**Structure**

**❏ Does the code completely and correctly implement the design?**

1. Yes, the code seems to implement the design of user authentication and registration. The login functionality verifies the user’s credentials, and the registration functionality launches a registration activity.

**❏ Does the code conform to any pertinent coding standards?**

1. The code follows standard Java conventions for Android development. However, there are minor improvements that could be made, such as ensuring proper commenting style and maintaining uniformity in variable naming.

❏ **Is the code well-structured, consistent in style, and consistently formatted?**

1. The code is mostly well-structured and formatted. However, there are some areas where readability could be improved, such as adding more comments and possibly breaking down some functions into smaller, more manageable ones.

❏ **Are there any uncalled-for or unneeded procedures or any unreachable code?**

1. The code does not seem to contain any unneeded or unreachable code. Every function defined serves a purpose (authentication, handling the button clicks, starting activities, etc.).

**❏ Are there any leftover stubs or test routines in the code?**

1. No. The code does not contain any leftover stubs or test routines.

**❏ Can any code be replaced by calls to external reusable components or library functions?**

1. The authentication logic could potentially be refactored or modularized, but it's straightforward and self-contained. Libraries like Firebase Authentication or other authentication solutions could be integrated, reducing the need for manual database verification.

❏ **Are there any blocks of repeated code that could be condensed into a single procedure?**

1. The code does not appear to contain repetitive blocks. The flow between the login and registration activities is distinct, and no obvious refactoring opportunities stand out.

❏ **Is storage use efficient?**

1. There's no clear indication of inefficient storage usage, though this code mainly interacts with user input and the database, which appears efficient.

❏ **Are symbolics used rather than “magic number” constants or string constants?**

1. The code uses string constants (e.g., "regUsername", "authUsername"). It would be better to define these as static constants, improving maintainability.

❏ **Are any modules excessively complex and should be restructured or split into multiple routines?**

1. The onCreate() method could be refactored into smaller, more focused functions. For example, the button click listeners and the ActivityResultLauncher setup could be moved into their respective methods to improve readability.

**Documentation**

❏ **Is the code clearly and adequately documented with an easy-to-maintain commenting style?**

1. The code has very few comments. Adding comments would improve clarity, particularly for people unfamiliar with the codebase.

❏ **Are all comments consistent with the code?**

1. There are no comments to check for consistency with the code.

**Variables**

**❏ Are all variables properly defined with meaningful, consistent, and clear names?**

1. Yes, the variables are clearly named (mUsername, mPassword, mAuthError, etc.) and convey their purposes well.

**❏ Do all assigned variables have proper type consistency or casting?**

1. Yes, the variables are of the correct types and don’t require casting.

**❏ Are there any redundant or unused variables?**

1. No, all variables appear to be used appropriately.

**Arithmetic Operations**

❏ Does the code avoid comparing floating-point numbers for equality?

The code does not involve any arithmetic operations, so this section is not relevant here.

❏ Does the code systematically prevent rounding errors?

❏ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?

❏ Are divisors tested for zero or noise? Loops and Branches

**❏ Are all loops, branches, and logic constructs complete, correct, and properly nested?**

1. There are no loops, but the if-else branches (in authUser()) are well-structured and function as expected.

**❏ Are the most common cases tested first in IF- -ELSEIF chains?**

1. There’s only one primary if-else condition in the authUser() method, which handles the authentication outcome. This logic is simple and works fine

❏ **Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?**

1. Yes, the authUser() method handles both authenticated and non-authenticated cases.

**❏ Does every case statement have a default?**

1. N/A (no switch or case blocks).

**❏ Are loop termination conditions obvious and invariably achievable?**

1. N/A (no loops).

**❏ Are indexes or subscripts properly initialized, just prior to the loop?**

1. N/A (no loops).

**❏ Can any statements that are enclosed within loops be placed outside the loops?**

1. N/A (no loops).

❏ **Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?**

1. N/A (no loops).

**Defensive Programming**

❏ **Are indexes, pointers, and subscripts tested against array, record, or file bounds?**

1. The code doesn't use arrays, so this is not applicable.

❏ **Are imported data and input arguments tested for validity and completeness?**

1. User inputs (username, password) are tested when performing authentication, but there's room for additional validation (e.g., non-empty username and password checks).

❏ **Are all output variables assigned?**

1. Yes, the output variables (such as regName in the result handler) are properly assigned.

**❏ Are the correct data operated on in each statement?**

1. Yes, the data is correct in each statement.

**❏ Is every memory allocation deallocated?**

1. No explicit memory allocation/deallocation is needed here since Android's garbage collector manages memory.

**❏ Are timeouts or error traps used for external device accesses?**

1. No timeouts or error traps are used for external device access. However, it would be good practice to add error handling for database interactions and network requests (if applicable).

❏ **Are files checked for existence before attempting to access them?**

1. The code doesn't handle file access, so this is not applicable.

**❏ Are all files and devices left in the correct state upon program termination?**

1. The app seems to clean up appropriately by finishing the activity when authentication is successful or when the registration activity is completed.