**# Cypress\_API\_Automation**

This Repo provides an overview of API automation testing, focusing on the fundamental HTTP methods: GET, POST, PUT, and DELETE. In API automation, we validate the functionality, performance, and reliability of APIs programmatically, ensuring they behave as expected.

**\*\*Table of Contents\*\***

Introduction

Testing HTTP Methods

GET Requests

POST Requests

PUT Requests

DELETE Requests

Conclusion

\*\*Introduction

\*\*API (Application Programming Interface) automation is a critical part of modern software development. It involves automating the testing of APIs, which are the building blocks of many applications. API tests ensure that the endpoints exposed by your application or service work correctly, deliver the expected responses, and handle various scenarios gracefully.

**\*\*Testing HTTP Methods\*\***

**\*\*GET Requests\*\***

GET requests are used to retrieve information from the server. In API automation, you should test GET requests to:

Verify that the endpoint returns the expected data.

Check for proper error handling when a resource is not found.

Test performance by measuring response times for different payloads.

**\*\*POST Requests\*\***

POST requests are used to create new resources on the server. In API automation, you should test POST requests to:

Ensure that data sent in the request body is correctly processed and stored on the server.

Verify that the server responds with the appropriate status code (e.g., 201 Created) and a location header pointing to the newly created resource.

Test error handling when invalid data is provided.

**\*\*PUT Requests\*\***

PUT requests are used to update existing resources on the server. In API automation, you should test PUT requests to:

Confirm that the resource is updated correctly with the data sent in the request body.

Verify that the server responds with the appropriate status code (e.g., 200 OK) when the update is successful.

Test error scenarios, such as attempting to update a nonexistent resource.

**\*\*DELETE Requests\*\***

DELETE requests are used to remove resources from the server. In API automation, you should test DELETE requests to:

Ensure that the specified resource is deleted from the server.

Verify that the server responds with the appropriate status code (e.g., 204 No Content) when the deletion is successful.

Test error scenarios, such as attempting to delete a resource that doesn't exist.

**\*\*Conclusion\*\***

API automation testing is a crucial aspect of ensuring the quality and reliability of your software applications.

By rigorously testing GET, POST, PUT, and DELETE requests, you can identify and address issues early in the development cycle, leading to more robust and stable applications. Adopting best practices and using appropriate tools will streamline your API testing efforts and contribute to the overall success of your project.

///////////////////////////////////////

**# Cypress\_CrossBrowserTesting**

Cross Browser Testing is the practice of testing your website's functionality and appearance across various web browsers to ensure that it works optimally on all platforms.

It is essential because different browsers have different rendering engines to display web content. For example, Chrome uses the Blink rendering engine while Firefox uses Gecko and Safari uses the WebKit rendering engine.

I have shared below ways to perform cross browser testing in Cypress.

1). Cross Browser Testing using Cypress test runner

2). Cross Browser Testing using Cypress test file

3). Cross Browser Testing using Cypress CLI

4). Cross Browser Testing Using Script

5). Cross Browser Testing using Cloud Platform

6). Cross Browser Testing using GitHub Actions

7). Cross Browser Testing using GitHub Actions

8). Cross Browser Testing using Docker

For a more detailed understanding of cross browser

testing in Cypress, please refer to the blog provided below.

Blog : https://www.lambdatest.com/learning-hub/crossbrowser-testing-with-cypress

**# Cypress-Viewport-Testing**

Testing responsiveness of the web application.

<img width="970" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/43ddd4fd-b863-4063-adf6-a92c5c991271">

(1) Testing by passing different widths and height using the below syntax

<img width="645" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/c786723e-0aef-4791-b7bf-6e0c46874f91">

(2) Testing viewport by passing presets

<img width="348" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/dbf35b35-332d-4961-b133-5d8b48db2bf0">

Example:

<img width="334" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/22ba7847-de83-41fa-8acb-f9d062b8dd17">

(3) Testing viewports by passing orientation

Example

<img width="905" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/e7f69477-ce4b-40b0-bbb8-3a7cfcbfa0fd">

(4) Configuring viewport at the project level

Example:

<img width="1007" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/3f549144-3e83-411f-9b55-c27a0b19cf33">

(5) Testing resolution on cloud platform

<img width="153" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/a832df89-0d5e-4d55-9012-9ae97aa9da28">

(6). Testing in Full Mode

<img width="1119" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/9daf0564-c205-41d3-b529-5fed30ac92a0">

(7). Override Viewport Options At the Run Time

     Sample Command :

     npx cypress run --browser chrome --config viewportWidth=1280,viewportHeight=720

     PS - You can change the viewport width and height as per your requirement

 (8). Testing Viewport Using Environment Variables

<img width="372" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-Viewport-Testing/assets/10338077/544b941f-1db7-4ace-b4db-3f125a522443">

**# Cypress-E2E**

Cypress-E2E

Automation Framework using below components:

Page Object Model (Page Class + Test Files)

Commands.js

Fixtures

Mochawesome HTMl Report

Customized Scripts in Package.json

Running using headless mode

GitHub Actions

<img width="1394" alt="image" src="https://github.com/Anshita-Bhasin/Cypress-E2E/assets/10338077/7e322357-a5b8-4dd4-8082-8c6a989a419f">

**# Cypress\_Read\_Data\_From\_CSV**

Cypress\_Read\_Data\_From\_CSV

This project demonstrates how to automate the user registration process in Cypress.

The test data for the registration form is sourced from a CSV (Comma-Separated Values) file, enabling us to test multiple scenarios with ease.

Additionally, I have used the Faker API from the Faker.js library to generate unique and random email addresses for each test run.

For the set- up, please refer to the medium blog here -

https://medium.com/@anshita.bhasin/read-csv-data-in-cypress-57b1792c6945

Ref — https://www.npmjs.com/package/@faker-js/faker, https://npm.io/package/neat-csv

Practise Website - https://naveenautomationlabs.com/opencart/index.php?route=account/register

![image](https://github.com/Anshita-Bhasin/Cypress\_Read\_Data\_From\_CSV/assets/10338077/fbbacc1f-49b4-4a53-a276-c970621d54c1)

![image](https://github.com/Anshita-Bhasin/Cypress\_Read\_Data\_From\_CSV/assets/10338077/a1583b5f-2327-4c4b-ba38-f74db694a54e)

**# Cypress-Docker-Testing**

This repository provides a sample project that demonstrates how to use Cypress tests with Docker containers. By running Cypress in a Docker container, you can easily set up a consistent and reproducible environment for your tests.

As of April 2023, Cypress supports below docker images

<img width="862" alt="image" src="https://user-images.githubusercontent.com/10338077/232113265-97572a4c-b9fd-4cd1-a762-f8c8e9f21c3b.png">

**# Prerequisites**

**=================**

Install:

Docker

Node.js

**# Run Headless Cypress test locally using Docker Image**

Docker Image contains both Cypress and OS dependencies which can be used to run Cypress test.As of today ( March 2023), there are 4 images provided for running Cypress in Docker.

cypress/factory

cypress/base

cypress/browsers

cypress/included

There are multiple ways in which Cypress tests can be run in Docker containers. One of the ways is to run using the following command from the root of your project.

"<docker run -it -v $PWD:/e2e -w /e2e cypress/included:12.8.1>"

**# Run Cypress on Chrome and Firefox Browser Using Docker Image**

Running Cypress tests on specific browser versions can be necessary to ensure compatibility with the target environment. Different browsers like Firefox and Chrome may have varying feature sets and behaviors that can affect the test results.

By default, Cypress runs the tests on Electron browser but if you want to run Cypress tests on Chrome and Firefox browsers using a Docker image. Then, you can follow the below commands.

"< docker run -it -v $PWD:/e2e -w /e2e cypress/included:12.8.1 --browser firefox >"

**# Run Cypress tests in headless mode using Dockerfile**

Running Cypress tests with Docker can help catch and fix issues early in the development process, saving time and resources down the line.if you want to use a customized script and run it using Docker. In that case, you can use Dockerfile.

You can refer to Dockerfile created at the root level. After creating a Dockerfile, next process is

1) Building a Dockerfile

To build the Docker image, navigate to the directory containing your Dockerfile and run the following command:

" <docker build -t <image\_name> . >"

2) After the docker image is built, you can verify by running it,by running the command

    "<docker images>"

3) Run the docker image after building it by using the command :

    "<docker run imagename:tagename .>"

**# Run Cypress test on multiple browsers Using Docker Compose**

The Docker Compose file is a YAML file that defines how Docker containers should be built, configured, and run as a single service or application. With the Docker Compose file, you can define multiple containers and their configuration in a single file, making it easier to manage and deploy complex Cypress testing environments.

Once you create docker compose file . You can run the test in docker container using the command

"< docker-compose up >"

**# Run Cypress test on Cloud Grid Using Docker file**

Cypress testing on a cloud platform (like LambdaTest) helps you attain accelerated test execution as well as improved browser coverage. There are multiple ways to perform Cypress testing on a cloud testing platform like LambdaTest. One of them is using Docker.

To use Dockerfile, follow below steps

1) Build Docker Image  "docker build -t <image\_name> . "

2) Docker Login : docker login

3) Docker Tag : docker tag imagename:tag dockerUsername/imagename:tag

4) Docker Push : docker push dockerUsername/image-name:v1

5) Docker Run : docker run <image\_name>:<tag\_name>

**# Run Cypress Test in Docker Container on CI.CD(GitHub Actions)**

GitHub Actions is a powerful CI/CD platform that enables you to automate workflows for building, testing, and publishing software across multiple platforms, operating systems, and programming languages. With GitHub Actions, you can even see the status checks displayed within the pull request.

One of the significant advantages of using GitHub Actions is that it allows you to execute Cypress tests in a Docker container, providing a self-contained and repeatable environment for testing.

There is an existing Docker image that already includes Chrome version 87 and Firefox version 82 pre-installed, which you can use to run Cypress tests. To leverage this image, you can pass the container image as an argument. Once the test cases are executed, the container is automatically stopped.

Create .yml in ".github/workflows/build.yml" and add the code for running test case in docker container

"<container: cypress/browsers:node12.13.0-chrome78-ff70>"

**# Bonus: Top Docker Commands**

Sharing some useful Docker commands.

<img width="665" alt="image" src="https://user-images.githubusercontent.com/10338077/232119538-c6602213-a57e-4716-a1d6-74d35f0b56c2.png">