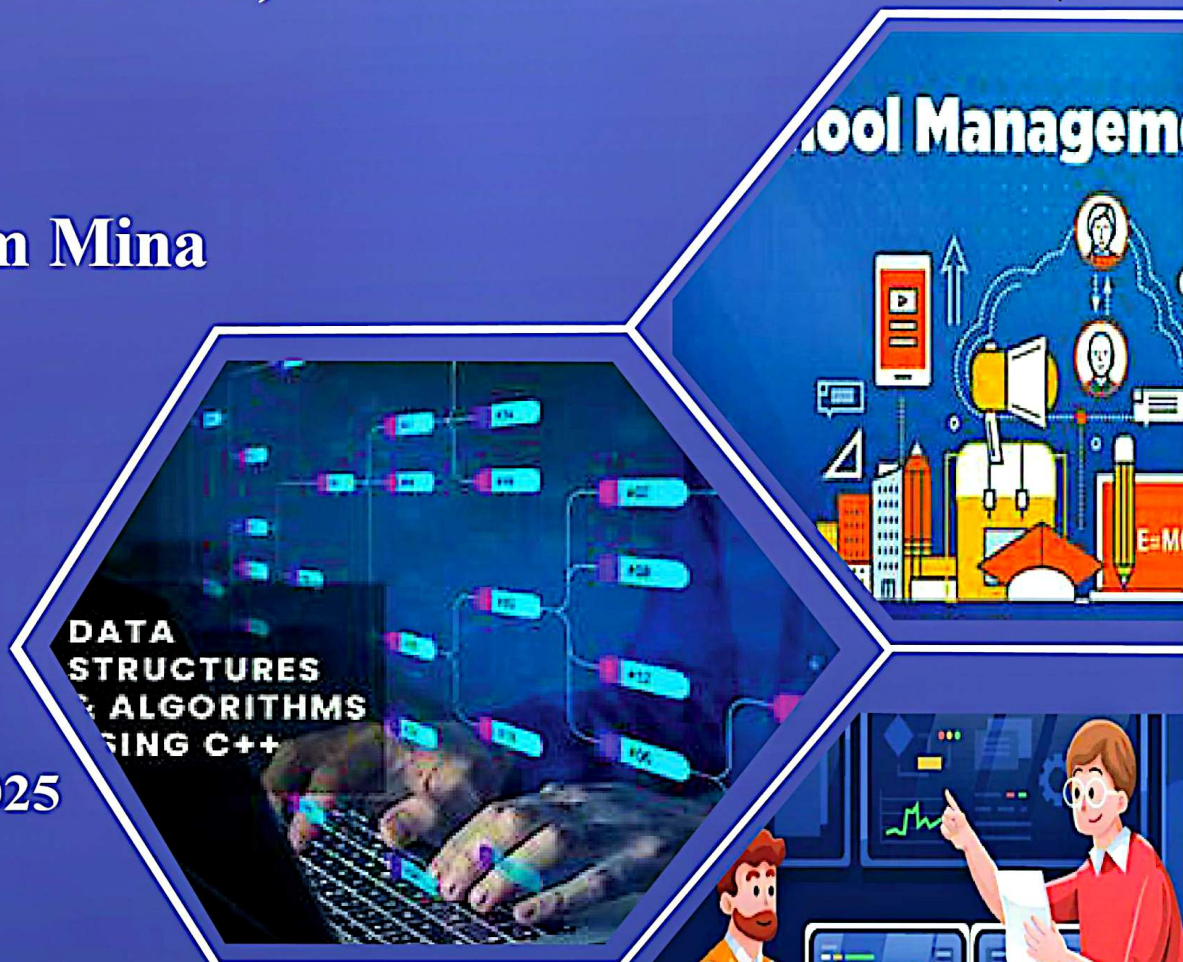
Department of Computer Science

COMSATS UNIVERSITY ISLAMABAD SAHIWAL CAMPUS

2025
SEMESTER PROJECT
ON
SCHOOL MANAGEMENT SYSTEM

Submitted By:
Alihassan (Sp24-Bcs-115)

Submitted To:
MS.CS, Mam Mina



Data Submission:
02-December-2025

SEMESTER PROJECT

SCHOOL MANAGEMENT SYSTEM

Feature:

Login

Logout

Student Manage

Teacher Manage

Attendence Manage

Exam/Grade Manage

Section/Class/Timetable Manage

Fee Manage

Library Manage

Transport Manage

Notification/Event

Online Fees Payment

Student Performance

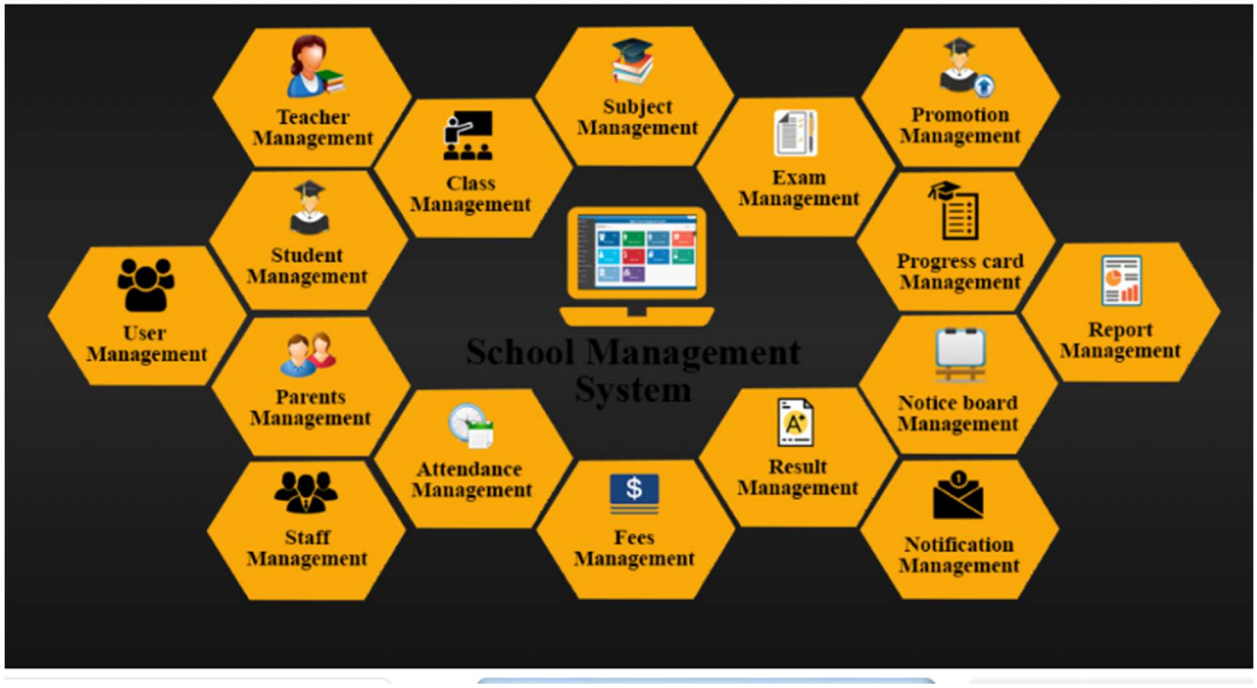
RFID Entry

Discipline Entry

School 3D Maps

Discussion Group

Hostel Management System



(Data Structure Design and Pseudocode Development)

Objective

This task focuses on designing the **internal working of your project** using **appropriate Data Structures (DS)**.

- Each group member contributes by selecting operations/modules.
- This week involves **short live class activity**, but **no formal presentation** is required.

1. Data Structure Planning Table:

Data Structure Five most important Operation about Project

1. Add New Student (StudentGUI)

DS: ArrayList / LinkedList
Reason: Students dynamic hain, add/remove hotay rehte hain.

2. Mark Student Attendance (AttendanceGUI)

DS: HashMap (key = studentID, value = attendance record)
Reason: Attendance search bohat fast honi chahiye → O(1)

3. Issue Book in Library (LibraryGUI / BookGUI)

DS: Queue (for book issue request)
Reason: First Come First Serve issuing system hota hai.

4. Search Teacher by ID (TeacherGUI)

DS: HashMap
Reason: Teacher ko ID se fast search karna hota hai.

5. Generate Student Performance Report (StudentPerformanceGUI)

DS: Tree (Binary Search Tree)
Reason: Sorted performance, fast insertion, fast searching.

Planning Table:

Operation	Data Structure Used	Reason for Choosing the DS (Friendly Words)
1. Add Student Record	**ArrayList**	The student list keeps growing, and ArrayList automatically increases size. Adding or removing students is easy.
2. Mark Attendance	**HashMap (key = StudentID)**	Attendance can be quickly accessed using Student ID. HashMap is very fast (O(1)).
3. Manage Library Books	**LinkedList**	Books are added/removed like a queue. LinkedList doesn't need shifting, so it's fast.
4. Online Fee Payment Record	**Queue**	Fee requests are handled in order (first come, first serve). Queue is perfect for this.
5. Student Performance Analysis	**Tree (BST)**	Marks need to be kept sorted (topper, lowest, average). Tree makes searching and sorting easy.

2. Operation Flow Explanation:

Choose 3 operation about project

1. Add Student Performance
2. Search Teacher by ID
3. Generate Student Attendance Report

1. Operation: Add Student Performance

✓ What this operation does:
The teacher enters a student's performance score (marks, grade, comments).
The system stores this data so that the student's report and progress can be viewed later.

✓ Which Data Structure supports it:
`HashMap<String, Performance>`
(where `String = studentID`)

✓ Why this DS is suitable:

- Direct access through studentID key
- Fast insert & update — average O(1) time
- Each student's performance can be easily updated
- Duplicate studentID automatically avoided

✓ How the user interacts with it:

1. Open the GUI form
2. Teacher selects the studentID
3. Enter marks, grade, and comments
4. Press "Save Performance"
5. Data is stored in the HashMap

2. Operation: Search Teacher by ID

✓ What this operation does:
Admin enters a teacher's ID, and the system returns the teacher's details (name, subject).

✓ Which Data Structure supports it:
`HashMap<String, Teacher>`

✓ Why this DS is suitable:

- Teacher ID is a unique key
- Direct, fast search — O(1) lookup
- No need to scan the full list
- Provides fast performance for real-time GUI

✓ How the user interacts with it:

1. Admin enters teacherID in the search bar
2. Clicks the search button

3. System matches the ID in the HashMap
4. Teacher's details are shown on the GUI

3. Operation: Generate Student Attendance Report

✓ What this operation does:
Generates a summary report for a student using attendance records (present days, absent days).

✓ Which Data Structure supports it:
`ArrayList<AttendanceRecord>`

✓ Why this DS is suitable:

- Attendance is recorded daily → sequential storage is best
- Easy to loop through and calculate totals
- Adding new attendance is simple — O(1) amortized
- Flexible and dynamic size

✓ How the user interacts with it:

1. Teacher/administrator selects a student in the GUI
2. Clicks "Generate Attendance Report"
3. Program loops through the ArrayList to calculate totals
4. Displays the result in a graph or table

4. Pseudocode Writing:

Write pseudocode for at least five functions from your system

1. Add Student (StudentGUI)

Data Structure: LinkedList

```
Operation: Add Student
Input: studentID, name, class
Process:
  1. Create new Student node with studentID, name, class
  2. If head == null
       head = new Student node
     Else
       Traverse LinkedList to end
       Add new node at end
Output: "Student Added Successfully"
```

2. Add Teacher (TeacherGUI)

Data Structure: HashMap (Key: teacherID, Value: Teacher object)

```
Operation: Add Teacher
Input: teacherID, name, subject
Process:
  1. Create Teacher object
  2. Add object to HashMap with teacherID as key
Output: "Teacher Added Successfully"
```

3. Record Attendance (AttendanceGUI)

Data Structure: Queue (FIFO)

```
Operation: Record Attendance
Input: studentID, status (Present/Absent)
Process:
  1. Create Attendance record
  2. Enqueue record into Attendance Queue
Output: "Attendance Recorded Successfully"
```

4. Student Performance Report (StudentPerformanceGUI)

Data Structure: Binary Search Tree (BST) on studentID

```
Operation: Generate Student Performance
Input: studentID
Process:
  1. Search studentID in BST
  2. If found
       Retrieve marks and performance
     Else
       Output "Student Not Found"
Output: Display student performance report
```

5. Online Fee Payment (OnlineFeesPaymentGUI)

Data Structure: Stack (for transaction history)

```
Operation: Pay Fees
Input: studentID, amount
Process:
  1. Create FeeTransaction record
  2. Push record onto Transaction Stack
  3. Update student balance
Output: "Payment Successful" + Updated Balance
```

Visual Diagram

Project School Management System flowchart

School Online Fees Payment

Flow Chart
(School Online fee Payment)