PROGRAMMING FUNDAMENTALS

FINAL PROJECT

SCHOOL MANAGEMENT SYSTEM

SUBMITTED BY:

M. AHSAN SADIQ	M.KAMRAN
2024-SE-40	2024-SE-24

SUBMITTED TO:

MISS. ZOHA SOHAIL

Project Overview

The School Management System is a C++ application designed to automate and streamline key school operations, including managing student and teacher records, scheduling classes, tracking attendance, and handling examinations. It uses file handling for data persistence, allowing long-term storage and retrieval of records. The system facilitates tasks like assigning students to classes, scheduling timetables, marking attendance, and generating performance reports, all through a modular structure that supports scalability. By centralizing these operations, the application aims to reduce manual work, improve accuracy, and provide a reliable and accessible platform for school administrators and teachers.

Key Features:

1. Student Management
2. Teacher Management
3. Class Management
4. Attendance Tracking
5. Examination Module
6. Administrative Tools

Phase 1:

The first phase in developing a School Management System involves planning and designing the system's foundation. This includes gathering requirements, defining the scope, and dividing the system into key modules like Student Management, Teacher Management, and Attendance Tracking. Basic class structures and relationships are designed using UML or ER diagrams, focusing on entities like `Student`, `Teacher`, and `Classroom`. The development begins with setting up the environment, implementing core classes, and building basic functionalities such as CRUD operations for one module (e.g., Student Management) with file handling for data persistence. A simple console-based interface is created to test functionality, forming a prototype that serves as the base for future expansion.

Data Validations:

- **Student/Teacher ID:** Check for correct format (e.g., "STU-001" for students).
- **Name:** Ensure only alphabets and spaces are used.
- **Age:** Validate age within a specific range (e.g., 5-18 for students).

- **Grade:** Ensure grades are between 0 and 100.
- **Attendance:** Validate date format and status ("Present" or "Absent").
- **Class ID and Section:** Ensure proper format (e.g., "CLS-001" and valid sections like 'A').
- **Timetable:** Validate class times are within school hours and do not overlap.
- **Marks:** Ensure marks are numeric and within the 0-100 range.
- **Exam Dates:** Confirm exam dates do not conflict with weekends or holidays.
- **File Formats:** Validate correct file structure and data integrity.
- **Duplicate Entries:** Prevent duplicate student or teacher IDs.
- **Empty Fields:** Ensure mandatory fields (e.g., name, ID) are not left blank.
- **Input Types:** Validate data types for fields (numeric for age, marks; alphabetic for names).

Functions:

Following are some functions (Predefined and User-defined) that can be used while developing this Management System. Names of the functions cannot be changed and the case sensitivity of the functions must be kept in mind otherwise the program will NOT RUN.

1. Menu Functions

These functions handle navigation within the application:

- ***void displayMainMenu();*** Displays the main menu with options like Student Management, Teacher Management, etc.
- ***void handleUserSelection(int choice);*** Handles user input to navigate the appropriate module.

---

2. Student Management Functions

- ***void addStudent();*** Allows the user to add a new student to the system.
- ***void viewAllStudents();*** Displays a list of all students.
- ***void searchStudentByID(int studentID);*** Searches for a student using their unique ID.
- ***void updateStudent(int studentID);*** Updates student information, such as grades, attendance, or personal details.
- ***void deleteStudent(int studentID);*** Removes a student from the system.
- ***void recordAttendance(int studentID);*** Marks attendance for a specific student.
- ***void generateStudentReport(int studentID);*** Generates and displays a performance report for a student.

---

3. Teacher Management Functions

- *void addTeacher();* Adds a new teacher to the system.
- *void viewAllTeachers();* Displays a list of all teachers.
- *void searchTeacherByID(int teacherID);* Searches for a teacher using their unique ID.
- *void updateTeacher(int teacherID);* Updates a teacher's details, such as assigned subjects or schedules.
- *void deleteTeacher(int teacherID);* Removes a teacher from the system.

---

4. Class and Timetable Management

- *void addClass();* Creates a new class or section.
- *void viewAllClasses();* Displays details of all classes and their respective students and teachers.
- *void assignTeacherToClass(int classID, int teacherID);* Assigns a specific teacher to a class.
- *void assignStudentToClass(int classID, int studentID);* Assigns a student to a class.
- *void createTimetable (int classID);* Allows the user to define the timetable for a specific class.
- *void viewTimetable (int classID);* Displays the timetable of a specific class.

---

5. Examination and Grade Management

- *void scheduleExam(int classID, string subject);* Schedules an exam for a specific class and subject.
- *void recordExamResult(int studentID, string subject, int marks);* Inputs the exam result for a student.
- *void viewStudentGrades(int studentID);* Displays all grades of a specific student.
- *void generateClassPerformanceReport(int classID);* Generates a performance report for the entire class.

---

6. Attendance Management

- *void markStudentAttendance(int classID, int studentID, string date, bool present);* Marks attendance for a specific student on a specific date.
- *void viewAttendance(int studentID);* Displays the attendance record of a student.
- *void viewClassAttendance(int classID);* Shows attendance for all students in a class.

7. File Management Functions

- *void saveDataToFile();*Saves all records (students, teachers, classes, etc.) to a file.
- *void loadDataFromFile();*
  Loads data from files when the system starts.
- *void backupData();*
  Creates a backup of all data.

8. Utility Functions

- *int generateUniqueID();*
  Generates unique IDs for students, teachers, or classes.
- *void validateInput();* Ensures user inputs are valid (e.g., numeric values for IDs, string for names).
- *void pauseAndClear();* Pauses the system and clears the screen for better user experience.

Global constants:

All these constants are of integer data and are declared as integers.

- **MAX_STUDENTS:** Sets the maximum number of students the system can handle (e.g., 1000).
- **MAX_TEACHERS:** Limits the number of teachers that can be added (e.g., 100).
- **DEFAULT_GRADE:** Provides an initial grade value (e.g., 0) for new students.
- **DEFAULT_ATTENDANCE:** Sets the starting attendance percentage for students (e.g., 0%).
- **PASSING_MARKS:** Defines the minimum marks required to pass an exam (e.g., 40).
- **GRADE_A:** Represents the cutoff for Grade A in performance metrics (e.g., 90).
- **STUDENT_FILE:** Specifies the file path where student records are saved (e.g., `students.dat`).
- **ID_SEPARATOR:** A character (e.g., '-') used to format IDs like "STU-001".
- **CLASS_START_HOUR:** Determines the start time of classes (e.g., 8 AM).
- **SCHOOL_NAME:** Stores the name of the school (e.g., "ABC International School").