UserChecksExtendedTest Documentation

UserChecksExtendedTest Documentation

Overview:

This document summarizes unit test cases implemented in UserChecksExtendedTest for validating user input handling.

These tests are designed to catch weak or malformed data early in the system and return proper feedback to frontend clients.

Test Class Structure:

Dependencies:

- JUnit 5 for lifecycle and assertions
- Mockito for mocking user service interactions
- Spring ResponseEntity for HTTP-style validation responses

Mocked Components:

- UserRepository (mocked for future extensibility)
- UserService: Handles business logic like email availability

Test Setup:

- Mocks are initialized with MockitoAnnotations
- UserChecks class is injected with mocks

Test Cases:

1. testEmailChecks_EmptyEmail

Purpose: Ensure that submitting an empty email string is handled appropriately.

Scenario:

- An empty string is passed to emailChecks

Assert:

- Response body indicates invalid format

- Status code is 400 Bad Request

2. testPasswordChecks_TooShortPassword

Purpose: Validate that very short passwords are caught by security policy.

Scenario:

- A password shorter than minimum length ("P1!") is submitted

Assert:

- Error message contains the word "password"

- Status is 400 Bad Request

3. testPasswordChecks_MissingSpecialCharacter

Purpose: Enforce strong password rules including at least one special character.

Scenario:

- A password with letters and numbers but no symbols is submitted

Assert:

- Response contains feedback about password rules

- Status is 400 Bad Request

Conclusion:

These tests cover critical validation paths for form inputs before account creation or login.

They reduce error propagation by enforcing consistent input standards at the earliest validation layer.