

DSA (LAB)

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Course Code:\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Prof:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



[Date]

University of management and technology

**Attendance Management System for Teachers**  
**Final Project Report**

**1. Introduction**

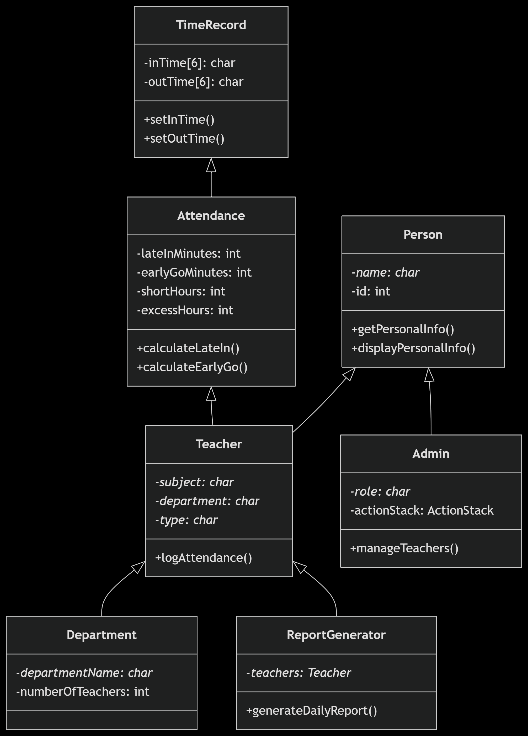
**Purpose:**  
The Attendance Management System automates faculty attendance tracking in universities, monitoring in/out times, late arrivals, early departures, and work hour compliance. It enforces policy-based rules for permanent, visiting, and administrative staff while demonstrating OOP principles and DSA applications.

**Key Objectives:**

* Implement inheritance hierarchies (single, multiple, multilevel, hierarchical, hybrid)
* Utilize data structures (BST, Stack, Queue, Graph) for efficient operations
* Enable file-based data persistence
* Provide role-based access (Teachers, Admins)

**2. System Architecture**

**Class Hierarchy**



**Inheritance Models**

1. **Single Inheritance:** Admin → Person
2. **Multiple Inheritance:** Teacher → Person + Attendance
3. **Multilevel:** Attendance → TimeRecord
4. **Hierarchical:** Department → Teacher
5. **Hybrid:** ReportGenerator → Teacher

**3. Core Components**

**A. Key Classes**

1. **Person (Base Class)**
   * **Purpose:** Base for all human entities
   * **Data:** name, id
   * **Methods:** Input/output personal info
2. **TimeRecord**
   * **Purpose:** Track check-in/out times
   * **Data:** inTime, outTime (HH:MM format)
3. **Attendance**
   * **Purpose:** Calculate attendance metrics
   * **Algorithms:**

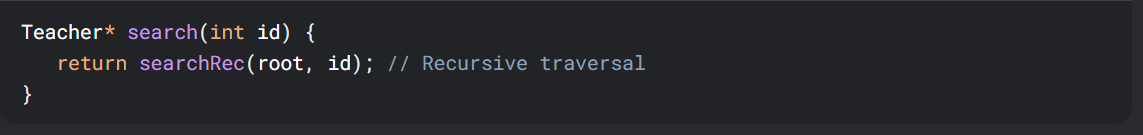
A screen shot of a computer code

AI-generated content may be incorrect.

1. **Teacher**
   * **Purpose:** Handle faculty-specific data and attendance logging
   * **Workflow:**
     1. Input in/out times
     2. Automatically calculate metrics
     3. Append to attendance\_log.txt
2. **Admin**
   * **Purpose:** Manage teacher records
   * **Features:**
     1. Undo actions using ActionStack (Stack LIFO)
     2. CRUD operations for teachers
3. **ReportGenerator**
   * **Purpose:** Generate daily reports
   * **Output:** summary\_report.txt with teacher IDs, names, and times

**Data Structures**

1. **Binary Search Tree (BST)**
   * **Purpose:** Efficient teacher search by ID
   * **Complexity:** O(log n) search time



1. **ActionStack (Stack)**
   * **Purpose:** Enable undo for admin actions
   * **Implementation:** Linked-list based stack
2. **LoginQueue (Queue)**
   * **Purpose:** Process teacher login requests
   * **Implementation:** FIFO linked-list
3. **DepartmentGraph (Graph)**
   * **Purpose:** Model department relationships
   * **Storage:** Adjacency list of connected departments

**C. File Handling**

1. **teachers.txt**
   * **Format:** ID,Name,Subject,Department,Type
   * **Purpose:** Persistent teacher data storage
2. **attendance\_log.txt**
   * **Format:** ID,In-Time,Out-Time
   * **Purpose:** Daily attendance records
3. **summary\_report.txt**
   * **Sample:**

A screen shot of a computer

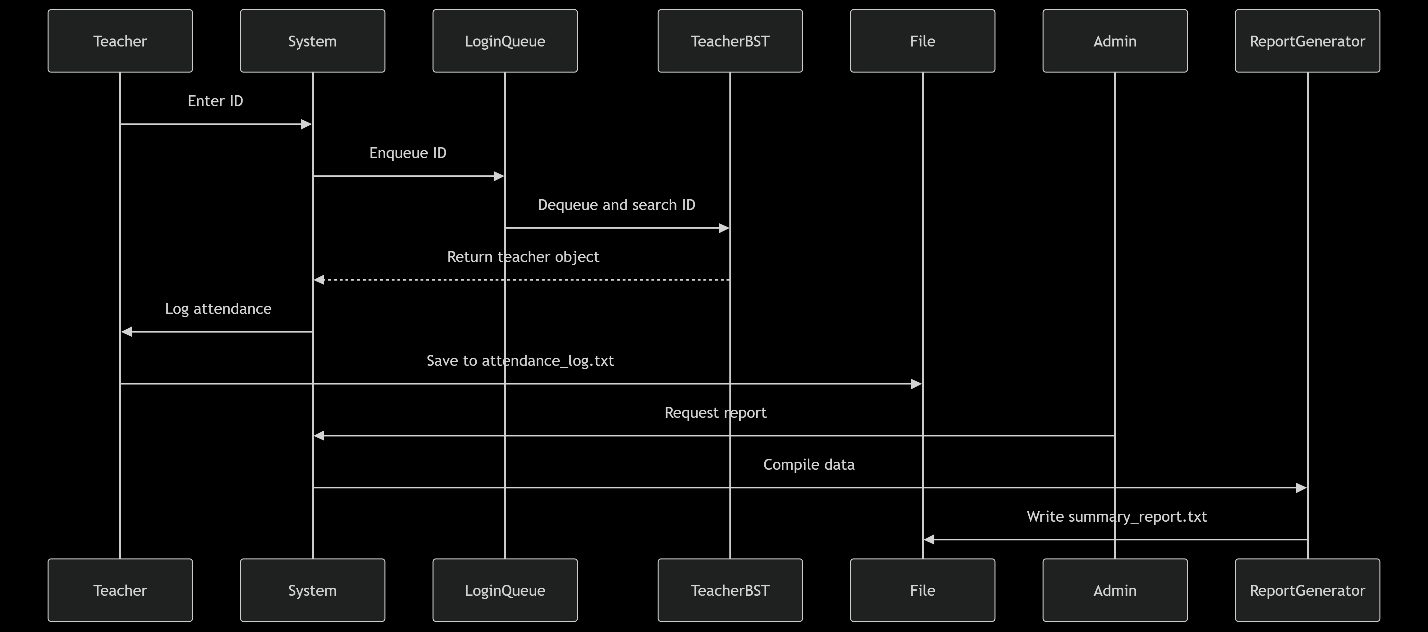
AI-generated content may be incorrect.

**4. Functionality & Usage**

**User Roles**

1. **Teacher**
   * Log in/out times
   * View personal attendance reports
2. **Admin**
   * Add/remove teachers
   * View department summaries
   * Generate reports
   * Undo accidental deletions

**Workflow**



**Algorithms**

1. **Time Conversion**
   * **Input:** "HH:MM" → **Output:** Total minutes

A computer screen shot of a computer code

AI-generated content may be incorrect.

1. **Attendance Calculations**
   * Late minutes = max(0, actual\_in - standard\_in)
   * Short hours = max(0, 8\*60 - (outTime - inTime))
2. **BST Search**
   * Recursive tree traversal for O(log n) lookups

**6. Error Handling**

* **Memory Management:** All dynamic memory deallocated in destructors
* **Input Validation:** Checks for valid time formats and IDs
* **File Operations:** Verify file open success before I/O
* **Edge Cases:**
  + Empty stacks during undo
  + Full teacher capacity
  + Invalid department lookups

**7. Conclusion**

**Achievements:**  
✅ Implemented all 7 required classes with proper inheritance  
✅ Integrated 6 data structures with custom implementations  
✅ Achieved persistent storage via file handling  
✅ Delivered policy-based attendance tracking

**Future Enhancements:**

* Monthly analytics with visualization
* Mobile app integration
* Biometric authentication

**Final Code Stats:**

* **Lines:** 850+
* **Files:** 1 main.cpp + 3 output .txt files
* **Dependencies:** Pure C++ (No external libraries)