**PRE-SUBMISSION REPORT**

**Title: Student Attendance Calculator**

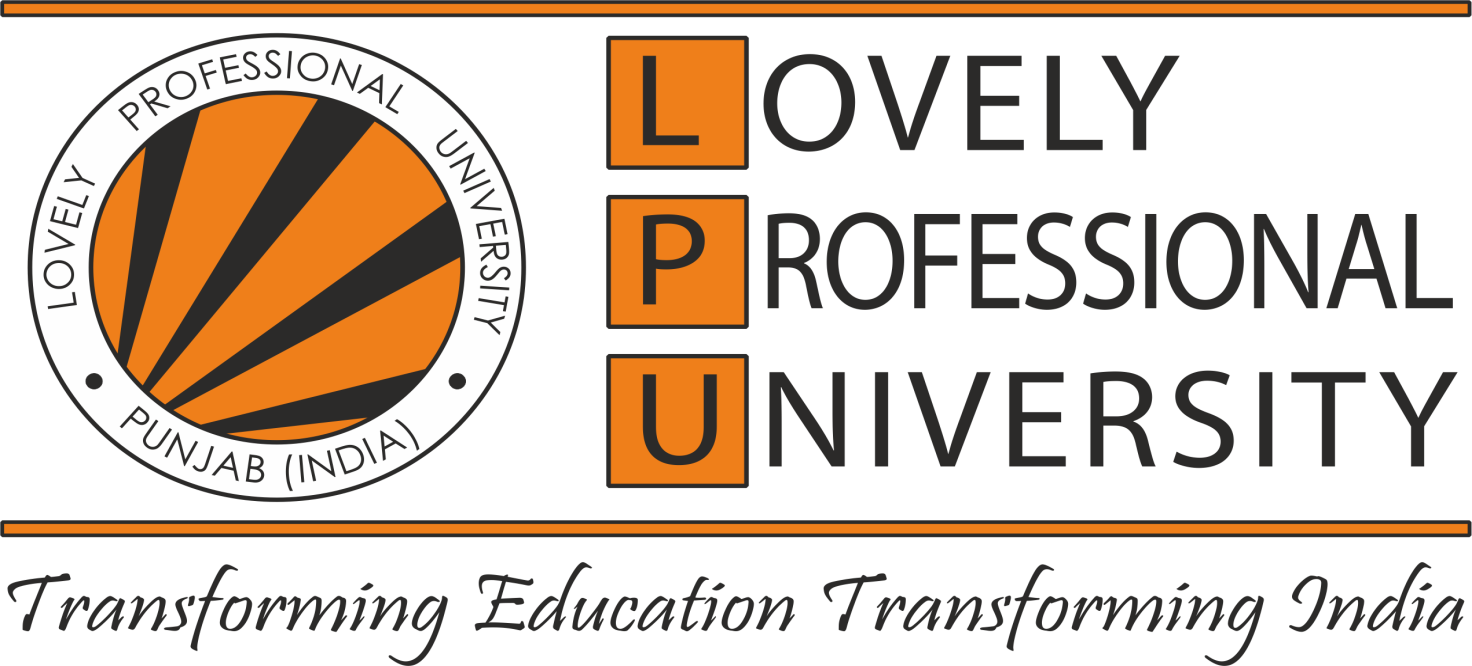
**Section – K22MT**

As a Code based project for the Course

**CSE 101**

### **By**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Registration No | Name of Student | Roll No. |
| **1** |  |  |  |



### **Submitted To Richa Jain Ma’am**

Lovely Professional University

ACKNOWLEDGEMENT

My profound gratitude goes out to , the CSE 101 instructor, for giving me the chance to work on the project of creating a student attendance calculator.

She was a great help during the project, and I want to thank her for that. Her knowledge of the topic, along with her readiness to offer suggestions and respond to our inquiries, enabled me to effectively complete the project.

Finally, I want to thank the university for giving me the tools I needed to complete this research. I am appreciative that I had the chance to work on this project because it has been a great opportunity for me to learn.

INTRODUCTION OF THE PROJECT

**Title – Student attendance calculator**

Objective – This program is a simple student attendance management system implemented in C language. It allows the user to insert new student records, display all existing records, search and modify records by roll number or name, and delete records by roll number. The system requires the user to log in using a predefined username and password, and only the admin can access the system's features.

The program uses a struct to store student information such as name, roll number, attendance for each course, and total attendance. The program can calculate the total attendance of each student by taking the sum of the attended classes divided by the total classes for each course and calculating the average.

The program provides the option to search for students by roll number or name and display their attendance information. It also provides a warning message for students with attendance below 75%. The program uses arrays to store and manipulate student information, with a maximum of 100 students and 8 courses.

The program uses basic input/output functions such as printf, scanf, and gets to interact with the user. It also uses string and math libraries for string and arithmetic operations. The code is well-structured and organized with meaningful function names and comments. Overall, this program provides a useful tool for tracking and managing student attendance records.

IMPLEMENTATION

**Methodology**

A simple system for managing student attendance is implemented in the provided code. The code consists of numerous functions for carrying out various tasks, such as adding student information, showing all student information, looking up a student by name or roll number, removing a student record, and changing the information for an already-existing student.

1. **Libraries**
   1. Three common C libraries are used in the code: string.h, stdlib.h, and stdio.h.
   2. Input and output operations like printf() and scanf are performed using stdio.h ().
   3. Stdlib.h is used to provide functions for dynamic memory allocation, like malloc ().
   4. String manipulation techniques like strcmp are performed using the string.h library ().
2. **Constants**
   1. Two constants, MAX\_STUDENTS and MAX\_COURSES, with the values 100 and 8, respectively, are defined in the code. These constants are used to determine how many students can be added to the system at once and how many classes can have attendance taken.
3. **Structures**
   1. A structure called student is defined in the code, and it maintains information about each student, such as name, roll number, attendance for each course, and overall attendance. The information about every student introduced to the system is contained in the struct student, which is an array of structures students.
4. **Variables**
   1. The integer variable total \_students store the overall number of students added to the system.
   2. The login credentials are stored in character arrays called username and password.
   3. Integer variables utilized in various functions include I j, total classes, missed classes, attended classes, total attended classes, and total\_total \_classes.
   4. The float variables attendance and total attendance are utilized in a variety of functions.

**Functions**

The code consists of several functions, each performing a specific task.

**1. login ()**

The user's login credentials are verified using the login() function. It asks the user to provide a username and password and then confirms that the entered information corresponds to the pre-defined information. It returns 1 to indicate a successful login if the credentials match; else, it returns 0 to indicate an unsuccessful login.

**2. add\_student ()**

To enter a new student's information into the system, use the insert student() function. For each course, the user is prompted to enter the student's name, roll number, and attendance. The total attendance field of the student's structure is where it maintains the attendance total for each student after computing it. It displays a warning message if the student has less than 75% overall attendance. The student's information is added to the students array.

**3. display\_students()**

All of the student information entered into the system is displayed using the display students() function. It displays the total new enrollment, along with each student's name, roll number, overall attendance, and attendance for each course.

**4. search\_student\_by\_roll\_number()**

To find a student's information by their roll number, use the search\_student \_by \_roll \_number() function. The user is prompted to provide the roll number before it searches the students array for the student's information. When a student is located, their name, roll number, and attendance for each course are displayed. If not, a notice stating that the student was not found is displayed.

**5. student\_search\_by\_name()**

To look up a student's information by name, use the search student by name() function. It asks for the user's name and then looks up the student's information in the students array. When a student is located, their name, roll number, and attendance for each course are displayed. If not, a notice stating that the student was not found is displayed.

**6. update\_attendance()**

A student's attendance for a particular course can be updated using the update attendance() function. While updating attendance for a course, the user is prompted to enter the course code and the student's roll number. After then, the user is prompted to enter the updated attendance value. The student's attendance for the chosen course is updated, and the student's overall attendance is recalculated, if the student is located and the new attendance figure is valid. A warning notice is sent if the revised attendance figure results in a decrease in the student's overall attendance below 75%.

**7. delete\_student()**

To remove a student's information from the system, use the delete student() function. The user is prompted for the student's roll number before it searches the student array for the student's information. The student's information is deleted from the students array if the student is located.

**8. exit\_program()**

To end the programme, use the exit program() function. The software is terminated once a notice stating that it is quitting is displayed.

**9.File handling**

The software can use file handling to save and load student data to and from files. The programmer stores the student data in a CSV (comma-separated values) file. The student's name, roll number, and attendance for each course are all kept in the CSV file.

The programmer first determines whether the current working directory has the file "students.csv" when it launches. If the file is there, the application uses file handling routines to load the student data into the students array. The programmer initialises the students array to an empty list if the file does not exist.

When a user uses the insert student() function to add a new student, the software appends the new student's data to the students array and then uses file handling functions to write the updated data to the CSV file. Similar to this, the software receives the data from the CSV file using file handling functions and prints the data to the console when the user examines the student data using the display students() function.

The software may persist the data even after it is closed and restarted by using file handling to store and retrieve the student data to and from a file.

CONCLUSION

Ultimately, these features combine to offer a fundamental student management system. The user has access to updating attendance, adding new students, seeing all students, searching for students by name or roll number, deleting students, and exiting the software after logging in. While being a simple system, it serves as an excellent foundation for creating more intricate student management systems.