**Project: Student Management System**

Project Requirements:

**Student Data:**

Each student will have the following attributes:

* Student ID (integer)
* Name (string)
* Age (integer)
* GPA (float)

**Functionality:**

* Add a New Student:
  + Prompt the user to enter student details (ID, name, age, GPA) and add the student to the system.
* Delete a Student by ID:
  + The user can remove a student from the system by entering the student ID.
* Search for a Student by ID:
  + Enable the user to search for a student by entering the student ID and display the student's details if found.
* Display All Students:
  + Show a list of all students currently in the system with their details.
* Save All Student Data to a File:
  + Save the current list of students to a text file. Each student's details should be on a separate line, with attributes separated by commas.
* Load Student Data from a File:
  + Load student data from a text file into the system, replacing any existing data in memory.

**File Handling:**

Use a text file named students.txt to save and load student data.

The format for the text file should be one student per line, with attributes separated by commas, e.g., ID, Name, Age, GPA.

**User Interface:**

Create a console-based menu to interact with the system. The menu should include options for each functionality:

\* Add Student

\* Delete Student

\* Search Student

\* Display All Students

\* Save to File

\* Load from File

\* Exit

**Data Structures:**

Use a linked list to store the student data in memory.

**Error Handling:**

Handle errors such as invalid inputs, attempting to delete or search for a non-existent student, and file read/write errors gracefully.

**Additional Requirements:**

Ensure the program can handle invalid user inputs without crashing.

Remember that memory is managed correctly, especially when adding or deleting students.

Provide clear prompts and messages to the user for each action.

**Summary:**

The Student Management System should allow users to manage student records by adding, deleting, searching, displaying, saving, and loading student data. The system should be robust, handle errors gracefully, and use a linked list for data storage. Data should be persisted in a text file named students.txt. The user interface should be console-based, providing a menu for interaction.