Test Strategy

1. Objective:

The objective of this test strategy is to ensure robust testing coverage for the Campaign API, validate its functionality, and provide recommendations for a stable test environment that supports both manual and automated testing.

---

2. Testing Layers:

2.1 Unit Tests:

- Test individual components or functions of the API to ensure they perform as expected.

2.2 Integration Tests:

- Validate interactions between different modules of the API.
- Mock external services (like third-party APIs) where applicable to isolate the API functionality.

2.3 API Contract Tests:

- Ensure the API adheres to predefined contract and schema expectations.
- Verify that API responses are consistent with the schema and meet the defined expectations.
- **Framework**: Postman

2.4 End-to-End (E2E) Tests:

- Simulate real-world scenarios, covering user workflows and end-user interactions.
- Focus initially on critical endpoints such as POST /campaign and GET /campaign.
- **Framework**: Cypress

---

3. Environment Recommendations (for DevOps):

3.1 Stable Staging Environment:

- Use Docker to containerize the API and database for a consistent testing environment.
- Implement CI/CD pipelines for automated deployment and testing (e.g., using GitHub Actions).

3.2 Mocking External Dependencies:

- Use tools like Wire Mock to mock external services, ensuring consistent responses during testing.

---

4. Documenting and executing the API Tests :

4.1 Tool Selection:

- **Postman** will be used for detailed API documentation, making it easier for developers and product owners to interact with the API.

4.2 Implementation Steps:

1. Overview of API Testing Approach:

- Testing the following CRUD operations on the Campaign API:
    - **Create Campaign** (POST)
    - **Get Campaign** (GET)
    - **Update Campaign** (PUT)
    - **Delete Campaign** (DELETE)
- These tests will be implemented using Postman and automation tools to ensure reliable and repeatable testing.

2. Pre-requisites for Testing:

- Postman should be installed and configured.
- Set up environment variables in Postman:
    - baseUrl: API base URL (e.g., http://localhost:8081)
    - campaignId: Dynamically created campaign ID
    - campaignData: JSON object with campaign details for creating a campaign.
    - updatedCampaignData: JSON object for updating the campaign.

5. Postman Request and Test Details:

5.1 Create Campaign (POST Request):

- **Method**: POST
- **Endpoint**: {{baseUrl}}/campaign
- **Test Implementation**:
  - Verify response status code is 201 (Created).
  - Verify the response contains a campaign ID.
  - Verify response body matches the input data (e.g., name, client, category).

5.2 Get Campaign (GET Request):

- **Method**: GET
- **Endpoint**: {{baseUrl}}/campaign/{{campaignId}}
- **Test Implementation**:
  - Verify response status code is 200 (OK).
  - Verify the campaign name matches the original name passed in the creation request.

5.3 Update Campaign (PUT Request):

- **Method**: PUT
- **Endpoint**: {{baseUrl}}/campaign/{{campaignId}}
- **Request Body**: Use environment variables for passing updated campaign data.
- **Test Implementation**:
  - Verify response status code is 200 (OK).
  - Verify the updated campaign name is as expected.

5.4 Delete Campaign (DELETE Request):

- **Method**: DELETE
- **Endpoint**: {{baseUrl}}/campaign/{{campaignId}}
- **Test Implementation**:
  - Verify response status code is 200 (OK) indicating successful deletion.

5.5 Verify Campaign Deletion (GET Request):

- **Method**: GET
- **Endpoint**: {{baseUrl}}/campaign/{{campaignId}}
- **Test Implementation**:
  - Verify response status code is 404 (Not Found) after deletion.

---

6. Test Automation Execution:

6.1 Manual Execution:

- Tests can be executed manually by running the collection directly from Postman.

6.2 Automated Execution:

- Use **Newman** (a command-line collection runner for Postman) to run the Postman collection in an automated manner.
  - **Example command**:

```
newman run campaign-api-collection.json -e environment.json
```

---

7. Continuous Integration (CI) Integration:

7.1 CI Pipeline Integration:

- Install **Newman** to run the tests in the CI pipeline (e.g., using Jenkins, GitLab CI, or GitHub Actions).
  - **Example Jenkins command**:

```
node --max_old_space_size=8192 $(npm bin)/newman run campaign-api-collection.json -e environment.json
```

---

8. Reporting and Logs:

- Generate detailed logs during test execution using **Newman**.
    - **Example command to generate an HTML report**:

```
newman run campaign-api-collection.json -e environment.json -r html
```

- The output report provides insights into the status of each request, enabling easier troubleshooting.

---

9. Additional Considerations:

9.1 Data Cleanup:

- Ensure any created test data is cleaned up after each run to avoid conflicts with subsequent tests.

9.2 Environment and Configuration:

- Ensure the correct API environment (e.g., staging or production) and configurations are set before running tests.

9.3 Error Handling:

- Implement robust error handling in Postman scripts to ensure graceful test failures when unexpected conditions occur.

---

10. Implementing Cypress Automated Tests:

10.1 Framework: Cypress

- Cypress is ideal for API testing, combining simplicity and power.
    - Test for edge cases (e.g., missing or invalid fields).
    - Framework for automating the tests for the POST /campaign endpoint.

10.2 Test Steps:

- **POST /campaign**:
    - Make a POST request to /campaign with campaign data.
    - Verify the response status code is 201 (Created).
    - Ensure the response body contains the campaign details and that the id field matches the generated ID.
    - Ensure the response body contains the correct name, client, category, and status.
- **GET /campaign/{id}**:
    - Description: This endpoint retrieves a campaign by its ID.
    - Verify response status code is 200 (OK).
    - Validate the response schema and ensure the returned campaign has the expected details.
- **PUT /campaign/{id}**:
    - Description: This endpoint updates an existing campaign by its ID.
    - Verify response status code is 200 (OK).
    - Ensure the updated campaign data is as expected.
- **DELETE /campaign/{id}**:
    - Description: This endpoint deletes a campaign by its ID.
    - Verify response status code is 200 (OK).
- **GET /campaign/{id} (after deletion)**:
    - Ensure that after deletion, a 404 status is returned, indicating the campaign was deleted.
- **GET /campaign (for duplicates)**:
    - Verify no duplicate ids exist in the response list.
- **POST /campaign (negative case)**:
    - Test the creation of a campaign with missing required fields.
    - Verify that the response status code is 400 (Bad Request), and the appropriate error message is returned.

10.3 Error Handling:

- Simulate invalid input (e.g., missing required fields) and ensure the correct error status codes (e.g., 400 Bad Request) are returned.

10.4 Test Automation Execution:

- Use npx cypress run for headless test execution.
- Integrate tests into the CI pipeline to ensure consistent testing after every code change.

11. DevOps Recommendations for API Test Automation:

11.1 Test Environment Management:

- Set up isolated and consistent test environments that mimic production.
- Use environment variables for configuration (e.g., API keys, base URLs).

11.2 CI/CD Integration:

- Integrate automated tests into the CI/CD pipeline.
- Use parallel test execution to reduce test runtime.

11.3 Test Reporting and Monitoring:

- Generate detailed reports with logs, screenshots, and videos for failures.
- Set up alerts for test failures to notify relevant stakeholders.

11.4 Version Control and Test Data Management:

- Maintain versioning for test scripts, ensuring synchronization with application changes.
- Use mock or test-specific data for tests.

11.5 Security and Compliance:

- Avoid hardcoding sensitive data in scripts; use secure storage (e.g., Vault, AWS Secrets Manager).
- Implement security scans to identify vulnerabilities.

11.6 Test Automation Maintenance:

- Regularly review and update test cases to reflect changes in the API.
- Clean up test data to prevent conflicts during subsequent runs.

11.7 Collaboration and Feedback Loop:

- Foster communication between DevOps, Development, and QA teams.
- Provide quick feedback to developers to address issues promptly.

12. Conclusion:

This strategy focuses on validating the functionality of the **different campaign** endpoint, ensuring that the API behaves as expected. By using **Postman**, **Cypress**, and continuous integration tools, we can automate testing, streamline the development workflow, and maintain robust API quality standards.