API testing for the https://reqres.in/ API using RestAssured, Cucumber, Jenkins, and TestNG:

**Test Scenarios for ReqRes API**

Here are more scenarios you can implement:

**1. GET /api/users?page=2**

* **Scenario**: Retrieve users on page 2.
* **Expected Results**:
  + Status code should be 200.
  + The response should contain a list of users.
  + The page field in the response should be 2.

**2. GET /api/users/{id} with valid ID**

* **Scenario**: Retrieve a user by a valid ID (e.g., ID = 2).
* **Expected Results**:
  + Status code should be 200.
  + The response should contain the user details with the specified ID.
  + The data field should contain id, email, first\_name, last\_name, and avatar.

**3. GET /api/users/{id} with invalid ID**

* **Scenario**: Attempt to retrieve a user with an invalid ID (e.g., ID = 9999).
* **Expected Results**:
  + Status code should be 404.
  + The response body should be empty.

**4. POST /api/users**

* **Scenario**: Create a new user with valid data.
* **Expected Results**:
  + Status code should be 201.
  + The response should contain the user details including name, job, and an id.
  + The createdAt field should reflect the current timestamp.

**5. POST /api/users with missing fields**

* **Scenario**: Attempt to create a user without specifying mandatory fields (e.g., missing name).
* **Expected Results**:
  + Status code should be 400 or 422.
  + The response should indicate that the name field is missing.

**6. PUT /api/users/{id}**

* **Scenario**: Update an existing user's details.
* **Expected Results**:
  + Status code should be 200.
  + The response should reflect the updated name and job.
  + The updatedAt field should reflect the current timestamp.

**7. PUT /api/users/{id} with invalid ID**

* **Scenario**: Attempt to update a user with an invalid ID.
* **Expected Results**:
  + Status code should be 404.
  + The response should indicate that the user could not be found.

**8. DELETE /api/users/{id}**

* **Scenario**: Delete a user by ID.
* **Expected Results**:
  + Status code should be 204.
  + The response body should be empty.

**9. GET /api/unknown**

* **Scenario**: Retrieve a list of unknown resources.
* **Expected Results**:
  + Status code should be 200.
  + The response should contain a list of resources with fields like id, name, year, color, and pantone\_value.

**10. GET /api/unknown/{id}**

* **Scenario**: Retrieve a specific unknown resource by ID.
* **Expected Results**:
  + Status code should be 200.
  + The response should contain the details of the resource with the specified ID.

**11. POST /api/register**

* **Scenario**: Register a new user with valid email and password.
* **Expected Results**:
  + Status code should be 200.
  + The response should contain a token.

**12. POST /api/register with invalid data**

* **Scenario**: Attempt to register a user with missing or invalid email/password.
* **Expected Results**:
  + Status code should be 400.
  + The response should indicate that email or password is missing.

**13. POST /api/login**

* **Scenario**: Login a user with valid credentials.
* **Expected Results**:
  + Status code should be 200.
  + The response should contain a token.

**14. POST /api/login with invalid credentials**

* **Scenario**: Attempt to login a user with invalid credentials.
* **Expected Results**:
  + Status code should be 400.
  + The response should indicate that the email or password is incorrect.

**15. GET /api/unknown/{id} with invalid ID**

* **Scenario**: Attempt to retrieve an unknown resource with an invalid ID.
* **Expected Results**:
  + Status code should be 404.
  + The response body should be empty.

**16. GET /api/users?delay=3**

* **Scenario**: Test delayed response by fetching users with a delay of 3 seconds.
* **Expected Results**:
  + Status code should be 200.
  + The response should be delayed by approximately 3 seconds.
  + The response should contain a list of users.

**17. POST /api/users with large payload**

* **Scenario**: Create a new user with a large payload (e.g., long strings for name and job).
* **Expected Results**:
  + Status code should be 201.
  + The response should contain the user details and an id.
  + The createdAt field should reflect the current timestamp.

**18. PUT /api/users/{id} with partial data**

* **Scenario**: Attempt to update a user with only one field (e.g., update name only).
* **Expected Results**:
  + Status code should be 200.
  + The response should reflect the partial update.
  + The updatedAt field should reflect the current timestamp.

**19. DELETE /api/users/{id} with invalid ID**

* **Scenario**: Attempt to delete a user with an invalid ID.
* **Expected Results**:
  + Status code should be 404.
  + The response should indicate that the user could not be found.

**20. Comprehensive Health Check**

* **Scenario**: Perform a comprehensive health check by sending requests to all endpoints (/users, /register, /login, /unknown).
* **Expected Results**:
  + Status code should be 200 for all requests.
  + Each endpoint should return valid and correctly structured responses.
  + Validate that the API is functioning as expected.

**Execution and Reporting**

* After running tests, Jenkins will generate TestNG and Cucumber reports for review.
* Monitor Jenkins builds to ensure the tests are running smoothly.