Nunit + Jasmin +Karma Testing Assignment :- 2

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

- Karma testing tool is good for my project.

- Karma 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. Finally, you can easily control your workflow both from your terminal or your IDE. No need to run a standalone application or another tool.

- 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.

- Karma is a task runner for our tests. It uses a configuration file in order to set the startup file, the reporters, the testing framework, the browser among other things.

- Karma is a direct product of the AngularJS team from struggling to test their own framework features with existing tools. As a result of this, they made Karma and have transitioned it to Angular as the default test runner for applications created with the Angular CLI.

Write five sample unit test cases in Karma or Jest.

1 :-

```
it('should change title to Angular Training', async(() => {

 const fixture = TestBed.createComponent(AppComponent);
 fixture.nativeElement.querySelector('button').click();
 fixture.detectChanges();
expect(fixture.nativeElement.querySelector('h1').textContent).toEqual('Unit Test App');
}));
```

2 :-

```
 it("should create a post in an array", () => {
  const qouteText = "This is my new post";
service.addNewQuote(qouteText);
expect(service.quoteList.length).toBeGreaterThanOrEqual(1);
});
```

3 :-

```
it("should remove a created post from the array of posts", () => {
   service.addNewQuote("This is my new post");
   service.removeQuote(0);
 expect(service.quoteList.length).toBeLessThan(1);
});
```

4 :-

```
it("should create Quote component", () => {
  expect(component).toBeTruthy();
 });
```

5 :-

```
 it("should use the quoteList from the service", () => {
  const quoteService = fixture.debugElement.injector.get(QuoteService);
  fixture.detectChanges();
 expect(quoteService.getQuote()).toEqual(component.quoteList);
});
```