Problem Statement:

Rainbow School is in the process of developing a comprehensive software solution for school management. After successfully creating the database structure, the next critical step is to prototype the user interface (UI) and navigation flow. The goal is to provide stakeholders with a tangible representation of what the final application will look like and how it will function. This prototype will be focused on the UI and navigation flow only, without any backend code.

Solution:

Step 1: Project Setup

- Create a New ASP.NET WebForms Project:
- Open Visual Studio and create a new ASP.NET WebForms project.
- Project Structure:
- Organize your project with folders for different sections of the prototype. Create folders such as "Views," "Students," "Teachers," "Courses," and a "Styles" folder for CSS.

Step 2: Home Page

- Create Home Page (Home.aspx):
- Inside the Views folder, create a file named Home.aspx.
- Design the Home Page
- Provide an overview of the prototype's features and navigation options.
- Include links/buttons for different sections like Students, Teachers, Courses, etc.
- Sample Data on Home Page
- Add some sample data or placeholders to demonstrate information related to Rainbow School, like the number of students, teachers, and courses.

Step 3: Student Management

- Student List Page (StudentList.aspx):
- In the Students folder, create a file named StudentList.aspx.
- Design this page to simulate displaying a list of students.
- Manually create placeholders for student names, IDs, etc. to represent sample data.
- Student Details Page (StudentDetails.aspx):
- In the Students folder, create a file named StudentDetails.aspx.
- Design this page to display detailed information about a selected student.
- Manually create placeholders for student details such as name, age, class, etc.

- Add/Edit Student Page (AddEditStudent.aspx):
- In the Students folder, create a file named AddEditStudent.aspx.
- Design this page to simulate adding or editing student records.
- Include form elements for name, age, class, etc.

Step 4: Teacher Management

- Teacher List Page (TeacherList.aspx):
- In the Teachers folder, create a file named TeacherList.aspx.
- Design this page to simulate displaying a list of teachers.
- Manually create placeholders for teacher names, IDs, subjects, etc.
- Teacher Details Page (TeacherDetails.aspx):
- In the Teachers folder, create a file named TeacherDetails.aspx.
- Design this page to display detailed information about a selected teacher.
- Manually create placeholders for teacher details such as name, subject, etc.
- Add/Edit Teacher Page (AddEditTeacher.aspx):
- In the Teachers folder, create a file named AddEditTeacher.aspx.
- Design this page to simulate adding or editing teacher records.
- Include form elements for name, subject, etc.

Step 5: Course Management

- Course List Page (CourseList.aspx):
- In the Courses folder, create a file named CourseList.aspx.
- Design this page to simulate displaying a list of courses.
- Manually create placeholders for course codes, titles, instructors, etc.
- Course Details Page (CourseDetails.aspx):
- In the Courses folder, create a file named CourseDetails.aspx.
- Design this page to display detailed information about a selected course.
- Manually create placeholders for course details such as code, title, instructor, etc.
- Add/Edit Course Page (AddEditCourse.aspx):
- In the Courses folder, create a file named AddEditCourse.aspx.
- Design this page to simulate adding or editing course records.
- Include form elements for course code, title, instructor, etc.

Step 6: Mock Data

- Create a static class called MockData that provides mock data for the prototype. You can use this data to populate the pages with sample information.

Step 7: Navigation Flow

- Ensure that navigation links/buttons on each page are correctly connected to the corresponding pages.
- Include "Back" or "Cancel" buttons to simulate navigating back in the flow.

Step 8: Styling

- Create a CSS file (e.g., styles.css) in the "Styles" folder.
- Apply basic CSS styles to improve the appearance of your prototype. You can also use a CSS framework like Bootstrap for faster styling.
- Maintain a consistent design theme across all pages to provide a unified experience.

Step 9: Testing

- Manually test the navigation flow by clicking through the different sections and ensuring that the links/buttons work as intended.

Step 10: Feedback and Iteration

- Collect feedback on the prototype from stakeholders.
- Use their input to refine the UI, navigation flow, and overall user experience.
- Iterate on the prototype as needed to address feedback and make improvements.
- By following these steps, you can create a functional ASP.NET WebForms prototype for managing a school database, focusing on the UI and navigation flow, which will help Rainbow School get a clear idea of what is required for the actual application.

Step 11: Upload on Github

- Git Hub Repository Link:
- **https://github.com/rastogi102/Phase3_Practice-Projects/tree/d40fd980758365a1eaa87950bc7dc29aa21e7f75/Section%205/WebAppSchoolManagement**