Test Plan - Online Medical System

Test File: RegisterTest.php

Identifier: SIGNUP-SUCCESS-TEST

Test Case: The system shall successfully register a new user account when valid input values are provided.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testSuccess()` method in the `RegisterTest` class.

Expected Results: The system should return `True`, indicating a successful registration.

Postconditions: A new user account is created in the database.

Identifier: SIGNUP-FAILURE-INVALID-NAME-TEST

Test Case: The system shall reject registration when the user provides an invalid name.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"

2. Middle name: 1 (invalid value)
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_1()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to an invalid name.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-INVALID-PHONE-TEST

Test Case: The system shall reject registration when the user provides an invalid phone number.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+123456789" (invalid format)
6. Email: "john.doe@example.com"
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_2()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to an invalid phone number.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-INVALID-EMAIL-TEST

Test Case: The system shall reject registration when the user provides an invalid email address.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "johndoe@example" (invalid format)
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_3()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to an invalid email address.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-INVALID-BIRTHDATE-TEST

Test Case: The system shall reject registration when the user provides an invalid birthdate (in the future).

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "2050-05-01" (future date)
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_4()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to an invalid birthdate.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-INVALID-USERNAME-TEST

Test Case: The system shall reject registration when the user provides an invalid username.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: -1 (invalid value)
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_5()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to an invalid username.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-PASSWORD-MISMATCH-TEST

Test Case: The system shall reject registration when the user provides mismatching password

and confirmation password.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "DifferentPassword123" (mismatch)

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_6()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to a password mismatch.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-EXISTING-USERNAME-TEST

Test Case: The system shall reject registration when the user provides an already existing username.

Preconditions:

1. The database is available and can be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.
4. There is already a user account with the provided username in the database.

Input Values:

1. First name: "John"
2. Middle name: "Doe"
3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: "existinguser123" (already exists)
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_7()` method in the `RegisterTest` class.

Expected Results: The system should return `False`, indicating a failed registration due to an existing username.

Postconditions: No new user account is created in the database.

Identifier: SIGNUP-FAILURE-DATABASE-ERROR-TEST

Test Case: The system shall handle database errors during the registration process.

Preconditions:

1. The database is not available or cannot be accessed.
2. The required PHP dependencies are properly configured.
3. The necessary web server environment is set up.

Input Values:

1. First name: "John"
2. Middle name: "Doe"

3. Last name: "Smith"
4. Birth: "1990-05-01"
5. Phone: "+1234567890"
6. Email: "john.doe@example.com"
7. Username: "johndoe123"
8. Password: "P@ssw0rd123"
9. Confirm Password: "P@ssw0rd123"

Execution Steps:

1. Ensure the necessary environment and dependencies are set up.
2. Execute the `testFailure_8()` method in the `RegisterTest` class.

Expected Results: The system should handle the database error gracefully and return `False`, indicating a failed registration.

Postconditions: No new user account is created in the database.

Test Identifier	Expected Outcome	Actual Outcome
SIGNUP-SUCCESS-TEST	True	True
SIGNUP-FAILURE-INVALID-NAME-TEST	False	False
SIGNUP-FAILURE-INVALID-PHONE-TEST	False	False
SIGNUP-FAILURE-INVALID-EMAIL-TEST	False	False
SIGNUP-FAILURE-INVALID-BIRTHDATE-TEST	False	False
SIGNUP-FAILURE-INVALID-USERNAME-TEST	False	False
SIGNUP-FAILURE-PASSWORD-MISMATCH-TEST	False	False
SIGNUP-FAILURE-EXISTING-USERNAME-TEST	False	False
SIGNUP-FAILURE-DATABASE-ERROR-TEST	False	False