**Angular Use cases**

**1. Create a Simple Counter Component**

**Description:**  
Create a counter component with "Increment" and "Decrement" buttons. The counter value should be displayed dynamically.

**Requirements:**

* Use an Angular component to display the counter.
* Create two buttons to increase and decrease the count.
* Display the current count in the UI.
* Use event binding ((click)) to update the counter value.

**2. Create a Simple Input Binding Example**

**Description:**  
Build a component where a user types in an input field, and the entered text is displayed dynamically.

**Requirements:**

* Use [(ngModel)] for two-way data binding.
* Display the input value in real-time.
* Ensure the form field is required.

**3. Implement a Custom Structural Directive**

**Description:**  
Create a custom directive called appHighlight that changes the background color of an element when hovered.

**Requirements:**

* Use @Directive to create the directive.
* Use HostListener to listen for mouse events.
* Apply the directive to text or button elements in a template.

**Intermediate Exercises**

**4. Build a Todo List with Reactive Forms**

**Description:**  
Create a simple Todo List where users can add and remove tasks using **Reactive Forms**.

**Requirements:**

* Use FormGroup and FormControl for form handling.
* Validate input (minimum 3 characters, required field).
* Display a list of tasks dynamically.
* Implement a delete button for each task.

**5. Fetch Data from a REST API using HttpClient**

**Description:**  
Build a service that fetches user data from a fake API and displays it in a component.

**Requirements:**

* Use HttpClientModule to make API requests.
* Fetch user data from <https://jsonplaceholder.typicode.com/users>.
* Display the user list in a table using \*ngFor.
* Handle errors if the API call fails.

**6. Implement Routing with Lazy Loading**

**Description:**  
Set up Angular routing with lazy loading for different feature modules.

**Requirements:**

* Create two feature modules (Dashboard and Profile).
* Configure lazy loading for each module in app-routing.module.ts.
* Implement navigation using routerLink.
* Add a wildcard (\*\*) route to show a "Page Not Found" message.

**7. Create a Shopping Cart using Services**

**Description:**  
Build a basic shopping cart where users can add products, view the cart, and remove items using an Angular service.

**Requirements:**

* Use an Angular service (CartService) to manage cart items.
* Display available products in a component.
* Allow users to add/remove products to/from the cart.
* Show the cart items on a separate page using Angular routing.