# Report on Authentication System

## Overview

The Authentication System is a Java-based program that allows users to register, log in, and search user data stored in a CSV file. It ensures secure handling of passwords through hashing and provides essential functionalities for managing user accounts.

## Functionalities

### 1. Registration

The registration process allows a user to create an account by providing a username and password. The password is hashed using the SHA-256 algorithm for security and stored alongside the username in a CSV file.

- Process:

1. 1. The user enters a username and password.  
   2. The password is hashed using the `hashPassword` method.  
   3. The username and hashed password are appended to the CSV file.

Code Explanation:

**public static void register(Scanner scanner) throws IOException, NoSuchAlgorithmException {  
 System.out.print("Enter username: ");  
 String username = scanner.nextLine(); // Read username from the user.  
  
 System.out.print("Enter password: ");  
 String password = scanner.nextLine(); // Read password from the user.  
  
 String hashedPassword = hashPassword(password); // Hash the password using SHA-256.  
  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(USER\_FILE, true))) {  
 writer.write(username + "," + hashedPassword); // Append username and hashed password to the file.  
 writer.newLine(); // Add a new line for the next entry.  
 System.out.println("Registration successful!");  
 }  
}**

### 2. Login

The login process verifies a user’s credentials by comparing the provided username and hashed password with the entries in the CSV file.

- Process:

1. 1. The user enters a username and password.  
   2. The password is hashed using the same hashing method as during registration.  
   3. The system checks if the username and hashed password match any entry in the CSV file.

Code Explanation:

**public static void login(Scanner scanner) throws IOException, NoSuchAlgorithmException {  
 System.out.print("Enter username: ");  
 String username = scanner.nextLine(); // Read username input.  
  
 System.out.print("Enter password: ");  
 String password = scanner.nextLine(); // Read password input.  
  
 String hashedPassword = hashPassword(password); // Hash the entered password.  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(USER\_FILE))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(","); // Split the line into username and hashed password.  
 if (parts[0].equals(username) && parts[1].equals(hashedPassword)) { // Check for a match.  
 System.out.println("Login successful!");  
 return;  
 }  
 }  
 }  
 System.out.println("Invalid username or password. Please try again.");  
}**

### 3. Search Functionality

Users can search for entries in the CSV file using keywords.

- Process:

1. 1. The user provides a keyword.  
   2. The system reads the CSV file line by line and checks if the keyword exists in any line.  
   3. Matching entries are displayed, or a message is shown if no matches were found.

Code Explanation:

**public static void searchEntries(String keyword) throws IOException {  
 try (BufferedReader reader = new BufferedReader(new FileReader(USER\_FILE))) {  
 String line;  
 boolean found = false;  
  
 while ((line = reader.readLine()) != null) {  
 if (line.contains(keyword)) { // Check if the line contains the keyword.  
 System.out.println("Match: " + line); // Print matching line.  
 found = true;  
 }  
 }  
  
 if (!found) {  
 System.out.println("No matches found.");  
 }  
 }  
}**

## Key Components

### 1. Password Security

The program uses the `MessageDigest` class to hash passwords with the SHA-256 algorithm. This ensures that passwords are stored securely and cannot be easily retrieved even if the CSV file is compromised.

Code Explanation:

**private static String hashPassword(String password) throws NoSuchAlgorithmException {  
 MessageDigest md = MessageDigest.getInstance("SHA-256"); // Create a MessageDigest instance for SHA-256.  
 byte[] hashedBytes = md.digest(password.getBytes()); // Hash the password bytes.  
  
 StringBuilder sb = new StringBuilder();  
 for (byte b : hashedBytes) {  
 sb.append(String.format("%02x", b)); // Convert each byte to a hexadecimal string.  
 }  
 return sb.toString(); // Return the hashed password as a string.  
}**

### 2. File I/O

The system uses `BufferedReader` and `BufferedWriter` for reading and writing to the `users.csv` file. This ensures efficient handling of file operations.

### 3. CSV File Format

User data is stored in the CSV file with the following structure:  
```  
**Username,HashedPassword  
user001,<hashed\_password>  
user002,<hashed\_password>**```

### 4. Error Handling

The system gracefully handles common errors, such as invalid credentials during login and missing data during searches.

## Conclusion

The Authentication System is a simple yet effective program for managing user accounts. It incorporates secure password hashing and basic file-based storage for user data. While functional, enhancements such as password validation, duplicate checks, and improved search capabilities can make the system more robust and user-friendly.