**Software Sharks**
**Department of Computer Science**
**University of Pretoria**
Lynwood Road
Hatfield, Pretoria
0002
012 420 4111

# Testing Policies

**for**

# BISI - Image APP

**Version 2.0 Draft**

**Prepared by:**

| | |
|---|---|
| Mark Coetzer | u14044537 |
| Orisha Orrie | u13025199 |
| Tobias Bester | u14041368 |
| Len Bekker | u11026953 |
| Mukundi Matodzi | u16091265 |
| Jonathan Lew | u13318765 |

**On:**

**10th April 2018**

# Table of Contents

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide a general guide as to the style and practices used when implemented unit and end-to-end test cases.

## 1.2 Definitions, Acronyms and Abbreviations

Table 1: Definitions

| Term | Definition |
| --- | --- |
| e2e | End-to-End Test |

## 1.3 References

Karma Test Runner: https://karma-runner.github.io/1.0/index.html

Jasmine Test Framework: https://jasmine.github.io/2.4/introduction.html

## 2. Technologies

### 2.0 Environment

- **Name**: Angular
- **Version**: v5.2.9

### 2.1 Test Framework


- **Name**: Jasmine
- **Version**: v2.4.1

### 2.2 Test Runner


- **Name**: Karma
- **Version**: v2.0.0

# 3. Procedures

## 3.1 Setup

### Installation

The project has been setup to be tested with the Jasmine Test Framework via the Angular CLI. No external downloads are required after initial project setup.

### Run Tests

To build the app and launch it in watch mode with the Karma Test Runner:
Execute the following command:

```
ng test
```

### Configuration

The default configuration provided by the Angular CLI is acceptable for this projects test environment.

### Test File Names

Test file names follow the syntax:

```
<appName>.<component>.spec.ts
```

They are located in the same directory as the relative component (sibling file) with the file name:

```
<appName>.<component>.ts
```

### Test Locations'

Each test is located within its respective component directory.

i.e:

```
ss-imagerec-webapp > src > app > app.component.spec.ts
```

## 3.2 Test Examples

**Note**: All examples are taken directly from angular.io documentation.

### Basic Service

```
// Straight Jasmine testing without Angular's testing support
describe('ValueService', () => {
  let service: ValueService;
  beforeEach(() => { service = new ValueService(); });

  it('#getValue should return real value', () => {
    expect(service.getValue()).toBe('real value');
  });

  it('#getObservableValue should return value from observable',
    (done: DoneFn) => {
    service.getObservableValue().subscribe(value => {
      expect(value).toBe('observable value');
      done();
    });
  });

  it('#getPromiseValue should return value from a promise',
    (done: DoneFn) => {
    service.getPromiseValue().then(value => {
      expect(value).toBe('promise value');
      done();
    });
  });
});
```

### Basic Component Test

```
describe('LightswitchComp', () => {
  it('#clicked() should toggle #isOn', () => {
    const comp = new LightswitchComponent();
    expect(comp.isOn).toBe(false, 'off at first');
    comp.clicked();
    expect(comp.isOn).toBe(true, 'on after click');
    comp.clicked();
    expect(comp.isOn).toBe(false, 'off after second click');
  });

  it('#clicked() should set #message to "is on"', () => {
    const comp = new LightswitchComponent();
    expect(comp.message).toMatch(/is off/i, 'off at first');
    comp.clicked();
    expect(comp.message).toMatch(/is on/i, 'on after clicked');
  });
});
```

## HTTP Services

```
let httpClientSpy: { get: jasmine.Spy };
let heroService: HeroService;

beforeEach(() => {
  // TODO: spy on other methods too
  httpClientSpy = jasmine.createSpyObj('HttpClient', ['get']);
  heroService = new HeroService(<any> httpClientSpy);
});

it('should return expected heroes (HttpClient called once)', () => {
  const expectedHeroes: Hero[] =
    [{ id: 1, name: 'A' }, { id: 2, name: 'B' }];

  httpClientSpy.get.and.returnValue(asyncData(expectedHeroes));

  heroService.getHeroes().subscribe(
    heroes => expect(heroes).toEqual(expectedHeroes, 'expected heroes'),
    fail
  );
  expect(httpClientSpy.get.calls.count()).toBe(1, 'one call');
});

it('should return an error when the server returns a 404', () => {
  const errorResponse = new HttpErrorResponse({
    error: 'test 404 error',
    status: 404, statusText: 'Not Found'
  });

  httpClientSpy.get.and.returnValue(asyncError(errorResponse));

  heroService.getHeroes().subscribe(
    heroes => fail('expected an error, not heroes'),
    error  => expect(error.message).toContain('test 404 error')
  );
});
```