What is Unit Testing?

When you test your codebase, you take a piece of code — typically a function — and verify it behaves correctly in a specific situation. Unit testing is a structured and automated way of doing this. As a result, the more tests you write, the bigger the benefit you receive. You will also have a greater level of confidence in your codebase as you continue to develop it.

What is Integration testing?

It is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Setting up the tools: -

For this we are going to use mocha and chai.

**Mocha** is a feature-rich JavaScript test framework running on [Node.js](https://nodejs.org/) and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

Mocha comes with tons of great features, the website shows a long list but here are the ones I like the most:

* simple async support, including promises.
* async test timeout support.
* before, after, before each, after each hooks (very useful to clean the environment where each test!).
* use any assertion library you want, Chai

Install node.js for it. Give `**npm install`** command.

If you want to test Node.js code, just run ` **npm install -g mocha`**. If you want to test code in the browser, run `**npm install mocha chai --save-dev`.**

**Chai** is the assertion library**.** With Mocha we actually have the environment for making our tests but how do we do test HTTP calls for example? Moreover, How do we test whether a GET request is actually returning the JSON file we are expecting, given a defined input? We need an assertion library, that's why mocha is not enough.So here it is Chai, the assertion library for it. To install chai use command, `**npm install -g chai`**.

Also, install virtual alexa, `**npm install virtual-alexa --save-dev`**

How to write test cases?

Every test case file follows the same basic pattern. First, you have a describe block:

describe('AnyName', function done() {

// Further code for tests goes here

});

describe is used to group individual tests.

Secondly, inside the describe, we’ll have it blocks:

describe('AnyName', function done() {

it('test\_name', async () => {

// Test implementation goes here

});

// We can have more its here

});

It is used to create the actual tests. The first parameter to it should provide a human-readable description of the test.

All Mocha tests are built from these same building blocks, and they follow this same basic pattern.

* First, we use describe to say what we’re testing – for example, “describe conversation flow”.
* Then, we use a number of it functions to create the individual tests – each it should explain one specific behavior.

To test the test cases, run mocha folder\_name.js

For more details regarding chai assertion library go to https://www.chaijs.com

**If your skill is in English in region, then even your bespoken should also be in English in region. If you want to change the language you have to go the dropdown of the language, then add language. You will get a screen that have all languages, click your language and add. Then you have to add json and then build and save your skill.**

Create lambda function for alexa skill

Go to dxctechlabs, click onservices, then search lambda

Then create blueprints. Then select alexa-skill-kit-sdk-factskill. Then configure. Give give name for the lambda. In role, choose an existing role.

Existing role- lambda\_exec\_role