* Unit tests
* Integration tests
* End-to-end (E2E)

WDIO

Mocha

**C8 coverage**

Chai.js

**Test runners** (unit testing frameworks) are tools that help us to structure and run test scenarios. Some of the listed tools have build-in assertion libraries and some of them requires additional ones (such as Chai JS).

To understand what tools are needed for unit testing, one needs to remember what kind of tests there are. As a rule, these are tests that check the functions of the application under test, in other words, they check that the function will return the expected result if it is called.

Mocha

**Assertion libraries**

The main goal of any test is to perform a certain check. Checks/Asserts can be built using if-else statement, but this is not convenient enough and one needs to write a lot of repetitive code. NodeJS has a built-in module for asserts. But it is not very convenient for complex checks. As a result, the community solved this problem by creating third-party libraries that allow us to write human-readable checks, as well as any complexity. There are some test runners that already include functions for assertions, but they may also not be enough, and you can connect third-party ones for this.

Chai.js

**UI testing requires** the largest number of tools used, such as: a framework / library for working with a browser, a test runner (if it is not included in the framework for interacting with the browser), and an assertion library (if it is not included in the framework for interacting with the browser or if it is not enough)

WDIO

**Unit test** - They are tests that are aimed at testing the smallest testable parts of the software.

## **Why are unit tests necessary?**

* They allow you to be sure your code is doing exactly what it is supposed to do.
* They allow you to be sure that you are not breaking something when changing existing code.
* They act as low-level documentation for your code.
* They help you detect problems with your code early on.
* They help you with debugging by providing you with information about what exactly went wrong.
* They improve code quality: if you can’t cover a part of your code with unit tests you should rewrite it so that you can
* **Fast** - A developer should not hesitate to run the tests as they are slow. All of these including setup, the actual test and tear down should execute fast (milliseconds) as you may have thousands of tests in your entire project.
* **Isolated** - No order-of-run dependency, do not do any more actions after the assert statement(s), preferably a single logical assert.
* **Repeatable** – Tests should return the same result if no changes to the tested code were made.
* **Self-Validating** – No manual inspection is required to check whether the test has passed or failed.
* **Thorough** - Tests for corner/edge/boundary values. Test for exceptions and errors. Test for illegal arguments or bad inputs.

**C8 coverage**

**Reporter – mochaawesome**

[WebdriverIO](https://webdriver.io/) is a progressive automation framework built to automate modern web and mobile applications. It simplifies the interaction with your app and provides a set of plugins that help you create a scalable, robust and stable test suite.

### It is designed to be:

* **Extendable** - Adding helper functions, or more complicated sets and combinations of existing commands is simple and really useful
* **Compatible** - WebdriverIO can be run on the WebDriver Protocol for true cross-browser testing as well as Chrome DevTools Protocol for Chromium based automation using Puppeteer.
* **Feature Rich** - The huge variety of built-in and community plugins allows you to easily integrate and extend your setup to fulfil your requirements.

The Document Object Model ([DOM](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)) is the data representation of the objects that comprise the structure and content of a document on the web. In this guide, we'll briefly introduce the DOM. We'll look at how the DOM represents an HTML or XML document in memory and how you use APIs to create web content and applications

**Commands**

The [$](https://webdriver.io/docs/api/element/$/) command is a short way to call the findElement command in order to fetch a single element on the page.

**Layers are a fundamental architectural style in test automation. The main idea is the encapsulation of application logic in distinct layers.**

We should strictly adhere to the following rule: the higher layers should use the APIs of the lower layers, while the lower layers should not depend on the higher layers. This architectural pattern is a reflection of the Dependency Inversion Principle from the SOLID principles.

**Hierarchy of Layers**

* **Tests Layer:** This layer contains automated tests which are unaware of the page's structure and interact solely with the API provided by the Page Object Model (POM) layer.
* **Business Layer:** The POM resides here and should know nothing about the automated tests or be dependent on them in any way.
* **Core Layer:** The POM utilizes APIs from libraries to interact with the browser, network, etc. Ideally, abstract wrapper methods should be used to work with these libraries. This way, if there is a need to switch libraries, only these methods would require changes, leaving the higher-level logic unchanged.

Additionally, there exists a **Service Layer** which contains the configuration settings of the project, repository, and all necessary dependencies. This layer is separate because it is essential for each repository/project, regardless of the programming language used.