Phase 4

Testing and deploying the application.

Phase 4

Testing tools.

Jasmine and Karma

Mocha and Chai

Superset testing tool

JEST

Graph QL : Self learning

Docker

Jenkin

Cloud computing

AWS Cloud : EC2, S3 EBS Volume Etc

**Day 1**

Input

Process

Output

MVC

App.js Router, controller, model database

If we want to test controller functionality

Controller must be call from router and router must be call from App.js. We can test the application using some testing tool which verify the application running properly or not.

Testing is use to find the error or defect or bugs in the application or software.

function chckUser(name,pass) {

if(name!=”Raj” && pass !=”123”) {

return “success”;

}else {

return “success”;

}

}

Testing: Testing mainly divided into two types

Black box testing

Input ---🡪 Process -----🡪 Output

A =10 display value

B=20 B is 20

Actual

Expect the output : 30

White box testing

Input---------🡪Process-------------🡪Output

A=10

B=20 coding output put 30

Actual

Expected

Both must be match

Manual testing : we run the testing and manually write the result in testing template.

Automation testing : tool generate the reports. Less interaction from programmer or tester for the loop.

Unit testing: unit testing is testing which help to do the testing for smallest piece for code that can logically verified. Unit also known as function/ method/ class/ modules etc.

Unit is testing is a type of white box testing.

Unit testing is a kind of software testing method is which each individual and independent part of the source code is tested to determine that it is good enough for the user.

When we perform unit testing on the front end (client side) of the software. It is called or known as front end unit testing.

Jasmine : Jasmine is a open source testing framework which help to do the unit testing on front end side technologies.

Using jasmine we can do Client side as well as server side(node JS) testing.

Jasmine provide pre-defined functions which help to do the testing.

1. **Test suite:** Test suite is technically contains more than one test case as well as another test suite. To write test suite testing framework provide pre-pre-definedfunction ie describe.

**Syntax**

describe(“suiteName”,callbackfucntion)

describe(“message”,()=> {

})

1. **Test spec :** Test spec is known as test case which actually help to do the testing. Test spec contains more than one expect which help to check actual and expected output.

Syntax

it(“spectName”,callback)

it(“addition”,()=> {

})

1. **expect :** this construct aids in testing if the expectation from the software are fulfilled or not.

**Testing hook or life cycle functions.**

**beforeEach() : it call again and again before every it functions.**

**afterEach() : it call again and again after every it function.**

**beforeAll() : it call only once before all it functions.**

**afterAll() : it call only once after all it function.**

**fetch() :**

**in react js using axios we call rest API.**

**fetch() is a pre-defined function provide by JavasScript which help to call REST API without depends upon any third party library.**

**Fetch() function return type is promise. Then and catch()**

**Fetch is a pre-defined function part of JavaScript we can use fetch function in Normal JavaScript code.**

**In Fetch we have to use first then() function convert data in json or text format mandatory.**

**Axios is third party library we have to install using node js and we can use in react js application. But in axis it return by default json.**

**Node JS (Server side JavaScript Testing) using jasmine tool.**

**Create folder Server Side JavaScript (Node JS)**

**Create the package.json file**

**Using npm init command**

**We have to install two external modules**

**ie**

**testing dependencies always must be available in development mode.**

**npm install jasmine-node -D**

**npm install jasmine -D**

**or**

**npm install jasmine-node –-save-dev**

**npm install jasmine –save-dev**

**after added dependencies**

**create spec directory using**

**jasmine init command**

**if you get the error jasmine command not recognize.**

**npm install jasmine –g**

**now create src folder**

**Express JS Testing**

**Jasmine with external node Js module ie superTest which help to do the testing for Express JS application.**

**First folder express js testing**

**Package.json file**

**Npm init**

**npm install express**

**npm install jasmine –D**

**npm install supertest –D**

**jasmine init : This command is use to create spec folder**

**create folder src**

**Mocha : Mocha is a feature – rich test framework running on node js as well as client side JavaScript programs.**

**Mocha make asynchronous testing very simple.**

**Jasmine is a big framework has almost everything built into in including assertion/ expectation.**

**Mocha only provide test runner function like describe and it but depends upon third party module like node js assert module or other assertion modules like should.js expect.js or chai.js**

**Mocha with Chai :**

**Mocha with Node JS (node js assertion module)**

**Create folder mocha with node js**

**Create package.json file using npm init**

**npm install mocha –g**

**npm install mocha –D**

**Mocha allow you to use any assertion library you wish,**

**Should.js**

**Expect.js**

**Chai.js**

**Chai.js : chai js is a assertion library which provide set of pre-defined functions which help to check expectation and actual output.**

**Chai with assertion style : The assert style is exposed through assert interface. This provides the classical assert-dot notation similar to assert module in node js.**

**Chai folder**

**Create assert style folder.**

**Create package.json file**

**npm install chai -D**

**Chai with expect style exposed through expect interface. In this scenario you can use natural language assertion.**

**Chai with expect style**

**Create expect style folder**

**Create package.json file**

**Npm install chai –D**

**Chai with should style**

**This style allow for the same chainable assertion as the expect interface. However it extends each object with a should property to start your chain.**

**Create should style folder**

**Create package.json file**

**npm install chai –D**

**Angular -🡪 Jasmine and Karma**

**Client Side JavaScript -🡪 jasmine or Mocha**

**React JS -🡪 JEST Testing framework**

**Node JS (Server sider JavaScript) -🡪Jasmine or Mocha with Chai with style.**

**Node JS with Mocha and Chai**

**Phase 4**

**02-09-2021**

**Create folder mocha with chai**

**Create package.json file**

**npm install mocha –D**

**npm install chai –D**

**create src folder and spec folder**

**Mocha Chai with Express JS Testing**

**Create folder mocha chai with express Js**

**Create package.json file using npm init**

**Npm install mocha –D**

**Npm install chai –D**

**Npm install chai-http –D it is like a supertest**

**Npm install express**

**Create two src and spec**

**JavaScript (Client Side ) ---jasmine**

**Node JS (Server side)**

**Jasmine with supertest for Express**

**Or**

**Mocha and chai with chai-http for express**

**React JS JEST**

**JEST is an open source testing framework build on JavaScript, designed majority to work with React and React Native application.**

**JEST JavaScript testing framework with focus on simplicity. JEST was created facebook.**

**JESt also provide testing runner function like describe, it and expect.**

**Create folder as JEST and insider JavaScript folder**

**Create package.json file**

**npm install jest –D**

**npm install jest –g**

**Create src folder and \_\_tests\_\_ (it is like a spec folder).**

**All testing file must be inside \_\_test\_\_ folder.**