactive menu for user interaction.

- **Error Handling:** The system includes error-handling mechanisms to gracefully handle invalid inputs, preventing crashes or incorrect data processing.

## 5. Code Documentation

### 5.1. Structures

The program defines the following structures for student and employee accounts:

- struct StudentAccount: Represents student account data, including student ID, name, fees paid, and fees due.

- struct EmployeeAccount: Represents employee account data, including employee ID, name, and salary.

### 5.2. Arrays

- struct StudentAccount students[MAX_ACCOUNTS]: Array to store student accounts.

- struct EmployeeAccount employees[MAX_ACCOUNTS]: Array to store employee accounts.

## 5.3. Functions

### 5.3.1. Student Account Functions

- void insertStudent(): Inserts a new student account.

- void modifyStudent(int studentID): Modifies an existing student account.

- void searchStudent(int studentID): Searches for a student account.

- void deleteStudent(int studentID): Deletes a student account.

### 5.3.2. Employee Account Functions

- void insertEmployee(): Inserts a new employee account.

- void modifyEmployee(int employeeID): Modifies an existing employee account.

- void searchEmployee(int employeeID): Searches for an employee account.

- void deleteEmployee(int employeeID): Deletes an employee account.

# 6. Sample Usage

Here are some examples of how to use the program:

**Step 1: Insert a Student Account:**

Select option 1 from the menu.

Enter student information as prompted.

**Step 2: Modify a Student Account**:

Select option 2 from the menu.

Enter the Student ID to modify.

Enter the new fee information.

**Step 3: Search for a Student Account:**

Select option 3 from the menu.

Enter the Student ID to search for.

**Step 4: Delete a Student Account:**

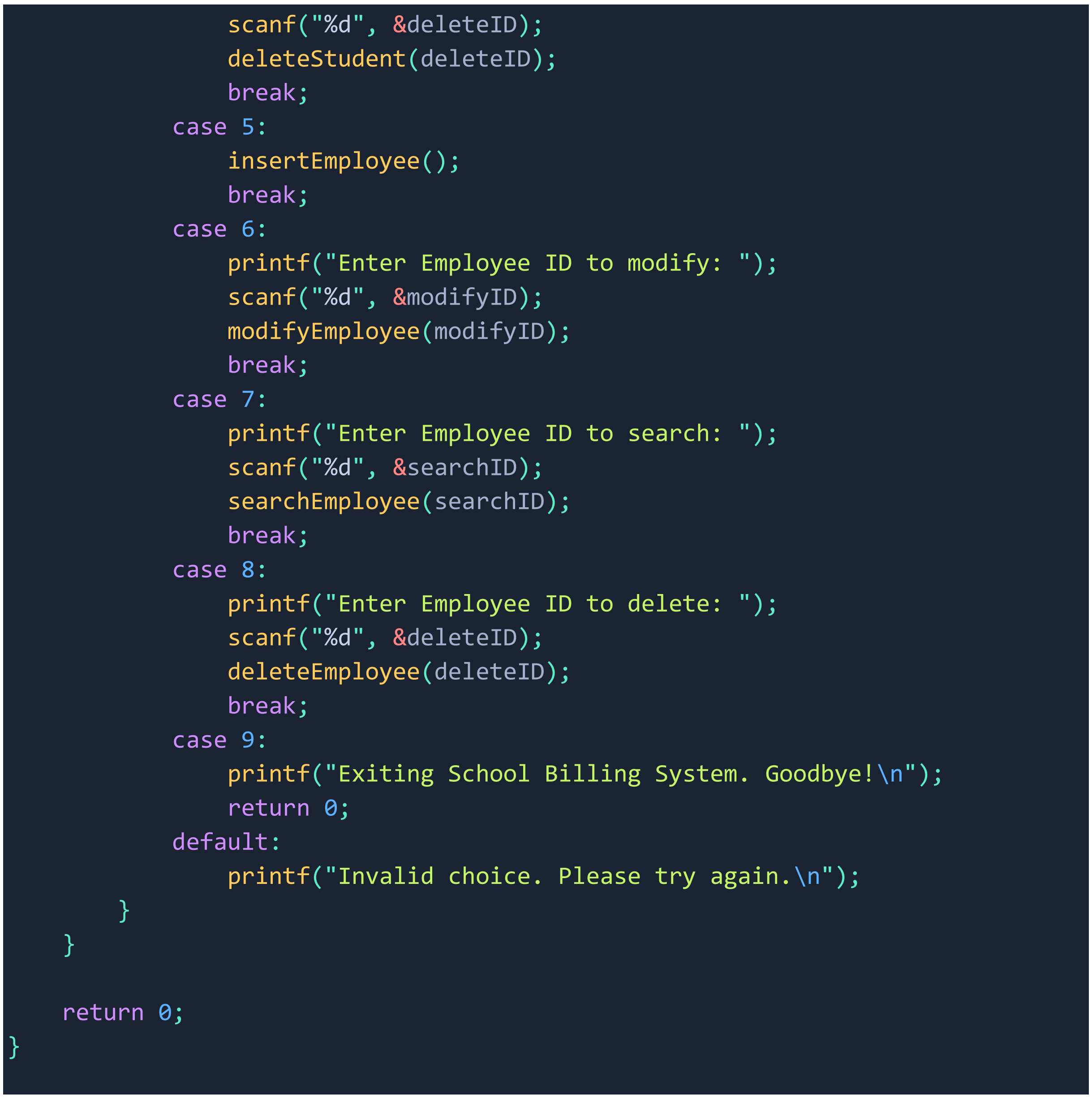
Select option 4 from the menu.

Enter the Student ID to delete.

(Repeat the above steps for employee accounts, using options 5-8.)

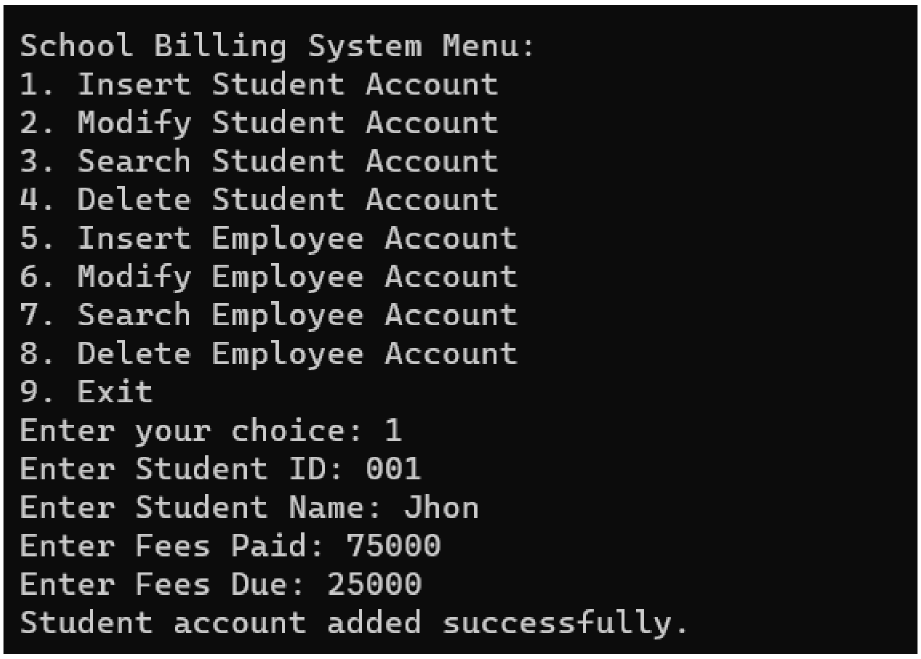
## 7. Appendix

```c
#include <stdio.h>
#include <string.h>

#define MAX_ACCOUNTS 100

struct StudentAccount {
    int studentID;
    char name[50];
    float feesPaid;
    float feesDue;
};


struct EmployeeAccount {
    int employeeID;
    char name[50];
    float salary;
};


struct StudentAccount students[MAX_ACCOUNTS];
struct EmployeeAccount employees[MAX_ACCOUNTS];

int numStudents = 0;
int numEmployees = 0;

// Function to insert a new student account
void insertStudent() {
    if (numStudents < MAX_ACCOUNTS) {
        struct StudentAccount newStudent;
        printf("Enter Student ID: ");
        scanf("%d", &newStudent.studentID);
        printf("Enter Student Name: ");
        scanf("%s", newStudent.name);
        printf("Enter Fees Paid: ");
        scanf("%f", &newStudent.feesPaid);
        printf("Enter Fees Due: ");
        scanf("%f", &newStudent.feesDue);

        students[numStudents++] = newStudent;
        printf("Student account added successfully.\n");
    } else {
        printf("Maximum number of student accounts reached.\n");
    }
}

void modifyStudent(int studentID) {
    for (int i = 0; i < numStudents; i++) {
        if (students[i].studentID == studentID) {
            printf("Enter new Fees Paid: ");
            scanf("%f", &students[i].feesPaid);
            printf("Enter new Fees Due: ");
            scanf("%f", &students[i].feesDue);
            printf("Student account modified successfully.\n");
            return;
        }
    }
    printf("Student with ID %d not found.\n", studentID);
}

void searchStudent(int studentID) {
    for (int i = 0; i < numStudents; i++) {
        if (students[i].studentID == studentID) {
            printf("Student ID: %d\n", students[i].studentID);
            printf("Student Name: %s\n", students[i].name);
            printf("Fees Paid: %.2f\n", students[i].feesPaid);
            printf("Fees Due: %.2f\n", students[i].feesDue);
            return;
        }
    }
    printf("Student with ID %d not found.\n", studentID);
}

void deleteStudent(int studentID) {
    for (int i = 0; i < numStudents; i++) {
        if (students[i].studentID == studentID) {
            for (int j = i; j < numStudents - 1; j++) {
                students[j] = students[j + 1];
            }
            numStudents--;
            printf("Student account deleted successfully.\n");
            return;
        }
    }
    printf("Student with ID %d not found.\n", studentID);
}

void insertEmployee() {
    if (numEmployees < MAX_ACCOUNTS) {
        struct EmployeeAccount newEmployee;
        printf("Enter Employee ID: ");
        scanf("%d", &newEmployee.employeeID);
        printf("Enter Employee Name: ");
        scanf("%s", newEmployee.name);
        printf("Enter Salary: ");
        scanf("%f", &newEmployee.salary);

        employees[numEmployees++] = newEmployee;
        printf("Employee account added successfully.\n");
    } else {
        printf("Maximum number of employee accounts reached.\n");
    }
}

void modifyEmployee(int employeeID) {
    for (int i = 0; i < numEmployees; i++) {
        if (employees[i].employeeID == employeeID) {
            printf("Enter new Salary: ");
            scanf("%f", &employees[i].salary);
            printf("Employee account modified successfully.\n");
            return;
        }
    }
    printf("Employee with ID %d not found.\n", employeeID);
}

void searchEmployee(int employeeID) {
    for (int i = 0; i < numEmployees; i++) {
        if (employees[i].employeeID == employeeID) {
            printf("Employee ID: %d\n", employees[i].employeeID);
            printf("Employee Name: %s\n", employees[i].name);
            printf("Salary: %.2f\n", employees[i].salary);
            return;
        }
    }
    printf("Employee with ID %d not found.\n", employeeID);
}

void deleteEmployee(int employeeID) {
    for (int i = 0; i < numEmployees; i++) {
        if (employees[i].employeeID == employeeID) {
            for (int j = i; j < numEmployees - 1; j++) {
                employees[j] = employees[j + 1];
            }
            numEmployees--;
            printf("Employee account deleted successfully.\n");
            return;
        }
    }
    printf("Employee with ID %d not found.\n", employeeID);
}

int main() {
    int choice;

    while (1) {
        printf("\nSchool Billing System Menu:\n");
        printf("1. Insert Student Account\n");
        printf("2. Modify Student Account\n");
        printf("3. Search Student Account\n");
        printf("4. Delete Student Account\n");
        printf("5. Insert Employee Account\n");
        printf("6. Modify Employee Account\n");
        printf("7. Search Employee Account\n");
        printf("8. Delete Employee Account\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertStudent();
                break;
            case 2:
                printf("Enter Student ID to modify: ");
                int modifyID;
                scanf("%d", &modifyID);
                modifyStudent(modifyID);
                break;
            case 3:
                printf("Enter Student ID to search: ");
                int searchID;
                scanf("%d", &searchID);
                searchStudent(searchID);
                break;
            case 4:
                printf("Enter Student ID to delete: ");
                int deleteID;
```

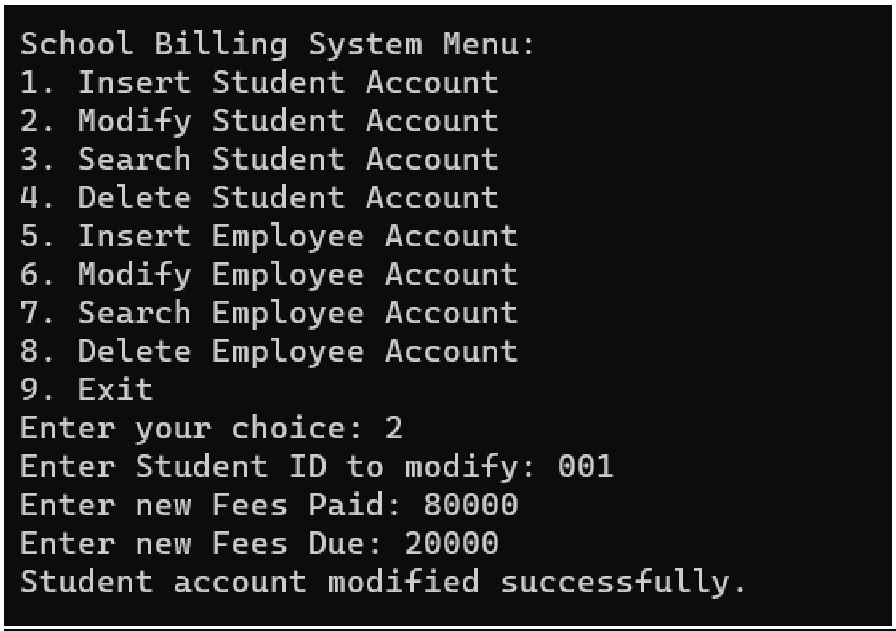


```c
                scanf("%d", &deleteID);
                deleteStudent(deleteID);
                break;
            case 5:
                insertEmployee();
                break;
            case 6:
                printf("Enter Employee ID to modify: ");
                scanf("%d", &modifyID);
                modifyEmployee(modifyID);
                break;
            case 7:
                printf("Enter Employee ID to search: ");
                scanf("%d", &searchID);
                searchEmployee(searchID);
                break;
            case 8:
                printf("Enter Employee ID to delete: ");
                scanf("%d", &deleteID);
                deleteEmployee(deleteID);
                break;
            case 9:
                printf("Exiting School Billing System. Goodbye!\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

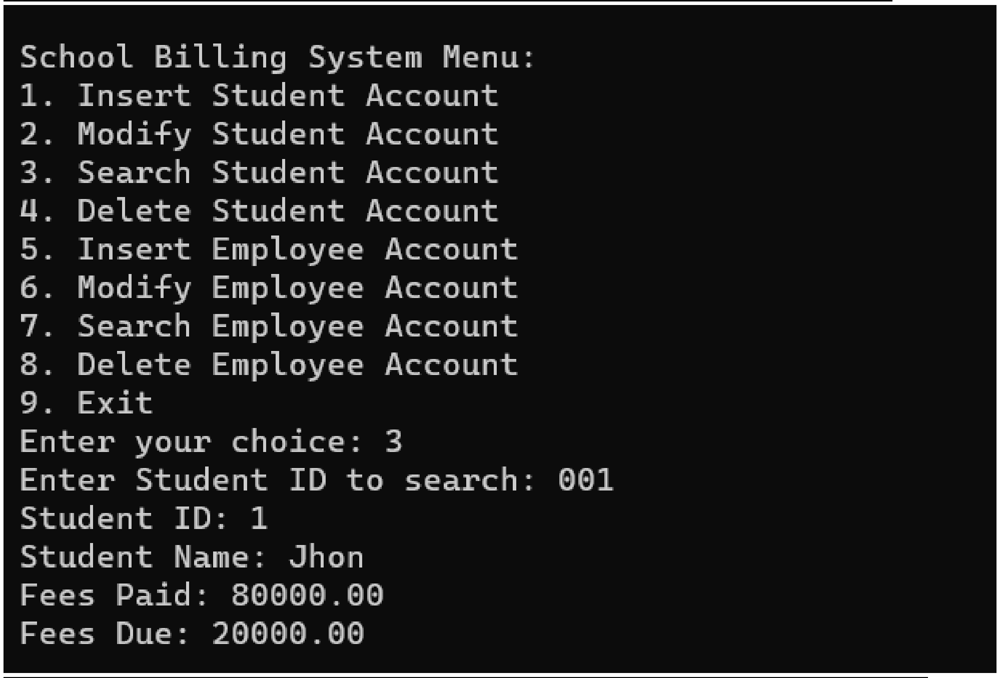
**OUTPUT :**



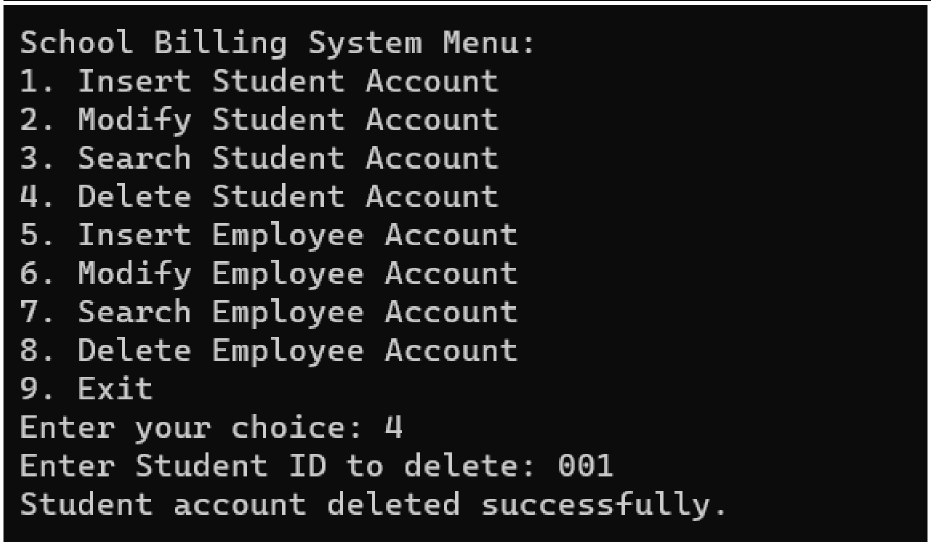
```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 1
Enter Student ID: 001
Enter Student Name: Jhon
Enter Fees Paid: 75000
Enter Fees Due: 25000
Student account added successfully.
```



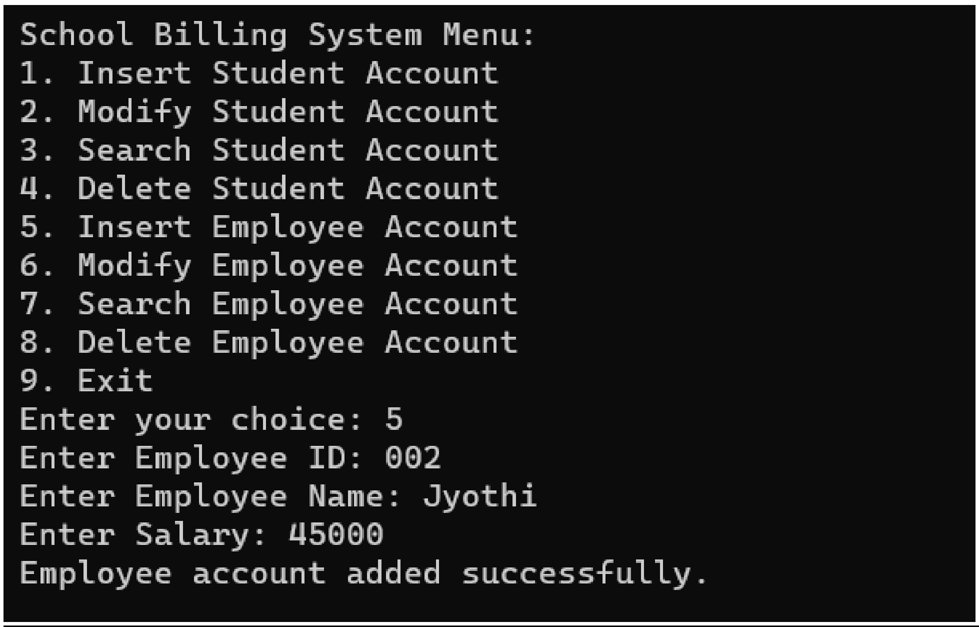
```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 2
Enter Student ID to modify: 001
Enter new Fees Paid: 80000
Enter new Fees Due: 20000
Student account modified successfully.
```



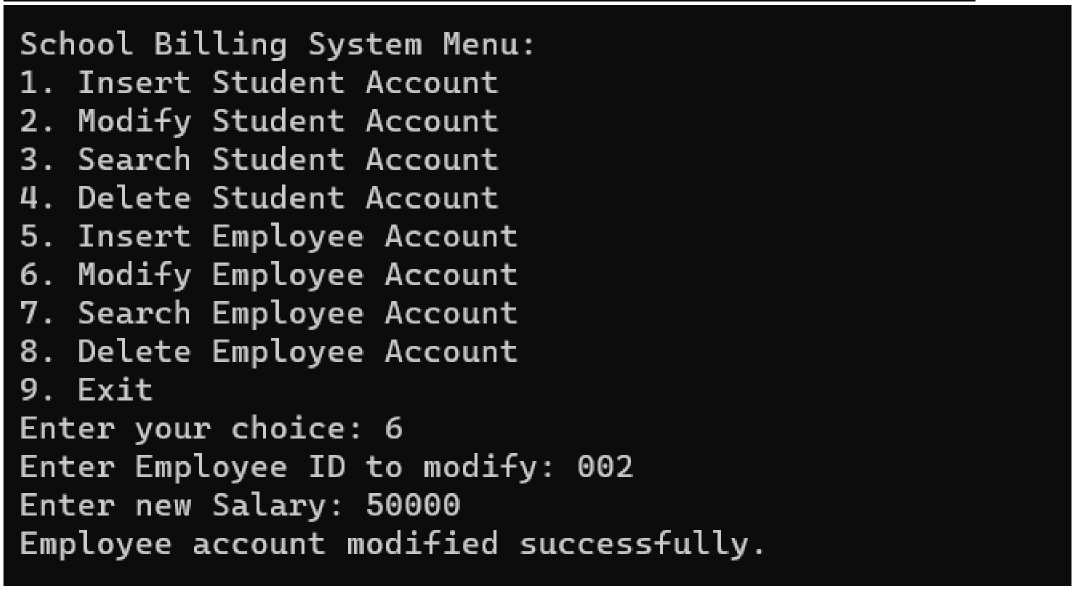
```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 3
Enter Student ID to search: 001
Student ID: 1
Student Name: Jhon
Fees Paid: 80000.00
Fees Due: 20000.00
```



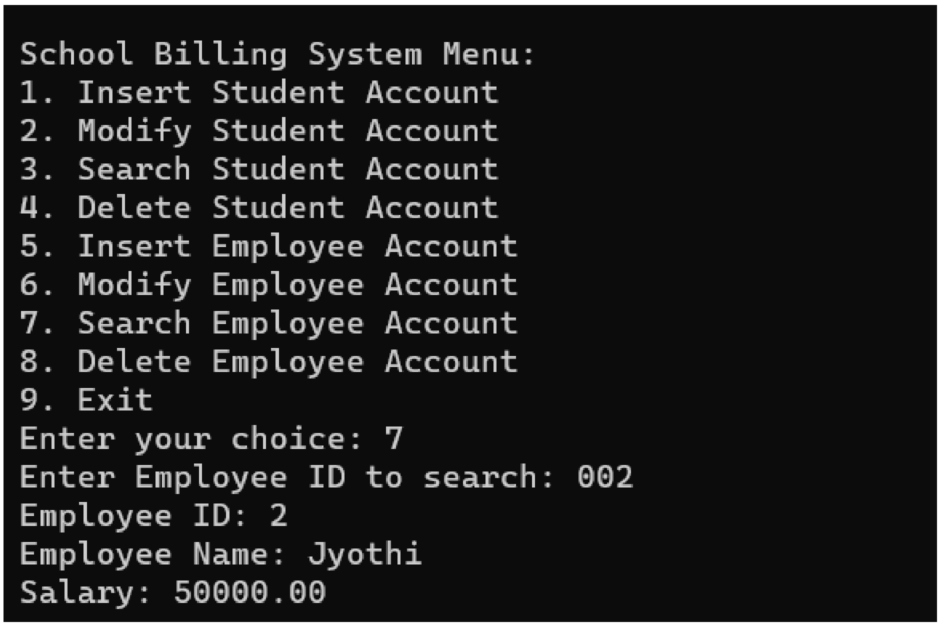
```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 4
Enter Student ID to delete: 001
Student account deleted successfully.
```



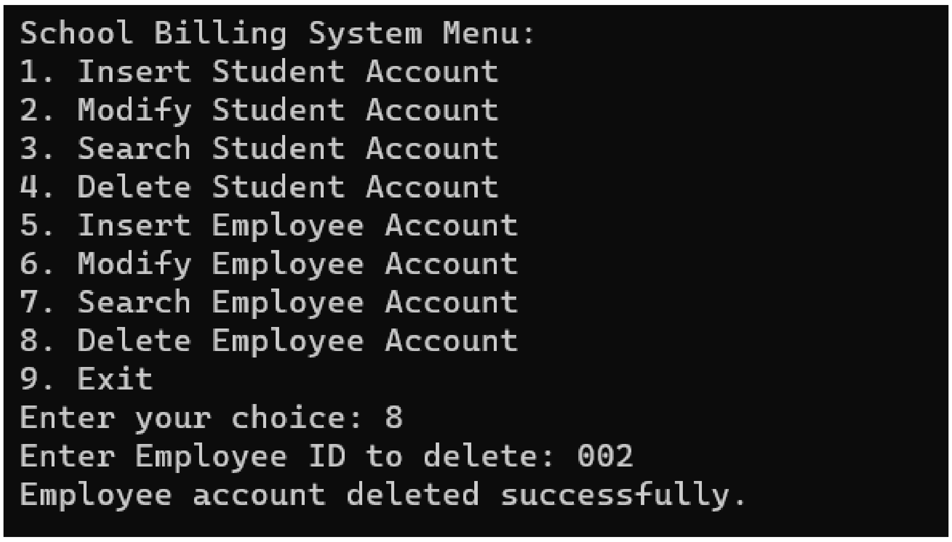
```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 5
Enter Employee ID: 002
Enter Employee Name: Jyothi
Enter Salary: 45000
Employee account added successfully.
```



```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 6
Enter Employee ID to modify: 002
Enter new Salary: 50000
Employee account modified successfully.
```



```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 7
Enter Employee ID to search: 002
Employee ID: 2
Employee Name: Jyothi
Salary: 50000.00
```



```
School Billing System Menu:
1. Insert Student Account
2. Modify Student Account
3. Search Student Account
4. Delete Student Account
5. Insert Employee Account
6. Modify Employee Account
7. Search Employee Account
8. Delete Employee Account
9. Exit
Enter your choice: 8
Enter Employee ID to delete: 002
Employee account deleted successfully.
```

## 8. Conclusion

The School Billing System offers an efficient and user-friendly solution for managing student and employee accounts in a school setting. It streamlines administrative tasks and improves data accuracy.