PBL REPORT

SUBMITTED BY	:	**ESHA ARAIN**
SUBMITTED T0	:	**ENGR ABDUL AHAD**
ROLL NO	:	**2023-CPE-09**
SEMESTER	:	**4th**
SESSION	:	**2023-2027**
COURSE TITLE	:	**DSA**
COURSE CODE	:	**CPE-221P**
PBL TITLE	:	**Attendance Tracker**
DATE	:	**15th May, 2025**

Department Of Computer Engineering, Multan

Course Title	**DATA STRUCTURES AND ALGORITHMS**
Course Code	CPE-221
Credit Hours	04 (3+1)
Pre-requisite	Computer Fundamentals (CPE-111)
Course In-charge	Dr. Mudasir Khalil
Lab Engineer	Engr. Abdul Ahad

Course Information:

Problem based learning:

An activity is said to be stimulating students' problem-based learning if it requires

Mandatory

- sufficient knowledge of elementary concepts directly related to the given problem (depth)
- students to make a choice amongst a pool of possible solutions/methods/theorems/tools available for the given problem
- a healthy team/individual effort

Optional

- usage of modern tools and already gained sufficient hands-on experience of those tools
- investigation of experimental data and verification of the achieved results by means of simulations, tests, and\or formal methods, if applicable
- substantial knowledge of various subjects that lay the foundation of engineering (breadth)

An Exercise Problem Based Learning

1:

Problem:
Create an attendance manager where you can add, search, and list students marked present or absent**.**
Features: • Add student name/roll and mark status • Search if a student is present/absent • View full attendance list • Count Total count of present & absent .
DSA Concepts: • Array / Vector – To store attendance records • Linear Search – To check if student is marked • Sorting – To alphabetically organize list

Summary:

Following are salient outcomes of the practical problem in terms of problem-based learning:

• Brainstorming exercise forced them to explore the surrounding environment to sort out the problems to be solved using image processing.

• Problem formulation enhances their ability to gather real-time requirements and address conflicts/constraints.

• Design/Implementation gave them a chance to go through the in-depth engineering knowledge to solve the problem and analyze in an effective way.

DECLARATION

I hereby declare that this project work entitled "**ATTENDANCE TRACKER**" has been prepared by me during the year 2025 under the guidance of **Engr Abdul Ahad**, Department of Computer Engineering, Bahauddin Zakariya University, Multan, Pakistan, in the partial fulfillment of BSc. degree prescribed by the university.

I also declare that this project is the outcome of my own effort, that it has not been submitted to any other university for the award of any degree.

Name Of Student	**Esha Arain**
Roll No	**2023-CPE-09**
Semester	**4th**
Session	**2023-2027**

Engr. Abdul Ahad
(Laboratory Engineer)
Dept. of Computer Engineering,
BZU, Multan

Dr. Mudassir Khalil
(Course Instructor)
Dept. of Computer Engineering,
BZU, Multan

ACKNOWLEDGEMENT

I wish to express my deepest gratitude to Laboratory Engineer **Engr. Abdul Ahad** who guided me on every step of making this project. It is whole-heartedly appreciated that your great advice for my project proved monumental towards the success and cooperation of this project.

I would like to express my gratitude towards my parents & friends for their kind cooperation and encouragement which help me in completion of this project.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped us out with their abilities

ABSTRACT

The Attendance Management System (AMS) is a console-based C++ application designed to automate attendance tracking in educational institutions. Key features include:

- Course information management
- Student record maintenance (add/delete/sort)
- Daily attendance marking (Present/Absent)
- Attendance report generation
- Data persistence using file handling

The system demonstrates practical implementation of OOP concepts, file handling, and data structures, offering an efficient alternative to manual attendance tracking.

Table of Contents

LIST OF FIGURES

ATTENDANCE MANAGEMNET SYSTEM

1. Introduction

The **Student Attendance Management System** is a console-based application developed in C++ to manage and track student attendance for academic courses. The system provides functionalities for course information management, student record keeping, attendance marking, and report generation.

1.1 Objectives

- To automate the attendance tracking process.
- To maintain student records efficiently.
- To generate attendance reports for specific dates.
- To allow easy updates and modifications to attendance records.
- To provide a user-friendly interface for educators.

1.2 Scope

- Designed for educational institutions to manage course-wise attendance.
- Supports adding, deleting, and sorting student records.
- Allows marking and updating attendance for different dates.
- Generates attendance reports in a structured format.

CHAPTER 2: THEORETICAL FOUNDATION

2.1 OOP Concepts

- Encapsulation (structs for data)
- File handling for persistence

2.2 Data Structures

- Arrays for student storage
- String manipulation for data processing

2.3 System Architecture

Three-tier design:

1. **Presentation Layer:** Console interface
2. **Logic Layer:** C++ functions
3. **Data Layer:** Text files

CHAPTER 3: IMPLEMENTATION

3.1 Data Structures

The system uses the following data structures:

1. CourseInfo **Struct** – Stores course-related details:
    - University name
    - Department
    - Course name and code
    - Semester and session
    - Instructor name
2. Student **Struct** – Stores student details:
    - Name
    - Roll number

3.2 File Handling

- course.dat – Stores course information.
- students.dat – Stores student records.
- attendance_<date>.txt – Stores attendance records for a specific date.

3.3 Functions

3.3.1 Course Management

- setCourseInfo() – Sets and saves course details.
- loadCourseInfo() – Loads course details from file.

3.3.2 Student Management

- addStudent() – Adds a new student.
- deleteStudent() – Deletes a student record.

CODE:

```
void deleteStudent() {
char c;
do{
    if(totalStudents == 0) {
        cout << "No students to delete!\n";
        return;
    }
viewStudents();
    char rollNo[20];
    cout << "Enter Roll Number to delete: ";
    cin >> rollNo;

    int foundIndex = -1;
    for(int i = 0; i < totalStudents; i++) {
        if(strcmp(students[i].rollNo, rollNo) == 0) {
            foundIndex = i;
            break;
        }
    }

    if(foundIndex == -1) {
        cout << "Student not found!\n";
        return;
    }

    for(int i = foundIndex; i < totalStudents-1; i++) {
        strcpy(students[i].name, students[i+1].name);
        strcpy(students[i].rollNo, students[i+1].rollNo);
    }

    totalStudents--;
    saveStudentsToFile();
    cout << "Student deleted successfully!\n";

    cout<<"Remaining students are :\n";
    viewStudents();
    cout<<"to continue delete press c or any other key to exit:\n";
    c=getch();
    if(tolower(c)=='c')
    clearScreen();
}while(tolower(c)=='c');

}
```

Figure 1del student code

- viewStudents() – Displays all students.
- sortStudents() – Sorts students alphabetically.

CODE:

```
void sortStudents() {


    if(totalStudents == 0) {
        cout << "No students to sort!\n";
        return;
    }

    for(int i = 0; i < totalStudents-1; i++) {
        for(int j = 0; j < totalStudents-i-1; j++) {
            if(strcmp(students[j].name, students[j+1].name) > 0) {
                Student temp = students[j];
                students[j] = students[j+1];
                students[j+1] = temp;
            }
        }
    }

    saveStudentsToFile();
    cout << "Students sorted alphabetically!\n";
    viewStudents();
}
```

Figure 2 sort students code

3.3.3 Attendance Management

- markAttendance() – Records attendance for a given date.
- searchUpdateAttendance() – Searches and updates attendance records.
- displayDayAttendance() – Displays attendance for a specific day.

3.3.4 Utility Functions

- clearScreen() – Clears the console.
- displayHeader() – Displays course information.
- saveStudentsToFile() / loadStudentsFromFile() – Manages student data persistence.

CHAPTER 4: WORK FLOW

4.1 Main Workflow

4.1.1 Course Setup

- The instructor enters course details (university, department, course name, etc.).

CODE:

```
struct CourseInfo {
    char university[50];
    char department[50];
    char courseName[50];
    char courseCode[20];
    char semester[20];
    char session[20];
    char instructor[50];
};

struct Student {
    char name[50];
    char rollNo[20];
};
```

Figure 3 structure code

- The data is saved in course.dat

.

CODE:

```
void setCourseInfo() {
    cout << "\nEnter University Name: ";
    cin.ignore();
    cin.getline(currentCourse.university, 50);

    cout << "Enter Department: ";
    cin.getline(currentCourse.department, 50);

    cout << "Enter Course Name: ";
    cin.getline(currentCourse.courseName, 50);

    cout << "Enter Course Code: ";
    cin.getline(currentCourse.courseCode, 20);

    cout << "Enter Semester: ";
    cin.getline(currentCourse.semester, 20);

    cout << "Enter Session: ";
    cin.getline(currentCourse.session, 20);

    cout<< "Enter Instructor Name:";
     cin.getline(currentCourse.instructor,50);

    courseInfoSet = true;

    ofstream courseFile("course.dat");
    courseFile << currentCourse.university << "\n"
               << currentCourse.department << "\n"
               << currentCourse.courseName << "\n"
               << currentCourse.courseCode << "\n"
               << currentCourse.semester << "\n"
               << currentCourse.session<<"\n"
               << currentCourse.instructor;
    courseFile.close();

    cout << "\nCourse information saved successfully!\n";
}
```

```
void loadCourseInfo() {
    ifstream courseFile("course.dat");
    if(!courseFile) return;

    courseFile.getline(currentCourse.university, 50);
    courseFile.getline(currentCourse.department, 50);
    courseFile.getline(currentCourse.courseName, 50);
    courseFile.getline(currentCourse.courseCode, 20);
    courseFile.getline(currentCourse.semester, 20);
    courseFile.getline(currentCourse.session, 20);
    courseFile.getline(currentCourse.instructor,50);

    courseInfoSet = true;
    courseFile.close();
}
```

Figure 4 save/load code

4.1.2 Student Management

- Students are added with their name and roll number.
- Records are stored in students.dat.

CODE:

```
void saveStudentsToFile() {
    ofstream outFile("students.dat");
    for(int i = 0; i < totalStudents; i++) {
        outFile << students[i].rollNo << "," << students[i].name << "\n";
    }
    outFile.close();
}

void loadStudentsFromFile() {
    ifstream inFile("students.dat");
    if(!inFile) return;

    char line[100];
    while(inFile.getline(line, 100)) {
        char* rollNo = strtok(line, ",");
        char* name = strtok(NULL, ",");

        if(rollNo && name) {
            strcpy(students[totalStudents].rollNo, rollNo);
            strcpy(students[totalStudents].name, name);
            totalStudents++;
        }
    }
    inFile.close();
}
```

Figure 5 save students code

4.1.3 Attendance Marking

- The instructor selects a date and marks attendance (Present/Absent).
- Records are saved in attendance_<date>.txt.

CODE:

```
void markAttendance() {
    if(!courseInfoSet) {
        cout << "Please set course information first!\n";
        return;
    }

    if(totalStudents == 0) {
        cout << "No students added yet!\n";
        return;
    }

    char date[20];
    cout << "Enter date (DD-MM-YYYY): ";
    cin >> date;

    char filename[50] = "attendance_";
    strcat(filename, date);
    strcat(filename, ".txt");

    ofstream file(filename);

    file << "University: " << currentCourse.university << "\n";
    file << "Department: " << currentCourse.department << "\n";
    file << "Course: " << currentCourse.courseName << " (" << currentCourse.courseCode << ")\n";
    file << "Semester: " << currentCourse.semester << " | Session: " << currentCourse.session << "\n";
    file << "Attendance Date: " << date << "\n";
    file << "---------------------------------\n";
    file << "RollNo\t\tName\t\t\tStatus\n";
```

Figure 6

```
    for(int i = 0; i < totalStudents; i++) {
        char status;
        bool validInput = false;

        while(!validInput) {
            cout << students[i].name << " (Roll No: " << students[i].rollNo << ") - Present (P/A)? ";
            cin >> status;

            // Convert to uppercase for case-insensitive comparison
            status = toupper(status);

            if(status == 'P' || status == 'A') {
                validInput = true;
                file << students[i].rollNo << "\t" << students[i].name << "\t";

                if(status == 'P')
                    file << "Present\n";
                else
                    file << "Absent\n";
            } else {
                cout << "Invalid entry! Please enter P for Present or A for Absent.\n";
                // Clear any remaining input in case user entered multiple characters
                //cin.ignore(numeric_limits<streamsize>::max(), '\n');
            }
        }
    }

    file.close();
    cout << "\nAttendance saved to " << filename << "\n";
}
```

Figure 7 mark attendance code

4.1.4 Attendance Reports & Updates

- View attendance for any date.
- Search and update attendance records.

CODE:

```
void displayDayAttendance() {


    char date[20];
    cout << "Enter date to view (DD-MM-YYYY): ";
    cin >> date;

    char filename[50] = "attendance_";
    strcat(filename, date);
    strcat(filename, ".txt");

    ifstream file(filename);
    if(!file) {
        cout << "No attendance record for " << date << "!\n";
        return;
    }

    cout << "\nATTENDANCE RECORD FOR " << date << "\n";
    cout << "---------------------------------\n";

    char line[100];
    while(file.getline(line, 100)) {
        cout << line << "\n";
    }

    file.close();
}
```

Figure 8 attendance sheet display code

4.2 Key Features

✓ **Dynamic Student Entry** – Allows adding multiple students in one session.
✓ **Persistent Storage** – All data is saved in files for future access.
✓ **Error Handling** – Ensures valid inputs (e.g., only 'P' or 'A' for attendance).
✓ **Sorting & Searching** – Students can be sorted alphabetically, and attendance can be searched by date.

CHAPTER 5: USER INTERFACE

The system follows a **menu-driven** approach:

CODE:

```
int main() {
    loadCourseInfo();
    loadStudentsFromFile();

    int choice;
    while(1) {
        clearScreen();
        displayHeader();

        cout << "MAIN MENU\n";
        cout << "1. Course Information\n";
        cout << "2. Add Student\n";
        cout << "3. Delete Student\n";
        cout << "4. Mark Attendance\n";
        cout << "5. View Students\n";
        cout << "6. Search/Update Attendance\n";
        cout << "7. View Day's Attendance\n";
        cout << "8. Sort Students\n";
        cout << "9. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        clearScreen();
        displayHeader();

        switch(choice) {
            case 1: setCourseInfo(); break;
            case 2: addStudent(); break;
            case 3: deleteStudent(); break;
            case 4: markAttendance(); break;
            case 5: viewStudents(); break;
            case 6: searchUpdateAttendance(); break;
            case 7: displayDayAttendance(); break;
            case 8: sortStudents(); break;
            case 9:
                cout << "Exiting program...\n";
                saveStudentsToFile();
                return 0;

            default: cout << "Invalid choice!\n";
        }

        cout << "\nPress Enter to continue...";
        cin.ignore();
        cin.get();
    }

}
```

Figure 9 code main menue

CHAPTER 6: TESTING AND VALIDATION

6.1 Test Cases

Function	Test Case	Expected Result
addStudent()	Add a new student	Student appears in students.dat
deleteStudent()	Delete a student by roll no.	Student removed from file
markAttendance()	Mark attendance for a date	File attendance_DD-MM-YYYY.txt created
searchUpdateAttendance()	Update attendance status	Attendance record modified
sortStudents()	Sort students by name	Students displayed in alphabetical order

6.2 Limitations

- **No graphical interface** (Console-based only).
- **No password protection** (Lacks user authentication).
- **Limited error handling** (Basic input validation only).

CHAPTER 7: FUTURE ENHANEMENTS:

- **Graphical User Interface (GUI)** – Using Qt or Windows Forms.
- **Database Integration** – Replace file storage with SQLite/MySQL.
- **Multi-user Login** – Separate access for teachers and admins.
- **Automated Reports** – Generate PDF attendance summaries.
- **Mobile App Integration** – For remote attendance marking.

CHAPTER 8: SOURCE CODE:

The complete C++ code is provided in the project file. Key dependencies:

- <iostream> (Input/output)
- <fstream> (File handling)
- <cstring> (String operations)
- <conio.h> (Console input, e.g., getch())

CHAPTER 9: CONSOLE DISPLAY:

9.1 MAIN MENUE

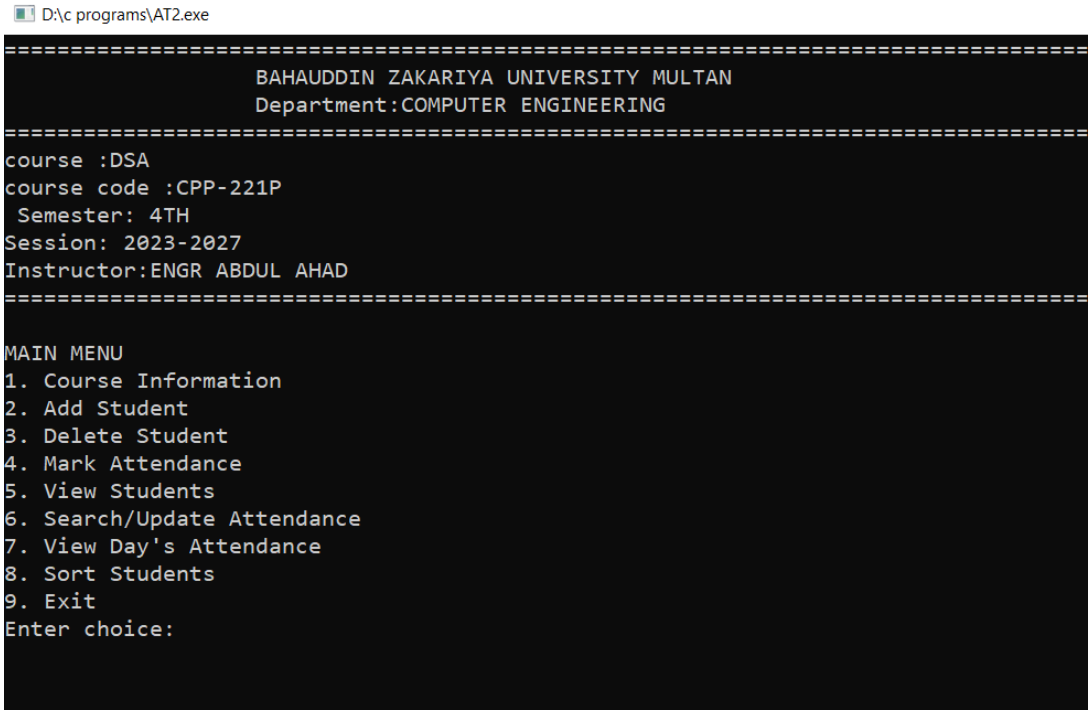

Figure 10 main menue display

9.2 ADD STUDENT

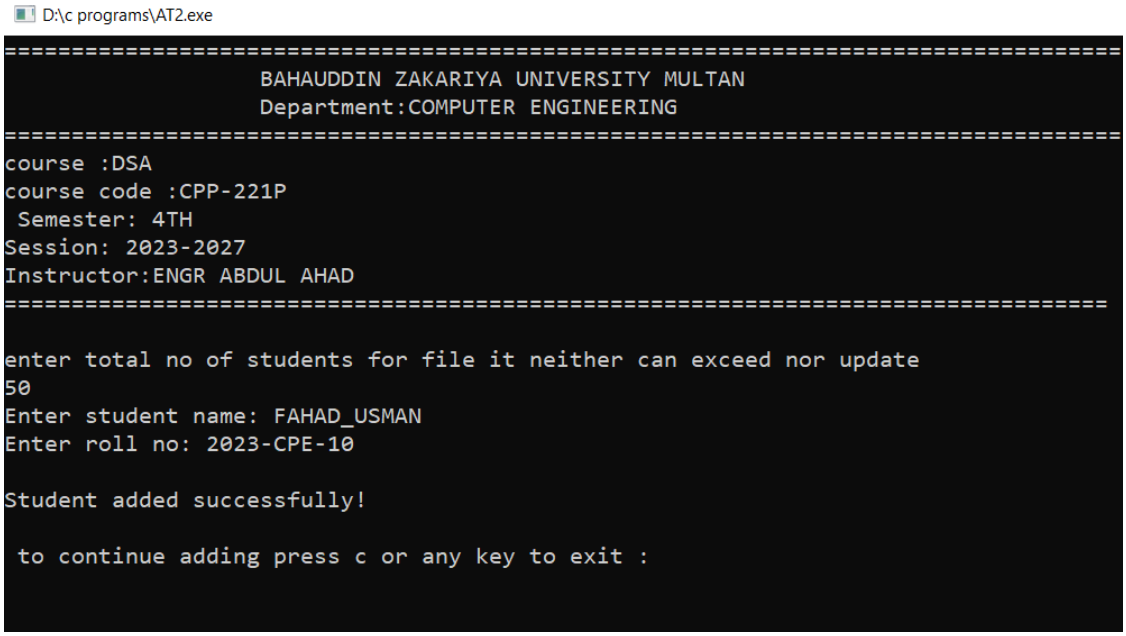

Figure 11add student display

9.3. DELETE STUDENT

```
================================================================

        STUDENT LIST
--------------------------------
Roll No         Name
------------    -------------------
2023-CPE-08     ALEENA_KHAN
2023-CPE-03     ANIQA_IMRAN
2023-CPE-01     AYESHA_NOREEN
2023-CPE-05     BUSHRA_KANOOZ
2023-CPE-09     ESHA_ARAIN
2023-CPE-02     M.RAZA
2023-CPE-04     UQAB_HAIDER
2023-CPE-10     FAHAD_USMAN
Enter Roll Number to delete: 2023-CPE-10
Student deleted successfully!
Remaining students are :
        STUDENT LIST
--------------------------------
Roll No         Name
------------    -------------------
2023-CPE-08     ALEENA_KHAN
2023-CPE-03     ANIQA_IMRAN
2023-CPE-01     AYESHA_NOREEN
2023-CPE-05     BUSHRA_KANOOZ
2023-CPE-09     ESHA_ARAIN
2023-CPE-02     M.RAZA
2023-CPE-04     UQAB_HAIDER
to continue delete press c or any other key to exit:
```

Figure 12 delete student display

Figure 13

9.4 MARK ATTENDANCE

```
================================================================
                  BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN
                  Department:COMPUTER ENGINEERING
================================================================
course :DSA
course code :CPP-221P
 Semester: 4TH
Session: 2023-2027
Instructor:ENGR ABDUL AHAD
================================================================

Enter date (DD-MM-YYYY): 12-05-2025
ALEENA_KHAN (Roll No: 2023-CPE-08) - Present (P/A)? P
ANIQA_IMRAN (Roll No: 2023-CPE-03) - Present (P/A)? P
AYESHA_NOREEN (Roll No: 2023-CPE-01) - Present (P/A)? P
BUSHRA_KANOOZ (Roll No: 2023-CPE-05) - Present (P/A)? P
ESHA_ARAIN (Roll No: 2023-CPE-09) - Present (P/A)? P
M.RAZA (Roll No: 2023-CPE-02) - Present (P/A)? A
UQAB_HAIDER (Roll No: 2023-CPE-04) - Present (P/A)? P

Attendance saved to attendance_12-05-2025.txt

Press Enter to continue...
```

Figure 14 mark attendance display

Figure 15

9.5 VIEW STUDENTS

```
================================================================
                  BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN
                  Department:COMPUTER ENGINEERING
================================================================
course :DSA
course code :CPP-221P
 Semester: 4TH
Session: 2023-2027
Instructor:ENGR ABDUL AHAD
================================================================

         STUDENT LIST
--------------------------------
Roll No         Name
------------    -------------------
2023-CPE-08     ALEENA_KHAN
2023-CPE-03     ANIQA_IMRAN
2023-CPE-01     AYESHA_NOREEN
2023-CPE-05     BUSHRA_KANOOZ
2023-CPE-09     ESHA_ARAIN
2023-CPE-02     M.RAZA
2023-CPE-04     UQAB_HAIDER

Press Enter to continue...
```

Figure 16 vie students display

Figure 17

9.6 SEARCH/UPDATE

```
================================================================
                  BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN
                  Department:COMPUTER ENGINEERING
================================================================
course :DSA
course code :CPP-221P
 Semester: 4TH
Session: 2023-2027
Instructor:ENGR ABDUL AHAD
================================================================

Enter date to search (DD-MM-YYYY): 12-05-2025
Enter roll number to search: 2023-CPE-01

Record found: 2023-CPE-01       AYESHA_NOREEN   Present
Update status (P/A): A

Attendance record updated successfully!

Press Enter to continue...
```

Figure 18 sorting display

Figure 19

9.7 VIEW ATTENDANCE SHEET

```
Enter date to view (DD-MM-YYYY): 12-05-2025

ATTENDANCE RECORD FOR 12-05-2025
---------------------------------
University: BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN
Department: COMPUTER ENGINEERING
Course: DSA (CPP-221P)
Semester: 4TH | Session: 2023-2027
Attendance Date: 12-05-2025
---------------------------------
RollNo          Name                    Status
2023-CPE-08     ALEENA_KHAN     Present
2023-CPE-03     ANIQA_IMRAN     Present
2023-CPE-01     AYESHA_NOREEN   Present
2023-CPE-05     BUSHRA_KANOOZ   Present
2023-CPE-09     ESHA_ARAIN      Present
2023-CPE-02     M.RAZA  Absent
2023-CPE-04     UQAB_HAIDER     Present

Press Enter to continue...
```

Figure 20 attendance sheet

9.8 SORTING STUDENTS

```
============================================================
                  BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN
                  Department:COMPUTER ENGINEERING
============================================================
course :DSA
course code :CPP-221P
 Semester: 4TH
Session: 2023-2027
Instructor:ENGR ABDUL AHAD
============================================================

Students sorted alphabetically!
         STUDENT LIST
--------------------------------
Roll No         Name
-----------     ------------------
2023-CPE-08     ALEENA_KHAN
2023-CPE-03     ANIQA_IMRAN
2023-CPE-01     AYESHA_NOREEN
2023-CPE-05     BUSHRA_KANOOZ
2023-CPE-09     ESHA_ARAIN
2023-CPE-02     M.RAZA
2023-CPE-04     UQAB_HAIDER

Press Enter to continue...
```

Figure 21 sorting of students

CHAPTER 10: CONCLUSION

This enhanced implementation provides robust attendance management with careful attention to console UX and data integrity. The system demonstrates effective use of structured file storage while maintaining simplicity suitable for educational environments.