THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

ELECTENG 304
Computer Systems 2E

Development of a Conference Attendance Information System

Authors : Thusitha Mabotuwana (9790416)
Duleepa Thrimawitharana (9801736)

Department: Electrical and Electronic Engineering.

Date: October 24, 2003.

We hereby declare that the following work is our own unaided work and was not copied from or written in collaboration with any other person.

Signed: _____

_____

Table of Contents

1.0 Introduction

The aim of this project was to design and develop a Conference Attendance Information System using the C++ language, that would monitor the delegates entering and leaving the Conference and produce a report, to the screen which includes the attendance at each presentation, on a session by session basis, session total and any absences of chairperson or presenters within a session.
The delegates' details were to be loaded using the *delegates.txt* file which contains each delegate's name and ID. The *sessions.txt* file, which includes each session's name, day, start time, room number, chairperson ID and presentation details (presentation name and presenter ID) was to be used to create the sessions list.
This report gives a brief overview of how the required functionalities were implemented to meet the project specifications. It includes flowcharts showing flow of information and various aspects considered in the development process. The C++ code written is attached as an Appendix.

2.0 Implementation of the Design

- At startup, the default delegate and session details are loaded to two vectors using *delegates.txt* and *sessions.txt* files. These input files can be changed later if necessary.
- The Main Menu consisting of 7 options is then loaded. The Menus are:
    1. New Delegate Entering A Room
    2. Delegate Leaving A Room
    3. Generate Attendance Log
    4. Generate Security Log (System Violations)
    5. Load New File With Delegate Details
    6. Load New File With Session Details
    7. Quit From Conference Attendance Information System
- When the first option is selected and details correctly entered, the instance of the DELEGATE Class (the entered ID is checked against the IDs taken from the input file) is updated with the Session the delegate entered into along with entered time.
- When a delegate is leaving a room, the time he/she was present is considered (Please refer to Figure 02 to see the time line considered) and the 2 *in-out* lists (which contain the IDs of delegates who entered and left rooms) are updated. The 2 attributes, session name and entered time in the DELEGATE class are also updated for the left delegate's instance.
- When a Session Attendance Log is requested, the *in-out* lists for each presentation are compared and if a delegate was present in both (meaning that the delegate was there from beginning to end of presentation), the details are displayed. It also checks if the Presenter and/or Chairperson was there and gives appropriate messages if absent. If the presenter was absent, the presentation total will be set to zero. This Menu also produces a log with delegates in current presentations which is generated using only the *in* list.
- All errors encountered during execution (such as incorrect IDs entered, incorrect in and/or out times, delegates not scanned out etc.) are appended to *errorlog.txt* file and a copy of this file is opened for the user on request.
- When the user wants to change the default delegate and/or session input files, the old vector is deleted and a new delegate and/or session vector created provided that the new file was available in the system.

3.0 Classes Used and Their Functionalities

Three main Classes have been used in the development process. The functionality of each class is briefly explained in this section followed by a diagram that shows the private variables used in these Class along with the methods used.

The DELEGATE Class: This Class is used to store the details about each delegate. The delegate's ID and name are stored when the input file is read. The other three attributes, entered time, entered session and left time are updated as the delegate enters and/or leaves presentations, so that they can be compared to see if a delegate was duly present for a presentation.

The SESSION Class: This Class holds all the details required about a session. All the attributes are initialised when the input file is read except for the Boolean variable used to check if the Chairperson was present. This attribute is updated if the Chairperson was recorded as *entered* at the beginning of the desired presentation.

The PRESENTATION Class: All the integer attributes in this Class are initialised when the input file is read. The Boolean variable is made *true* or *false* depending on whether the Presenter was recoded as *entered* at the beginning of the presentation. This Class also has 2 vectors that store the IDs of the delegates who enter and/or leave the presentation.

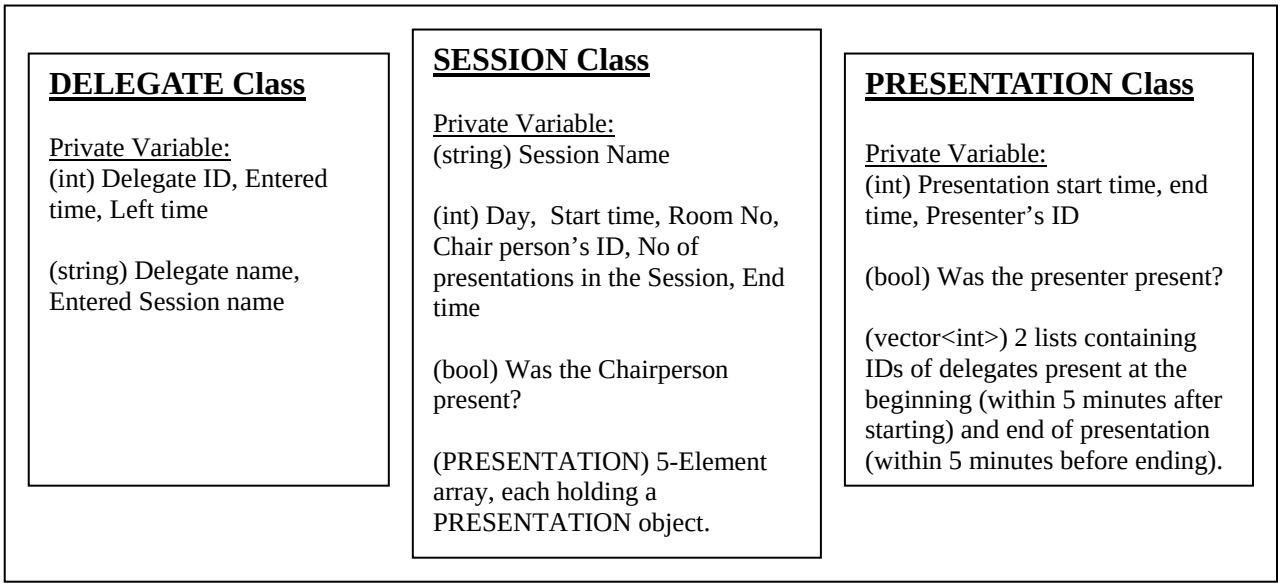

Figure 01: Diagram showing Classes used and their private attributes.

4.0 Possibilities Considered in the Development Process

The following possibilities were considered in order to make the system more realistic and practical.

- Lets consider the time limits first. Using a Session that runs from 0900-1030 (ie. 3 presentations in this session) as an example:

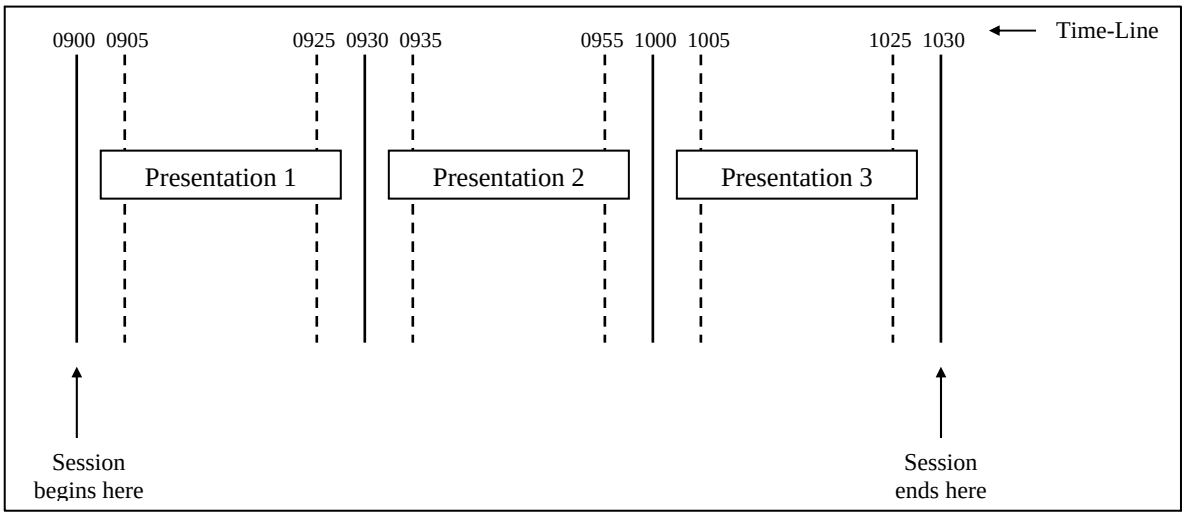

Figure 02: Example to demonstrate time-line considerations

- A delegate is considered to have entered (ie. Delegate entered the room. Could or could not be present) if entered within 5 minutes after the presentation starts. Delegate is considered to have left the room if left within 5 minutes before the presentation finishes. Some of the possibilities for delegate present/absent status for the above-drawn example are shown in Table 01.

In Time	Out Time	Presentation 1	Presentation 2	Presentation 3
0900	0930	Present	Not-Present	Not-Present
0900	1030	Present	Present	Present
0900	0920	Not-Present	Not-Present	Not-Present
0900	0955	Present	Present	Not-Present
0905	0930	Present	Not-Present	Not-Present
0910	0920	Not-Present	Not-Present	Not-Present
0902	0904	Not-Present	Not-Present	Not-Present
0910	0945	Not-Present	Not-Present	Not-Present
0910	1025	Not-Present	Present	Present

Table 01: Delegate In/Out status for example in Figure 02.

However, if invalid times, for example if delegate in and out times are the same, delegate entered time later than the time he/she left or delegate left after the session end-time, are entered, appropriate messages will be displayed on the screen notifying the user of the current entry.

- Since the system is simulated using manual entry, there is a possibility that invalid characters can be entered. When numbers are to be entered the following are considered:

Entered	Value
(35)	(35)
(35)	(35)
(35)	(35)
(35hu)	(35)
(35 38)	(35)
(hu)	Invalid entry. Message displayed

Table 02: Valid Vs. invalid entries for numerical inputs

- Since delegates can stay in the same room for another Session, they are considered as *entered* for the next Session provided that there is another Session in the same room, although they might not have left the room.
- If the Chairperson was not present for a session a warning message is displayed. However if the presenter was absent from the presentation, the total number of delegates present for that presentation is considered zero.

5.0 Assumptions Made

The following assumptions were made in order to narrow down the possibilities as in a real-world situation and make the development process easier.

- The entering and leaving times for delegates will be entered in a 24-hour format. For example to enter 9.30am, 0930; 2.30pm, 1430 should be entered.
- The input files are stored in the system, hence are assumed to be in the right format. However error checking is used when manual entries are loaded.
- A session can progress even though the chairperson may not be present.
- The total number of delegates for a presentation will be zero if presenter was absent.
- A session or a presentation will always start and finish on the half-hour.
- The Presenter or Chairperson will be present for the whole presentation, if present at the start of presentation.
- Delegates can enter and/or leave presentations at any time they wish.
- Delegates can stay in a room for as long as they want (For example, a delegate can stay for 3 presentations without leaving or stay continuously for multiple Sessions).

6.0 Using the System

The developed systems consists of 5 main menus. These menus along with their submenus (or the functionality of the submenu) are shown in Figure 03 below.

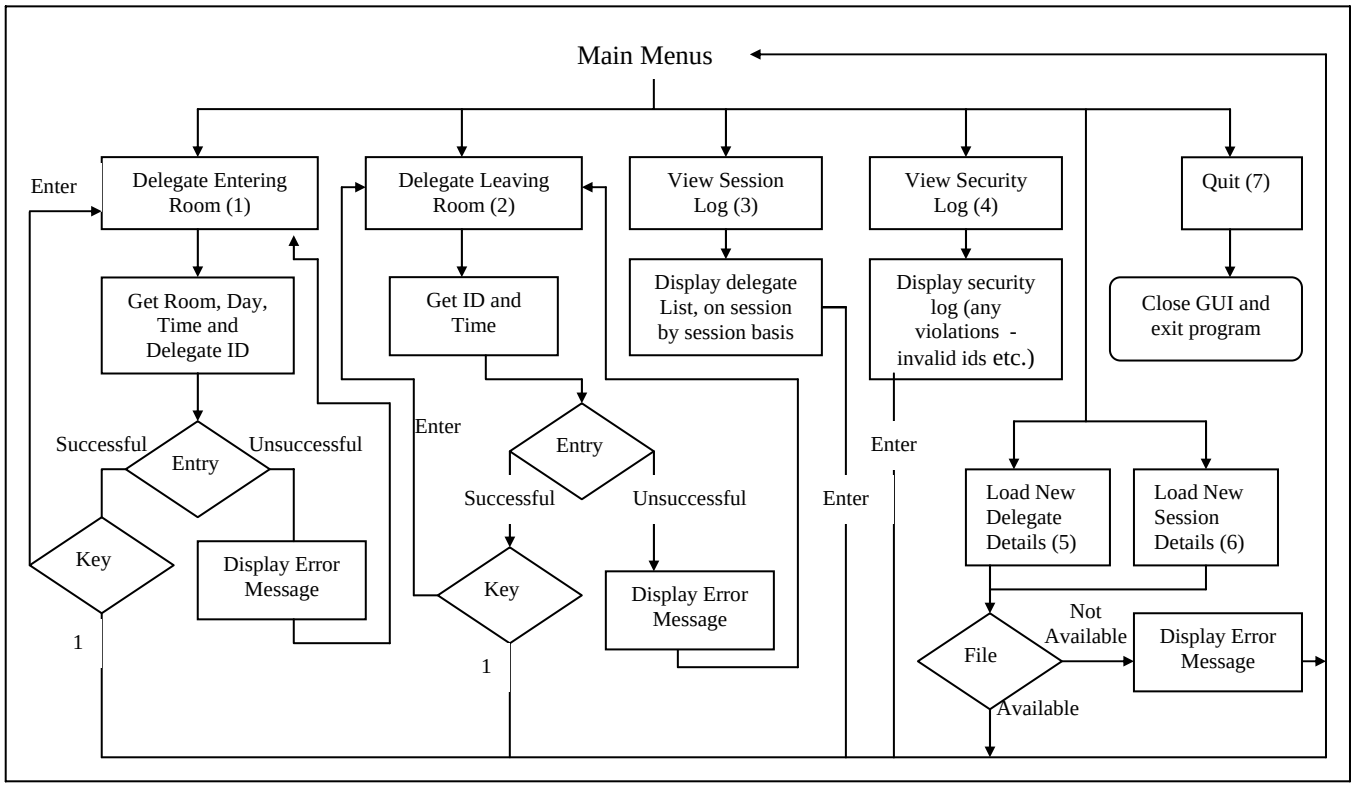

Figure 03: Menus, submenus and their relationships.

7.0 Conclusions

The Conference Attendance Information System has successfully been implemented to meet all the project specifications. Some extra features have also been added to make the system more user-friendly, stable and realistic. A considerable amount of time has been spent testing and debugging the system which has proven to be satisfactory, provided that the delegate in and/or out times are entered in the 24-hour time format and the two input files are in the required format.

Appendix: C++ Code Implementation for the Design

File Main.cpp

```
//-------------------------------------------------------------------------------------------
//Inclusion of header files
//-------------------------------------------------------------------------------------------
#include "CreateDelNSesLists.h"
//-------------------------------------------------------------------------------------------
//Function prototypes
//-------------------------------------------------------------------------------------------

void MenuIn(vector <DELEGATE> &, vector <SESSION> &);
void MenuOut(vector <DELEGATE> &,vector <SESSION> &);
void MenuSesLog(vector <DELEGATE> &,vector <SESSION> &);
int getIntVal(char *);
void displayHeader(int, string);
int appendDelegate(vector <DELEGATE> &,vector <SESSION> &, int, int, int,int);
int addOutTime(vector <DELEGATE> &,vector <SESSION> &, int, int);
void createInOrOutList(vector <DELEGATE> &,vector <SESSION> &, int, int);


//-------------------------------------------------------------------------------------------
//Functions to control GUI - invoked by main()
//-------------------------------------------------------------------------------------------

//-----------------Main GUI function - contains main menu and calls----------------------------
//-----------------appropriate functions according to user input-------------------------------

void GUIInput(){
        int intVal, enteredVal;
        string inputfile;
        char * inputStrPtr = new char[20];

        vector <DELEGATE> delegatesVec;
        vector <SESSION> sessionsVec;
        delegatesVec = CreateDelList(delegatesVec,"delegates.txt");
        sessionsVec = CreateSesList(sessionsVec,"sessions.txt");

        do{
                system("cls");
        //Clear screen
                displayHeader(57,"Developed by Duleepa and Thusitha");

                        //Displays default header
                cout << '\n' << "Please Select The Action Number From Following List" << endl << endl;

                        //Display menu options
                cout << "1. New Delegate Entering A Room" << endl;
                cout << "2. Delegate Leaving A Room" << endl;
                cout << "3. Generate Attendance Log" << endl;
                cout << "4. Generate Security Log (System Violations)" << endl;
                cout << "5. Load New File With Delegate Details" << endl;
                cout << "6. Load New File With Session Details" << endl;
                cout << "7. Quit From Conference Attendence Information System" << endl;
                cout << '\n' << "Enter The Option (1, 2, 3, 4, 5, 6 or 7)? ";

                cin.getline(inputStrPtr,10,'\n');                                       //Get user option as a
string to avoid crashes
                intVal = getIntVal(inputStrPtr);                                        //Convert it to an int

                do{
                        //Call appropriate sub-menu
                        enteredVal = 0;
        //Keep track on incorrect entries

                        switch(intVal){

                                case 1:     MenuIn(delegatesVec, sessionsVec);          //call sub-
menu
                                                enteredVal = 1;
```

```
        //Set to indicate correct entry
                                                break;

                                        case 2:    MenuOut(delegatesVec, sessionsVec);                  //call sub-
menu
                                                enteredVal = 1;
        //Set to indicate correct entry
                                                break;

                                        case 3:    MenuSesLog(delegatesVec,sessionsVec);
        //call sub-menu
                                                enteredVal = 1;
        //Set to indicate correct entry
                                                break;

                                        case 4: system("cls");
        //Clear screen
                                                //call sub-menu
                                                enteredVal = 1;
        //Set to indicate correct entry
                                                break;

                                        case 5:    system("cls");
        //Clear screen
                                                delegatesVec.clear();
                                                cout << "Enter New Delegate File: ";
                                                cin.getline(inputStrPtr,20,'\n');
                                                inputfile = inputStrPtr;
                                                delegatesVec = CreateDelList(delegatesVec,inputfile);
                                                enteredVal = 1;
        //Set to indicate correct entry
                                                break;

                                        case 6:    system("cls");
        //Clear screen
                                                sessionsVec.clear();
                                                cout << "Enter New Session File: ";
                                                cin.getline(inputStrPtr,20,'\n');
                                                inputfile = inputStrPtr;
                                                cout << inputfile;
                                                sessionsVec = CreateSesList(sessionsVec,inputfile);
                                                system("pause");
                                                enteredVal = 1;
        //Set to indicate correct entry
                                                break;

                                        case 7:    cout << endl << "Exiting" << endl << endl;
                                                enteredVal = 1;
        //Set to indicate correct entry
                                                break;

                                        default:
                                                if(!enteredVal){
        //If incorrect entry loop until a

                                //correct entry is made
                                                        cout << "Invalid Option" << endl << endl;
                                                        cout << '\n' << "Enter The Option (1, 2, 3, 4, 5, 6
or 7)? ";

                                                        cin.getline(inputStrPtr,10,'\n');
                                                        intVal = getIntVal(inputStrPtr);
                                                }
                                                break;
                                }

                        } while(!enteredVal);
        //Loop until correct value entered
```

```
	} while (intVal!=7);                                                    //If option
5 selected exit
}


//-------------Sub menu 1 function (MenuIn) - Contains function calls to-----------------------
//--------------get information from user and append to delegate class-------------------------

void MenuIn(vector <DELEGATE> & delegates, vector <SESSION> & sess){

	int rmNo, day, time, delID, menuOption;
	char * inputStrPtr = new char[20];

	do{
		system("cls");
	//Clear screen
		displayHeader(59,"Sub Menu 1 - Delegate Entering A Room");

			//Display Header
		cout << '\n';
		cout << "Enter Room		: ";		//Get infromation about the delegate
		cin.getline(inputStrPtr,10,'\n');                                       //Entering a room
		rmNo = getIntVal(inputStrPtr);

		cout << "Enter Day		: ";
		cin.getline(inputStrPtr,10,'\n');
		day = getIntVal(inputStrPtr);

		cout << "Enter Time (24 Hour Clock-HHMM) : ";
		cin.getline(inputStrPtr,10,'\n');
		time = getIntVal(inputStrPtr);

		cout << "Enter Delegate ID		: ";
		cin.getline(inputStrPtr,10,'\n');
		delID = getIntVal(inputStrPtr);

		if(appendDelegate(delegates,sess,rmNo,day,time,delID))

			//If the entered information match a
			cout << '\n' << "Your Entry Was Successful" << endl << endl;

			//delegate and session display message
		else
		//Else display unsuccessful message
			cout << '\n' << "Your Entry Was Not Successful" << endl << endl;

			//Give option to stay in the same menu
		cout << "Enter \"1\" to Exit To Main Menu Or Else Press Enter To Continue ";
		cin.getline(inputStrPtr,10,'\n');                                       //Or to exit to main
menu
		menuOption = getIntVal(inputStrPtr);
	} while(menuOption!=1);
	//Loop until user chose to leave to main menu
}


//--------------Sub menu 1 function (appendDelegate) - Add information to----------------------
//--------------------------------the delegate class-----------------------------------------

int appendDelegate(vector <DELEGATE> & delegates,vector <SESSION> & sess, int rmNo, int day, int inTime,int
delID){
	string sessName;
	char *ptrSesName = new char[20];
	int idFound = 0;

	for (int j=0; j<delegates.size(); j++){
		if(delegates[j].getId()==delID){                                        //First check for
matching ID
```

```
                        idFound = 1;
                        cout << endl << endl << "Matching ID Found For: " << delegates[j].getDelName() <<
endl;

                        for (int k=0; k<sess.size(); k++){                              //If ID match, check
for matching session
                                sessName = sess[k].compareSession(day, inTime, rmNo);
                                ptrSesName = (char*) sessName.c_str();

                                if(sessName!="\0"){
        //If compareSession return a matching session
                                        if (strlen(delegates[j].getSessionName())){

                        //Check whether delegate was already in

                                                cout << "ERROR: USER ALREADY IN ROOM" << endl;

                        //If so generate error message
                                                addOutTime(delegates,sess,inTime,delID);//----------------
check return

                        //Call out function to sign out

                        //delegate from last session
                                                //Generate security log-----------------------------------
                                        }

                                        delegates[j].setInTime(inTime);                 //Set
delegate details appropriately
                                        delegates[j].setSessionName(ptrSesName);
                                        delegates[j].setme(ptrSesName);
                                        cout << "Matching Session Found As: " << ptrSesName <<endl;

                                        return 1;
                                }
                        }
                }
        }
        if(!idFound){
        //If ID match not successful generate error message
                cout << endl << "ERROR: INVALID ID ENTERED" << endl;
                //Generate security log-----------------------------------
        }
        else{
                //Else session match was not successful
                cout << endl << "ERROR: INVALID SESSION ENTERED" << endl;
        }
                        //Generate error message
        return 0;
}


//--------------Sub menu 2 function (MenuOut) - Get information from user----------------------
//----------------and set appropriate vriables in delegate to sign out-------------------------

void MenuOut(vector <DELEGATE> & delegates,vector <SESSION> & sess){
        int time, delID, menuOption = 0;
        char * inputStrPtr = new char[20];

        do{
                system("cls");
        //Clear screen
                displayHeader(59,"Sub Menu 1 - Delegate Leaving A Room");

                //Print header
                cout << '\n';
        //Get details from user
                cout << "Enter Delegate ID (1-9999)      : ";
```

```
                cin.getline(inputStrPtr,10,'\n');
                delID = getIntVal(inputStrPtr);

                cout << "Enter Time (24 Hours Clock-HHMM) : ";
                cin.getline(inputStrPtr,10,'\n');
                time = getIntVal(inputStrPtr);

                addOutTime(delegates, sess, time, delID);                       //Add the out time to delegate

                //Give user the option to stay at current menu
                cout << "Enter \"1\" to Exit To Main Menu Or Else Press Enter To Continue ";
                cin.getline(inputStrPtr,10,'\n');                                       //Or to exit to main menu
                menuOption = getIntVal(inputStrPtr);
        } while(menuOption!=1);                                                                 //Loop
until user wants to exit

}


//--------------Sub menu 2 function (addOutTime) - Adds out time to delegate--------------------
//-----------------and do manipulations to add delegate to in out lists-----------------------

int addOutTime(vector <DELEGATE> & delegates,vector <SESSION> & sess, int outTime, int delID){
        string sessName;
        char *ptrSesName = new char[20];
        int idFound = 0;

        for (int j=0; j<delegates.size(); j++){                                 //Find matching delegate
                if(delegates[j].getId()==delID){
                        idFound = 1;
                        delegates[j].setOutTime(outTime);                       //Set delegate details
                        createInOrOutList(delegates, sess, j, outTime);         //Add to histry and remove old
details
                        cout << endl << "Delegate Signed Out From Following Sessions: " << endl;
                        return 1;
                }
        }
        if(!idFound) cout << endl << "ERROR: INVALID ID ENTERED" << endl;
        return 0;
}


//--------------Sub menu 2 function (createInOrOutList) - Creates in and out--------------------
//------------------for presentation objects and also cleans delegate-------------------------

void createInOrOutList(vector <DELEGATE> & delegate,vector <SESSION> & sess, int vecPos, int outTime){

        string sessNameDel;
        int temp=0;
        sessNameDel = delegate[vecPos].getSessionName();

        for (int k=0; k<sess.size(); k++){                                              //Find session for delegate, by
matching session name
                if(sess[k].getSessionName()==sessNameDel){              //Check which presentation attended
                        do{

        if(sess[k].checkTimeLimits(delegate[vecPos].getInTime(),delegate[vecPos].getOutTime(),delegate[vecPos].ge
tId())==2){
                                                delegate[vecPos].emptySesDetails();
                                                temp =
appendDelegate(delegate,sess,sess[k].getRoom(),sess[k].getSesDay(),sess[k].getEndTime()+1,delegate[vecPos].getId());
                                                if(!temp){
                                                        cout << "ERROR: Session Ended Before Delegate Left***"
<< temp << endl;
                                                        break;
                                                }
                                                else
```

```
		if(sess[k].checkTimeLimits(sess[k].getEndTime(),outTime,delegate[vecPos].getId())!=2)
                                                                        break;
                                        }
                                        else{
                        //Also adds to in and out lists in presentation object
                                                delegate[vecPos].emptySesDetails();                     //Empty
details about current session in delegate
                                                break;
                                        }
                                }while(true);
                        }

                }
}


//--------------Sub menu 3 function (MenuSesLog) - Creates lists of delegats-------------------
//------------------presented for presentations and currently loged on-------------------------

void MenuSesLog(vector <DELEGATE> & delegates,vector <SESSION> & sess){

                char *ignore = new char[20];
                int menuOption = 0;
                int day, time, countCurDels = 1;
                char * inputStrPtr = new char[20];

                do{
                        system("cls");
                //Clear screen

                        cout << "View Session Log Upto Day  : ";
                        cin.getline(inputStrPtr,10,'\n');
                        day = getIntVal(inputStrPtr);

                        cout << "Until (24 Hours Clock-HHMM): ";
                        cin.getline(inputStrPtr,10,'\n');
                        time = getIntVal(inputStrPtr);

                        if (inputStrPtr=="" || day==0){
                                system("cls");
                                cout << "Invalid Entry. Day and Time cannot be equal to zero" << endl;
                                cout << "\a" << endl;
                        }
                        else{
                                system("cls");
                                cout << "Delegates Present In Current Sessions (as at ";
                                cout << time <<" hours on day " << day  << ")" << endl;

                                for (int j=0; j<delegates.size(); j++){
                                        if (strlen(delegates[j].getSessionName()) && delegates[j].getInTime() <= time){

                                                cout << endl << countCurDels++ << ".) Delegate " <<
delegates[j].getDelName() << "(";
                                                cout << delegates[j].getId() << ") Present In Session " <<
delegates[j].getSessionName();
                                        }
                                }
                                if(countCurDels<=1)
                                        cout << "There Are No Current Delegates Present In Sessions";
                                cout << endl << endl << "Delegates Present In All Sessions Held So Far" << endl;
                                for (int i=0; i<sess.size(); i++){
                                        sess[i].presentationLog(delegates,day,time);
                                }
                        }
                                //Give user the option to stay at current menu

                        cout << "Enter \"1\" to Exit To Main Menu Or Else Press Enter To Continue ";
                        cin.getline(inputStrPtr,10,'\n');                                       //Or to exit to main menu
```

```
                menuOption = getIntVal(inputStrPtr);
        } while(menuOption!=1);                                                         //Loop
until user wants to exit

}


//--------------Function (getIntVal) - Convert input string to an int safely-------------------

int getIntVal(char *enteredStr){

        char val[5];
        int count=0, started=0;

        for (int i=0;i<strlen(enteredStr);i++){                         //Delete all leading spaces and trailing
chars

                if (isdigit(enteredStr[i])){
                        started = 1;
                        val[count++]=enteredStr[i];
                }

                else if (isspace(enteredStr[i]) && !started);
                else if (started) break;
                else return 0;
        //Error: return o

        }

        return atoi(val);                                                               //Else
integer
}


//--------------------Function (displayHeader) - Displays Header--------------------------------

void displayHeader(int in2Width, string msg){
        cout.width(76);
        cout.fill('*');
        cout << "" << endl << endl;
        cout.width(60);
        cout.fill(' ');
        cout.setf(ios::right,ios::adjustfield);
        cout << "Conference Attendence Information System" << endl;
        cout.width(in2Width);
        cout.fill(' ');
        cout.setf(ios::right,ios::adjustfield);
        cout << msg << endl << endl;
        cout.width(76);
        cout.fill('*');
        cout << "" << endl;
}

//-----------------------------------------------------------------------------------------
//Main Function
//-----------------------------------------------------------------------------------------

int main(){

        system("cls");

        GUIInput();
        //Controled by the GUI

        return EXIT_SUCCESS;

}
File CreateDelNSesLists.h
```

```
#include "Delegates.h"
#include "Presentations.h"
#include "Sessions.h"

bool compare(DELEGATE, DELEGATE);

//-----------------------------------------------------------------------------------------
//Session Vector to store input from session file
//-----------------------------------------------------------------------------------------

vector <SESSION> CreateSesList(vector <SESSION> sessions, string fileStr){

	string inputLine, sesName, ignore;
	char comma;
	int sesDay, sesTime, sesRm, sesChId, sesPresId;
	int endOfFile, countPresentations;
	SESSION *ses;

	endOfFile = 0;                                                          //Open file
	ifstream fin((char*) fileStr.c_str());

	if(!fin){                                                               //If no file error exit
		cerr << "Can't open file" << endl;
		exit (EXIT_FAILURE);
	}

		do{
	//Else read data
		getline(fin, sesName, '\n');
		if(fin.eof()) break;
		cerr << sesName << endl;

		fin >> sesDay;
		getline(fin, ignore, '\n');
		cerr << sesDay << endl;

		fin >> sesTime;
		getline(fin, ignore, '\n');
		cerr << sesTime << endl;

		fin >> sesRm;
		getline(fin, ignore, '\n');
		cerr << sesRm << endl;

		fin >> sesChId;
		getline(fin, ignore, '\n');
		cerr << sesChId << endl;

		countPresentations = 0;

		ses = new SESSION;                                                  //Append details to
object
		ses -> setSessionName(sesName);
		ses -> setDay(sesDay);
		ses -> setStartTime(sesTime);
		ses -> setRoomNo(sesRm);
		ses -> setChairId(sesChId);

		do{
	//Get presentations until finds ";"
			comma = ' ';
			fin >> comma;
			cerr << comma << endl;
			if(comma == ';'){
				ses -> setEndTime(countPresentations);
				break;                                                      //Append
details to object
```

```
                        }

                        getline(fin, ignore, '\n');                 //Do a loop
                        cerr << ignore << endl;
                        if(fin.eof()) break;

                        fin >> sesPresId;
                        getline(fin, ignore, '\n');
                        if(fin.eof()) break;
                        cerr << sesPresId << endl;                  //Append details to object
                        ses -> setPresenters(countPresentations,sesPresId);
                        countPresentations ++;

                }while(true);

                sessions.push_back(*ses);                           //Push object to vector

                getline(fin, ignore, '\n');                         //Do a loop until end of file
                cerr << ignore << endl;
                if(fin.eof()) break;

        }while(true);

        return sessions;                                                    //Return vector

}
```

File Delegates.h

```
//-------------------------------------------------------------------------------------------
//Delegate Vector to store input from delegate file
//-------------------------------------------------------------------------------------------

bool compare(DELEGATE a, DELEGATE b){                       //Sorting algorithm for delegate vector
        return a.getId() < b.getId();
}

vector <DELEGATE> CreateDelList(vector <DELEGATE> delegates, string fileStr){

        string in;
        int personID;
        DELEGATE *person;

        ifstream fin((char*) fileStr.c_str());

        if(!fin){                                                                   //Open file
                cerr << "Can't open file" << endl;          //If no file error exit
                exit (EXIT_FAILURE);
        }

        do{
        //Else get details from file
                fin >> personID;
                if(fin.eof()) break;
                person = new DELEGATE;
                person ->setId(personID);
                getline(fin,in,'\t');
                getline(fin,in,'\n');
                person -> setDelName(in);                           //Appen details to object
                delegates.push_back(*person);                       //Push to vector
        }while(true);

        for (int i=0; i<delegates.size(); i++)          //Test print
                cout << delegates[i].getId() << ' ' << delegates[i].getDelName() << endl;

        sort(delegates.begin(), delegates.end(),compare);

        //Sort in order of ids
```

```
	for (int j=0; j<delegates.size(); j++)
		cout << delegates[j].getId() << ' ' << delegates[j].getDelName() << endl;

	return delegates;                                                    //Return vector

}

#include <fstream>
#include <ostream>

using namespace std;

#include <ctype.h>
#include <time.h>
#include <stdlib.h>


//-----------------------------------------------------------------------------------------
//Class definitions and member methods
//-----------------------------------------------------------------------------------------


//-----------Delegate class - Holds data about delegates name, ID, currently--------------------
//------------attended session and entered and out times for that session----------------------

class DELEGATE{
	int id, inTime, outTime;
	string delName, sesName;
	char *trial;

	public:
		DELEGATE() {id=0;inTime=0;outTime=0;delName='\0';sesName='\0';trial = new char[5];}

	//Default constructor to intialize the variables
		void setId(int s) {id=s;}                                    //Sets the delegate ID
		void setDelName(string s) {delName=s;}          //Sets delegate name-deep copy
		void setSessionName(char* sesN) {strcpy((char*) sesName.c_str(),sesN);}

	//Sets session name-deep copy (deleted after out)
		void setInTime(int inT) {inTime=inT;} //Sets current in time (deleted after out)
		void setOutTime(int outT) {outTime=outT;}

	//Sets current out time (deleted after out)
		void emptySesDetails() {inTime=0;outTime=0;sesName='\0';}

	//Deletes current session and times
		int getId() { return id;}                                    //Returns delegate ID
		string getDelName() {return delName;}//Returns delegate name
		char *getSessionName() {return (char*) sesName.c_str();}

	//Returns session name
		int getInTime() {return inTime;}                 //Returns entered time
		int getOutTime() {return outTime;}               //Returns exit time

		void setme(char *in) {strcpy(trial,in);}//----------------------------
		char *getme() { return trial;}//----------------------
};
```

File Sessions.h

```
//-----------------------------------------------------------------------------------------
//Inclusion of header files
//-----------------------------------------------------------------------------------------

#include <iostream>
#include <algorithm>
```

```
#include <vector>
#include <string>
#include <fstream>
#include <ostream>

using namespace std;

#include <ctype.h>
#include <time.h>
#include <stdlib.h>


//-----------------------------------------------------------------------------------------
//Class definitions and member methods
//-----------------------------------------------------------------------------------------


//---------Session class - Holds data about session name, day, times, room details,----------
//----------chairperson and presentation details for each session object--------------------

class SESSION{
        string sessionName;
        int day;
        int startTime;
        int roomNo;
        int chairId;
        int noOfPres;
        int endTime;
        bool chairIn;
        PRESENTATION *presentations[5];

        public:
                SESSION() {sessionName='\0'; day=0; startTime=0; roomNo=0; chairId=0; noOfPres=0;
endTime=0; chairIn=false;}
                void setSessionName(string sesName) {sessionName = sesName;}

        //Sets session name (deep copy)
                void setDay(int d) {day = d;}                           //Set day
                void setStartTime(int time) {startTime = time;}

        //Set starting time
                void setRoomNo(int room) {roomNo = room;}   //Set room
                void setChairId(int Id) {chairId = Id;}             //Set chaiperson ID
                void setPresenters(int prNo,int prId);              //Add presenters to the array
                void setEndTime(int presNo);                            //Set end time
                string getSessionName() {return sessionName;}

        //Return session name
                int getNoOfPres() {return noOfPres;}               //Return number of presentations
                int getSesDay() {return day;}
                int getRoom() {return roomNo;}
                int getEndTime() {return endTime;}
                void presentationLog(vector <DELEGATE> dels, int uptoDay, int uptoTime);

        //Creates a attendence log for each presenatation
                int checkTimeLimits(int inTime, int outTime,int id);

        //Check presentations delegate id was present

        //and append to presentation in or out lists
                string compareSession(int d, int Time, int rmNo);

        //Check whether the input parameters belong

        //to this session and if match return sesName else NULL
};
```

```
//----------Session member function (presentationLog) - prints the presentees for each----------
//----------------------presentation if within the time limits---------------------------------

void SESSION::presentationLog(vector <DELEGATE> dels, int uptoDay, int uptoTime){
	int countP = 1;

	if (day<=uptoDay && uptoTime>=startTime){			//Print all delegate details before
required day
		//cout.uppercase(strlen((char*) sessionName.c_str()));
		cout << endl << "SESSION NAME IS: " << sessionName << endl;
		cout.width(strlen((char*) sessionName.c_str())+strlen("SESSION NAME IS: "));
		cout.fill('=');
		cout << "";

	//Print the session name
		cout << endl;
		for (int i=0; i<noOfPres;i++){					//For number of presentations
			if(presentations[i]->getEndTime()<=uptoTime)
				presentations[i]->printList(dels, sessionName, countP++, chairIn, chairId);

	//Print the presentees list for each presentation

		}
	}
}


//----------Session member function (checkTimeLimits) - appends delegates to in or out----------
//--------------------lists for each presentation depending on time limits-----------------------

int SESSION::checkTimeLimits(int inT, int outT,int id){
	char *strTime = new char[5];					//Time int converted to string to get last
2 digits
	int started = 0, ended = 0, alreadyDone = 0;	//Flags to indicate in and out status
	int curTime = startTime;					//Start Time for presentation 1
intitialised to session start time
	chairIn = false;
	if(inT==outT){							//If deleget
enter and leave at the same time mark as nor present
		cout << "ERROR: Delegate Entered And Left At The Same Time, Not Recorded As Present" <<
endl;
		return 0;						//And exit
	}
	else if(inT>outT){					//Else if incorrect out
time generate error message
		cout << "ERROR: Delegate Left Before Enterering Session" << endl;
		return 0;						//And exit
	}

	if(inT>=curTime && inT<=(curTime+5)){			//If start time of delegate is within first 5
mins
		started = 1;						//of
presentation start time regard as in inlist
		cout << endl << "Entered in presentation 1" << endl;
		chairIn = true;
		presentations[0]->createListIn(id);		//Push to list in histry
	}

	for(int i=1; i<(noOfPres+1); i++){			//Calculate start and end time for each
presentation

	//And check which presentations deleget attended to
		sprintf(strTime,"%d",curTime);			//Convert to a string number to
compare last 2 digits

		if(strTime[strlen(strTime)-2]=='3'){		//If the current time is in format HH30
			curTime = curTime + 70;				//Start time for
presentation i and end time for i-1
```

```
                        if(inT>=curTime && inT<=(curTime+5) && i<=noOfPres){
                                started = 1;                                    //Compare
whether matched start time if so push to inlist
                                alreadyDone = 1;                                //Set already done to
avoid an extra entry
                                cout << endl << "Entered in presentation " << (i+1) <<endl;
                                chairIn = true;
                                presentations[i]->createListIn(id);     //Push to list in histry
                        }

                        if(inT>(curTime-65) && inT<(curTime-40) ){
                                started = 1;                                    //If entered
after presentation started but before ended
                                ended = 0;
        //Set started flags
                        }

                        if(outT<=curTime && outT>=(curTime-45) && i<=noOfPres){
                                ended = 1;
        //Compare whether matched end time if so push to outlist
                                cout << "Out from presentation " << (i) <<endl;
                                presentations[i-1]->createListOut(id);
                                //chairIn = false;                              //Push to list out
histry
                        }

                        if(outT>(curTime-70) && outT<(curTime-45) ){
                                started = 0;                                    //If exit
after presentation started but before ended
                                ended = 1;
        //Set ended flag
                                //chairIn = false;
                                break;
        //And break the loop
                        }

                        if(ended)                                               //If ended
set break from loop (ie. deleget left session)
                                break;
                }

                else{
        //If the current time is in format HH00
                        curTime = curTime + 30;                                 //Start time for
presentation i and end time for i-1

                        if(inT>=curTime && inT<=(curTime+5) && i<=noOfPres){
                                started = 1;                                    //Compare
whether matched start time if so push to inlist
                                alreadyDone = 1;                                //Set already done to
avoid an extra entry
                                cout << endl << "Entered in presentation " << (i+1) <<endl;
                                chairIn = true;
                                presentations[i]->createListIn(id);     //Push to list in histry
                        }

                        if(inT>(curTime-25) && inT<(curTime)){
                                started = 1;                                    //If entered
after presentation started but before ended
                                ended = 0;
        //Set started flags
                        }

                        if(outT<=curTime && outT>=(curTime-5) && i<=noOfPres){
                                ended = 1;
        //Compare whether matched end time if so push to outlist
                                cout << "Out in presentation " << (i) <<endl;
```

```
                                        presentations[i-1]->createListOut(id);
                                        //chairIn = false;                                      //Push to list out
histry
                                }

                                if(outT>(curTime-30) && outT<(curTime-5)){
                                        started = 0;                                                    //If exit
after presentation started but before ended
                                        ended = 1;
        //Set ended flag
                                        //chairIn = false;
                                        break;
        //And break the loop
                                }

                                if(ended)                                                               //If ended
set break from loop (ie. deleget left session)
                                        break;
                        }

                        if(started==1 && ended==0 && i<noOfPres){     //If started set at the end of each presentation
deleget

        //was present at the end of presentation i and beginning

        //of presentation i+1
                                if(alreadyDone)                                                         //If already
done by the above loop ignore once
                                        alreadyDone = 0;
                                else{
        //Else push to the appropriate list
                                        cout << "Exit from presentation " << (i) <<endl;
                                        chairIn = true;
                                        presentations[i-1]->createListOut(id);

        //Push to list in histry
                                        cout << "Entered in presentation " << (i+1) <<endl;
                                        presentations[i]->createListIn(id);         //Push to list out histry

                                }
                        }
                }

        if(outT>endTime){                                                                       //Else delegate
stayed after session entered
                        cout << "Exit from presentation " << (i-1) << endl;
                        presentations[i-2]->createListOut(id);
                        return 2;                                                                       //And exit
        }

        return 1;
}


//----------Session member function (setPresenters) - sets presenter ID-------------------------

void SESSION::setPresenters(int prNo,int prId){

        //Sets the presenter ID for each presentation
        presentations[prNo]=new PRESENTATION;
        presentations[prNo]->setId(prId);
        cout << "PRESENTATION ID: " << presentations[prNo]->getId() << endl;
}


//-----------------Session member function (setEndTime) - Sets the end time and-----------------
//---------------------------number of presentations for each session--------------------------
```

```
void SESSION::setEndTime(int presNo){
	char *strStartTime = new char[5];
	sprintf(strStartTime,"%d",startTime);
	int preTime = startTime;

	noOfPres = presNo;                                                    //Set the
number of presentations

	if(strStartTime[strlen(strStartTime)-2]=='3'){	//If the current time is in format HH30
		switch(presNo){                                                   //Calculate
end time appropriately
			case 3:
				endTime = startTime + 170;		//Ex. 0930+170 =>
9.30+0.30*3 = 1100
				break;
			case 4:
				endTime = startTime + 200;
				break;
			case 5:
				endTime = startTime + 270;
				break;
			default:
				endTime = startTime;
				break;
		}
	}
	else{
	//If the current time is in format HH00
		switch(presNo){                                                   //Calculate
end time appropriately
			case 3:

				endTime = startTime + 130;		//Ex. 0900+130 =>
9.30+0.30*3 = 1030
				break;
			case 4:
				endTime = startTime + 200;
				break;
			case 5:
				endTime = startTime + 230;
				break;
			default:
				endTime = startTime;
				break;
		}
	}
	for(int i=0; i<presNo; i++){                          //Add presentation times as well
		presentations[i]->setStartTime(preTime);
		sprintf(strStartTime,"%d",preTime);
		if(strStartTime[strlen(strStartTime)-2]=='3')
			preTime += 70;
		else
			preTime += 30;
		presentations[i]->setEndTime(preTime);
	}
}


//----------Session member function (compareSession) - check whther given details---------------
//-----------------------------the current sessin object---------------------------------------

string SESSION::compareSession(int d, int Time, int rmN){

	if(day==d && roomNo==rmN)                                         //If day, room are equal and
		if(Time>=startTime && Time <= endTime)	//within the time limits return presentation name
			return sessionName;

	return "\0";                                                      //Else return NULL
```

```
}
```

File Presentation.h

```
//Class definitions and member methods
//-------------------------------------------------------------------------------------------


//--------Presentation class - Holds data about presenter ID, start and end time----------------
//------------and a list of attended delegates for each presentation object---------------------

class PRESENTATION{
	int presenterId;
	int startTime;
	int endTime;
	bool presenterIn;
	vector <int> presentIn;
	vector <int> presentOut;

	public:
		PRESENTATION() {presenterId=0; startTime=0; endTime=0; presenterIn=false;}
		void setId(int s) {presenterId=s;}			//Set presenter ID
		void setStartTime(int stTime) {startTime=stTime;}

	//Set start time
		void setEndTime(int enTime) {endTime=enTime;}

	//Set end time
		void createListIn(int id) {presentIn.push_back(id); if(id==presenterId) presenterIn=true;}

	//Enter new delegate attended at the beginning

	//to the vector "presentIn"
		void createListOut(int id) {presentOut.push_back(id);}

	//Enter new delegate attended at the end

	//to the vector "presentIn"
		void printList(vector <DELEGATE> dels,  string sesName, int CP,  bool chairI, int chair);

	//Print a list of delgates present for this presentation
		int getId() {return presenterId;}			//Returns the presenter ID
		int getStartTime() {return startTime;}	//Returns the start time
		int getEndTime() {return endTime;}			//Returns the end time
};


//-------Presentation memeber method (printList) - Prints the name and ID of the----------------
//------------------------delegates present at that presentation-------------------------------

void PRESENTATION::printList(vector <DELEGATE> dels, string sesName, int CP,  bool chairI, int chair){
	int small, countPres = 1;
	//Size of the small vector
	string delSess;
	bool presenterPresCList = false;
	bool chairPresent = false;

	if (presentIn.size()>presentOut.size())		//Get the size of the smallest vector
		small=presentOut.size();
	else
		small=presentIn.size();

	//----------------Check whether the presenter was present------------------------------

	if(!presenterIn || !chairI){					//If presenter was not present
		for (int j=0; j<dels.size(); j++){			//may be in the current list (waited for next
presentation)
```

```
	//Check current list for presenter
				delSess = dels[j].getSessionName();
				if ((delSess == sesName) && (dels[j].getInTime()<=(startTime+5)) &&
presenterId==dels[j].getId())
					presenterPresCList = true;		//If in the current list and stayed at the
begining

	//of the presentation and matches the ID set this

	//dels[j].getInTime()>=startTime &&
				if ((delSess == sesName) && (dels[j].getInTime()<=(startTime+5)) &&
chair==dels[j].getId())
					chairPresent = true;
			}
	}

	if(presenterPresCList || presenterIn){		//If presenter in print details
		cout << endl << "Delegate List For Presentation (" << CP << ") Hosted By: " << presenterId <<
endl;

	//Print the presenter Name----------
		if(!(chairPresent || chairI))
			cout << "Chairperson (" << chair << ") Not Present For This Presentation" << endl;

		for (int i=0; i<small;i++){				//Find the delgates presented at the start
			for (int j=0; j<small;j++) {
				if (presentIn[i] == presentOut[j]){

	//Create the list
					for (int k=0;k<dels.size();k++){

	//Find the name of that delegate
						if (dels[k].getId()==presentIn[i]){

	//Print details
							cout << countPres++ << ". " <<
dels[k].getDelName();
							cout << " (" << dels[k].getId() << ")";
							break;
						}
					}
					cout << endl;
					break;
				}
			}

		}
	}
	else {
		cout << endl << "Delegate List For Presentation (" << CP << ") Hosted By: " << presenterId <<
endl;
		cout << "No Delegates Present [Presenter (ID:" << presenterId << ") Was Absent]" << endl << endl;
	}
}
```