**Design**

Consistent Styling: Ensure consistent styling across your application. For example, you have used both inline styles (style="margin-left: 58px;") and external styles (CSS classes). Consider moving all styles to external stylesheets for better maintainability.

Reusability: If you find that certain components or styles are repeated across different parts of your application, consider creating reusable components or styles to avoid code duplication.

Button Styles: Consider using Angular Material buttons (<button mat-button>) instead of the regular <button> for a more consistent look and feel.

Accessibility: Ensure that your application is accessible. This includes providing appropriate labels, keyboard navigation, and ensuring that your UI is readable and usable by people with disabilities.

Comments: Add comments where necessary, especially for complex logic or sections that might not be immediately clear to someone else reading the code.

Testing: Consider adding unit tests for your components, especially if they contain complex logic or interactions.

Pagination Logic: Ensure that the pagination logic is working correctly, considering edge cases and potential issues that might arise with different data sets.

**Error Handling**

Consider providing more specific error messages or logging additional information when an error occurs during the login process. This can be helpful for debugging and user support.

.**Logging**

Remove the console.log(); statement from your code. While it can be useful for debugging during development, it's not recommended to leave such statements in production code.

**Unsubscribe from Observables**

**Use of Explicit Type**

**Route Redirection with Query Parameters:**

Consider redirecting the user to the login page with a query parameter indicating the intended destination. This way, after successful login, you can navigate the user back to the initially requested page.

Consider using more meaningful names for services, variables, methods etc.

**Centralized API Configuration:** If your application has multiple services, you might consider centralizing the API configuration (base URL, headers) in a dedicated service or utility function to avoid duplicating the configuration logic across services.