Introduction to API Testing

Agenda

In this session, we will discuss:

- Introduction to API
- Introduction to API Testing
- Tools and Environment Setup
- Authentication and Authorization
- Testing methods
- API Testing Approach and its Test Cases
- Automation Frameworks
- How to Test API
- Data Validation and performance testing

Agenda

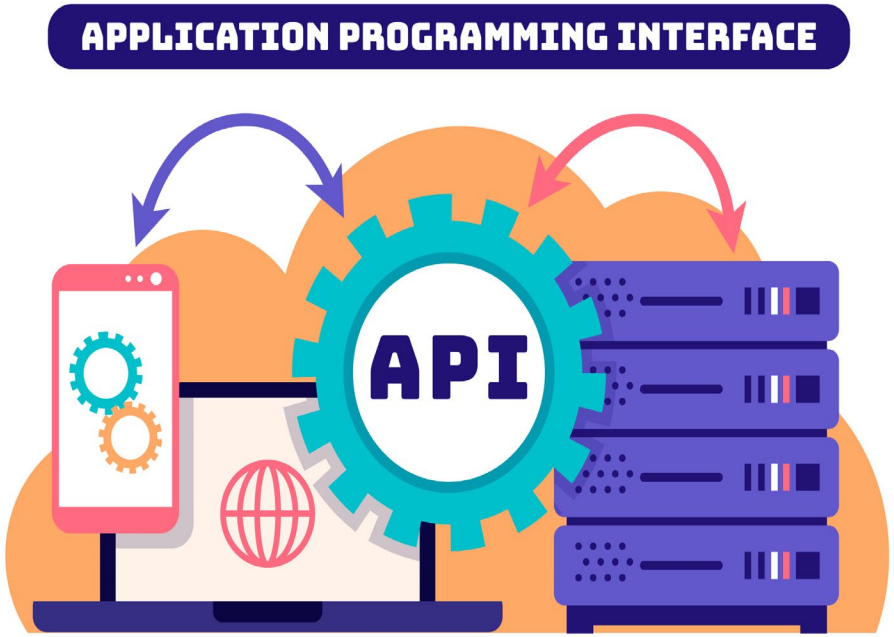
In this session, we will discuss:

- Error Handling and Security Testing
- Documentation and Reporting
- Best Practices of API Testing
- Difference between API Testing and Unit Testing
- Challenges and Pitfalls
- Ethical and Legal Considerations
- Future Trends and Emerging Technologies

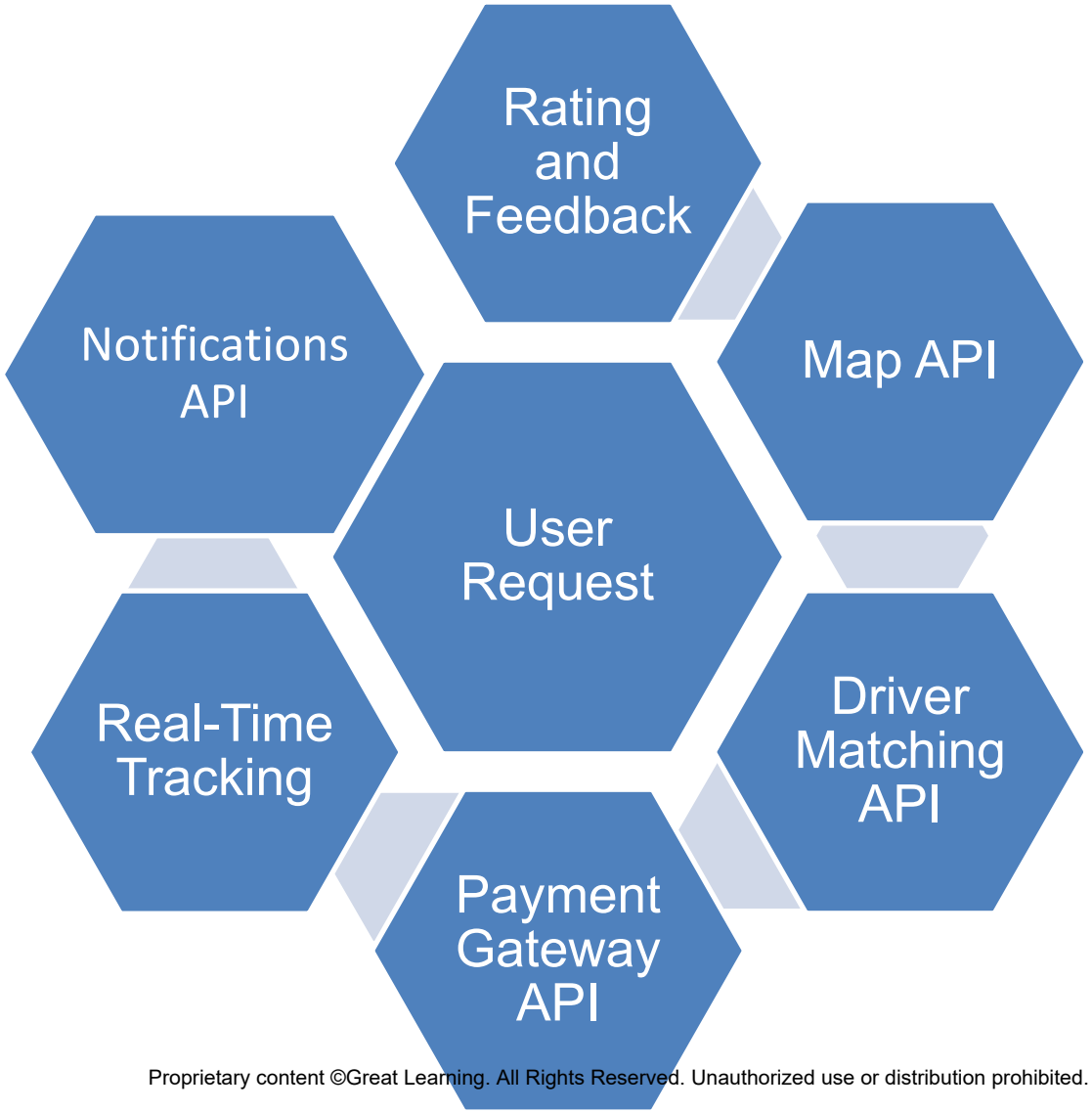
Introduction to API

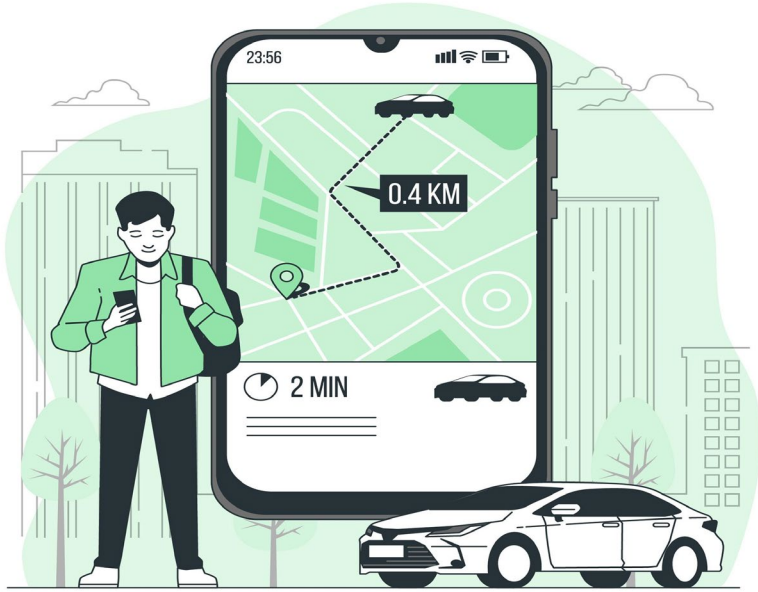
Introduction to API

- APIs are tools that facilitate communication between two software components by providing a predefined set of definitions and protocols.
- APIs provide a set of rules, protocols, and tools that developers can use to access the functionality and data of another software system, service, or platform.
- APIs helps in structuring the requests and responses, making it possible for developers to integrate various services and functionalities into their own applications without having to build everything from scratch.



Working of API

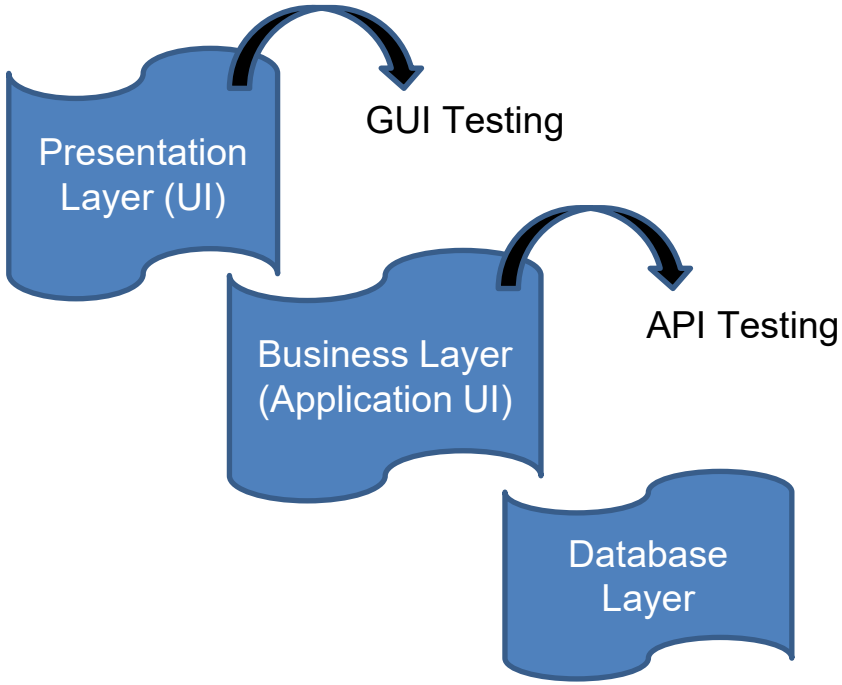




Introduction to API Testing

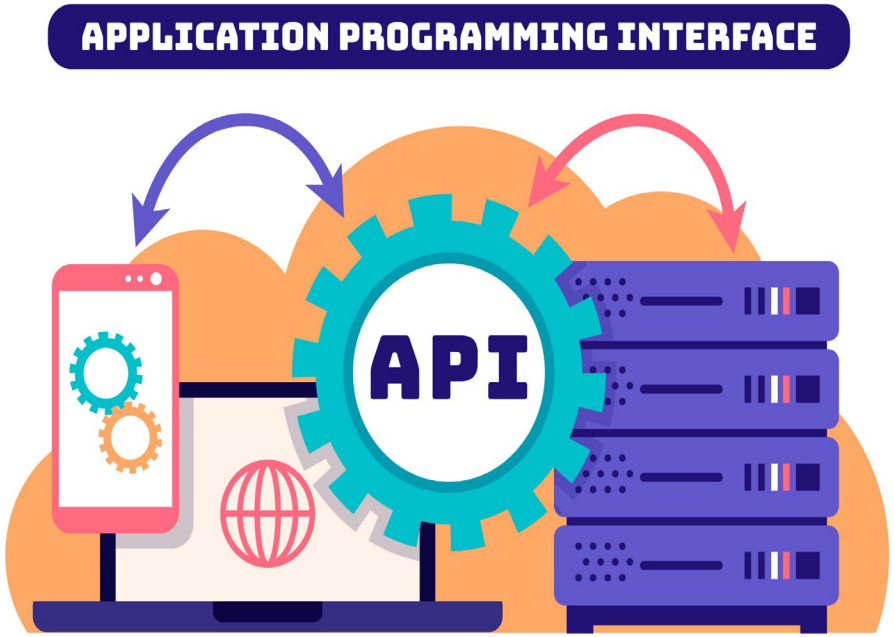
Introduction to API Testing

- It is a software that validates the Application Programming Interfaces(APIs).
- The main use of API testing is to check for functionality, reliability, performance, and security of the programming interfaces.
- Instead of standard inputs and outputs, a software can be used to send calls to the API, get output, and note down the system's response.
- A model-driven development approach is also used to map out how the software system should work before the actual coding begins.



RESTful API

- REST is an architectural style and a set of constraints for designing networked applications.
- In RESTful APIs each request from a client to a server must contain all the information needed to understand and process the request.
- In REST, the resources are identified by unique URLs (Uniform Resource Locators).
- REST uses standard HTTP methods (GET, POST, PUT, DELETE, etc.) to perform actions on resources.
- RESTful APIs are relatively simple to understand and use.

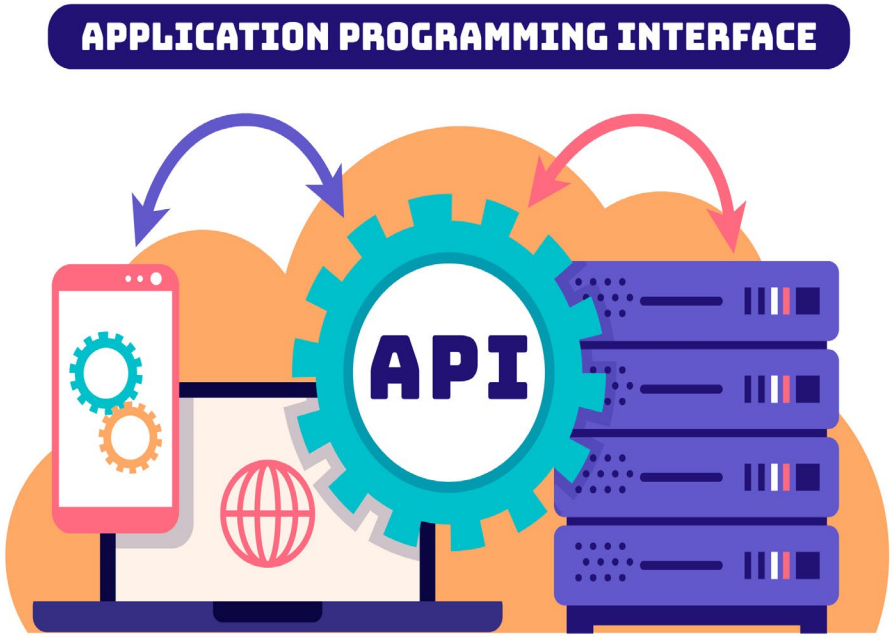


RESTful API Testing

- Testing a REST API is an essential part of software development.

Best Practices for testing a REST API:

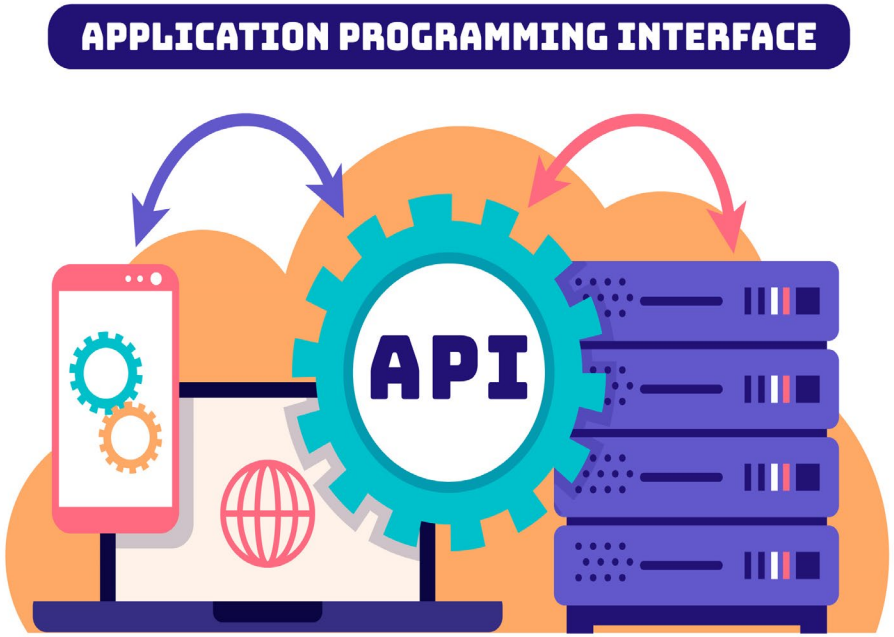
- Understanding the API
- Set Up the Testing Environment
- Testing Types
- API Authentication



RESTful API Testing

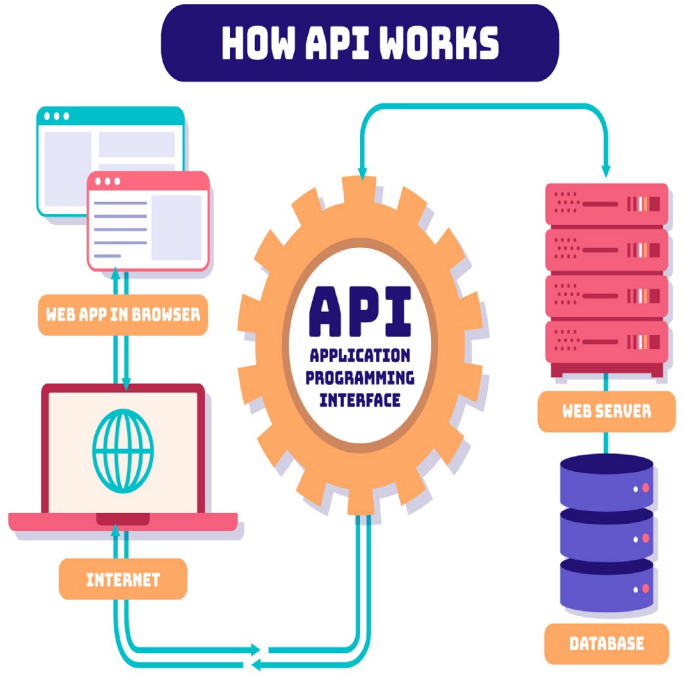
Best Practices for testing a REST API:

- API Testing Techniques
- Data Validation
- Error Handling and Performance Testing
- Continuous Testing and Security Testing
- Documentation and Reporting



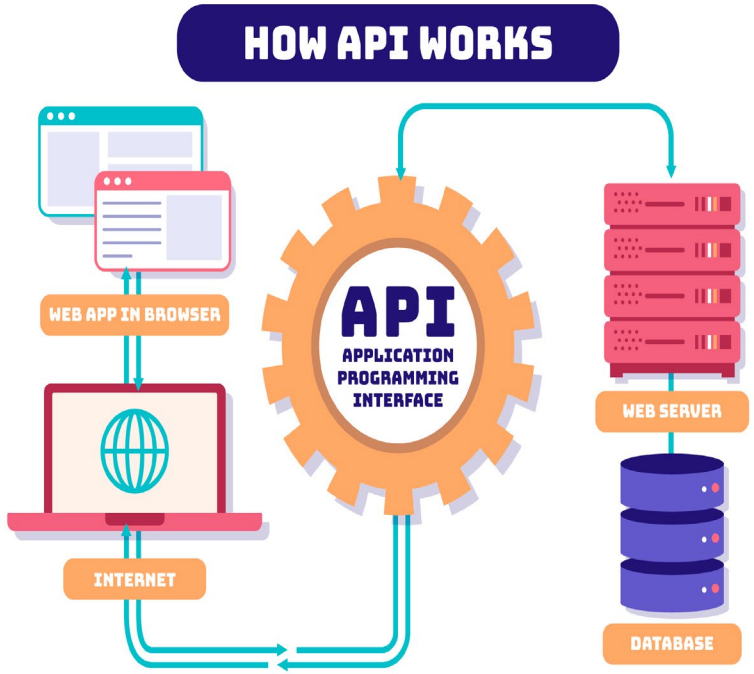
SOAP API

- SOAP (Simple Object Access Protocol) is a protocol for exchanging structured information in the implementation of web services.
- A SOAP API (Application Programming Interface) is an interface that allows applications to communicate with each other using the SOAP protocol.
- SOAP is used for sending messages between applications over various protocols, including HTTP, SMTP, and more.
- SOAP provides mechanisms for security, including encryption and authentication, which can be essential in enterprise-level applications.



SOAP API Testing

- Testing a SOAP API is essential to ensure the correct functioning of a SOAP-based web service.
- SOAP is a protocol for exchanging structured information in the implementation of web services.



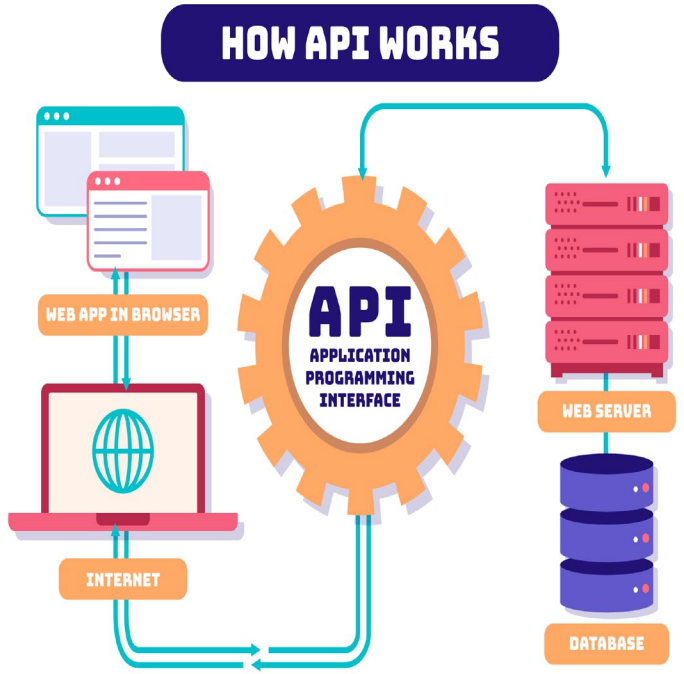
Best Practices for testing a SOAP API:

- Understanding the API
- Set Up the Testing Environment
- Testing Types
- Authentication

SOAP API Testing

Best Practices for testing a SOAP API:

- SOAP Request Structure
- SOAP Response Structure
- Data Validation and Error Handling
- Continuous Testing and Security Testing
- Documentation and Reporting



Difference Between SOAP and REST

Feature	SOAP	REST
Full Form	Simple Object Access Protocol	Representational State Transfer
Purpose	SOAP is a protocol for communication between applications	REST is an architecture style for designing communication interfaces.
Design	SOAP API exposes the operation.	REST API exposes the data
Protocol	SOAP is independent and can work with any transport protocol.	REST works only with HTTPS

Difference Between SOAP and REST

Feature	SOAP	REST
Data Format	Transports data in standard XML format.	REST supports XML, JSON, plain text, HTML
Performance	SOAP messages are larger, which makes communication slower.	REST has faster performance due to smaller messages and caching support.
Scalability	SOAP is difficult to scale	REST is easy to scale
Security	SOAP supports encryption with additional overheads.	REST supports encryption without affecting performance.

Tools and Environment Setup

Tools and Environment Setup

Setting up the testing environment for API testing involves preparing the necessary tools, configurations, and resources to facilitate efficient testing.

1. Choosing an API Testing Tool:

- **Postman**: Used to create and send API requests, manage collections of requests, and automate testing.
- **Insomnia**: Similar features to Postman, but its an alternative for those who prefer different interfaces.
- **cURL:** A command-line tool for making HTTP requests. It helps to work with APIs via the terminal.

Tools and Environment Setup

2. Installation and Configuration:

- Depending on the chosen tool**,** the local machines are installed and configured.
- This helps in setting up environment variables, authentication credentials, and other preferences.

3. API Access and Authorization:

- When working with real-world APIs, the API access credentials (e.g., API keys, tokens, OAuth) are obtained from the API provider.

Tools and Environment Setup

4. Test Environment Setup:

Test environment setup focuses on configuring the chosen testing tool to work with the specific API they are testing:

- **Defining the API's base URL:** The root URL where API endpoints are located.
- **Configuring headers:** Setting up request headers, including authentication tokens, content types (e.g., JSON, XML), and any custom headers required by the API.
- **Variables and data:** Setting up variables for dynamic data, like timestamps or user IDs, to be used in test cases.
- **Environment management:** Managing different environments (e.g., development, staging, production) to facilitate easy switching between API instances.

Tools and Environment Setup

5. Version Control:

- Git is used to track changes in API test scripts, collaborate with team members, and maintain a history of script versions.

6. Test Data Management:

- Data used in API tests should always be available and consistent.
- It includes creating test datasets, using databases, or employing data generation tools

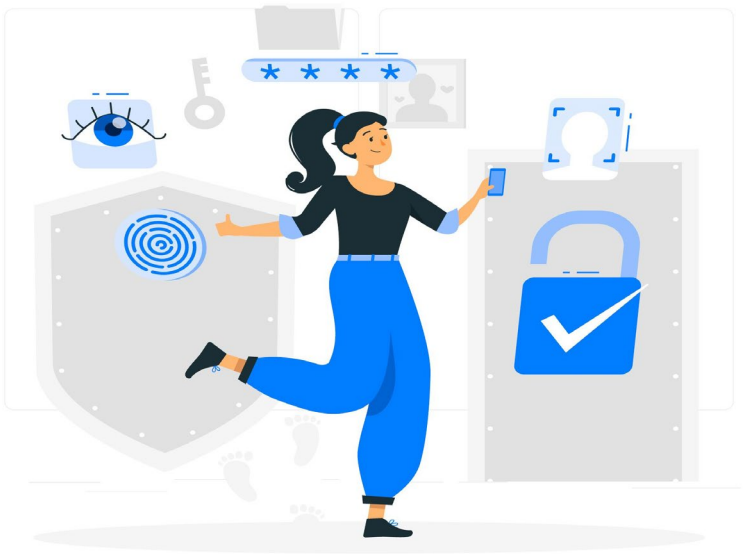
7. Collaboration and Teamwork:

- API test collections, collaborating with team members, and integrating API testing with project management and issue tracking tools like Jira, GitLab, or Trello should be known.

Authentication and Authorization

Authentication

Authentication is the process of verifying the identity of a user or application trying to access an API.

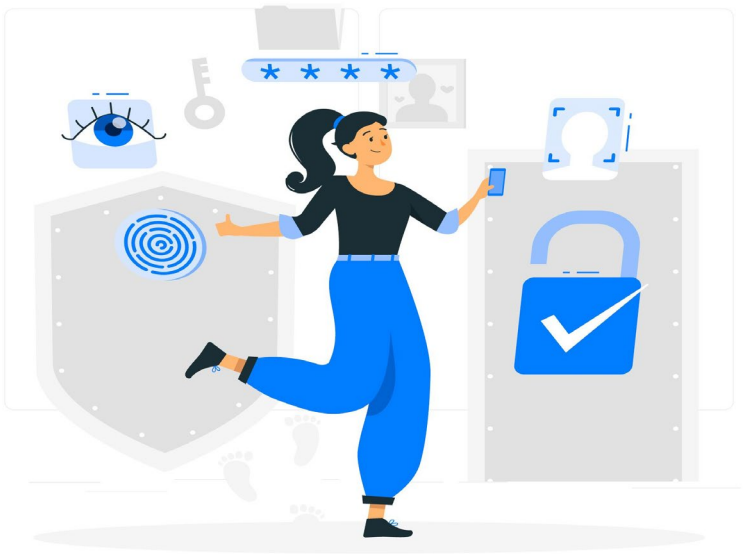


Authentication Methods:

- **API Keys:** A simple and common authentication method where a unique key is provided to the client. The key is included in the request header to authenticate the client.
- **OAuth (OAuth 2.0):** A more secure and flexible method for authentication. It allows applications to access APIs on behalf of a user without exposing the user's credentials.

Authorization

Authorization determines what actions a user or application is allowed to perform once they've been authenticated.



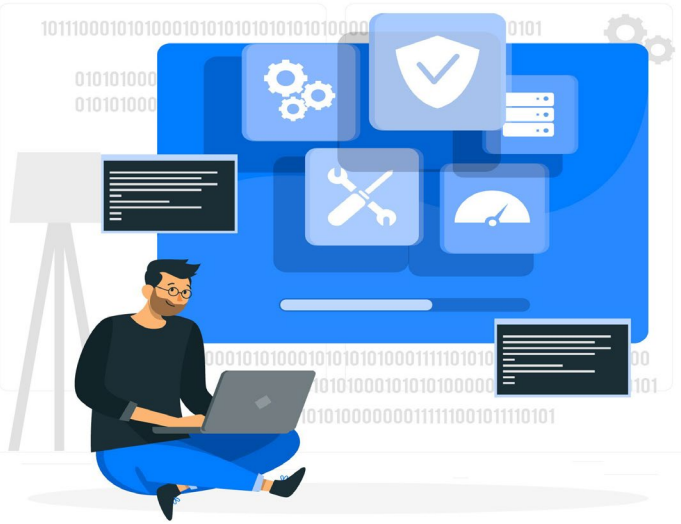
Authorization and Access Control:

- **Access Control Lists (ACLs):** These are lists of permissions associated with API endpoints, specifying who can perform specific actions (e.g., read, write, delete).
- **Role-Based Access Control (RBAC):** A more advanced authorization mechanism that assigns roles to users or applications and controls access based on those roles.

Testing methods

Manual Testing of APIs

Manual API testing is a method where testers interact with an API directly without the use of automated tools or scripts.



- Testers manually verify the behavior of the API by crafting and sending various types of requests to different endpoints.
- Providing different input data to observe how the API responds.
- Checking for correct status codes, response data, and adherence to the API documentation.
- Manually evaluating the API's performance, error handling, and security.

Automated Testing of APIs

Automated API testing is the process of using specialized testing frameworks and scripts to perform testing tasks automatically.

- **Postman:** Postman allows users to create automated test scripts using JavaScript. Testers can set up collections of requests and write test scripts to validate responses and verify the API's functionality.
- **RestAssured:** A Java-based testing framework for REST APIs. It provides a set of methods for sending requests, validating responses, and writing test cases in a programmatic way.
- **Karate:** A framework that combines API testing with a BDD (Behavior-Driven Development) approach. It uses a simple syntax and allows writing test scenarios in a human-readable format.

Advantages of Automated Testing

Repeatability

Automated tests can be run as often as needed, ensuring consistent and repeatable results.

Speed

Automation can execute a large number of test cases much faster than manual testing

Advantages of Automated Testing

Continuous Integration

Automated API tests can be integrated into CI/CD pipelines for continuous validation of API changes.

Regression Testing

Automated tests are valuable for checking that new changes to the API don't break existing functionality

API Testing Approach and its Test Cases

API Testing Approach

- The API Testing Approach is a predetermined strategy or method employed by the QA team to conduct testing on the API once the build is prepared.

- This form of testing does not involve examining the source code.

- The API testing approach aids in gaining a deeper comprehension of the functionalities, testing methods, input parameters, and the process of executing test cases.

API Testing Approach

Gaining insight into the API program's operation and precisely outlining the program's scope.

Utilize testing methodologies like equivalence classes, boundary value analysis, and educated guessing to formulate test cases for the API.

It's crucial to carefully plan and establish the input parameters for the API.

Perform the test cases and the expected outcomes with the real results.

Test Cases for API Testing

Return value based on input condition

It is relatively easy to test, as input can be defined and results can be authenticated

Does not return anything

When there is no return value, a behavior of API on the system to be checked

Trigger some other API

If an output of an API triggers some event or interrupt, then those events and interrupt listeners should be tracked

Test Cases for API Testing

Update data structure

Updating data structure will have some outcome or effect on the system, and that should be authenticated

Modify certain resources

If API call modifies some resources then it should be validated by accessing respective resources

Automation Frameworks

Automation Frameworks

API testing frameworks are pre-built structures that offer reusable components, guidelines, and a structured approach to creating automated API tests.

Frameworks help testers efficiently to create and maintain automated tests.

The types of API testing frameworks are data-driven, keyword-driven, and behavior-driven development (BDD) frameworks.

Frameworks like Postman, RestAssured, Karate, or tools like Selenium are a few popular frameworks.

How to Test API

How to Test API

Exploratory Testing

The testing team should manually perform a series of API calls as outlined in the API documentation. This involves confirming that a specific resource exposed by the API can be appropriately listed, created, and deleted.

Usability Testing

This type of testing aims to determine whether the API is not only fully functional but also user-friendly. It assesses how well the API integrates with other platforms.

Security Testing

This phase of testing involves evaluating the authentication methods required by the API and whether sensitive data is securely encrypted when transmitted over HTTP or other protocols.

How to Test API

Automation Testing

The API testing process should conclude with the development of a set of scripts or a testing tool that can be used for routine API testing.

Documentation Validation

The testing includes ensuring that the documentation accompanying the API is comprehensive and provides sufficient information for users to effectively interact with the API.

Security Testing

This phase of testing involves evaluating the authentication methods required by the API and whether sensitive data is securely encrypted when transmitted over HTTP or other protocols.

Data Validation and Performance Testing

Data Validation

- **Response Data Validation:** In API testing, it's crucial to validate the data received in API responses.

Schema Validation

Ensuring that the response adheres to a predefined schema or contract, such as a JSON Schema or XML Schema.

Data Type Validation

Verifying that data types (e.g., integers, strings) in the response match the expected types.

Field Existence and Content

Checking if specific fields exist in the response and that their content is as expected.

Data Validation

Boundary Values

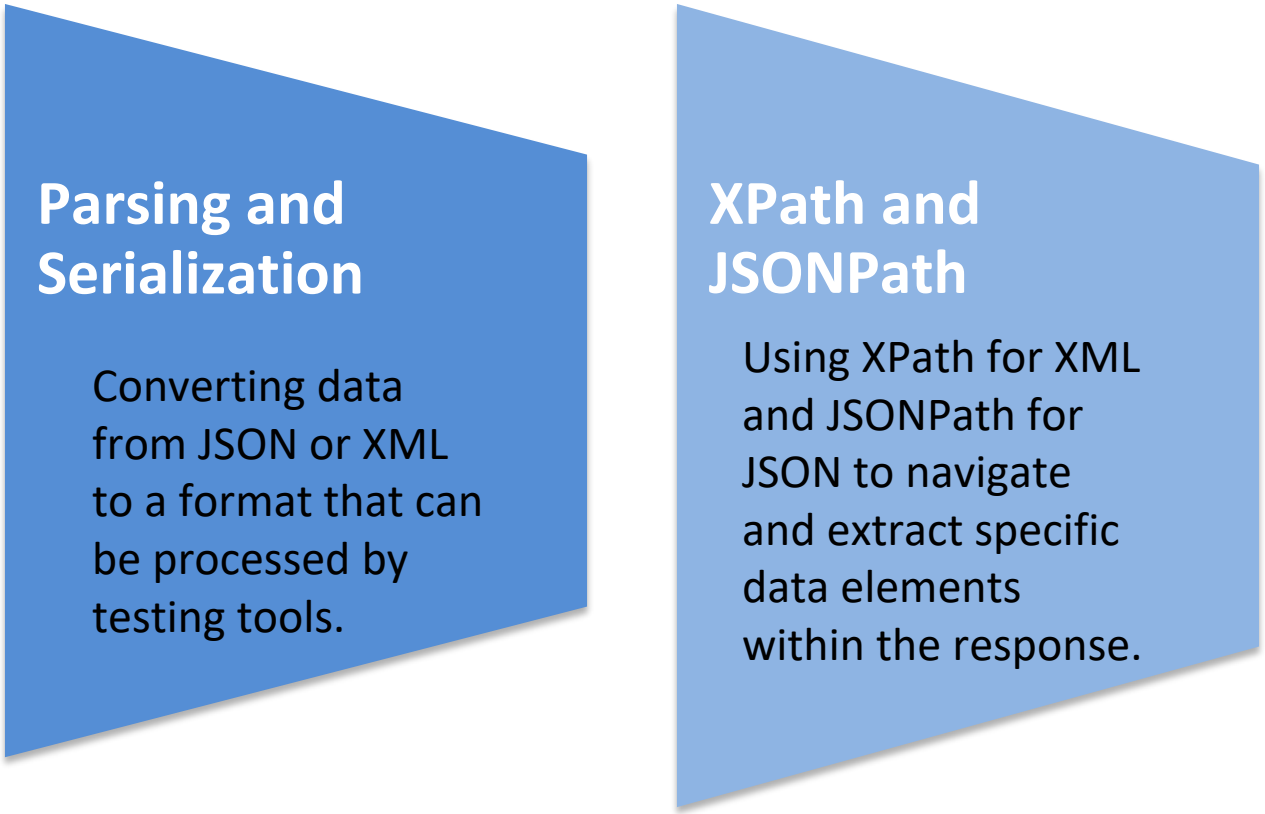
Validating values within specified boundaries.

Error and Exception Handling

Testing the response for error messages or exceptions that indicate issues with the API.

Data Validation

- **Handling JSON and XML Data:** APIs often use JSON and XML formats for data exchange.



Data Validation

Data Transformation

Converting data from one format (e.g., XML) to another (e.g., JSON) if needed.

Data Comparison

Comparing expected and actual data to validate correctness.

Performance Testing

- Load Testing : Load testing involves assessing the API's performance under expected load conditions, which helps identify bottlenecks and performance issues.

Load Levels

Gradually increasing the number of concurrent users or requests to gauge how the API behaves under normal load.

Stress Testing

Subjecting the API to loads beyond its capacity to identify breaking points and potential failures.

Virtual Users

Using tools to simulate virtual users making requests to the API, mimicking real-world usage.

Performance Testing

- Monitoring and Analyzing API Performance: API performance is monitored and analyzed.

Set Performance Metrics

Define key performance indicators (KPIs) such as response times, throughput, and error rates.

Continuous Monitoring

Continuously observe API performance and analyze trends over time.

Profiling and Profiling Tools

Profile the API to identify resource-intensive operations and bottlenecks.

Performance Testing

Load Balancing

Verify the efficiency of load balancing mechanisms in distributing incoming requests.

Scalability Testing

Test how well the API scales by adding resources or replicating components under load.

Error Handling and Security Testing

Error Handling

Handling API Errors and Exceptions

- API testing should include scenarios to validate how the API handles errors and exceptions.
- This involves intentionally triggering errors by providing invalid data or making incorrect requests to the API.

Appropriate Error Codes and Messages

- Error responses from an API should provide clear and meaningful information to both developers and end-users.
- This helps in troubleshooting and resolving issues efficiently.

Security Testing

Identifying Security Vulnerabilities

- **SQL Injection:** Assess if the API is vulnerable to SQL injection attacks, which can manipulate or retrieve sensitive data from the database.
- **Data Exposure:** Check if the API inadvertently exposes sensitive information, such as personal or confidential data, to unauthorized users.

Securing API Endpoints

- Security testing also involves verifying that security measures are in place to protect API endpoints.

Documentation and Reporting

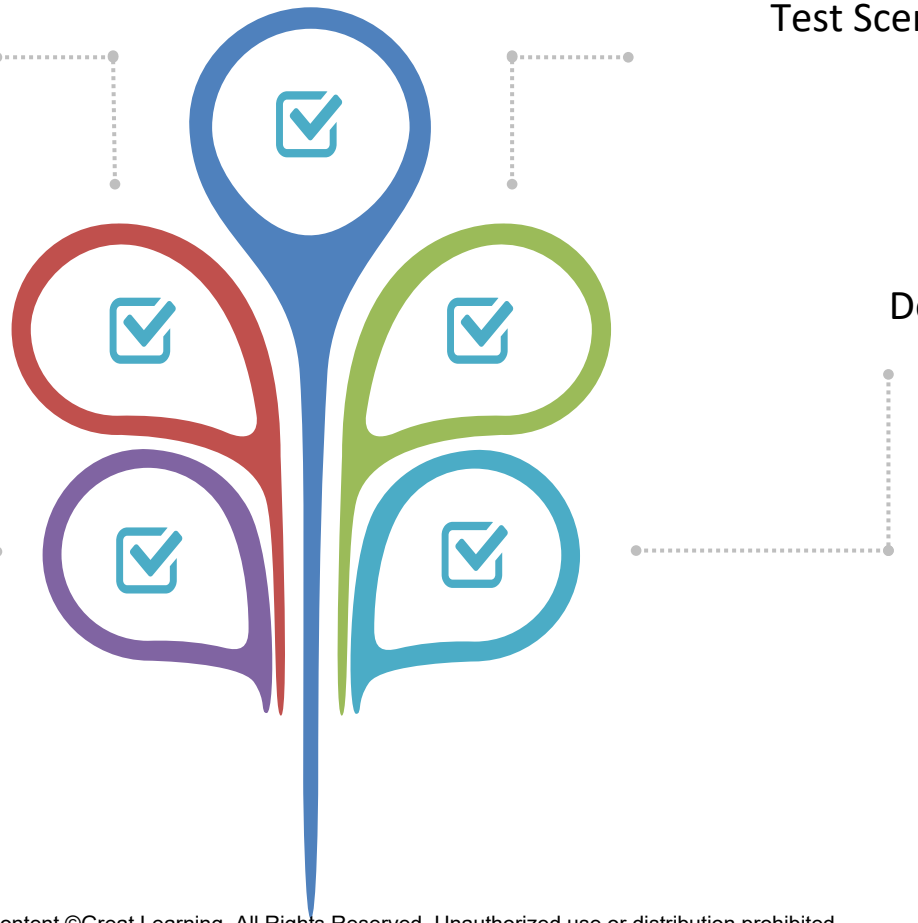
Documentation

Test Cases

Test Data

Test Scenarios

Test Plan



Documentation of API Endpoints

Reporting

Test Logs

Test Metrics

Test Summary Report

Test Execution

Continuous Monitoring

Best Practices of API Testing

Best Practices

API Test cases

Organize API test cases into distinct categories for clarity.

Declarations

Begin each test with clear declarations of the APIs being utilized.

Parameters

Explicitly state the selected parameters within each test case.

Best Practices

Prioritization

Prioritize the sequencing of API function calls for the convenience of testers.

Independent

Strive to ensure that each test case is as self-contained as possible, minimizing dependencies.

Development Process

Refrain from creating interlinked "test chaining" within your development process.

Best Practices

Handling Functions

Exercise extra caution when handling one-time call functions like "Delete" or "CloseWindow."

Execution

Plan and execute call sequences thoughtfully and methodically.

Test Coverage

To guarantee comprehensive test coverage, create API test cases that encompass all potential input combinations for the API.

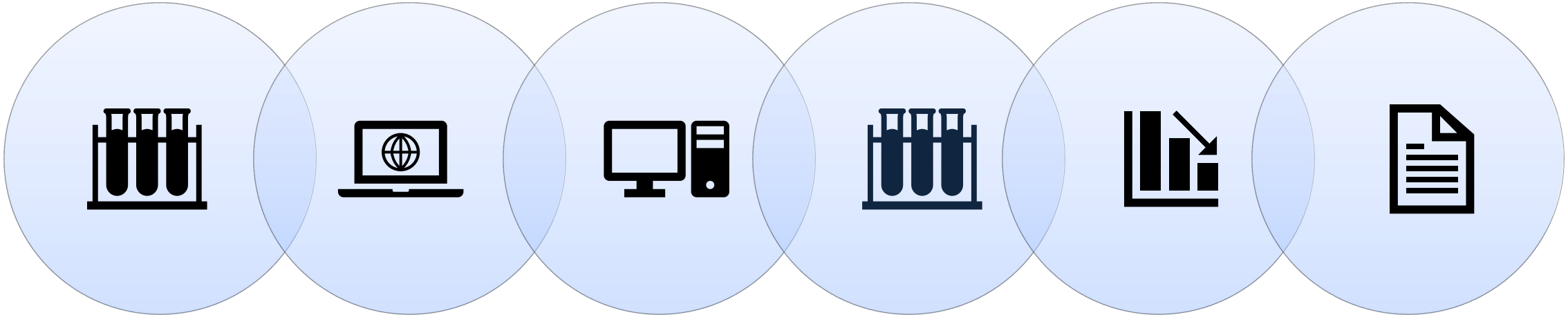
Difference between API Testing and Unit Testing

Authentication

Unit Testing	API Testing
Unit testing is performed by developers	API testing is performed by testers
In unit testing separate functionality is tested	In API testing End-to-end functionality is tested
The source code can be accessed by the developer	The source code can not be accessed by the tester
UI testing is also included in unit testing	Only API functions will be tested in API testing
Only the basic functionalities are tested in unit testing	All the functional issues are tested in API testing
The unit testing is limited in scope	The API testing is broader in scope

Challenges and Pitfalls

Challenges



Parameter Combination, Selection and Call Sequencing

Absence of GUI

Verifying and Validating on different Systems

Parameter Selection and Categorization

Exceptional Handling Function

Coding Knowledge

Ethical and Legal Considerations

Ethical and Legal Considerations

Respecting API Usage Policies

- API providers often have terms of service and usage policies
- To avoid any violations that may lead to legal consequences or the suspension of API.

Data Privacy and Security

- Testers should be cautious when working with sensitive or personal data through APIs.
- It's essential to handle data in compliance with privacy regulations, such as GDPR or HIPAA, and to avoid data breaches or exposure.

Ethical and Legal Considerations

Rate Limiting

- Many APIs have rate limits to prevent abuse and ensure fair usage.
- Testers must be mindful of these limits to avoid overloading the API server and potentially disrupting its services.

Consent and Permissions

- If the API involves user data, it's essential to obtain proper consent and permissions from users before testing.

Ethical and Legal Considerations

Data Ownership

- Understand who owns the data being accessed through the API.
- Testers must be aware of copyright and intellectual property rights, and respect data ownership rules.

Compliance with Regulations

- Different industries and regions have specific regulations and compliance requirements.
- Testers must be aware of these and ensure that the API testing process aligns with legal obligations.

Ethical and Legal Considerations

Data Masking and Anonymization

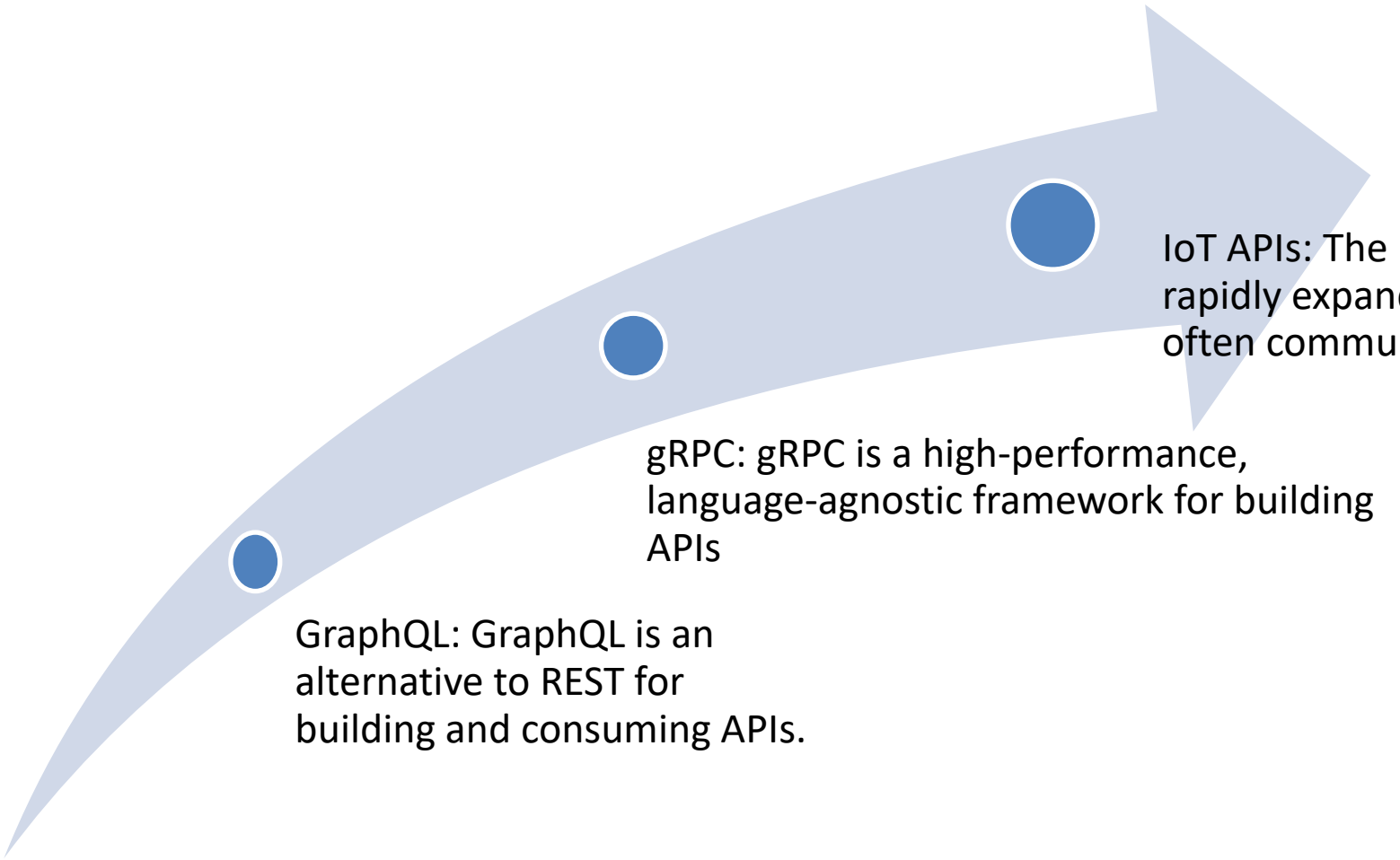
- To protect sensitive data, consider using data masking or anonymization techniques in test environments to prevent exposure of confidential information.

Reporting Vulnerabilities Responsibly

- If testers discover security vulnerabilities during testing, it's important to report them to the API provider promptly and responsibly rather than exploiting them.

Future Trends and Emerging Technologies

Future Trends



IoT APIs: The Internet of Things (IoT) is rapidly expanding, and IoT devices often communicate via APIs.

gRPC: gRPC is a high-performance, language-agnostic framework for building APIs

GraphQL: GraphQL is an alternative to REST for building and consuming APIs.

References

- https://dummy.restapiexample.com/
- https://jsonplaceholder.typicode.com/
- https://gorest.co.in/
- https://restful-booker.herokuapp.com/

Summary

Here’s a brief recap:

- APIs are software tools that enable communication between different components. To use them effectively, you need to set up the right environment, handle authentication, choose testing methods, and use automation frameworks to create and manage automated tests efficiently.
- In API testing, it's crucial to focus on error handling and security testing. Additionally, thorough documentation and transparent reporting are vital. By following best practices, you can ensure comprehensive and secure API testing, leading to effective testing methods and clear reporting.
- The differences between API Testing and Unit Testing is understood. API Testing has its unique challenges, ethical and legal aspects to consider, and it's important to keep an eye on future trends and emerging technologies in this field.