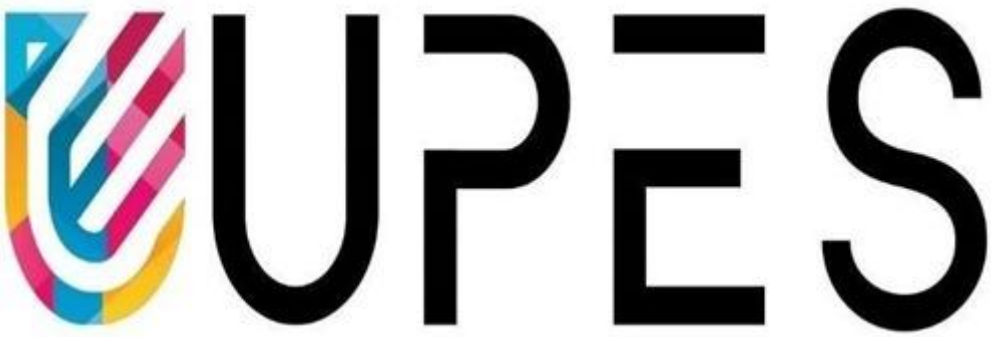

TOPIC:-

University Timetable Management System

Course: Programming In C

(Course Code:- CSEG1041_5)

Submitted by

Om Adlakha

(Sap_Id:- 590025655)

Submitted To

Mr. Mohsin

Furkh Dar

Date of Submission:- 01/12/2025

Problem Statement:-

In many educational institutions, creating a schedule by hand is a time-consuming and prone to error. It becomes difficult to effectively manage subject distribution, faculty availability, and student batches. The project's objective is to develop and implement a C program that arranges and methodically automates the faculty management, student record management, and schedule creation processes.

Objective:-

- To develop a C-based system for automatic timetable generation.
- To manage student records including name, ID, date of birth, and batch.
- To manage faculty records with subject allocation.
- To generate a conflict-free timetable using randomization.
- To allow searching of student and faculty timetables.

Software / Tools Used:-

Operating System:- Windows 11

Compiler:- GCC (MinGW) Compiler

Algorithm of Project:-

1. START
2. INPUT: Load student data, faculty data, and timetable from files.
3. INITIALIZE: Start the random number generator using current time.
4. DISPLAY: Show the main menu with the following options:
    - Add Student
    - Add Faculty
    - Generate Timetable
    - Search Student
    - Search Faculty
    - Exit
5. INPUT: Read the user's choice.
6. PROCESS:
    - If choice = 1, accept and store student details.
    - If choice = 2, accept and store faculty details.
    - If choice = 3, generate the timetable automatically.
    - If choice = 4, search and display student timetable.
    - If choice = 5, search and display faculty timetable.
7. REPEAT: Return to the main menu until the user selects Exit.
8. SAVE: Store all updated student, faculty, and timetable data into files.
9. OUTPUT: Display program exit message.
10. STOP

Pseudocode of Project:-

```
BEGIN TimetableSystem

INPUT current_time
SET random_seed = current_time

CALL LoadStudents
CALL LoadFaculty
CALL LoadTimetable

REPEAT
    DISPLAY "1. Add Students"
    DISPLAY "2. Add Faculty"
    DISPLAY "3. Generate Timetable"
    DISPLAY "4. Search Student"
    DISPLAY "5. Search Faculty"
    DISPLAY "6. Exit"

    INPUT choice

    IF choice == 1 THEN
        CALL AddStudents
    END IF

    IF choice == 2 THEN
        CALL AddFaculty
    END IF

    IF choice == 3 THEN
        CALL GenerateTimetable
    END IF

    IF choice == 4 THEN
        CALL SearchStudent
    END IF

    IF choice == 5 THEN
        CALL SearchFaculty
    END IF

UNTIL choice == 6

CALL SaveStudents
CALL SaveFaculty
CALL SaveTimetable

OUTPUT "Program Ended Successfully"

END TimetableSystem
```

Output Of Program (Screenshots:-)

1:- Adding Student Details:-

```
PS D:\C Language\C PROJECT> gcc src/*.c -I include -o timetable_app
PS D:\C Language\C PROJECT> ./timetable_app.exe

=== Timetable System ===
1. Add Students
2. Add Faculty
3. Generate Timetable
4. Search Student
5. Search Faculty
6. Exit
Choose: 1
How many students? 3
Enter name: Ujjawal Chaurasia
DOB (DD-MM-YYYY): 23-09-2007

Enter name: Aarav Rana
DOB (DD-MM-YYYY): 15-06-2007

Enter name: Jaideep Singh
DOB (DD-MM-YYYY): 12-09-2007
```

2:- Adding Faculty Data:-

```
=== Timetable System ===
1. Add Students
2. Add Faculty
3. Generate Timetable
4. Search Student
5. Search Faculty
6. Exit
Choose: 2
How many faculty? 3
Enter name: Mr. Ashish Bhatnagar
Choose subject:

1. Advanced Engineering Mathematics
2. Programming in C
3. Linux Lab
4. Problem Solving
5. E.V.S

Enter Subject Number the Faculty teaches: 2

Enter name: Mr. Sunil Pujari
Choose subject:

1. Advanced Engineering Mathematics
2. Programming in C
3. Linux Lab
4. Problem Solving
5. E.V.S

Enter Subject Number the Faculty teaches: 3

Enter name: Ms. Damanpreet Kaur
Choose subject:

1. Advanced Engineering Mathematics
2. Programming in C
3. Linux Lab
4. Problem Solving
5. E.V.S

Enter Subject Number the Faculty teaches: 4
```

3:- Generating Timetable:-

```
=== Timetable System ===
1. Add Students
2. Add Faculty
3. Generate Timetable
4. Search Student
5. Search Faculty
6. Exit
Choose: 3
Timetable generated.
```

4:- Searching Student Timetable:-

```
=== Timetable System ===
1. Add Students
2. Add Faculty
3. Generate Timetable
4. Search Student
5. Search Faculty
6. Exit
Choose: 4
Enter student name: Aarav Rana

Student: Aarav Rana(Id: 1013), Batch 1

Monday:
  14:00-15:00  Linux Lab (Mr. Sunil Pujari)
  17:00-18:00  Problem Solving (Mr. Baldivya Mitra)

Tuesday:
  10:00-11:00  Programming in C (Mr. Bhuvam Bam)
  11:00-12:00  Linux Lab (Mr. Sunil Pujari)
  14:00-15:00  Advanced Engineering Mathematics (Ms. Rishu Gandhi)
  15:00-16:00  Advanced Engineering Mathematics (Ms. Rishu Gandhi)

Wednesday:
  08:00-09:00  Programming in C (Mr. Ashish Bhatnagar)
  10:00-11:00  Linux Lab (Dr. Supreet Singh)
  16:00-17:00  Programming in C (Ms. Roshni Singh)
  17:00-18:00  Programming in C (Mr. Mohsin Furkh Dar)

Thursday:
  08:00-09:00  Programming in C (Mr. Ashish Bhatnagar)
  11:00-12:00  Problem Solving (Ms. Damanpreet Kaur)
  15:00-16:00  Problem Solving (Ms. Madhuri Mishra)
  17:00-18:00  E.V.S (Mr. Vaibhav Saini)

Friday:
  09:00-10:00  Linux Lab (Mr. Jitendra Yadav)
  12:00-13:00  Linux Lab (Mr. Jitendra Yadav)
  15:00-16:00  Programming in C (Mr. Bhuvam Bam)

Saturday:
  08:00-09:00  E.V.S (Mr. Vaibhav Saini)
  09:00-10:00  Problem Solving (3)
  12:00-13:00  Linux Lab (Dr. Supreet Singh)
```

5:- Searching Faculty Timetable:-

```
=== Timetable System ===
1. Add Students
2. Add Faculty
3. Generate Timetable
4. Search Student
5. Search Faculty
6. Exit
Choose: 5
Enter faculty name: Mr. Sunil Pujari

Faculty: Mr. Sunil Pujari (Subject: Linux Lab)
Monday:
  08:00-09:00  Linux Lab (Batches: 13)
  12:00-13:00  Linux Lab (Batches: 5, 8)
  14:00-15:00  Linux Lab (Batches: 1)
  17:00-18:00  Linux Lab (Batches: 4, 20)

Tuesday:
  08:00-09:00  Linux Lab (Batches: 5)
  11:00-12:00  Linux Lab (Batches: 1, 13)
  15:00-16:00  Linux Lab (Batches: 9)
  16:00-17:00  Linux Lab (Batches: 8)

Wednesday:
  12:00-13:00  Linux Lab (Batches: 12)
  15:00-16:00  Linux Lab (Batches: 12)

Thursday:
  08:00-09:00  Linux Lab (Batches: 3)
  10:00-11:00  Linux Lab (Batches: 6)
  11:00-12:00  Linux Lab (Batches: 19)
  12:00-13:00  Linux Lab (Batches: 9)
  16:00-17:00  Linux Lab (Batches: 10)

Friday:
  10:00-11:00  Linux Lab (Batches: 8)
  12:00-13:00  Linux Lab (Batches: 21)
  13:00-14:00  Linux Lab (Batches: 19)
  16:00-17:00  Linux Lab (Batches: 13)

Saturday:
  09:00-10:00  Linux Lab (Batches: 6)
  15:00-16:00  Linux Lab (Batches: 15)
```

6:- Saving all changes and Exiting the program

```
=== Timetable System ===
1. Add Students
2. Add Faculty
3. Generate Timetable
4. Search Student
5. Search Faculty
6. Exit
Choose: 6
Saved and exiting.
PS D:\C Language\C PROJECT>
```

Source Code (All .c files):-

1:- main.c

src > C main.c > ...

```c
// Including all standard library and user_defined library.

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

#include "student.h"
#include "faculty.h"
#include "timetable.h"

int main(){
    srand((unsigned)time(NULL)); // Define in stdlib.h that makes your timetable different every time by randomizing subjects and slots.

    load_students();  // Read data from student, faculty and trimetable binary data.
    load_faculty();
    load_timetable();

    int ch;
    while (1){
        printf("\n=== Timetable System ===\n");
        printf("1. Add Students\n");
        printf("2. Add Faculty\n");
        printf("3. Generate Timetable\n");
        printf("4. Search Student\n");
        printf("5. Search Faculty\n");
        printf("6. Exit\n");
        printf("Choose: ");
        scanf("%d", &ch);

        if (ch == 1) add_students();
        else if (ch == 2) add_faculty();
        else if (ch == 3) generate_timetable();
        else if (ch == 4) search_student();
        else if (ch == 5) search_faculty();
        else if (ch == 6){
            save_students();  // Save all changes made in student, faculty and timetable binary file.
            save_faculty();
            save_timetable();
            printf("Saved and exiting.\n");
            break;
        }
        else {
            printf("Invalid Choice! Enter Again\n\n");
        }
    }

    return 0;
}
```

2:- student.c

src > C student.c > ...

```c
#include <stdio.h>
#include <string.h>
#include "student.h"
#include "timetable.h"

Student students[MAX_STUDENTS];
int student_count = 0;
int next_student_id = 1000;

static void flush_stdin(){   // It clears leftover input from the keyboard buffer so the next input works correctly.
    int c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

static int ci_equal(const char *a, const char *b){  // Compare strings and convert uppercase to lowercase.
    while (*a && *b){
        char x = *a, y = *b;
        if (x >= 'A' && x <= 'Z') x += 32;
        if (y >= 'A' && y <= 'Z') y += 32;
        if (x != y) return 0;
        a++; b++;
    }
    return *a == 0 && *b == 0;
}

void save_students(){
    FILE *f = fopen("studentsB.dat","wb");
    if (!f) return;
    fwrite(&student_count, sizeof(int), 1, f);
    fwrite(&next_student_id, sizeof(int), 1, f);
    fwrite(students, sizeof(Student), student_count, f);
    fclose(f);
}

void load_students(){
    FILE *f = fopen("studentsB.dat","rb");
    if (!f) return;
    fread(&student_count, sizeof(int), 1, f);
    fread(&next_student_id, sizeof(int), 1, f);
    fread(students, sizeof(Student), student_count, f);
    fclose(f);
}

```

src > C student.c > ...

```c
void add_students(){
    int n;
    printf("How many students? ");
    if (scanf("%d", &n) != 1){
        flush_stdin();
        return;
    }
    flush_stdin();

    for (int i=0;i<n;i++){
        if (student_count >= MAX_STUDENTS){
            printf("Limit reached.\n");
            break;
        }

        Student S;
        S.student_id = next_student_id++;

        printf("Enter name: ");
        fgets(S.name, MAX_NAME, stdin);
        S.name[strcspn(S.name, "\n")] = 0;  // strcspn() removes extra \n for clear result.

        printf("DOB (DD-MM-YYYY): ");
        scanf("%d-%d-%d", &S.dob.day, &S.dob.month, &S.dob.year);
        flush_stdin();

        S.batch = (student_count / MAX_STUDENTS_PER_BATCH) + 1;
        students[student_count++] = S;

        printf("\n");
    }

    save_students();
}

void search_student(){
    char student_name[MAX_NAME];
    printf("Enter student name: ");
    flush_stdin();
    fgets(student_name, MAX_NAME, stdin);
    student_name[strcspn(student_name, "\n")] = 0;

    for (int i=0;i<student_count;i++){
        if (ci_equal(student_name, students[i].name)){
            printf("\nStudent: %s(Id: %d), Batch %d\n", students[i].name, students[i].student_id, students[i].batch);
            printf("\n");
            print_batch_timetable(students[i].batch);
            return;
        }
    }
    printf("Not found.\n");
}
```

3:- faculty.c

src > C faculty.c > add_faculty()

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "faculty.h"
#include "timetable.h"

Faculty fac[MAX_FACULTY];
int faculty_count = 0;
int next_faculty_id = 1;

static void flush_stdin(){  // It clears leftover input from the keyboard buffer so the next input works correctly.
    int c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

static int randint(int a, int b){
    return a + rand() % (b - a + 1);
}

static int ci_equal(const char *a, const char *b){  // // Compare strings and convert uppercase to lowercase.
    while (*a && *b){
        char x = *a, y = *b;
        if (x >= 'A' && x <= 'Z') x += 32;
        if (y >= 'A' && y <= 'Z') y += 32;
        if (x != y) return 0;
        a++; b++;
    }
    return *a == 0 && *b == 0;
}

void save_faculty(){
    FILE *f = fopen("facultyB.dat","wb");
    if (!f) return;
    fwrite(&faculty_count, sizeof(int), 1, f);
    fwrite(&next_faculty_id, sizeof(int), 1, f);
    fwrite(fac, sizeof(Faculty), faculty_count, f);
    fclose(f);
}

void load_faculty(){
    FILE *f = fopen("facultyB.dat","rb");
    if (!f) return;
    fread(&faculty_count, sizeof(int), 1, f);
    fread(&next_faculty_id, sizeof(int), 1, f);
    fread(fac, sizeof(Faculty), faculty_count, f);
    fclose(f);
}
```

src > C faculty.c > add_faculty()

```c
void add_faculty(){
    int n;
    printf("How many faculty? ");
    scanf("%d", &n);
    flush_stdin();

    for (int i=0;i<n;i++){
        if (faculty_count >= MAX_FACULTY){
            printf("Limit reached.\n");
            break;
        }

        Faculty F;
        F.faculty_id = next_faculty_id++;

        printf("Enter name: ");
        fgets(F.name, MAX_NAME, stdin);
        F.name[strcspn(F.name, "\n")] = 0;

        printf("Choose subject:\n\n");
        for (int s=0;s<SUBJECT_COUNT;s++)
            printf("%d. %s\n", s+1, subjects[s]);

        printf("\n");
        int c;
        printf("Enter Subject Number the Faculty teaches: ");
        scanf("%d", &c);

        if (c > SUBJECT_COUNT) {
            printf("Invalid Option. Please Select Again\n");

            printf("Enter Subject Number the Faculty teaches: ");
            scanf("%d", &c);
        }
        flush_stdin();
        F.subject_id = c - 1;

        fac[faculty_count++] = F;

        printf("\n");
    }

    save_faculty();
}
```

src > C faculty.c > ...

```c
int random_faculty_for_subject(int subj){
    int list[MAX_FACULTY], count = 0;
    for (int i=0;i<faculty_count;i++){
        if (fac[i].subject_id == subj)
            list[count++] = fac[i].faculty_id;
    }
    if (count == 0) return 0;

    return list[rand() % count];
}

void search_faculty(){
    char faculty_name[MAX_NAME];
    printf("Enter faculty name: ");
    flush_stdin();
    fgets(faculty_name, MAX_NAME, stdin);
    faculty_name[strcspn(faculty_name, "\n")] = 0;

    printf("\n");

    for (int i=0;i<faculty_count;i++){
        if (ci_equal(faculty_name, fac[i].name)){
            print_faculty_timetable_grouped(fac[i].faculty_id);
            return;
        }
    }
    printf("Not found.\n");
}
```

4:- timetable.c

src > C timetable.c > ...

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

#include "timetable.h"
#include "faculty.h"
#include "student.h"
#include "common.h"

Cell timetable[MAX_BATCHES][DAYS][SLOTS];

/* Subject names */
const char *subjects[SUBJECT_COUNT] = {
    "Advanced Engineering Mathematics", "Programming in C", "Linux Lab", "Problem Solving", "E.V.S"
};

/* Day names */
const char *daynames[DAYS] = {
    "Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"
};

int randint(int a, int b){
    return a + rand() % (b - a + 1);
}

/* Count how many batches a faculty teaches at the same time */
int faculty_slot(int faculty_id, int day, int slot){
    int count = 0;

    for (int b = 0; b < MAX_BATCHES; b++){
        if (timetable[b][day][slot].faculty_id == faculty_id){
            count++;
        }
    }
    return count;     /* Maximum allowed = 2 */
}

void save_timetable(){
    FILE *f = fopen("timetable8.dat","wb");
    if (!f) return;

    fwrite(timetable, sizeof(Cell), MAX_BATCHES * DAYS * SLOTS, f);
    fclose(f);
}

void load_timetable(){
    FILE *f = fopen("timetable8.dat","rb");
    if (!f){
        /* initialize empty timetable */
        for (int b=0; b<MAX_BATCHES; b++)
            for (int d=0; d<DAYS; d++)
                for (int s=0; s<SLOTS; s++){
                    timetable[b][d][s].subject_id = -1;
                    timetable[b][d][s].faculty_id = 0;
                    timetable[b][d][s].hour = 8 + s;
                }
        return;
    }

    fread(timetable, sizeof(Cell), MAX_BATCHES * DAYS * SLOTS, f);
    fclose(f);
}

```

src > C timetable.c > ...

```c
void generate_timetable(){

    /* clear timetable first */
    for (int b=0; b<MAX_BATCHES; b++)
        for (int d=0; d<DAYS; d++)
            for (int s=0; s<SLOTS; s++){
                timetable[b][d][s].subject_id = -1;
                timetable[b][d][s].faculty_id = 0;
                timetable[b][d][s].hour = 8 + s;
            }

    /* generate timetable */
    for (int b=0; b<MAX_BATCHES; b++){
        for (int d=0; d<DAYS; d++){

            int periods = randint(MIN_PERIODS, MAX_PERIODS);
            int used[SLOTS] = {0};
            int done = 0;

            while (done < periods){
                int slot = randint(0, SLOTS-1);
                if (used[slot]) continue;
                used[slot] = 1;

                int subj = randint(0, SUBJECT_COUNT-1);
                int assigned_faculty = 0;

                /* Try to find a faculty who is free (max 2 loads allowed) */
                for (int try = 0; try < 20; try++){
                    int fid = random_faculty_for_subject(subj);
                    if (fid == 0) break;

                    if (faculty_slot(fid, d, slot) < 2){
                        assigned_faculty = fid;
                        break;
                    }
                }

                timetable[b][d][slot].subject_id = subj;
                timetable[b][d][slot].faculty_id = assigned_faculty;

                done++;
            }
        }
    }

    save_timetable();
    printf("Timetable generated.\n");
}
```

src > C timetable.c > ...

```c
void print_batch_timetable(int batch){
    if (batch < 1 || batch > MAX_BATCHES){
        printf("Invalid batch.\n");
        return;
    }

    int b = batch - 1;

    for (int d=0; d<DAYS; d++){
        printf("%s:\n", daynames[d]);

        int any = 0;

        for (int s=0; s<SLOTS; s++){
            if (timetable[b][d][s].subject_id != -1){
                any = 1;

                int hour = timetable[b][d][s].hour;
                int subj = timetable[b][d][s].subject_id;
                int fid = timetable[b][d][s].faculty_id;

                char fname[MAX_NAME] = "TBA";

                if (fid != 0){
                    for (int i=0; i<faculty_count; i++){
                        if (fac[i].faculty_id == fid){
                            strcpy(fname, fac[i].name);
                            break;
                        }
                    }
                }

                printf("  %02d:00-%02d:00  %s (%s)\n",
                       hour, hour+1, subjects[subj], fname);
            }
        }
        printf("\n");

        if (!any) printf("  No classes\n");
    }
}

```

src > C timetable.c > ...

```c

void print_faculty_timetable_grouped(int faculty_id){

    int fidx = -1;
    for (int i=0; i<faculty_count; i++){
        if (fac[i].faculty_id == faculty_id){
            fidx = i;
            break;
        }
    }

    if (fidx == -1){
        printf("Faculty not found.\n");
        return;
    }

    printf("Faculty: %s (Subject: %s)\n",
           fac[fidx].name,
           subjects[fac[fidx].subject_id]);

    for (int d=0; d<DAYS; d++){
        printf("%s:\n", daynames[d]);

        int any = 0;

        for (int s=0; s<SLOTS; s++){
            int batches[3];
            int count = 0;

            for (int b=0; b<MAX_BATCHES; b++){
                if (timetable[b][d][s].faculty_id == faculty_id){
                    if (count < 2){            /* show max 2 only */
                        batches[count] = b + 1;
                    }
                    count++;
                }
            }

            if (count > 0){
                any = 1;
                int hour = 8 + s;

                printf("  %02d:00-%02d:00  %s (Batches: ",
                       hour, hour+1, subjects[fac[fidx].subject_id]);

                int show = count;
                if (show > 2) show = 2;        /* print max 2 */

                for (int k=0; k<show; k++){
                    printf("%d", batches[k]);
                    if (k+1 < show) printf(", ");
                }

                printf(")\n");
            }
        }
        printf("\n");

        if (!any) printf("  No classes\n");
    }
}
```

Conclusion:-

The Timetable Management System was successfully designed and implemented using the C programming language. The program efficiently manages student and faculty records and generates an automatic schedule with suitable subject and faculty allocation. This project helped me understand file handling, structures, arrays, and modular programming in C. The system successfully accomplishes the project's original objectives.

Future Enhancements:-

- Adding a option to search timetable by entering batch number.
- Adding a option to show list of students in a batch number entered by user.
- Adding a option to show list of faculties by the subject they teach (subject was entered by user).