1. You are building a **Student Management System** in Ruby on Rails where users can add, view, update, and delete student records. Each student has the following attributes:

- name (string) – The full name of the student
- email (string) – The student's email address (must be unique)
- age (integer) – The student's age

Question:

1. **Form Implementation:**
    - Create a Rails **form** using form_with in the **view** to allow users to **create or update** a student.
    - Ensure that the form includes fields for name, email, and age.
    - The form should dynamically work for both creating a **new student** and **editing an existing student**.
2. **CRUD Operations:**
    - Implement the **create** action in StudentsController to save a new student record, handling any validation errors properly.
    - Implement the **update** action to modify an existing student record.
    - Implement the **destroy** action to delete a student from the database.
3. **Validation and Error Handling:**
    - Ensure that the name and email fields are **required**.
    - The email must be **unique** across students.
    - Handle validation errors and display appropriate **error messages** in the form.

Expected Output:

- A functional **form** that can be used to **add or edit** a student.
- Full **CRUD operations** implemented in the controller (create, update, destroy).
- Proper **error handling** and **validation messages** displayed to the user.

2. You are developing a **Library Management System** where users can manage books. Each book has the following attributes:

- title (string) – The name of the book
- author (string) – The author of the book
- published_year (integer) – The year the book was published

Question:

1. **Form Implementation:**
    - Create a **form** using form_with in the **view** to allow users to **create or update** a book.
    - Ensure that the form includes fields for title, author, and published_year.
    - The form should work for both **adding a new book** and **editing an existing book** dynamically.

2. **CRUD Operations:**
    - Implement the **create** action in BooksController to save a new book, handling validation errors.
    - Implement the **update** action to modify an existing book record.
    - Implement the **destroy** action to delete a book from the database.
3. **Validation and Error Handling:**
    - Ensure that title and author are **required fields**.
    - Ensure published_year is a **valid integer** and not in the future.
    - Handle validation errors and display appropriate **error messages** in the form.
4. **Index and Show Pages:**
    - Implement an **index page** that lists all books with options to edit or delete them.
    - Implement a **show page** to display details of a single book.

Expected Output:

- A **form** that can be used to **add or edit** a book.
- Complete **CRUD functionality** implemented in the controller (create, update, destroy).
- Proper **error messages** displayed in case of validation failures.
- An **index page** that lists all books and a **show page** to view book details.