■ Angular Testing Guidelines & Best Practices (Deep Dive)

Testing in Angular is not just about coverage numbers—it's about ensuring stability, confidence, and maintainability across upgrades and refactors. This document provides detailed best practices, along with working code templates for components, services, and pipes.

1■■ Keep Tests Small and Deterministic

Each test should focus on one clear behavior. Use mocks and dependency injection to isolate what you are testing.

```
it('should emit search event when query is entered', () => {
  component.query = 'angular';
  component.search();
  expect(component.searchEvent.emit).toHaveBeenCalledWith('angular');
});
```

2■■ Use Angular TestBed Wisely

```
beforeEach(async () => {
  await TestBed.configureTestingModule({
    imports: [UserListComponent],
    providers: [{ provide: UserService, useValue: mockUserService }]
  }).compileComponents();
});
```

3■■ Follow AAA Pattern (Arrange-Act-Assert)

```
it('should calculate total correctly', () => {
  const cart = new CartService();
  cart.add({ price: 100 });
  cart.add({ price: 50 });
  const total = cart.getTotal();
  expect(total).toBe(150);
});
```

■ Code Templates

■ Component Test Template

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { UserListComponent } from './user-list.component';
import { UserService } from '../../services/user.service';
import { of } from 'rxjs';

describe('UserListComponent', () => {
  let component: UserListComponent;
  let fixture: ComponentFixture<UserListComponent>;
  let userServiceMock: any;

  beforeEach(async () => {
    userServiceMock = { getUsers: jest.fn().mockReturnValue(of([{ id: 1, name: 'John' }])) };
    await TestBed.configureTestingModule({
      imports: [UserListComponent],
      providers: [{ provide: UserService, useValue: userServiceMock }]
    }).compileComponents();

    fixture = TestBed.createComponent(UserListComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
```

```
  });

  it('should create component', () => {
    expect(component).toBeTruthy();
  });

  it('should load users on init', () => {
    expect(userServiceMock.getUsers).toHaveBeenCalled();
    expect(component.users.length).toBe(1);
  });
});
```

■ Service Test Template

```
import { TestBed } from '@angular/core/testing';
import { UserService } from './user.service';
import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';

describe('UserService', () => {
  let service: UserService;
  let httpMock: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [UserService]
    });
    service = TestBed.inject(UserService);
    httpMock = TestBed.inject(HttpTestingController);
  });

  it('should fetch users', () => {
    const mockUsers = [{ id: 1, name: 'Alice' }];

    service.getUsers().subscribe(users => {
      expect(users).toEqual(mockUsers);
    });

    const req = httpMock.expectOne('/api/users');
    expect(req.request.method).toBe('GET');
    req.flush(mockUsers);
  });

  afterEach(() => {
    httpMock.verify();
  });
});
```

■ Pipe Test Template

```
import { CapitalizePipe } from './capitalize.pipe';

describe('CapitalizePipe', () => {
  const pipe = new CapitalizePipe();

  it('should capitalize the first letter', () => {
    expect(pipe.transform('angular')).toBe('Angular');
  });

  it('should return empty string for null input', () => {
    expect(pipe.transform(null)).toBe('');
  });
});
```

■ Coverage Strategy

Metric	Target
Statements	≥ 90%
Branches	≥ 85%
Functions	≥ 90%
Lines	≥ 90%