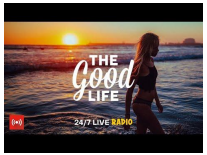
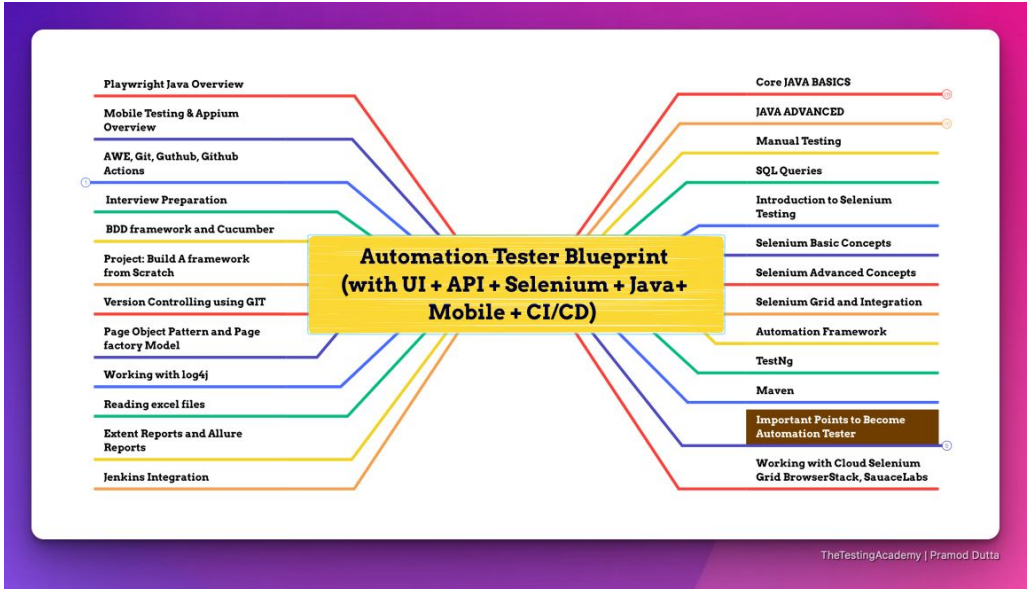
Meeting Etiquette

- Please be on Mute all the time.
- Turn off Video, So that we can save bandwidth.
- There is separate Q&A Section in End Please use that, Add questions in Google Form.
- Break at 5 min.
- Mute your microphone
- To help keep background noise to a minimum, make sure you mute your microphone when you are not speaking.
- Be mindful of background noise
- Avoid multitasking
- All links and Slides will be shared.



{ Automation Tester } BluePrint.

Pramod Dutta



Manual API Testing for Automation Tester.

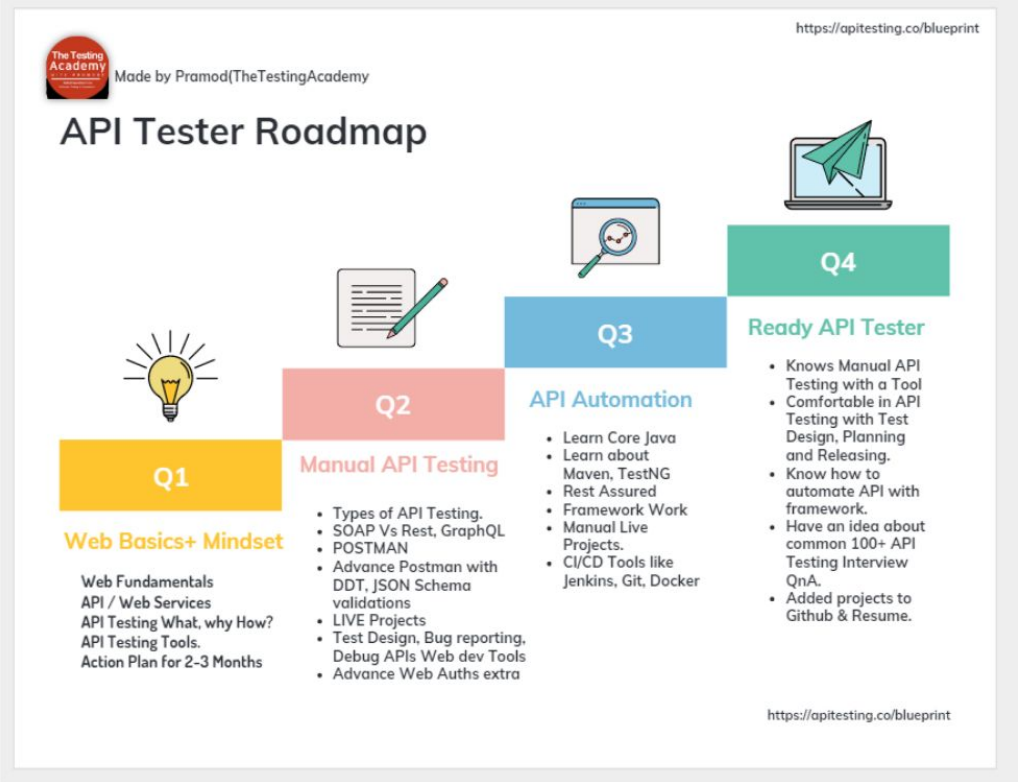
Before Starting

1. Manual Quiz Done?
2. Agile Testing Quiz Done?
3. API Testing DOne?

API Testing

Commitment!
Block at least
4 hour Per Week.

ROADMAP to Become an API Tester



Agenda of Session

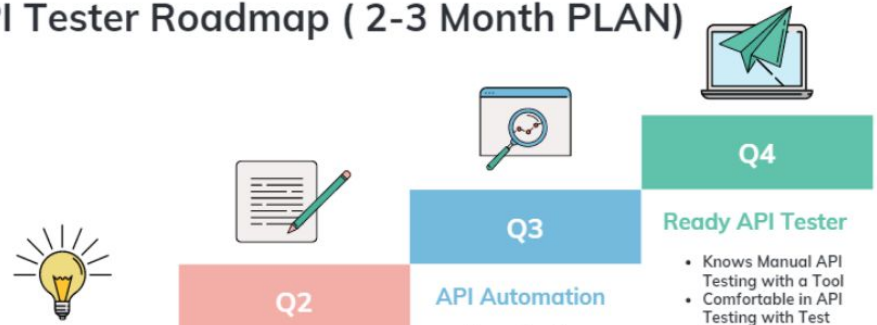
- POSTMAN OverView
- Import External Request.
- History
- WorkSpace in Postman
- Postman Tabs - Param, Auth,
- Headers, Body, Pre-req, Post Req(Test), Settings
- Postman Console.

API Tester ROADMAP

https://apitesting.co/blueprint

Made by Pramod(TheTestingAcademy

API Tester Roadmap (2-3 Month PLAN)



Q1

Web Basics+ Mindset

Web Fundamentals
API / Web Services
API Testing What, why How?
API Testing Tools.
Action Plan for 2-3 Months

Q2

Manual API Testing

- Types of API Testing.
- SOAP Vs Rest, GraphQL
- POSTMAN
- Advance Postman with DDT, JSON Schema validations
- LIVE Projects
- Test Design, Bug reporting, Debug APIs Web dev Tools
- Advance Web Auths extra

Q3

API Automation

- Learn Core Java
- Learn about Maven, TestNG
- Rest Assured
- Framework Work
- Manual Live Projects.
- CI/CD Tools like Jenkins, Git, Docker

Q4

Ready API Tester

- Knows Manual API Testing with a Tool
- Comfortable in API Testing with Test Design, Planning and Releasing.
- Know how to automate API with framework.
- Have an idea about common 100+ API Testing Interview QnA.
- Added projects to Github & Resume.

https://apitesting.co/blueprint

POSTMAN Basics

POSTMAN Basics

- OverView
- Import External Request.
- History
- WorkSpace in Postman
- Postman Tabs - Param, Auth, Headers, Body, Pre-req, Post Req(Test), Settings
- Postman Console.

POSTMAN Basics

- Import External Request.

POSTMAN Basics

- History

POSTMAN Basics

- WorkSpace in Postman

POSTMAN Basics

- Postman Tabs - Param, Auth, Headers, Body, Pre-req, Post Req(Test), Settings.

POSTMAN Basics

- Postman Console.

Authorization VS Authentication

Authorization VS Authentication

Authentication	Authorization
Authentication verifies who the user is.	Authorization determines what resources a user can access.
Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user.	Authorization works through settings that are implemented and maintained by the organization.
Authentication is the first step of a good identity and access management process.	Authorization always takes place after authentication.
Authentication is visible to and partially changeable by the user.	Authorization isn't visible to or changeable by the user.
Example: By verifying their identity, employees can gain access to an HR application that includes their personal pay information, vacation time, and 401K data.	Example: Once their level of access is authorized, employees and HR managers can access different levels of data based on the permissions set by the organization.

Authorization

- APIs use authorization to ensure that client requests access data securely
  - Authorization types
    - **No auth**
    - **API key**
    - Bearer token
    - **Basic auth**
    - Digest auth
    - OAuth 1.0
    - OAuth 2.0
    - Hawk authentication
    - AWS Signature
    - NTLM authentication
    - Akamai EdgeGrid

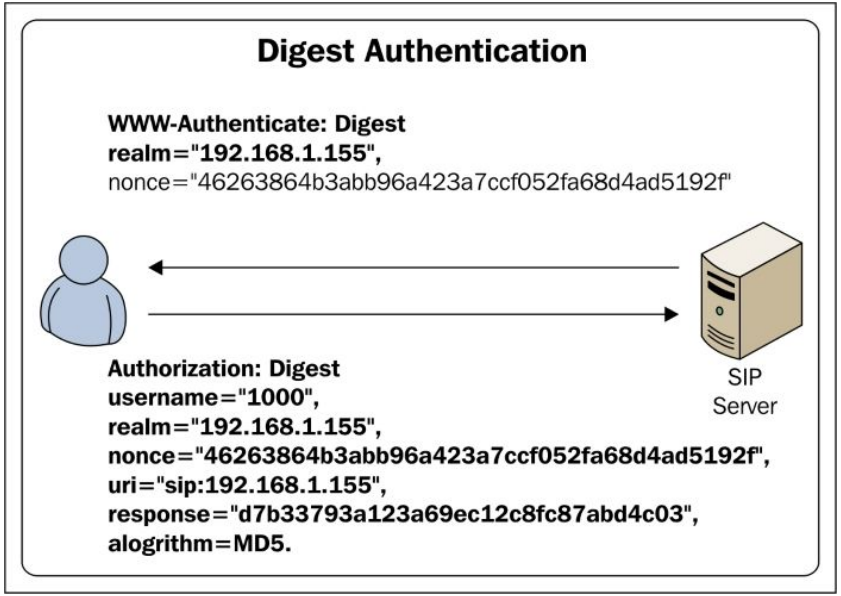
Basic auth

Basic authentication involves sending a verified username and password with your request. In the request Authorization tab, select Basic Auth from the Type dropdown list

Basic <Base64 encoded username and password>

Digest auth

With Digest auth, the client sends a first request to the API, and the server responds with a few details, including a number that can be used only once (a nonce), a realm value, and a 401 unauthorized response.



Bearer Authentication

Bearer authentication (also called token authentication) is an HTTP authentication scheme that involves security tokens called bearer tokens

Authorization: Bearer <token>

The name “Bearer authentication” can be understood as “give access to the bearer of this token.”

The bearer token is a cryptic string, usually generated by the server in response to a login request. The client must send this token in the `Authorization` header when making requests to protected resources:

JWT Token

JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

https://jwt.io/

JWTs are a convenient way to encode and verify claims.

A Bearer token is just string, potentially arbitrary, that is used for authorization.

https://stackoverflow.com/questions/40375508/whats-the-difference-between-jwts-and-bearer-token

API Keys

Some APIs use API keys for authorization. An API key is a token that a client provides when making API calls. The key can be sent in the query string:

GET /something?api_key=abcdef12345

or as a cookie:

GET /something HTTP/1.1

Cookie: X-API-KEY=abcdef12345

`X-API-KEY`

Cookie Authentication

Cookie authentication uses HTTP cookies to authenticate client requests and maintain session information.

The client sends a login request to the server.

On the successful login, the server response includes the Set-Cookie header that contains the cookie name, value, expiry time and some other info. Here is an example that sets the cookie named JSESSIONID:

Set-Cookie: JSESSIONID=abcde12345; Path=/; HttpOnly

The client needs to send this cookie in the Cookie header in all subsequent requests to the server.

Cookie: JSESSIONID=abcde12345

On the logout operation, the server sends back the Set-Cookie header that causes the cookie to expire.

OAuth 2.0

OAuth 2.0 is an authorization protocol that gives an API client limited access to user data on a web server. GitHub, Google, and Facebook APIs notably use it.

an OAuth 2.0 server issues access tokens that the client applications can use to access protected resources on behalf of the resource owner.

OAuth relies on authentication scenarios called *flows*,

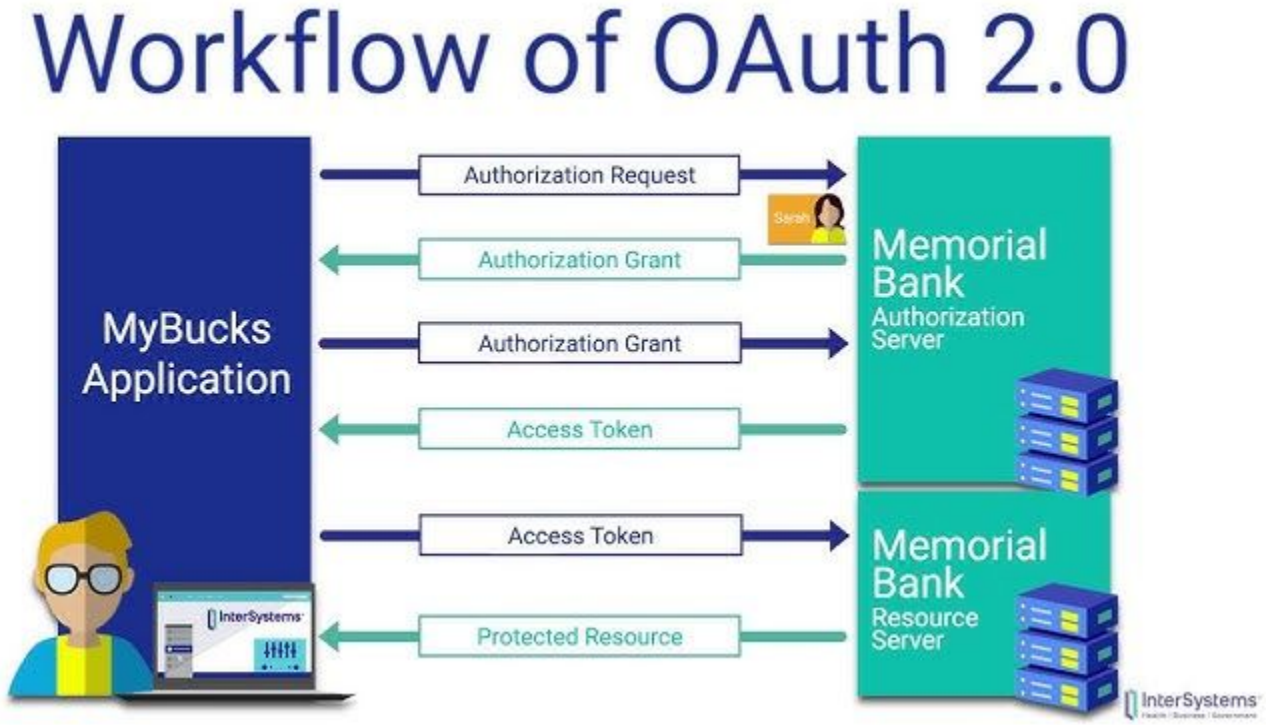
https://developer.spotify.com/console/get-album/

OAuth 2.0

An example OAuth 2.0 flow could run as follows:

- A client application makes a request for the user to authorize access to their data.
- If the user grants access, the application then requests an access token from the service provider, passing the access grant from the user and authentication details to identify the client.
- The service provider validates these details and returns an access token.
- The client uses the access token to request the user data with the service provider.

OAuth 2.0


Workflow of OAuth 2.0
MyBucks Application
Authorization Request
Authorization Grant
Authorization Grant
Access Token
Access Token
Protected Resource
Memorial Bank Authorization Server
Memorial Bank Resource Server
InterSystems

Demo OAuth 2.0

Quiz

Status Code for Invalid API Keys.

401
Response

Thanks, for attending Class

I hope you liked it. Say Thanks in Comment :)