Assignment Title: Simulating a Biometric Authentication System in Java

---

Assignment Description:

Biometric authentication is a cutting-edge technology that uses unique biological traits for identity verification. In this assignment, you will implement a simplified biometric authentication system in Java. Your program will simulate biometric data, such as fingerprints or voiceprints, for user authentication using hash-based or pattern-matching techniques.

Your program will:

1. Enroll users by capturing their biometric data (simulated as unique strings or patterns).
2. Store the biometric data securely.
3. Authenticate users by comparing new biometric inputs with stored data.

Requirements:

1. **User Enrollment**:
   - Allow users to enroll by providing a username and simulated biometric data (e.g., a string or numerical pattern).
   - Hash and securely store the biometric data in memory (simulating a database).
2. **Authentication**:
   - Allow users to authenticate by providing their username and simulated biometric data.
   - Compare the new biometric data with the stored data using pattern-matching or hash comparison.
   - Provide a success or failure message based on the result.
3. **Error Handling**:
   - Handle cases such as non-existent accounts, incorrect biometric inputs, or empty fields.

    - Provide meaningful error messages for invalid inputs.

4. **Security Measures**:
    - Use hashing and salting to securely store biometric data.
    - Ensure no plaintext biometric data is stored in memory.
5. **Biometric Matching Algorithm**:
    - Implement a simple matching mechanism that accounts for minor variations in the biometric input (e.g., allow small deviations in patterns or characters).
6. **Documentation and Testing**:
    - Include comments explaining the purpose of each major code section.
    - Provide at least three test cases, including successful authentication, failed authentication, and invalid input handling.
7. **Optional Enhancements** (Extra Credit):
    - Implement a more advanced matching algorithm for fuzzy comparisons (e.g., Levenshtein distance for string patterns).
    - Save user data to a file and reload it on program restart.

Deliverables:

1. **Java Source Code**:
    - Submit .java file(s), ensuring the code is well-structured, documented, and adheres to Java coding conventions.
2. **Test Results**:
    - Provide a document summarizing test cases with:
        - Enrolled users and their biometric data.
        - Outcomes of authentication attempts.
3. **Readme**:
    - Include a README file in Word or PDF format explaining how to run your program, its dependencies, and any optional features.

4. **Optional Enhancements** (Extra Credit):
   - Demonstrate advanced matching or file-based storage with additional test cases.

Submission Guidelines:

1. Submit your .java file(s) and test results via Blackboard.
2. Include a README file in Word or PDF format explaining how to run your program, its dependencies, and any optional features.

10-Point Rubric:

Criteria	**Points**	**Description**
Biometric Data Storage Security	2	Securely stores biometric data using hashing and salting.
User Enrollment Functionality	1	Allows users to enroll with unique simulated biometric data.
Authentication Functionality	1	Correctly authenticates users based on their biometric data.
Matching Algorithm	1	Implements a robust matching algorithm with some tolerance for input variation.
Error Handling	1	Provides meaningful error messages for invalid inputs or failed authentications.
Output Clarity	1	Clearly displays messages for success and failure during enrollment and authentication.
Documentation/Comments	1	Includes clear comments explaining major code sections and logic.
Test Cases and Results	1	Provides at least three test cases demonstrating the system's functionality.
Optional Enhancements	1	Implements advanced matching algorithms or file-based data storage. (Extra Credit)
Overall Functionality	1	Program runs correctly, achieving the goals of the assignment.