API Automation Framework

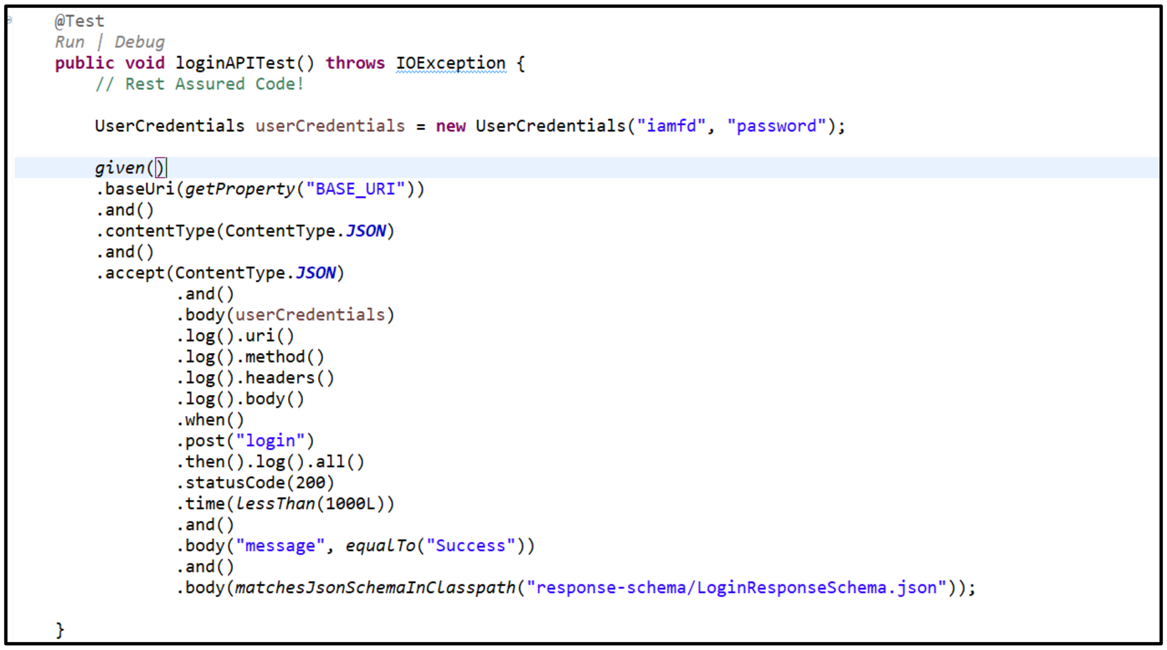
Requirement 6 From the Manager (Spec Builder)

Subject: Refactor the Rest Assured Code

Hi Team,

During our recent code review, we noticed a recurring issue across multiple API test classes. Many of the test scripts are repeating the same setup and validation code. For example:

- The BASE_URI is hardcoded in every test.
- contentType and accept headers are being declared again and again.
- Logging configuration (.log().uri(), .log().method(), etc.) is duplicated.
- Common checks like status code = 200, response time < 1000 ms, and content type = JSON are repeated in every test.

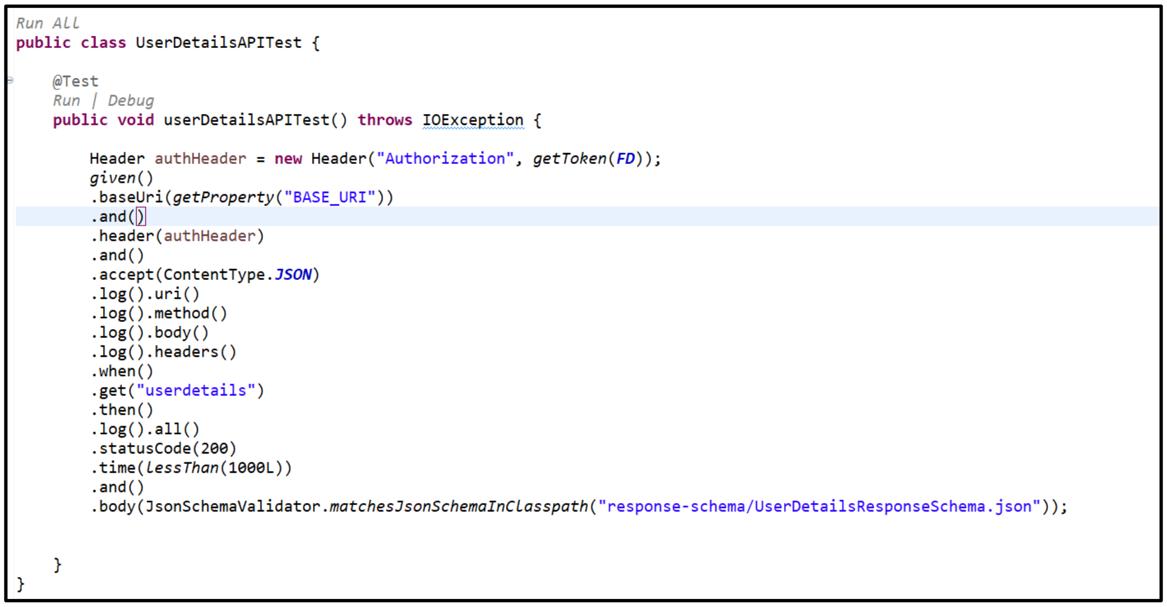


```java
@Test
Run | Debug
public void loginAPITest() throws IOException {
    // Rest Assured Code!

    UserCredentials userCredentials = new UserCredentials("iamfd", "password");

    given()
    .baseUri(getProperty("BASE_URI"))
    .and()
    .contentType(ContentType.JSON)
    .and()
    .accept(ContentType.JSON)
            .and()
            .body(userCredentials)
            .log().uri()
            .log().method()
            .log().headers()
            .log().body()
            .when()
            .post("login")
            .then().log().all()
            .statusCode(200)
            .time(lessThan(1000L))
            .and()
            .body("message", equalTo("Success"))
            .and()
            .body(matchesJsonSchemaInClasspath("response-schema/LoginResponseSchema.json"));

}
```



```
Run All
public class UserDetailsAPITest {

    @Test
    Run | Debug
    public void userDetailsAPITest() throws IOException {

        Header authHeader = new Header("Authorization", getToken(FD));
        given()
        .baseUri(getProperty("BASE_URI"))
        .and()
        .header(authHeader)
        .and()
        .accept(ContentType.JSON)
        .log().uri()
        .log().method()
        .log().body()
        .log().headers()
        .when()
        .get("userdetails")
        .then()
        .log().all()
        .statusCode(200)
        .time(lessThan(1000L))
        .and()
        .body(JsonSchemaValidator.matchesJsonSchemaInClasspath("response-schema/UserDetailsResponseSchema.json"));


    }
}
```

While these work correctly, they create unnecessary boilerplate. Every Tester who will write the test scripts needs to remember to use baseURI() and header and also the log().

If any of these settings need to change (for example, a new BASE_URI or a different response SLA), we will have to edit dozens of test methods individually.

This is not efficient and makes our framework harder to maintain.

To resolve this, we are introducing Requirement 6: ***Use RequestSpecBuilder and ResponseSpecBuilder.***

What needs to be done:

Create a new utility class SpecUtil.

Use RequestSpecBuilder to define all common request configurations in one place:

- ✓ baseUri (from ConfigManager.getProperty("BASE_URI"))
- ✓ contentType and accept as JSON
- ✓ logging for URI, method, headers, and body

Use ResponseSpecBuilder to standardize response expectations:

- ✓ Default status code (200)
- ✓ JSON content type
- ✓ Response time within SLA (e.g., less than 1000 ms)

Refactor existing test classes (LoginAPITest, UserDetailsAPITest, and others) to use these specs.

- Replace repeated setup with **`.spec(SpecUtil.requestSpec()).`**
- Replace repeated response checks with **`.spec(SpecUtil.responseSpecOk())`**
- Keep endpoint-specific validations (like JSON schema checks, or response body fields) inside the test itself.

Work on a new branch named feat/spec-builder, push your changes, and raise a Pull Request for review.

Benefits of this change:

- **Less boilerplate** → cleaner, shorter tests.
- **Consistency** → all tests follow the same logging and validation rules.
- **Easy maintenance** → if a global setting changes, we only update SpecUtil.
- **Scalability** → future APIs can be added with minimal setup.

Thanks,
Your Manager