**Project Title: Student Management System**

**Project Description:**

The Student Management System is a Java-based project that utilizes data structures and algorithms to manage student information efficiently. This system allows users to perform various operations such as adding new students, deleting existing ones, searching for students by ID or name, and displaying all student records. It also includes functionalities for sorting students based on different criteria and implementing basic algorithms for data manipulation.

**Key Features:**

Add a new student: Users can input details such as student ID, name, age, and grade to add a new student to the system.

Delete a student: Users can remove a student from the system by providing their ID.

Search for a student: Users can search for a student by their ID or name and view their details.

Display all students: Users can view a list of all students currently stored in the system.

Sort students: Implement various sorting algorithms (e.g., bubble sort, insertion sort, quicksort) to sort students based on their ID, name, age, or grade.

**Technologies and Concepts:**

Java programming language for the backend implementation.

Object-oriented programming (OOP) principles for designing classes and objects.

Data structures such as arrays, linked lists, and hash tables to store and organize student data.

Algorithms for operations like sorting (e.g., bubble sort, insertion sort, quicksort) and searching (e.g., linear search, binary search).

**Project Structure:**

Main.java: Entry point of the application containing the main method.

Student.java: Class representing the structure of a student object with attributes like ID, name, age, and grade.

StudentManagementSystem.java: Class containing methods for adding, deleting, searching, and displaying student records.

SortAlgorithms.java: Class implementing various sorting algorithms.

**Additional Considerations:**

Error handling: Implement error handling mechanisms to handle exceptions gracefully.

Efficiency: Optimize algorithms and data structures for efficient performance, especially for large datasets.

Documentation: Provide clear documentation for the codebase, including comments within the code and a README file explaining how to run the application and use its features.