Assignment-2

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

Jasmine is good for testing of the project beacause Jasmine is a JavaScript testing framework that supports a software development practice called Behaviour-Driven Development, or BDD for short. It's a specific flavour of Test-Driven Development (TDD). Jasmine, and BDD in general, attempts to describe tests in a human readable format so that non-technical people can understand what is being tested. However even if you *are* technical reading tests in BDD format makes it a lot easier to understand what's going on. Manually running Jasmine tests by refreshing a browser tab repeatedly in different browsers every time we edit some code can become tiresome. Karma is a tool which lets us spawn browsers and run Jasmine tests inside of them all from the command line.

karma is 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.

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. It's not necessary to know the internals of how Karma works. When using the Angular CLI it handles the configuration for us and for the rest of this section we are going to run the tests using only Jasmine.

Examples:

```
1)
  it('should create the app', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  }));
2)
  it(`should have as title 'angular-unit-test'`, async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app.title).toEqual('angular-unit-test');
  }));
```

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

```
1)
it("should be able to lower case a string",function() {
        expect(utils.toLowerCase).toBeDefined();
        expect(utils.toLowerCase("HELLO WORLD")).toEqual("hello world");
  });

2)
it("should be able to upper case a string",function() {
    expect(utils.toUpperCase).toBeDefined();
    expect(utils.toUpperCase("hello world")).toEqual("HELLO WORLD");
  });
```

3)

```
  it('should render title in a h1 tag', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('h1').textContent).toContain('Welcome to angular-
unit-test!');
  }));
```

4)

```
it('should truncate a string if its too long (>20)', () => {
    const pipe = new TroncaturePipe();
    const ret = pipe.transform('12345678901234567890123456789012345');
    expect(ret.length).toBeLessThanOrEqual(20);
  });
```

5)

```
it('should create', () => {

    const title = 'Hey there, i hope you are enjoying this article';
    const titleElement = element.querySelector('.header-title');
    component.title = title;
    fixture.detectChanges();
    expect(titleElement.textContent).toContain(title);
  });
```