Authentication Module Part 2 (Validations / Others)

validateUsername(username)

END

```
  // Length check
  IF username LENGTH < 3 OR username LENGTH > 15 THEN
    DISPLAY "Username must be between 3 and 15 characters." MESSAGE
    RETURN false
  END IF
  // Alphanumeric with underscores or hyphens
  IF NOT username MATCHES "[a-zA-Z0-9_\\-]+" THEN
    DISPLAY "Username can only contain letters, numbers, underscores, or
hyphens." MESSAGE
    RETURN false
  END IF
  // No spaces
  IF username CONTAINS " " THEN
    DISPLAY "Username cannot contain spaces." MESSAGE
    RETURN false
  END IF
  // First character must be a letter
  IF NOT Character.isLetter(username CHAR AT 0) THEN
    DISPLAY "Username must start with a letter." MESSAGE
    RETURN false
  END IF
  // Reserved words check
  SET reservedWords TO ["admin", "root", "null"]
  FOR EACH word IN reservedWords
    IF username TO LOWERCASE CONTAINS word THEN
```

```
        DISPLAY "Username cannot contain reserved words like 'admin', 'root', or 'null'." MESSAGE
        RETURN false
      END IF
  END FOR
  // Username already exists check
  SET filePath TO "src/storage/user.csv"
  IF usernameExistsInTheFile(username, filePath) THEN
      DISPLAY "Username already exists." MESSAGE
      RETURN false
  END IF
  // Valid username
  RETURN true
END
```

usernameExistsInTheFile(username, filePath)

START

```
  SET file TO new File(filePath)
  TRY
      OPEN scanner FOR file
      // Skip the header
      IF scanner HAS next line THEN
        scanner.nextLine()
      END IF
      // Iterate through each line in the file
      WHILE scanner HAS next line
        READ line FROM scanner
        SPLIT line BY "," INTO userDetails
        SET existingUsername TO userDetails[3]
        // Check if the username matches
        IF existingUsername EQUALS IGNORE CASE username THEN
          CLOSE scanner
          RETURN true // Username found
        END IF
      END WHILE
      CLOSE scanner
    CATCH FileNotFoundException
      DISPLAY "File not found: " + filePath MESSAGE
      PRINT stack trace
    END TRY
    RETURN false // Username not found
```
END

validateName(name, fieldName)

END

```
    IF name IS EMPTY THEN
      DISPLAY fieldName + " cannot be empty." MESSAGE
      RETURN false
    END IF
    SET nameRegex TO "^(?!.*\\s)(!?[A-Z][a-zA-Z]*)$"
    IF NOT name MATCHES nameRegex THEN
      DISPLAY "Only letters and '!' are allowed, and it must start with an uppercase letter." MESSAGE
      RETURN false
    END IF
    RETURN true
```

```
END

validatePasswordFormat(password)
START
  SET passwordRegex TO "^(?=.*[A-Z])(?=.*\\d)(?=.*[@$!%*?&])[A-Za-z\\d@$!%*?&]{8,}$"
  IF NOT password MATCHES passwordRegex THEN
    DISPLAY "Password must contain at least one uppercase letter, one number, one special character, and be at least 8 characters long." MESSAGE
    RETURN false
  END IF
  RETURN true
END

validatePasswordMatch(password, confirmPassword)
STAR
  IF NOT password EQUALS confirmPassword THEN
    DISPLAY "Passwords do not match." MESSAGE
    RETURN false
  END IF

  RETURN true
END

hashPassword(password)
START
  TRY
    SET md TO MessageDigest.getInstance("SHA-256")
    // Convert the password string to a byte array
    SET hashedBytes TO md.digest(password.getBytes())
    // Convert the hashed byte array to a hex string
    SET sb TO new StringBuilder()
    FOR EACH byte b IN hashedBytes
      APPEND String.format("%02x", b) TO sb
    END FOR
    RETURN sb.toString() // Return the hashed password as a hex string
  CATCH NoSuchAlgorithmException
    THROW new RuntimeException("Error hashing the password", e)
  END TRY
END
```

authenticateUser(username, password)

START

```
  SET file TO new File("src/storage/user.csv")
  TRY
    OPEN scanner FOR file
    // Skip the header line
    IF scanner HAS next line THEN
      scanner.nextLine()
    END IF
    // Iterate through the file and check for the username
    WHILE scanner HAS next line
      READ line FROM scanner
      SPLIT line BY "," INTO userDetails
      IF userDetails LENGTH >= 5 THEN
        SET fileUsername TO userDetails[3].trim()
        SET filePasswordHash TO userDetails[4].trim()
```

```
            // Check if the username matches
            IF fileUsername EQUALS username THEN
              // Hash the entered password
              SET enteredPasswordHash TO hashPassword(password)
              // Compare the entered password's hash with the stored password hash
              IF enteredPasswordHash EQUALS filePasswordHash THEN
                // If both username and password match, authentication is successful
                SET loggedInUserID TO userDetails[0]
                RETURN true
              END IF
            END IF
          END IF
        END WHILE
      CATCH FileNotFoundException
        DISPLAY "Error: User file not found!" MESSAGE
        PRINT stack trace
      END TRY
      // If username or password is incorrect, return false
      RETURN false
    END
```

validatePhoneNumber(phoneNumber)

```
START
  SET phoneRegex TO "^(\\+\\d{1,3}|0)\\d{7,12}$"
  IF NOT phoneNumber MATCHES phoneRegex THEN
    DISPLAY "Invalid phone number. It must start with +country code or 0 and be a valid length." MESSAGE
    RETURN false
  END IF
  RETURN true
END

validateEmail(email)
START
  SET emailRegex TO "^[a-zA-Z0-9+_.-]+@[a-zA-Z0-9.-]+$"
  IF NOT email MATCHES emailRegex THEN
    DISPLAY "Invalid email format." MESSAGE
    RETURN false
  END IF

  RETURN true
END

validatePhysicalAddress(address)
START
  IF address LENGTH < 5 THEN
    DISPLAY "Physical address must be at least 5 characters long." MESSAGE
    RETURN false
  END IF
  RETURN true
END

validateContactGroup(contactGroup)
START
  SET groupRegex TO "^[a-zA-Z ]+$"
  IF NOT contactGroup MATCHES groupRegex THEN
```

```
        DISPLAY "Invalid contact group name. Only letters and spaces are allowed."
MESSAGE
        RETURN false
    END IF
    RETURN true
END
```