Advanced API Testing and Automation Practices

5.1 Error Handling & Edge Case Testing

Learning objectives

- Understand why error-handling tests are critical
- Learn how Spring Boot handles errors by default
- Write Rest Assured tests for bad requests & invalid inputs
- Test negative paths (404, 400, 409, validation errors)
- Ensure your API behaves predictably under failure

Why Test Errors?

- Most bugs happen at the edges
- Users send invalid or unexpected data
- APIs must fail gracefully, not crash
- Error tests ensure stability, security, and clarity
- Helps catch missing validation or incorrect exception handling

Types of Error Scenarios to Test

- 1. Not Found (404) - Resource does not exist
- 2. Bad Request (400) - Invalid, malformed, or missing input
- 3. Conflict (409) - Duplicate entity or constraint violation
- 4. Server Error (500) - Unexpected failures (ensure not leaked to user)
- 5. Validation errors - Empty fields, invalid values, negative numbers

The Importance of Negative Testing

- Positive tests answer: “Does the API work when used correctly?”
- Negative tests answer: “Does the API handle mistakes correctly, WITHOUT crashing?”
- Examples:
  - Incorrect data types
  - Missing required fields
  - Invalid enum/category values
  - Wrong IDs
  - Violating business rules

Tips for Writing Good Negative Tests

- Only assert what is guaranteed
  (e.g., don’t assert timestamp values)
- Consider the API's intended behavior
- Think like a user making mistakes
- Use meaningful test names:
  - shouldReturn404WhenActivityNotFound()
  - shouldReturn400WhenActivityNameMissing()

Demo - Negative tests

- Activities
  - GET /persons/9999 → expect 404 Not Found
  - DELETE /persons/9999 → expect 404 Not Found
- Creating invalid activities
  - POST person with missing name → expect 400 Bad Request
  - POST person with negative age → expect 400 Bad Request

Demo - Validation

- 1. Add new dependency in pom.xml: spring-boot-starter-validation
- 2. Annotate the name in Person:
  - @NotBlank(message = "Name must not be blank")
- 3. Annotate the age in Person:
  - @Min(value = 0, message = "Age must be greater than or equal to 0")
- 4. Annotate the @RequestBody Person person with @Valid:
  - @Valid @RequestBody Person person

Summary

- Error handling tests are essential for production-ready APIs
- They ensure resilience, safety, and predictable behavior
- Rest Assured provides simple tools for negative-path testing
- This module prepares you for robust API validation in the final project