2023

Cypress Framework Setup and Execution Guide

TENPEARLS

MUBBASHIR SHAKIL - 3067

Table of Contents

Introduction

The Cypress Automation Framework is designed to facilitate end-to-end testing for both API and UI applications. It includes a well-organized directory structure and a set of utilities to simplify test creation, management, and execution.

Directory Structure

```
cypress/
├── e2e/
│   ├── Api/
│   │   ├── examples/
│   │   │   ├── createuser.cy.js
│   │   │   ├── deleteuser.cy.js
│   │   │   ├── getuser.cy.js
│   │   │   ├── registeruser.cy.js
│   │   │   └── states.cy.js
│   ├── config/
│   │   └── constants.js
│   └── UI/
│       ├── examples/
│       │   ├── pom-implementation/
│       │   │   └── logintest.cy.js
│       │   └── testdata-management/
│       │       ├── testdata-csv.cy.js
│       │       └── testdata-json.cy.js
│       └── pages/
│           ├── basePage.js
│           └── loginPage.js
├── fixtures/
│   └── examples/
│       ├── api-testdata.json
│       ├── userInfo.csv
│       └── userInfo.json
├── logs/
├── reports/
├── results/
├── screenshots/
├── support/
│   ├── node_fs/
│   │   ├── createDir.js
│   │   └── removeDir.js
│   ├── command.js
│   ├── e2e.js
│   └── utils/
│       ├── apiUtils.js
│       ├── driver.js
│       └── logging.js
├── cypress.config.js
├── cypress.env.json
├── docker-compose.yml
├── Dockerfile
```

```
└── package.json
```

Directory Description

- **e2e/**: Contains end-to-end tests for both API and UI applications.
    - **Api/**: Holds API test suites and configurations.
    - **config/**: Stores configuration files, such as constants used across tests.
    - **UI/**: Contains UI test suites and configurations.
- **fixtures/**: Stores static data used by tests, including JSON files and CSV files.
- **logs/**: Directory for storing log files generated during test execution.
- **reports/**: Location for test reports generated after test execution.
- **results/**: Stores the results of test execution, such as logs and screenshots.
- **screenshots/**: Stores screenshots captured during test execution.
- **support/**: Contains additional utilities and custom commands used in tests.
- **cypress.config.js**: Cypress configuration file.
- **cypress.env.json**: Cypress environment configuration file.
- **docker-compose.yml**: Docker Compose configuration file.
- **Dockerfile**: Dockerfile for creating a Docker image for the framework.
- **package.json**: Node.js package configuration file.

Framework Feature

- **API Testing**: Provides test suites and examples for testing APIs.
- **UI Testing**: Contains test suites and examples for testing UI applications, including Page Object Model (POM) implementation.
- **Test Data Management**: Includes utilities for managing test data from various sources like JSON and CSV files.
- **Logging and Reporting**: Logs test execution details and generates reports for easy analysis.
- **Custom Commands**: Extends Cypress with custom commands for enhanced test capabilities.
- **Continuous Integration**: Compatible with CI/CD pipelines for automated testing.

- **Docker Support**: Can be containerized using Docker for easy deployment and scalability.

Docker Configuration

To run your tests on docker container, simply follow the below instructions.

- Download and install Docker
- once installed simply go to your solution directory and write **`docker-compose up`** in **VScode** terminal and execution will start on docker container once the execution is done you can see the results under `docker-result/` directory.

Getting Started

To begin using the Cypress Automation Framework, follow these steps:

1. Clone the repository Repo Link.
2. Install dependencies using **`npm install`**.
3. Write test scripts in the appropriate directories under **`cypress/e2e/UI/test`**.
4. Configure test environments and settings in **`cypress.config.js`** and **`cypress.env.json`**.
5. **`cypress.env.json`** content will be provided on request.
6. Execute tests using following commands.
    1. **`npm run cypress:open`** to run tests from Cypress UI.
    2. **`npm run cypress:run`** to run tests headless.
    3. **`npm run test:e2e:chrome`** run all tests from heedless using chrome browser.
    4. **`npm run test:e2e:edge`** run all tests from heedless using edge browser.
    5. **`npm run test:api`** run API tests only.