**Problem 4: code reviwe**

@Component({

selector: 'app-user-panel',

template: `<div class="user-panel">{{ getUserName() }}</button>`,

})

export class UserPanelComponent {

private \_user;

constructor(

private \_authService: AuthService,

) {

this.\_authService

.user

.subscribe(user => {

this.\_user = user;

});

}

1) What’s wrong with this code snippet?

We have a UserPanelComponent that uses a function call getUserName() in its template to display the full name of a user that is passed in via the private \_user property.

The getUserName() function is executed every time Angular change detection runs. And that can happen many times!

Every time the button is clicked, the getUserName() function is executed.

Because Angular cannot predict whether the return value of getUserName() has changed, it needs to execute the function every time change detection runs.

So if change detection runs 100 times, the function is called 100 times, even if its return value never changes.

2) How can we improve it?

A first approach might be to reduce function calls is to use pure pipes.

By telling Angular that a pipe is pure, Angular knows that the pipe’s return value does not change if the pipe’s input does not change.

To avoid unnecessary function calls we can manually calculate the values we need in our view from within the component’s controller. In this case we can either declare a new public variable to store the full name inside this or we can change \_user property's access modifier from private to public to use it.

3) What benefits and drawbacks of each method?

1st Meethod:

Benefits:

We can use pure pipes to avoid multiple function calls in every Angular's change detection cycle.

Drawbacks:

Pure pipes only execute when their input values change.

2nd Method:

Benefits: By storing calculated value in a variable, we can easily use it in both component and template.

Drawbacks: have to first create a method to calculate the expected value and then store it in a declared variable.