Creating a lab exercise to practice the `@Input()` and `@Output()` decorators in Angular helps in understanding component interaction and data flow. This exercise will guide learners through building a simple Angular application that consists of a parent component and two child components, demonstrating how to pass data from the parent to the child and how to emit events from the child back to the parent.

Lab Exercise: Building a User Profile Application – Component Interaction

Objective

To understand and practice using `@Input()` and `@Output()` decorators in Angular for component communication.

Requirements

- Angular CLI installed on your machine.
- Basic understanding of Angular components, modules, and decorators.

Setup

1. Create a new Angular project: `ng new user-profile-app`.
2. Navigate to the project directory: `cd user-profile-app`.
3. Serve your application to verify its setup: `ng serve`.
4. Open your project in your preferred code editor.

Exercises

Exercise 1: Creating Components

1. Generate three components: `UserProfile`, `UserDetail`, and `UserEdit`.

```
ng generate component UserProfile
ng generate component UserDetail
ng generate component UserEdit
```

2. The `UserProfile` component will act as the parent component, while `UserDetail` and `UserEdit` will be child components.

Exercise 2: Using @Input()

1. In `UserDetail`, define an `@Input()` property named `userInfo` to receive user data (e.g., name, email) from the `UserProfile` component.
2. In `UserProfile`, create a user object with mock data and pass it to the `UserDetail` component in its template using property binding.

Exercise 3: Editing User Information

1. In `UserEdit`, define an `@Input()` property to receive the current user data from the `UserProfile` component.
2. Create a simple form in `UserEdit` with fields bound to the user data properties using two-way binding (you'll need to import `FormsModule` in your `app.module.ts`).
3. Add a save button to the form in `UserEdit`.

Exercise 4: Using @Output() and EventEmitter

1. In `UserEdit`, define an `@Output()` property named `updateUser` using `EventEmitter`.
2. Emit the updated user data back to the `UserProfile` component when the save button is clicked.
3. In `UserProfile`, handle the `updateUser` event to update the user object with the emitted data.

Exercise 5: Displaying Updated User Information

1. After updating the user data in `UserProfile`, ensure the `UserDetail` component reflects the updated information. This will test the dynamic data flow between parent and child components.

Conclusion

This lab exercise will have guided you through creating a basic user profile application that demonstrates component interaction in Angular using `@Input()` and `@Output()` decorators. You'll have practiced passing data from a parent component to child components and emitting events from child components back to the parent.