### **Project Title: Interactive Personal Portfolio Website**

#### **Objective Suggestion**

To build an advanced, interactive personal portfolio website showcasing your skills, projects, and personality. This project will solidify your understanding of responsive design with Flexbox/Grid, Bootstrap, animations, and JavaScript. You’ll also use GitHub Pages for deployment and practice structuring your code for readability and reusability.

#### **Project Requirements**

1. **Project Setup**
   * Use **HTML, CSS (including Flexbox/Grid), and JavaScript**.
   * Integrate **Bootstrap** for core styling, with additional custom CSS.
   * Deploy to **GitHub Pages** and ensure a clean, functional layout.
2. **Pages and Structure**
   * **Home Page**: Include an animated intro section with a welcome message, a brief description, and a “scroll down” or “explore” button that smoothly scrolls to the next section.
   * **About Page**: Display information about yourself, such as your skills, a timeline of your experiences, and hobbies. Include skill bars that fill up on scroll using animations.
   * **Projects Page**: Show at least 4-5 projects with images, descriptions, and live/demo links. Include a filter (using JavaScript) to sort projects by category (e.g., “JavaScript,” “HTML/CSS,” “Python”).
   * **Contact Page**: An interactive form with fields for Name, Email, Subject, and Message. Include enhanced form validation with error messages for each field if they’re left empty or formatted incorrectly.
3. **Styling Requirements**
   * Use **Flexbox/Grid** for structuring and aligning sections and cards.
   * Use **Bootstrap** for styling and customizing elements, but add a unique look with additional CSS styles.
   * Make use of **CSS animations** to add interest. For example, fade-in effects for sections as you scroll, hover effects on project cards, and smooth transitions between elements.
   * **Dark Mode Toggle**: Implement a light/dark theme toggle with JavaScript and save the user’s theme preference in localStorage so it persists across page loads.
4. **JavaScript Requirements**
   * **Dynamic Skill Bars**: Implement animated skill bars that fill up based on the user’s scrolling position.
   * **Project Filtering**: Add a filter button or dropdown on the Projects page to filter projects by type/category.
   * **Contact Form Validation**: Write JavaScript to validate each field individually with error messages and to submit the form if all fields are correct.
   * **Theme Toggle**: Implement the dark mode toggle as mentioned above.
5. **API Integration (Optional for Extra Challenge)**
   * **GitHub Repos**: Fetch and display your latest GitHub repositories on the Projects page using GitHub’s REST API. Show the repo name, description, and a link.
   * **Weather Widget**: Add a small weather widget using a free weather API. Display the current weather for your location or a location of your choice.
6. **Deployment**
   * Push code to a GitHub repository and publish it to **GitHub Pages**.
   * Update the **README.md** with a brief description, setup instructions, and a link to the live GitHub Pages site.

#### **Instructions**

1. **Project Setup**
   * Start by creating your repository on GitHub, clone it locally, and set up an organized file structure.
   * Link **Bootstrap** and any additional icons or libraries you need in your HTML.
   * Plan the layout and design for each page, making use of wireframes or mockups.
2. **Page Layouts**
   * **Home Page**: Use CSS animations or a JavaScript library like **GSAP** for intro animations.
   * **About Page**: Include skill bars that animate on scroll.
   * **Projects Page**: Use Flexbox/Grid to display projects in a responsive grid. Implement the filtering functionality to let users filter projects by category.
   * **Contact Page**: Implement custom validation and user-friendly error messages.
3. **JavaScript Functionality**
   * Implement the skill bar animations to activate when the About section comes into view.
   * Create an array of project objects and write functions to filter and display them based on user-selected categories.
   * Use event listeners to validate the contact form fields in real-time, displaying errors if fields are incomplete or incorrect.
   * Implement dark mode with a JavaScript toggle button and store the preference in localStorage.
4. **API Integration (Optional for Extra Challenge)**
   * Use GitHub’s REST API to fetch and display some of your recent repositories.
   * Use a weather API to display current weather information in a small widget on the home page.
5. **Deployment to GitHub Pages**
   * Push your final code to the GitHub repository.
   * Go to **Settings > Pages** in the repository and enable GitHub Pages.
   * Check the live URL and ensure everything is functioning as expected.

#### **Evaluation Criteria**

* **HTML Structure and Semantics**: Is the HTML organized, semantic, and structured?
* **CSS & Flexbox/Grid Usage**: Are Flexbox/Grid used effectively for a polished, responsive layout?
* **JavaScript Interactivity**: Are animations and interactive elements well-implemented?
* **API Integration**: Is the GitHub API and/or weather API integrated and functional?
* **Dark Mode**: Is the dark mode toggle implemented, and does it persist using localStorage?
* **Form Validation**: Is form validation effective, user-friendly, and comprehensive?
* **Deployment**: Is the project deployed on GitHub Pages and does the link work?