Student Management System - Part 3 & Part 5 Implementation

- Due 8 Mar by 23:59
- Points 80
- Submitting a file upload
- Available 18 Feb at 0:00 - 18 Apr at 23:59

This assignment was locked 18 Apr at 23:59.

Objective:
In this assignment, you will implement two key features of the Student Management System:

- **Part 3: Add Student Screen** – A form that allows users to input new student details.
- **Part 5: View and Search Students Screen** – A screen that displays a list of students and provides search functionality.

Since you haven't yet learned how to connect to a database, you may use synthetic data (hardcoded or in-memory collections) for this assignment.

Part 3: Add Student Screen (40 points)

Feature Details:

- **Purpose:** Create a form where users (e.g., teachers/admins) can enter new student details.
- **Controls to Use:**
  - **Labels:** For field descriptions.
  - **TextBoxes:** For student details like **Name** and **Age**.
  - **ComboBox:** For selecting the **Class** (or section).
  - **DateTimePicker:** For entering the student's enrollment or birth date.
  - **Button:** A **Save** button to submit the data.
- **Concepts to Implement:**
  - **Input Validation:** Ensure that the fields are correctly filled out. For example, the Name field should not be empty, Age should be numeric and within a reasonable range, and a Class should be selected.
  - **Form Handling:** Collect data from the controls and process it.
  - **Feedback:** Use a `MessageBox` to confirm that the data has been saved or to display error messages if validation fails.

Requirements:

- Design a clear and user-friendly Add Student form.

- Validate user inputs before “saving” the student data.
- On clicking the **Save** button:
  - If the input is valid, display a confirmation message (e.g., “Student record saved successfully!”).
  - If the input is invalid, display an appropriate error message.
- **Optional:** You can simulate saving by storing the data in a simple in-memory list (e.g., a `List<Student>`), though this is not mandatory if you wish to focus solely on the form and validation.

Part 5: View and Search Students Screen (40 points)

Feature Details:

- **Purpose:** Create a screen to display a list of student records and allow users to search/filter these records.
- **Controls to Use:**
  - **DataGridView:** To display all student records.
  - **TextBoxes/ComboBoxes:** For entering search criteria (e.g., Name, Age, or Class).
  - **Buttons:** A **Search** button to trigger the filtering action.
- **Concepts to Implement:**
  - **Data Binding:** Bind synthetic student data to the DataGridView. You may simulate student records using a collection such as `List<Student>`.
  - **Filtering Data:** Implement search functionality that filters the DataGridView based on user input.
  - **Search Functionality:** When the user clicks the **Search** button, the DataGridView should refresh to display only those records that match the search criteria.

Requirements:

- Populate the DataGridView with synthetic student data (e.g., a few pre-defined student records).
- Provide search controls to filter the data:
  - A **TextBox** for searching by **Name**.
  - A **ComboBox** for filtering by **Class**.
  - (Optional) Additional controls if you wish to filter by Age.
- On clicking the **Search** button:
  - Filter the displayed student records according to the entered search criteria.
  - If no records match, display a message indicating that no results were found.
- **Optional:** Include a **Reset** button to clear search filters and display all records again.

Student Management System - Part 3 & Part 5 Implementation (1)

Criteria	Ratings				Pts
Add Student Form Design Clear, user-friendly form layout with labels, textboxes, combo box, DateTimePicker, and a Save button.	**10 to >8.0 Pts** **Excellent**	**8 to >5.33 Pts** **Good**	**5.33 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
Input Validation (Add Student) Proper validation for each field (non-empty Name, numeric Age within range, selection in ComboBox). Displays appropriate error messages if validation fails.	**10 to >8.0 Pts** **Excellent**	**8 to >5.0 Pts** **Good**	**5 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
Save Functionality & Feedback The Save button triggers correct handling: shows a confirmation MessageBox on valid input and an error MessageBox on invalid input.	**10 to >8.0 Pts** **Excellent**	**8 to >5.33 Pts** **Good**	**5.33 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
DataGridView Population (View/Search) DataGridView is populated with synthetic student data (can be a hardcoded list).	**10 to >7.5 Pts** **Excellent**	**7.5 to >5.0 Pts** **Good**	**5 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
Search Controls & Functionality	**10 to >8.0 Pts** **Excellent**	**8 to >5.0 Pts** **Good**	**5 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts

Criteria	Ratings				Pts
Search controls (TextBox/ComboBox) are provided. The Search button filters the student list in the DataGridView based on input criteria.					
Data Binding & Filtering Logic Properly bind the synthetic data to the DataGridView and implement correct filtering logic to display matching records.	**10 to >8.0 Pts** **Excellent**	**8 to >5.0 Pts** **Good**	**5 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
Code Organization and Readability Code is well-organized, commented, and uses meaningful names for controls and variables.	**10 to >8.0 Pts** **Excellent**	**8 to >5.0 Pts** **Good**	**5 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
User Interface and Experience The overall UI is intuitive and visually appealing; forms are arranged logically, making it easy for the user to navigate and use the application.	**10 to >8.0 Pts** **Excellent**	**8 to >5.0 Pts** **Good**	**5 to >0.0 Pts** **Satisfactory**	**0 Pts** **Needs Improvement**	10 pts
					Total points: 80