General Structure of a Katalon Project for API Automation

1. **Test Cases (Test Cases/)**
   Description: Contains all the scripts that define test scenarios for your API testing.
   Contents:
   Each test case involves:
   - Sending API requests (sendRequest)
   - Verifying the HTTP response status code (verifyStatusCode)
   - Verifying specific values in the JSON response (verifyElementPropertyValue, assert JSON)
   - Validating the API response against a predefined JSON Schema (validate JSON Schema)

   Example Folder Structure:
   Test Cases/Forecast_API → Tests weather forecast API endpoints
   Test Cases/Air_Pollution → Tests air pollution API endpoints

2. **Object Repository (Object Repository/)**
   Description: Contains all saved Test Objects, which represent the API requests (endpoint + request setup).
   Contents:
   Each Test Object defines:
   - Endpoint URL (e.g., https://api.weather.com/forecast)
   - HTTP Method (GET, POST, PUT, DELETE, etc.)
   - Headers (Authorization, Content-Type, etc.)
   - Body (for methods like POST or PUT)
   - Query Parameters (optional filters)

   Example Folder Structure:
   Object Repository/get_details
   Object Repository/pollution_api
   Object Repository/weather_api

3. **Include/scripts/groovy/schemas/**
   Description: Contains additional supporting files, especially JSON Schema files, used for validating the structure of API responses.
   Contents:
   - JSON Schema Files for each type of expected response.
   - Validation uses these schemas to automatically check if fields, data types, and structures are correct.

   Example Files:
   Include/scripts/groovy/schemas/weather.schema.json
   Include/scripts/groovy/schemas/pollution.schema.json

4. **Profiles (Profiles/)**
   Description: Manages Global Variables like base URLs, API keys, or environment-specific settings (Dev, QA, Production), and in this project, Global Variables are not used.
   Contents:
   - Different Profiles for different environments (e.g., DevProfile, QAProfile, ProdProfile).
   - Variables like baseURL, apiKey, etc.

5. **Reports (Reports/)**
   Description: Auto-generated reports from each Test Suite execution. Provides complete visibility of testing status. You can export reports to PDF, HTML, CSV. And report generate with date timestamp.

   Contents:
   - Test execution results (Pass, Fail)
   - Error details if failed
   - Execution duration per test
   - Screenshots (optional, if API testing is combined with UI)

   Example Files :
   Reports/20250428_235444 → Result from running Test Case – Forecast_Weather
   Reports/20250428_235954 → Result from running Test Suites – Testing_API that contains 2 test case.

6. **Test Suites (Test Suites/)**
   Description: Organized groups of Test Cases bundled together to run as one package.
   Contents:
   Each Test Suite may include:
   - Multiple Test Cases.
   - Execution order setting (can be parallel or sequential).
   - Optional test data binding if using data-driven approach.

   Example Folder Structure:
   Test Suites/Testing_API → Executes all API Test Cases together.

7. **Data Files (Data Files/)**
   Description: Provides external data sources for data-driven testing, allowing you to run the same Test Case with multiple input data, and in this project, Data Files are not used.

Contents:
- CSV files
- Excel files (.xls or .xlsx)
- Internal test data tables

8. **Test Listeners**
   Description: Scripts that automatically run before or after Test Cases and Test Suites, and in this project, Test Listeners are not used.

   Contents:
   Examples of what you can automate with Test Listeners:
   - Clear console/logs before running tests
   - Set up authentication tokens dynamically
   - Clean up test data after execution
   - Retry failed tests automatically