**Q:1) Which testing tool is good for your project and why (write full description and give some example)?**

The tools used for testing in Angular are described below :

**1. Karma**

An open source framework developed and maintained by the GitHub community. This is the most compatible test runner for Angular known for testing on real devices. Karma provides continuous integration with various browsers with easy debugging directly from IDE.

It is a test runner for JavaScript. Along with Jasmine, Karma is one of the default testing tools for Angular. But nothing stops you from replacing Jasmine with another framework since Karma is testing framework agnostic. It’s also designed to offer simple integration with tools like Jenkins or Travis, allowing you to integrate it into your CI pipeline seamlessly.

Karma is a tool which lets us spawn browsers and run Jasmine tests inside of them all from the command line. The results of the tests are also displayed on the command line.

Karma can also watch your development files for changes and re-run the tests automatically.

Karma lets us run Jasmine tests as part of a development tool chain which requires tests to be runnable and results inspectable via the command line.

**2.Jasmine**

Jasmine is a Behaviour Driven Development (BDD) testing framework. Among its advantages, the main ones are that it doesn’t rely on other javaScript frameworks. It also doesn’t require DOM. One of the main goals of Jasmine is to provide an easy syntax, allowing you to write tests more easily. Jasmine is the default test framework used with Angular. It ships with Angular CLI by default.

**3.Jest**

Jest provides you with multiple layers on top of Jasmine and it is a very fast testing library that runs tests in parallel. It comes with minimum configuration setup, out of box mocking, and assertion support.

**4.Mocha**

Mocha is a feature-rich JavaScript test framework running on node.js and the browser, making asynchronous testing simple. Mocha tests run serially, allowing for flexible and accurate reporting while mapping uncaught exceptions to the correct test cases.

Mocha is a very flexible framework that gives the freedom to choose assertion, mocking libraries based on our requirements, and provides configurable test reporters.

**Example : Jasmine tests**

1) describe('Hello world', () => { (1)

it('says hello', () => { (2)

expect(helloWorld()) (3)

.toEqual('Hello world!'); (4)

});

});

2) describe('Hello world', () => {

let expected = "";

beforeEach(() => {

expected = "Hello World";

});

afterEach(() => {

expected = "";

});

it('says hello', () => {

expect(helloWorld())

.toEqual(expected);

});

});

**Q:2) Write five sample unit test cases in Karma or Jest.**

1)

describe('HelloComponent', () => {

test('should have a default name', () => {

expect(component.name).toBe('Hello');

});

});

2)

beforeEach(async(() => {

TestBed.configureTestingModule({

declarations: [ AppComponent ],

}).compileComponents();

}));

3)

it('should create the app', async(() => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.debugElement.componentInstance;

expect(app).toBeTruthy();

}));

4)

describe("Adding single number ", function () {

it("should add numbers",function() {

expect(nested.add(5)).toEqual(5);

expect(nested.add(5)).toEqual(10);

});

}

5)

describe("Finding a number in array",function () {

it("The Example of toContain() method",function () {

expect([1,2, 3, 4]).toContain(3);

});

});