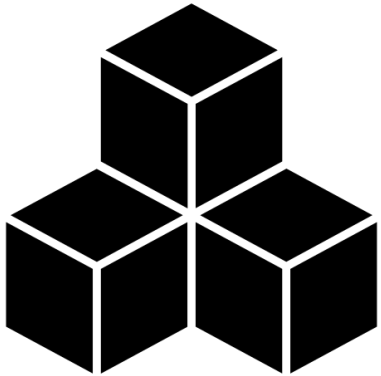
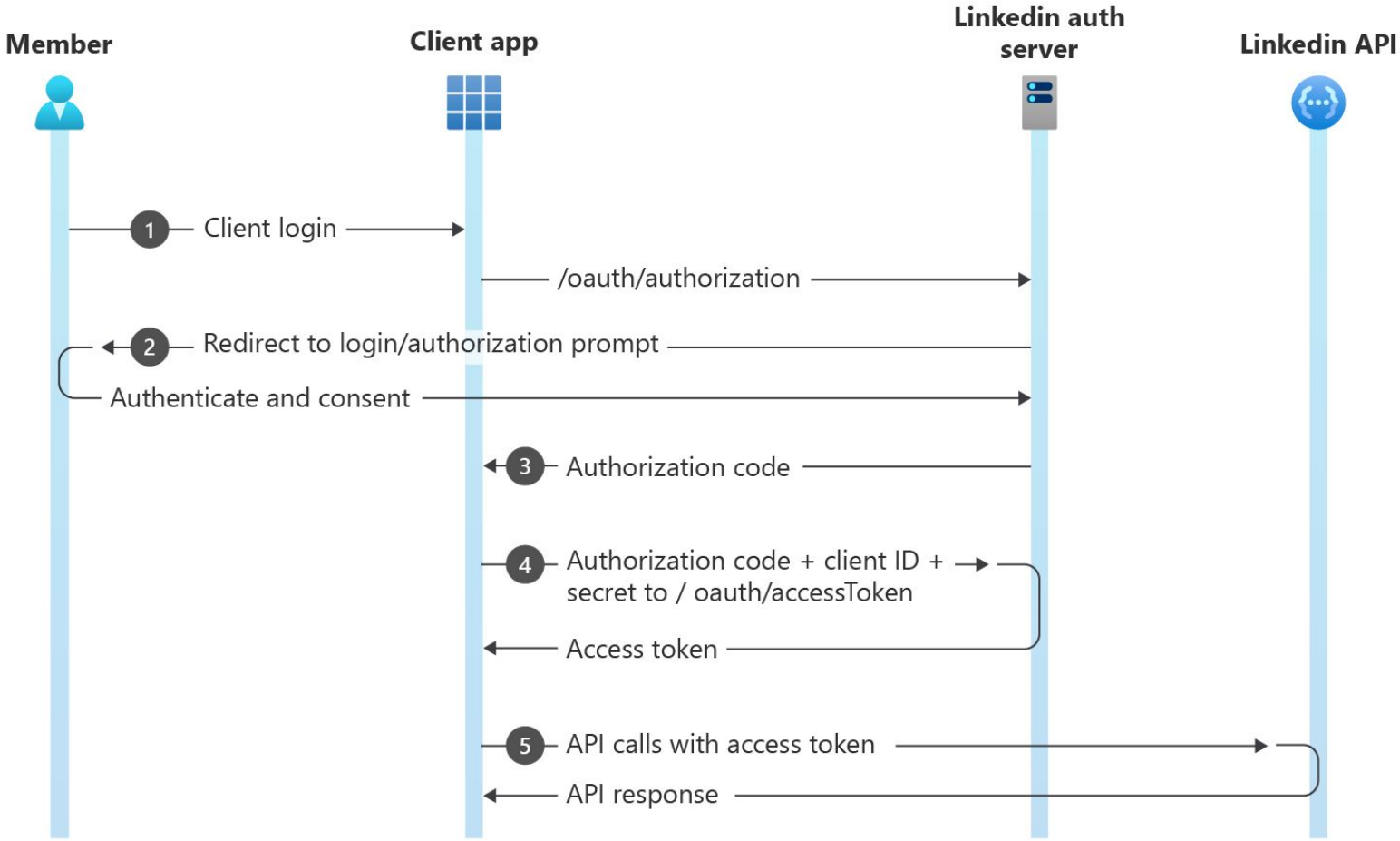


Hands on Workshop

On PostMan


Member
Client app
Linkedin auth server
Linkedin API
1
Client login
/oauth/authorization
2
Redirect to login/authorization prompt
Authenticate and consent
3
Authorization code
4
Authorization code + client ID + secret to / oauth/accessToken
Access token
5
API calls with access token
API response

Let’s explore the Authorization Code Flow

Rasel Hossain

Software Developer at Stack Learner

We Solved These Two Problems

- **Validation and Testing Gaps:** There are no solutions in place to validate or test their API responses.
- **Environment Mix-up:** They cannot easily manage their APIs in different environments (e.g., development, testing, production).

Focus on The Following Problems

- **Documentation Deficiency:** Their APIs lack proper documentation. For instance, details on how their partners should call API endpoints from different clients, the necessary payload, headers required for sending requests, etc., are missing.
- **Validation and Testing Gaps:** There are no solutions in place to validate or test their API responses.
- **Collaboration Challenges:** There is a problem with collaboration and easily sharing APIs across the development team.
- **Environment Mix-up:** They cannot easily manage their APIs in different environments (e.g., development, testing, production).

- **Performance Analysis Dilemma:** They lack an easy solution to analyze and identify performance issues, errors, or any unexpected behavior in their APIs, hampering improvements.
- **Frontend-Backend Coordination Issues:** They regularly observe that their frontend teams sometimes wait for the backend team to complete APIs, causing delays in completing projects on time.
- **Unclear authorization process:** Toto Company uses OAuth2 and OpenID Connect to secure their APIs. However, their backend team often receives questions from the frontend team, such as "Frontend team is unable to authorize and obtain a token from the authorization server." This is common for the frontend team because the backend team cannot demonstrate simple steps to work with their Auth server.

- **Data-Driven Testing Hurdles:** Sometimes they want to perform Data-Driven testing.
- **Automated Testing Goals:** They also aim to automate their API testing process, generating reports automatically.
- **API Data Flows:** Their backend team desires a system where they can directly manipulate APIs and observe the data flowing between them without creating a new application. This helps them create a mind map of their API functionality.

https://www.wallpaperflare.com/halloween-scary-pacific-forests-dark-nature-wallpaper-tjvvr/download

John & Michel

Developer of ToTo company

John: I know this forest like the back of my hand


SMOKEYBEAR.COM
SMOKEY
ONLY YOU
SMOKEY

https://www.wallpaperflare.com/halloween-scary-pacific-forests-dark-nature-wallpaper-tjvvr/download

We are back

John and Michel

Documentation

1. **Clear Instructions are Important:** They learned that documentation should be easy to understand, like the map they had. Confusing instructions can lead to mistakes.

2. **Documentation Must be Trustworthy:** They realized they needed reliable documentation they could trust, especially when things got tough.

3. **Documentation Helps in Confusion:** Documentation acts like a guide when things are uncertain, giving them direction when they felt lost.

4. **Avoiding Mistakes:** Just like the map helped them avoid wrong turns, good documentation can prevent errors and confusion in their work.

Let’s create API Documentation

OpenAPI Specification

- **Standardization:** OAS provides a standardized way to describe RESTful APIs. It offers a common language for developers, API providers, and consumers, tester, making it easier to understand, collaborate, and integrate APIs across different platforms and programming languages.
- **Documentation:** OAS serves as a comprehensive documentation for APIs. It describes endpoints, parameters, request/response payloads, authentication methods, and more in a structured format. This documentation helps developers understand how to interact with the API effectively.
- **Server Code Generation:** Similarly, OAS can be used to generate server-side code. By providing a detailed specification of the API, developers can automatically generate server implementations, reducing the manual effort required for coding. **#OpenAPI-Generator**

- **Testing and Validation:** OAS definitions can be used to validate API requests and responses. Testing tools can compare actual API behavior against the documented specification, ensuring that the API behaves as expected and adheres to the defined contract.
- **Tooling Ecosystem:** OAS has a vibrant ecosystem of tools and utilities built around it. There are editors for authoring OAS documents, validators for checking compliance with the specification, code generators for client/server implementations, testing frameworks, and more.

Let’s write OpenAPI Spec

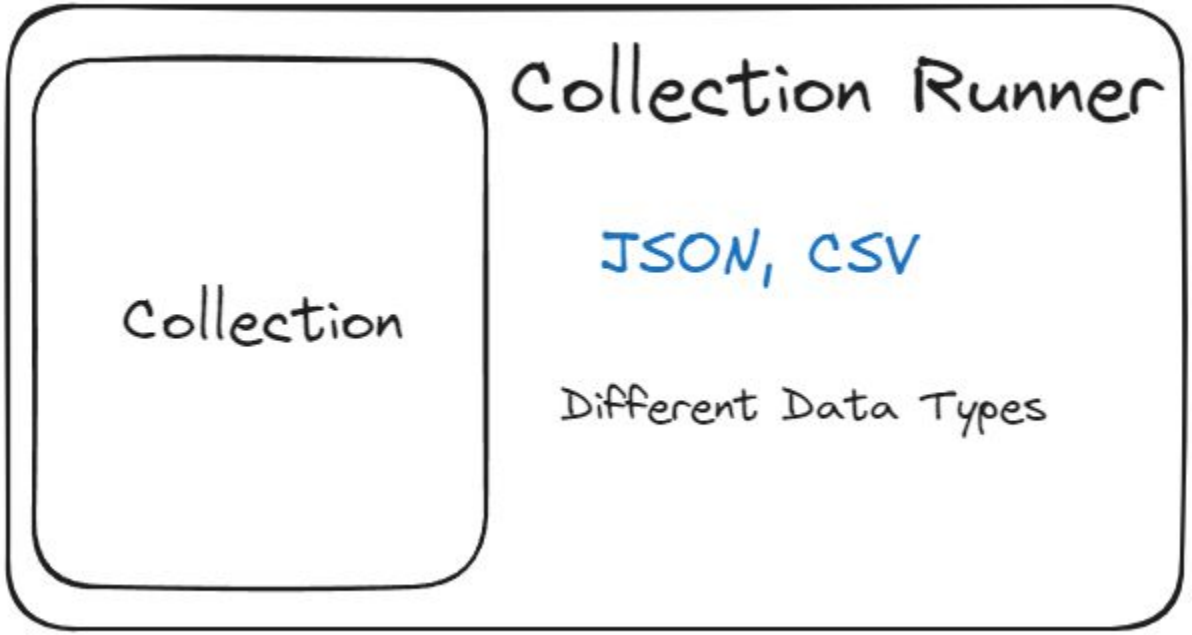
Authorization Code Flow

The Authorization Code Flow is an OAuth 2.0 authentication flow used by applications to obtain authorization to access resources on behalf of a user. It is commonly used in web applications where the client-side code runs in a web browser.

The Authorization Code Flow provides a secure and efficient way for applications to obtain access to protected resources on behalf of users without exposing sensitive credentials. It ensures that only authorized applications can access user data and that users have control over which resources the application can access.

Data Driven Testing

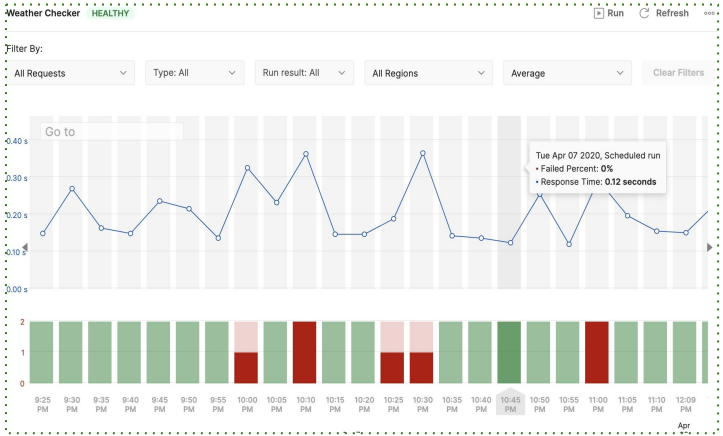
Data-driven testing in Postman is a technique where you use external data sources, such as CSV files, JSON files to drive your API tests. Instead of hardcoding test data directly into your requests, you dynamically replace variables in your requests with values from your data source during test execution. This allows you to run the same test scenario with different sets of data, making your tests more comprehensive and reusable.


Collection Runner
Collection
JSON, CSV
Different Data Types

Let’s explore Data Driven Testing

Monitoring

In Postman, monitoring refers to the process of continuously monitoring APIs for performance, availability, and functionality. Postman Monitoring allows you to schedule and run API tests at regular intervals from multiple global locations. This helps you ensure that your APIs are functioning as expected and meeting performance requirements.



Monitoring (Key Features)

Scheduled Tests

You can schedule API tests to run at specific intervals, such as every minute, hourly, daily, etc. This allows you to monitor your APIs continuously and detect any issues or performance fall over time.

Global Locations

Postman provides a network of global monitoring locations from which you can run your tests. This allows you to simulate requests from different geographic regions and ensure that your APIs perform well for users worldwide.

Performance Metrics

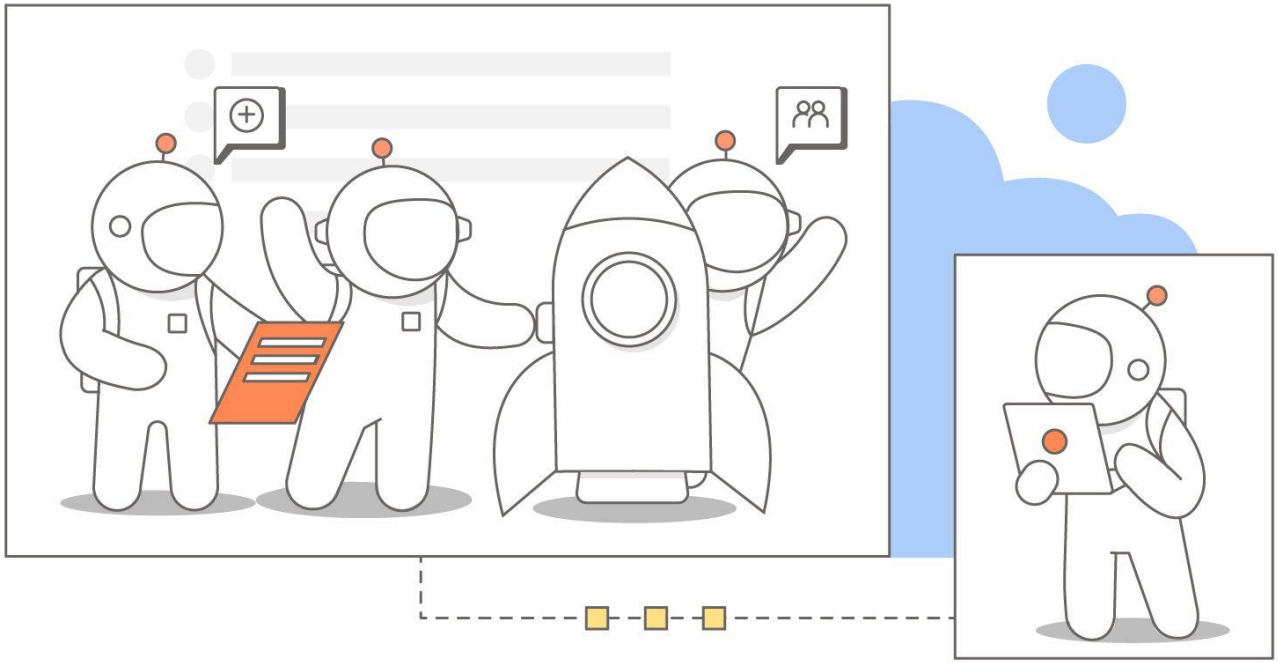
Postman Monitoring collects performance metrics such as response time, latency, and uptime for your APIs. You can view these metrics in dashboards and reports to analyze trends and identify areas for improvement.

Alerting

You can set up alerts. This allows you to receive notifications via email, Slack, or other channels when an API test fails or when performance metrics exceed specified limits.

Let’s Monitor Our API

Collaboration



Today we have solved

- **Data-Driven Testing Hurdles:** Sometimes they want to perform Data-Driven tasks.
- **Unclear authorization process:** Toto Company uses OAuth2 and OpenID Connect to secure their APIs. However, their backend team often receives questions from the frontend team, such as "Frontend team is unable to authorize and obtain a token from the authorization server." This is common for the frontend team because the backend team cannot demonstrate simple steps to work with their Auth server.

- **Documentation Deficiency:** Their APIs lack proper documentation. For instance, details on how their partners should call API endpoints from different clients, the necessary payload, headers required for sending requests, etc., are missing.
- **Performance Analysis Dilemma:** They lack an easy solution to analyze and identify performance issues, errors, or any unexpected behavior in their APIs, hampering improvements.
- **Collaboration Challenges:** There is a problem with collaboration and easily sharing APIs across the development team.

Thank You