Documentation for School Management System

1 Introduction to the Project

The School Management System is an Object-Oriented Programming (OOP) project designed to streamline the management of various educational administrative tasks. It provides a structured approach to manage students, teachers, courses, and other essential components of a school environment. This project encapsulates the principles of OOP, enabling efficient data management and user interaction.

2 What the Project Does

The School Management System allows users to perform a variety of functions including:

- Managing student profiles (creation, retrieval, updating, and deletion).
- Managing teacher profiles.
- Managing courses offered by the school.
- Viewing attendance and grades.
- Facilitating communication between students, teachers, and principals.
- Generating reports for administrative purposes.

3 Who the Project is Aimed At

The project is aimed at educational institutions, including:

- School administrators (Principals) who need to oversee the management of students and teachers.
- Teachers who require tools to manage their courses and student performance.
- Students who need to access their profiles, view attendance, and grades.

4 What the Project Uses to Store the Data

The project uses data structures defined in the `users.h` header file to manage user profiles and course information. The data can be stored in various formats such as JSON, CSV, or in a database, depending on the implementation. The use of the `nlohmann::json` library suggests a potential for JSON file storage.

5 Classes/Structures Used

The following classes and structures are defined in the `users.h` file:

- **Structs:**
  - `Course`: Represents a course with a code and name.
  - `StudentStats`: Holds statistics related to a student's performance, including course code, grade, and attendance.
- **Classes:**
  - `User` : An abstract base class for common user attributes and methods.
  - `Student`: Inherits from `User` and includes methods for managing student-specific data.
  - `Teacher`: Inherits from `User` and includes methods for managing teacher-specific data.
  - `Principal`: Inherits from `User` and includes methods for managing both student and teacher data, as well as course management.

6 Attributes and Behaviours in Those Data Structures

- **User Class:**
  - Attributes: `id`, `name`
  - Behaviours: `viewProfile()`, `getId()`, `getName()`
- **Student Class:**
  - Attributes: `stats` (pointer to `StudentStats`)
  - Behaviours: `viewProfile()`, `viewAttendances()`, `viewGrades()`
- **Teacher Class:**
  - Attributes: `courses` (pointer to `Course`)
  - Behaviours: `viewProfile()`, `updateAttendance()`, `updateGrade()`

- **Principal Class:**
  - Behaviours: Methods for creating, retrieving, updating, and deleting students and teachers, managing courses, and viewing all students and teachers.

7 How the Menu is Managed for Different Users

The menu for different users (students, teachers, and principals) is managed through polymorphism. Each user type has specific functionalities that are invoked based on their class type. The principal class provides a comprehensive set of methods for managing both students and teachers, while the student and teacher classes have specialized methods for their respective roles.

8 Concepts of OOPs Used

The project utilizes several OOP concepts:

- **Encapsulation:** Data and methods are encapsulated within classes, protecting the internal state.
- **Inheritance:** The `Student`, `Teacher`, and `Principal` classes inherit from the `User` base class, promoting code reuse.
- **Polymorphism:** The use of virtual functions allows for dynamic method resolution, enabling different behaviors for `viewProfile()` based on the user type.

9 Further Improvements Planned

Future enhancements may include:

- Implementing a database for persistent data storage.
- Adding a user authentication system for secure access.
- Implementing a user-friendly graphical interface (GUI).
- Integrating additional features like notifications, messaging, and scheduling.

10 Conclusion

The School Management System is a robust application designed to facilitate the management of educational institutions. By leveraging OOP principles, it provides a scalable and maintainable solution for managing users and their interactions within a school environment.

11 Outputs

The outputs of the application include:

- Student and teacher profiles displayed in a structured format.
- Attendance and grades reports.
- Course management outputs reflecting current course offerings.
- Logs of administrative actions taken by the principal.

12 How to Use the App

To use the School Management System:

1. Compile the project using a C++ compiler that supports the C++11 standard or later.
2. Run the compiled executable.
3. Log in using the appropriate credentials (if implemented).
4. Navigate through the menu options based on your user role.