আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Department of

Computer Science and Engineering

DS Final Project

Course Code	:	**CSE - 2322**
Course Title	:	**Data Structure Lab**
Section	:	**3AM**

Team Name	:	**"Team Console"**
Name of Team Leader	:	**Gazi Shaharabi Anwar Tuhin**
Team Leader ID	:	**C231035**
Name of Team Member 1	:	**SM Muntasir Sarwar**
Team Leader ID	:	**C231037**
Name of Team Member 2	:	**Adnan Mahmud Alvee**
Team Leader ID	:	**C231022**

Date of Submission	:	**25/06/2024**
Submitted To	:	**Mohammed Shamsul Alam**

Remark

GitHub : https://github.com/ShaharabiTuhin/Data_Structure_EMS

Lab Report : DS Final Project Report

Project Slide : Team Console

Presentation Video : Data Structure Final Project Video

Source Code :

```
#include <iostream>
#include <fstream>
#include <conio.h>
#include <windows.h>
#include <ctime>
using namespace std;
// Structure to store employee details
struct emp {
   string name, id, address;
   int salary, contact;
   string timestamp; // Added timestamp to store the time of data entry
};
// Node structure for linked list
struct Node {
   emp data;
   Node* next;
};
Node* head = nullptr; // Head pointer of the linked list
string username, password; // Variables to store username and password
string empFileName; // Employee file name for the current user
// Function declarations
void insertEmp(const emp& employee);
void saveEmpDataToFile();
void loadEmpDataFromFile();
void signUp();
bool logIn();
void empdata();
void show();
void Search();
void update();
void del();
void feature();
void sleepMilliseconds(int milliseconds);
string getCurrentTime(); // Function to get the current timestamp
// Function to insert employee data into the linked list
void insertEmp(const emp& employee) {
   Node* newNode = new Node;
   newNode->data = employee;
   newNode->next = nullptr;
   if (!head) {
      head = newNode;
   } else {
      Node* temp = head;
      while (temp->next) {
```

```
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
// Function to save employee data to file
void saveEmpDataToFile() {
    ofstream outFile(empFileName);
    if (outFile.is_open()) {
        Node* temp = head;
        while (temp) {
            outFile << temp->data.name << endl;
            outFile << temp->data.id << endl;
            outFile << temp->data.address << endl;
            outFile << temp->data.contact << endl;
            outFile << temp->data.salary << endl;
            outFile << temp->data.timestamp << endl; // Save timestamp
            temp = temp->next;
        }
        outFile.close();
    } else {
        cout << "Unable to open file for writing." << endl;
    }
}
// Function to load employee data from file
void loadEmpDataFromFile() {
    ifstream inFile(empFileName);
    if (inFile.is_open()) {
        while (!inFile.eof()) {
            emp newEmp;
            getline(inFile, newEmp.name);
            if (newEmp.name.empty()) {
                break;
            }
            getline(inFile, newEmp.id);
            getline(inFile, newEmp.address);
            inFile >> newEmp.contact;
            inFile >> newEmp.salary;
            inFile.ignore(); // Ignore newline character after reading integers
            getline(inFile, newEmp.timestamp); // Read timestamp
            insertEmp(newEmp);
        }
        inFile.close();
    } else {
        cout << "Unable to open file for reading." << endl;
    }
}
// Function to sign up a new user
void signUp() {
    cout << "\n\n\n\t\t----- Signup -----" << endl;
    cout << "\t\tEnter new username: ";
    cin >> username;
    cout << "\t\tEnter new password: ";
    cin >> password;
    empFileName = username + ".txt";
    ofstream userFile("users.txt", ios::app);
    if (userFile.is_open()) {
        userFile << username << " " << password << " " << empFileName << endl;
```

```
        userFile.close();
    } else {
        cout << "Unable to open user file for writing." << endl;
    }
    cout << "\n\t\tYour new id is creating please wait";
    for (int i = 0; i < 6; i++) {
        cout << ".";
        Sleep(500);
    }
    cout << "\n\n\t\tYour id created successfully";
    Sleep(2000);
}
// Function to log in a user
bool logIn() {
    ifstream userFile("users.txt");
    if (userFile.is_open()) {
        string usrn, pswd, file;
        cout << "\n\n\n\t\t   LOGIN" << endl;
        cout << "\t\tEnter username: ";
        cin >> usrn;
        cout << "\t\tEnter password: ";
        cin >> pswd;
        while (userFile >> username >> password >> empFileName) {
            if (usrn == username && pswd == password) {
                userFile.close();
                loadEmpDataFromFile();
                return true;
            }
        }
        userFile.close();
        cout << "Incorrect Username or Password" << endl;
    } else {
        cout << "Unable to open user file for reading." << endl;
    }
    return false;
}
// Function to insert employee data
void empdata() {
    int user = 0;
    cout << "How many employees data do you want to enter??" << endl;
    cin >> user;
    cin.ignore(); // Clear the input buffer before taking string input
    for (int i = 0; i < user; i++) {
        emp newEmp;
        cout << "Enter data of employee " << i + 1 << endl << endl;
        cout << "Enter employee name: ";
        getline(cin, newEmp.name); // Use getline to read the full name
        cout << "Enter id: ";
        getline(cin, newEmp.id);
        cout << "Enter address: ";
        getline(cin, newEmp.address);
        cout << "Enter contact: ";
        cin >> newEmp.contact;
        cout << "Enter salary: ";
        cin >> newEmp.salary;
        cin.ignore(); // Clear the input buffer after reading integers
        newEmp.timestamp = getCurrentTime(); // Get current timestamp
        insertEmp(newEmp);
```

```
    }
    saveEmpDataToFile(); // Save data to file after insertion
}
// Function to show all employee data
void show() {
    system("color F5"); // Change console color
    if (head) {
        Node* temp = head;
        while (temp != nullptr) {
            cout << endl;
            cout << "Data of employee " << endl;
            cout << "Name: " << temp->data.name << endl;
            cout << "ID: " << temp->data.id << endl;
            cout << "Address: " << temp->data.address << endl;
            cout << "Contact: " << temp->data.contact << endl;
            cout << "Salary: " << temp->data.salary << endl;
            cout << "Timestamp: " << temp->data.timestamp << endl; // Display timestamp
            temp = temp->next;
        }
    } else {
        cout << "No data is entered" << endl;
    }
}
// Function to Search for an employee by ID
void Search() {
    if (head) {
        string id;
        cout << "Enter id of employee which you want to search" << endl;
        cin >> id;
        Node* temp = head;
        while (temp) {
            if (temp->data.id == id) {
                cout << "Data of employee " << endl;
                cout << "Name: " << temp->data.name << endl;
                cout << "ID: " << temp->data.id << endl;
                cout << "Address: " << temp->data.address << endl;
                cout << "Contact: " << temp->data.contact << endl;
                cout << "Salary: " << temp->data.salary << endl;
                cout << "Timestamp: " << temp->data.timestamp << endl; // Display timestamp
                return;
            }
            temp = temp->next;
        }
        cout << "No such record found" << endl;
    } else {
        cout << "No data is entered" << endl;
    }
}
// Function to update employee data by ID
void update() {
    system("color F5"); // Change console color
    if (head) {
        string id;
        cout << "Enter id of employee which you want to update" << endl;
        cin >> id;
        Node* temp = head;
        while (temp) {
            if (temp->data.id == id) {
```

```
            cout << "Old data of employee " << endl;
            cout << "Name: " << temp->data.name << endl;
            cout << "ID: " << temp->data.id << endl;
            cout << "Address: " << temp->data.address << endl;
            cout << "Contact: " << temp->data.contact << endl;
            cout << "Salary: " << temp->data.salary << endl;
            cout << "Timestamp: " << temp->data.timestamp << endl; // Display old timestamp
            cout << "\nEnter new data" << endl;
            cout << "Enter employee name: ";
            cin.ignore(); // Clear the input buffer
            getline(cin, temp->data.name);
            cout << "Enter id: ";
            getline(cin, temp->data.id);
            cout << "Enter address: ";
            getline(cin, temp->data.address);
            cout << "Enter contact: ";
            cin >> temp->data.contact;
            cout << "Enter salary: ";
            cin >> temp->data.salary;
            temp->data.timestamp = getCurrentTime(); // Update timestamp
            saveEmpDataToFile(); // Save data to file after update
            system("CLS"); // Clear the console screen
            return;
        }
        temp = temp->next;
    }
    cout << "No such record found" << endl;
  } else {
    cout << "No data is entered" << endl;
  }
}
// Function to delete employee data
void del() {
  if (head) {
    int press;
    cout << "Press 1 to delete specific record" << endl;
    cout << "Press 2 to delete full record" << endl;
    cin >> press;
    if (press == 1) {
      string id;
      cout << "Enter id of employee which you want to delete" << endl;
      cin >> id;
      Node* prev = nullptr;
      Node* curr = head;
      while (curr) {
        if (curr->data.id == id) {
          if (prev)
            prev->next = curr->next;
          else
            head = curr->next;
          delete curr;
          cout << "Your required record is deleted" << endl;
          saveEmpDataToFile(); // Save data to file after deletion
          return;
        }
        prev = curr;
        curr = curr->next;
      }
```

```
            cout << "No such record found" << endl;
        } else if (press == 2) {
            Node* temp = head;
            while (head) {
                temp = head;
                head = head->next;
                delete temp;
            }
            cout << "All records are deleted" << endl;
            saveEmpDataToFile(); // Save data to file after deletion
        } else {
            cout << "Invalid Input" << endl;
        }
    } else {
        cout << "No data is entered" << endl;
    }
}
// Function to display features
void feature() {
    system("color F3"); // Change console color
    string text = "\n\t\t---- Features ----\n"
                  "\nUser Authentication:\n"
                  "\tSupports user signup (signUp()) and login (logIn()).\n"
                  "\tUser credentials are stored in users.txt with each user having a separate file (username.txt) to store employee
data.\n"
                  "\nEmployee Data Management:\n"
                  "Insert Data (empdata()):\n"
                  "\tAllows users to enter employee details such as name, ID, address, contact, and salary. Uses insertion sort to
maintain employees sorted by ID.\n"
                  "\nShow Data (show()):\n"
                  "\tDisplays all entered employee data if available.\n"
                  "\nSearch Data (search()):\n"
                  "\tAllows searching for an employee by ID using binary search.\n"
                  "\nUpdate Data (update()):\n"
                  "\tEnables updating employee details based on ID. After updating, the data is re-sorted to maintain order.\n"
                  "\nDelete Data (del()):\n"
                  "\tProvides options to delete specific employee records by ID or delete all records.\n"
                  "\nFile Handling:\n"
                  "\tEmployee data is stored in text files (username.txt).\n"
                  "\tUses file streams (ifstream and ofstream) for reading from and writing to files.\n"
                  "\nUser Interface:\n"
                  "\tUses console-based interface with options displayed using cout.\n"
                  "\tClear screen (system(\"CLS\")) to manage screen display.\n"
                  "\n\nHow It Works:\n"
                  "\tUser Management:\n"
                  "\tUsers can sign up with a new username and password.\n"
                  "\tExisting users can log in to access their specific employee data file (username.txt).\n"
                  "\nData Management:\n"
                  "\tUsers can add, view, search, update, and delete employee records.\n"
                  "\tData operations ensure that the employee records remain sorted by ID for efficient searching and
management.\n"
                  "\nFile Operations:\n"
                  "\tUpon login, the system loads existing employee data from the user's file (username.txt).\n"
                  "\tChanges (additions, updates, deletions) to employee records are saved back to the file to maintain
persistence.\n"
                  "\nError Handling:\n"
                  "\tIncludes basic error handling for file operations and user inputs to ensure smooth functioning of the
program.\n"
```

```
                "\nLimitations and Considerations:\n"
                "\tThe system assumes a single user per file (username.txt), limiting concurrent access.\n"
                "\tError handling is minimal and may need enhancement for robustness.\n"
                "\tInput validations (e.g., for ID uniqueness) are not explicitly handled in the provided code.\n"
                "\tOverall, this Employee Management System provides essential functionalities for managing employee data
through a console interface, leveraging file storage for persistence across sessions.\n";
    for (char c : text) {
        cout << c << flush;  // Print character without newline
        sleepMilliseconds(10);  // Adjust delay as needed
    }
}
// Function to introduce delay
void sleepMilliseconds(int milliseconds) {
    Sleep(milliseconds);
}
// Function to get the current timestamp
string getCurrentTime() {
    time_t now = time(0);
    tm* ltm = localtime(&now);
    char buffer[80];
    strftime(buffer, sizeof(buffer), "%d-%m-%Y %H:%M:%S", ltm);
    return string(buffer);
}
int main() {
    system("color F5"); // Change console color
    cout << "\n\n\t----- Employee Management System -----" << endl;
    while (true) {
        int choice;
        cout << "\n\n\t\tPress 1 to Sign Up" << endl;
        cout << "\t\tPress 2 to Log In\n" << endl;
        cout << "\t\tEnter your choice: ";
        cin >> choice;
        if (choice == 1) {
            signUp();
        } else if (choice == 2) {
            if (logIn()) {
                system("CLS"); // Clear the console screen
                char ch;
                while (true) {
                    system("color F8"); // Change console color
                    cout << "\n\nPress 1 to enter data" << endl;
                    cout << "Press 2 to show data" << endl;
                    cout << "Press 3 to search data" << endl;
                    cout << "Press 4 to update data" << endl;
                    cout << "Press 5 to delete data" << endl;
                    cout << "Press 6 to show feature" << endl;
                    cout << "Press 7 to logout\n" << endl;
                    cout << "Enter your choice: ";
                    cin >> ch;
                    system("CLS"); // Clear the console screen
                    system("color F5"); // Change console color
                    switch (ch) {
                        case '1':
                            empdata();
                            break;
                        case '2':
                            show();
                            break;
```

```
                case '3':
                    Search();
                    break;
                case '4':
                    update();
                    break;
                case '5':
                    del();
                    break;
                case '6':
                    feature();
                    break;
                case '7':
                    goto start;
                default:
                    system("color F5"); // Change console color
                    cout << "Invalid Input" << endl;
                    break;
            }
        }
    }
  } else {
    cout << "Invalid Input" << endl;
  }
 }
start:
 system("CLS"); // Clear the console screen
 return 0;
}
```

Online Link: https://onlinegdb.com/jwOyXpzqe