**1. Register Method**

**Phân tích giá trị biên (Boundary Value Analysis):**

* **Đầu vào request.phone:**
  + Số điện thoại hợp lệ có độ dài tối thiểu và tối đa (nếu có ràng buộc).
  + Số điện thoại có định dạng gần với biên hợp lệ (ví dụ: thiếu một ký tự, thừa một ký tự).

**Phân lớp tương đương (Equivalence Partitioning):**

* **Đầu vào request.phone:**
  + Số điện thoại hợp lệ chưa tồn tại.
  + Số điện thoại hợp lệ đã tồn tại.
  + Số điện thoại không hợp lệ (ví dụ: sai định dạng, chứa ký tự không phải số).

**Kiểm thử lỗi (Error Guessing):**

* Gửi request với số điện thoại null hoặc rỗng.
* Mô phỏng lỗi khi gọi các service phụ thuộc (userRepository, otpService, registerUserService, kafkaProducer).

**Test Cases:**

1. **Số điện thoại hợp lệ, chưa tồn tại:** Kiểm tra xem có tạo transactionId, lưu thông tin đăng ký và gửi OTP thành công hay không.
2. **Số điện thoại hợp lệ, đã tồn tại:** Kiểm tra xem có ném RuntimeException với thông báo "Đã tồn tại tài khoản với số điện thoại này" hay không.
3. **Số điện thoại không hợp lệ (ví dụ: chữ cái):** Kiểm tra xem phương thức normalizePhoneNumber có xử lý đúng không (mặc dù logic chính của register có thể không quan tâm đến định dạng trước khi normalize).
4. **Số điện thoại null:** Kiểm tra xem có lỗi xảy ra khi PhoneNumberUtils.normalizePhoneNumber(null) được gọi hay không.
5. **Lỗi khi lưu vào repository:** Mock userRepository.findByPhone để không tìm thấy user, nhưng mock registerUserService.insertNew để ném exception. Kiểm tra xem exception có bị xử lý không.
6. **Lỗi khi generate OTP:** Mock otpService.generateOtp để ném exception. Kiểm tra xem exception có bị xử lý không.
7. **Lỗi khi insert user đăng ký:** Mock registerUserService.insertNew để ném exception. Kiểm tra xem exception có bị xử lý không.
8. **Lỗi khi gửi Kafka message:** Mock kafkaProducer.sendMessage để ném exception. Kiểm tra xem exception có bị xử lý không.

**2. ChangePassword Method**

**Phân tích giá trị biên (Boundary Value Analysis):**

* **Đầu vào độ dài mật khẩu (request.password, request.confirmPassword, request.oldPassword):** Kiểm tra với độ dài tối thiểu, tối đa (nếu có ràng buộc) và các giá trị gần biên.

**Phân lớp tương đương (Equivalence Partitioning):**

* **Trường hợp số điện thoại:**
  + Số điện thoại tồn tại.
  + Số điện thoại không tồn tại.
* **Trường hợp mật khẩu:**
  + Mật khẩu mới và xác nhận mật khẩu khớp nhau.
  + Mật khẩu mới và xác nhận mật khẩu không khớp nhau.
* **Trường hợp người dùng đã xác thực (user.getIsVerified() là true):**
  + Mật khẩu cũ đúng.
  + Mật khẩu cũ sai.
* **Trường hợp người dùng chưa xác thực (user.getIsVerified() là false):**
  + Không cần kiểm tra mật khẩu cũ.

**Bảng quyết định (Decision Table Testing):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tồn tại SĐT** | **Mật khẩu khớp** | **Đã xác thực** | **Mật khẩu cũ đúng** | **Kết quả** |
| False | - | - | - | Trả về "Số điện thoại không tồn tại" |
| True | False | - | - | Trả về "Mật khẩu không khớp" |
| True | True | False | - | Cập nhật mật khẩu, isVerified = true, isRequiredChangePw = false, trả về "Mật khẩu đã được thay đổi" |
| True | True | True | False | Trả về "Mật khẩu cũ không đúng" |
| True | True | True | True | Cập nhật mật khẩu, isVerified = true, isRequiredChangePw = false, trả về "Mật khẩu đã được thay đổi" |

Xuất sang Trang tính

**Kiểm thử lỗi (Error Guessing):**

* Gửi request với các trường null hoặc rỗng.
* Mô phỏng lỗi khi lưu user vào repository.

**Test Cases:**

1. **Số điện thoại không tồn tại:** Kiểm tra response trả về "Số điện thoại không tồn tại".
2. **Mật khẩu mới và xác nhận không khớp:** Kiểm tra response trả về "Mật khẩu không khớp".
3. **Người dùng đã xác thực, mật khẩu cũ sai:** Kiểm tra response trả về "Mật khẩu cũ không đúng".
4. **Người dùng đã xác thực, mật khẩu cũ đúng, mật khẩu mới hợp lệ:** Kiểm tra xem mật khẩu, isVerified, và isRequiredChangePw được cập nhật đúng và response trả về "Mật khẩu đã được thay đổi".
5. **Người dùng chưa xác thực, mật khẩu mới và xác nhận khớp:** Kiểm tra xem mật khẩu, isVerified, và isRequiredChangePw được cập nhật đúng và response trả về "Mật khẩu đã được thay đổi" (không kiểm tra mật khẩu cũ).
6. **Request với phone là null:** Kiểm tra lỗi.
7. **Request với password là null:** Kiểm tra lỗi.
8. **Lỗi khi userRepository.save ném exception:** Mock repository để ném lỗi và kiểm tra xem lỗi có bị xử lý không.

**3. GenerateOtp Method**

**Phân tích giá trị biên (Boundary Value Analysis):**

* Kiểm tra kết quả trả về có độ dài đúng (6 ký tự số).
* Kiểm tra khả năng tạo OTP ở biên của khoảng số ngẫu nhiên (gần 000000 và 999999).

**Phân lớp tương đương (Equivalence Partitioning):**

* Mỗi lần gọi phương thức sẽ tạo ra một OTP khác nhau (tính ngẫu nhiên).

**Kiểm thử lỗi (Error Guessing):**

* Khó có lỗi logic trực tiếp ở đây, chủ yếu kiểm tra tính đúng đắn của định dạng.

**Test Cases:**

1. **Kiểm tra độ dài OTP:** Gọi phương thức nhiều lần và đảm bảo độ dài chuỗi trả về luôn là 6.
2. **Kiểm tra định dạng OTP:** Gọi phương thức nhiều lần và đảm bảo chuỗi trả về chỉ chứa các ký tự số.
3. **Kiểm tra tính ngẫu nhiên (ở mức độ nhất định):** Gọi phương thức nhiều lần và quan sát xem các OTP có xu hướng khác nhau hay không (không cần kiểm tra thống kê chặt chẽ).

**4. ResendOtp Method**

**Phân tích giá trị biên (Boundary Value Analysis):**

* **otpProperties.getResendDelay():** Kiểm tra các trường hợp ngay trước, bằng và sau thời gian chờ cho phép gửi lại.
* **otpProperties.getMaxAttempt():** Kiểm tra các trường hợp số lần gửi lại bằng, nhỏ hơn và lớn hơn giới hạn cho phép.

**Phân lớp tương đương (Equivalence Partitioning):**

* **Trường hợp registerUserEntity:**
  + Tồn tại.
  + Không tồn tại.
* **Trường hợp thời gian gửi lại:**
  + Trong thời gian chờ.
  + Ngoài thời gian chờ.
* **Trường hợp số lần gửi lại:**
  + Nhỏ hơn giới hạn.
  + Bằng giới hạn.
  + Lớn hơn giới hạn.

**Biểu đồ trạng thái (State Transition Testing):**

Trạng thái của một phiên OTP có thể bao gồm:

* **New:** Vừa được tạo.
* **WaitingResend:** Đang trong thời gian chờ gửi lại.
* **ResendAllowed:** Đã hết thời gian chờ, có thể gửi lại.
* **MaxResendReached:** Đã đạt số lần gửi lại tối đa.
* **Expired/Invalid:** Phiên đã hết hạn hoặc không tồn tại.

Các chuyển đổi trạng thái:

* New -> WaitingResend (sau lần gửi đầu tiên).
* WaitingResend -> ResendAllowed (sau khi hết thời gian chờ).
* ResendAllowed -> WaitingResend (sau khi gửi lại OTP).
* ResendAllowed -> MaxResendReached (sau khi đạt số lần gửi lại tối đa).
* Any state -> Expired/Invalid (nếu transactionId không tồn tại).

**Kiểm thử lỗi (Error Guessing):**

* Gửi request với transactionId là null hoặc rỗng.
* Mô phỏng lỗi khi cập nhật entity trong repository.

**Test Cases:**

1. **transactionId không tồn tại:** Kiểm tra xem có ném RuntimeException với thông báo "Phiên làm việc không tồn tại hoặc đã hết hạn" hay không.
2. **Gửi lại OTP quá sớm:** Gọi resendOtp lần thứ hai trong khoảng thời gian otpProperties.getResendDelay(). Kiểm tra xem có ném RuntimeException với thông báo "Vui lòng đợi..." hay không.
3. **Gửi lại OTP sau thời gian chờ:** Gọi resendOtp lần thứ hai sau khoảng thời gian otpProperties.getResendDelay(). Kiểm tra xem OTP mới có được tạo và trả về thành công hay không, và otpResendTime được cập nhật.
4. **Gửi lại OTP đạt giới hạn tối đa:** Gọi resendOtp nhiều lần cho đến khi đạt otpProperties.getMaxAttempt(). Lần gọi tiếp theo phải ném RuntimeException với thông báo "Vượt quá số lần gửi lại OTP" và entity bị xóa.
5. **Gọi resendOtp sau khi đạt giới hạn:** Sau khi test case 4 xảy ra, gọi lại resendOtp với cùng transactionId. Kiểm tra xem có ném "Phiên làm việc không tồn tại hoặc đã hết hạn" hay không.
6. **Lỗi khi cập nhật registerUserRepository:** Mock registerUserService.renew để ném exception. Kiểm tra xem exception có bị xử lý không.

**5. validateOtp Method**

**Phân tích giá trị biên (Boundary Value Analysis):**

* Không áp dụng trực tiếp cho giá trị đầu vào OTP (thường là chuỗi cố định độ dài).

**Phân lớp tương đương (Equivalence Partitioning):**

* **Trường hợp registerUserEntity:**
  + Tồn tại.
  + Không tồn tại.
* **Trường hợp OTP:**
  + OTP đúng.
  + OTP sai.
* **Trường hợp số lần nhập sai:**
  + Nhỏ hơn giới hạn.
  + Bằng giới hạn (sau lần nhập sai này, entity bị xóa).

**Biểu đồ trạng thái (State Transition Testing):**

Trạng thái của quá trình xác thực OTP:

* **Active:** OTP hợp lệ, chưa hết hạn.
* **InvalidOtp:** OTP không đúng.
* **MaxAttemptsReached:** Đã nhập sai quá nhiều lần.
* **Expired:** OTP đã hết hạn hoặc transactionId không tồn tại.
* **Validated:** OTP hợp lệ, người dùng đã được tạo.

Các chuyển đổi trạng thái:

* Active -> Validated (khi OTP đúng).
* Active -> InvalidOtp (khi OTP sai, số lần thử chưa đạt giới hạn).
* InvalidOtp -> MaxAttemptsReached (khi nhập sai đạt giới hạn).
* Any state -> Expired (nếu transactionId không tồn tại).

**Kiểm thử lỗi (Error Guessing):**

* Gửi request với transactionId hoặc otp là null hoặc rỗng.
* Mô phỏng lỗi khi truy vấn hoặc lưu dữ liệu vào repository.

**Test Cases:**

1. **transactionId không tồn tại:** Kiểm tra xem có ném RuntimeException với thông báo "OTP đã hết hạn hoặc không tồn tại" hay không.
2. **OTP đúng:** Kiểm tra xem user mới có được tạo, entity đăng ký bị xóa hay không.
3. **OTP sai, số lần thử nhỏ hơn giới hạn:** Kiểm tra xem otpFail được tăng lên và ném RuntimeException với thông báo "OTP không chính xác".
4. **OTP sai, số lần thử bằng giới hạn:** Gọi validateOtp với OTP sai cho đến khi đạt otpProperties.getMaxAttempt(). Kiểm tra xem entity đăng ký bị xóa và ném RuntimeException với thông báo "Vượt quá số lần nhập OTP".
5. **Gọi validateOtp sau khi đạt giới hạn nhập sai:** Sau test case 4, gọi lại validateOtp với cùng transactionId. Kiểm tra xem có ném "OTP đã hết hạn hoặc không tồn tại" hay không.
6. **Lỗi khi userRepository.save ném exception:** Mock repository để ném lỗi và kiểm tra xem lỗi có bị xử lý không.
7. **Lỗi khi registerUserRepository.deleteById ném exception:** Mock repository để ném lỗi và kiểm tra xem lỗi có bị xử lý không.

**6. NormalizePhoneNumber Method**

**Phân tích giá trị biên (Boundary Value Analysis):**

* Kiểm tra các tiền tố "+84" và "0" với các số điện thoại có độ dài hợp lệ.
* Kiểm tra số điện thoại không có tiền tố.

**Phân lớp tương đương (Equivalence Partitioning):**

* Số điện thoại bắt đầu bằng "+84".
* Số điện thoại bắt đầu bằng "0".
* Số điện thoại không có tiền tố "+84" hoặc "0".
* Số điện thoại null hoặc rỗng.

**Kiểm thử lỗi (Error Guessing):**

* Số điện thoại có định dạng không chuẩn khác (ví dụ: "+840...", "00...").

**Test Cases:**

1. **Số điện thoại bắt đầu bằng "+84":** Kiểm tra xem "+84123..." được chuyển thành "84123...".
2. **Số điện thoại bắt đầu bằng "0":** Kiểm tra xem "0987..." được chuyển thành "84987...".
3. **Số điện thoại không có tiền tố:** Kiểm tra xem "1900..." vẫn là "1900...".
4. **Số điện thoại null:** Kiểm tra kết quả trả về (có thể là null hoặc xử lý khác tùy logic).
5. **Số điện thoại rỗng:** Kiểm tra kết quả trả về (có thể là chuỗi rỗng hoặc xử lý khác).
6. **Số điện thoại bắt đầu bằng "+840":** Kiểm tra xem có được xử lý đúng không.
7. **Số điện thoại có ký tự không phải số (ở phần sau tiền tố):** Kiểm tra xem có gây ra lỗi không (phương thức này có vẻ chỉ xử lý tiền tố).