REAL-TIME FACE RECOGNITION

Technical Report

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF COMPUTER SCIENCE AND ENGINEERING

BACHELOR OF TECHNOLOGY

Submitted by:
Jaskirat Singh
2302566

Submitted to:
Er. Satinderpal Singh
Assistant Professor

Dept. of Computer Science Engineering
Guru Nanak Dev Engineering College,
Ludhiana, 141006

Problem Statement

The LIBRARY MANAGEMENT SYSTEM project has book and student class with data members like book no, bookname, authorname. Books record is stored in a binary file. A student can issue book and deposit it within 15 days. Student is allowed to issue only one book. Student Records are stored in binary file. Administrator can add, modify or delete record.

Solution

Source code

```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <iomanip>
#include <cstring>

using namespace std;

class Book {
public:
    char bookname[50], authorname[50];
    int book_id;


    void create_book() {
        cout << "Enter book id: ";
        cin >> book_id;
        cin.ignore();
        cout << "Enter book name: ";
        cin.getline(bookname, 50);
        cout << "Enter author name: ";
        cin.getline(authorname, 50);
    }

    void display_book() const {
        cout << left << setw(15) << book_id << setw(30) << bookname << setw(30) <<
            authorname << endl;
    }

    void write_book_file() {
        ofstream file("book.dat", ios::binary | ios::app);
        if (!file) {
            cout << "Unable to open file\n";
            return;
        }
        create_book();
        file.write(reinterpret_cast<const char*>(this), sizeof(*this));
        file.close();
        cout << "Record updated successfully!!\n";
    }

    void read_book_file() const {
        ifstream file("book.dat", ios::binary);
        if (!file) {
            cout << "Unable to open file\n";
            return;
        }
        cout << "\n\n\t\tBook List\n\n";
        cout << left << setw(15) << "Book Number" << setw(30) << "Book Name " <<
            setw(30) << "Author Name " << endl;
        cout << "
            ------------------------------------------------------------------------\
            n";
        while (file.read(reinterpret_cast<char*>(const_cast<Book*>(this)), sizeof(*
            this))) {
            display_book();
        }
        file.close();
        cout << "
            ------------------------------------------------------------------------\
            n";
    }

    void modify_book(int n) {
        fstream file("book.dat", ios::binary | ios::in | ios::out);
        if (!file) {
            cout << "File could not be opened! Press any key to exit.\n";
            return;
        }
        bool found = false;
        while (file.read(reinterpret_cast<char*>(this), sizeof(*this))) {
            if (book_id == n) {
                cout << "Enter new details for book having id " << n << endl;
                create_book();
                file.seekp(-static_cast<int>(sizeof(*this)), ios::cur);
                file.write(reinterpret_cast<const char*>(this), sizeof(*this));
                cout << "\nRecord modified!\n";
                found = true;
                break;
            }
        }
        if (!found) cout << "\nRecord not found!\n";
        file.close();
    }

    void delete_book(int n) {
        ifstream inFile("book.dat", ios::binary);
        if (!inFile) {
            cout << "File could not be opened! Press any key to exit.\n";
            return;
        }

        ofstream outFile("temp.dat", ios::binary);
        while (inFile.read(reinterpret_cast<char*>(this), sizeof(*this))) {
            if (book_id != n) {
                outFile.write(reinterpret_cast<const char*>(this), sizeof(*this));
            }
        }
        inFile.close();
        outFile.close();
        remove("book.dat");
        rename("temp.dat", "book.dat");
        cout << "\nRecord deleted!\n";
    }
    int bookid(){return book_id;}
};

class Student:public Book {
private:
    char student_name[50];
    int student_id;

public:
    Student(){
        strcpy(bookname,"no book issued");
        strcpy(authorname,"-");
        strcpy(student_name,"-");
        student_id = book_id = 0;
    }
    void create_student() {
        cout << "Enter student id: ";
        cin >> student_id;
        cin.ignore();
        cout << "Enter student name: ";
        cin.getline(student_name, 50);
    }

    void display_student() const {
        cout << left << setw(15) << student_id << setw(30) << student_name << endl;
    }

    void write_student_file() {
        ofstream file("student.dat", ios::binary | ios::app);
        if (!file) {
            cout << "Unable to open file\n";
            return;
        }
        create_student();
        file.write(reinterpret_cast<const char*>(this), sizeof(*this));
        file.close();
        cout << "Record updated successfully!!\n";
    }

    void read_student_file() const {
        ifstream file("student.dat", ios::binary);
        if (!file) {
            cout << "Unable to open file\n";
            return;
        }
        cout << "\n\n\t\tStudent List\n\n";
        cout << left << setw(15) << "Student id" << setw(30) << "Student Name " << endl;
        cout << "----------------------------------------------------------------------\n";
        while (file.read(reinterpret_cast<char*>(const_cast<Student*>(this)), sizeof(*this))) {
            display_student();
        }
        file.close();
        cout << "----------------------------------------------------------------------\n";
    }

    void delete_student(int n) {
        ifstream inFile("student.dat", ios::binary);
        if (!inFile) {
            cout << "File could not be opened! Press any key to exit.\n";
            return;
        }

        ofstream outFile("temp.dat", ios::binary);
        while (inFile.read(reinterpret_cast<char*>(this), sizeof(*this))) {
            if (student_id != n) {
                outFile.write(reinterpret_cast<const char*>(this), sizeof(*this));
            }
        }
        inFile.close();
        outFile.close();
        remove("student.dat");
        rename("temp.dat", "student.dat");
        cout << "\nRecord deleted!\n";
    }
    void issue_book(){
        int id;
        cout<<"\nEnter your Id : ";
        cin>>id;

        fstream file("student.dat", ios::binary | ios::in | ios::out);
        if (!file) {
            cout << "File could not be opened! Press any key to exit.\n";
            return;
        }
        bool found = false;
        while (file.read(reinterpret_cast<char*>(this), sizeof(*this))) {
            if (book_id && student_id == id) {book_issued_data(); break;}
            else if(student_id == id){
                int bookid;
                cout<<"\nEnter requested book id : ";
                cin>>bookid;
                if(book_finder(bookid)){
                file.seekp(-static_cast<int>(sizeof(*this)), ios::cur);
                file.write(reinterpret_cast<const char*>(this), sizeof(*this));
                cout << "\nBook is issued for 15 days!\n";
                found = true;
                book_issued_data();
                break;}
            }
            else{
                cout<<"User not found!!\n";
                found = true;
            }
        }
        if (!found) cout << "\nBook already issued!\n";
        file.close();

    }
    void return_book(){
         int id;
        cout<<"\nEnter your Id : ";
        cin>>id;

        fstream file("student.dat", ios::binary | ios::in | ios::out);
        if (!file) {
            cout << "File could not be opened! Press any key to exit.\n";
            return;
        }
        bool found = false;
        while (file.read(reinterpret_cast<char*>(this), sizeof(*this))) {
            if (book_id && student_id == id) {
                 strcpy(bookname,"no book issued");
                 strcpy(authorname,"-");
                 book_id = 0;
                file.seekp(-static_cast<int>(sizeof(*this)), ios::cur);
                file.write(reinterpret_cast<const char*>(this), sizeof(*this));
                cout << "\nBook is returned to library!\n";
                found = true;
                break;
            }
        }
        if (!found) cout << "\nBook not issued!\n";
        file.close();
    }

     bool book_finder(int n){
        fstream file("book.dat", ios::binary | ios::in | ios::out);
        if (!file) {
            cout << "File could not be opened! Press any key to exit.\n";
            return false;
        }
        bool found = false;
        Book b;
       while (file.read(reinterpret_cast<char*>(&b), sizeof(b))) {
       if(b.book_id == n){
           strcpy(this->bookname,b.bookname);
           strcpy(this->authorname,b.authorname);
           this->book_id = b.book_id;
           found = true;
           break;
       };
       }
        if(!found)cout<<"book not found!"<<endl;
        file.close();
        return found;}

        void book_issued_data(){
            cout << left << setw(15) << "Student id" << setw(30) << "Student Name " << setw(50)<<"Book Issued"<<endl;
        cout << "----------------------------------------------------------------------\n";
        cout << left << setw(15) << student_id << setw(30) << student_name << book_id<<": ("<<bookname<<" by "<<authorname<<")"<<endl;
        }
```

```cpp
};

int main() {
    Book b;
    Student student;
    int choice = 0, n;

    do {
        cout << "\n\n\t--- Administrator Menu ---";
        cout << "\n1. Add New Book";
        cout << "\n2. Display All Books";
        cout << "\n3. Modify Book Record";
        cout << "\n4. Delete Book Record";
        cout << "\n5. Add New Student";
        cout << "\n6. Display All Students";
        cout << "\n7. Delete Student Record";
        cout << "\n8. Issue book";
        cout << "\n9. return book";
        cout << "\n0. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            b.write_book_file();
            break;
        case 2:
            b.read_book_file();
            break;
        case 3:
            cout << "\nEnter Book Number to modify: ";
            cin >> n;
            b.modify_book(n);
            break;
        case 4:
            cout << "\nEnter Book Number to delete: ";
            cin >> n;
            b.delete_book(n);
            break;
        case 5:
            student.write_student_file();
            break;
        case 6:
            student.read_student_file();
            break;
        case 7:
            cout << "\nEnter Student ID to delete: ";
            cin >> n;
            student.delete_student(n);
            break;
        case 8: student.issue_book();break;
        case 9: student.return_book();break;
        case 0:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice\n";
        }
```

```c
    } while (choice != 0);

    return 0;
}
```