# Testing Policy
for
NINSHIKI

Version 0.4.0 - Demo #4 Draft

A detailed guide for the testing of the software

Compiled By

The Software Sharks Team
Coetzer MJ
Bester TJ
Orrie O
Lew J
Matodzi MC

Department of Computer Science
The University of Pretoria

For

Bramhope International School of Inovation

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

SOFTWARE
SHARKS

First release, April 2018

Contents

1 Introduction

This document describes the testing procedures used for the different subsystems of Ninshiki ("the system") from Software Sharks. The different subsystems are namely:

- Mobile App (Android)
- Web App (Angular)
- Backend (NodeJS and Python)

Following this, the specific unit tests written for each subsystem will be described for the respective subsystem, as well as the links to the location of test reports.

2 Testing procedures

2.1 General Testing Procedure

The development of the system follows the principles of Test-Driven Development. The general testing process is as follows:

1. For each use case, a test function is written that acts as a user completing a task.
2. Initially, all test functions fail. If they don't, they are rewritten so that they fail.
3. Starting from the highest priority use cases, the functional classes and programs that solve each use case are implemented, so that the respective test functions succeed.
4. This is repeated until all functions are implemented, all use cases are solved, and all test functions pass successfully.

This procedure is followed when writing test functions and test cases for each subsystem.

2.2 Travis CI

This process also makes use of continuous integration allowed by Travis CI:

1. Tests are run over the entire project automatically every time there is a push to the Travis CI branch.

2.3 Android specific testing

The Android testing procedure arranges the tests into three categories:

1. Small Tests: These comprise 70% of the overall tests and test each major component's functionality.
2. Medium Tests: These comprise 20% of the overall tests and integrate several components. They are run on real devices or emulators.
3. Large Tests: These comprise 10% of the overall tests and test integration and UI by running a UI workflow.

Small Tests are conducted in the development environment (Android Studio) and are done while functions are developed to ensure that certain functions work after others are added.

Medium Tests are conducted in the emulator, to make sure that different components work and interface together as expected. Different screen sizes and OS versions are emulated to ensure functionality across different devices.

Large Tests are conducted also in the emulator or a real device to ensure that the UI workflow is optimal and that the use cases are met.
Unit tests are written using Espresso.
Android Studio provides all the tools for each category of testing.

2.4 Angular Testing Procedure

There are two tools used in the Angular testing procedure: Jasmine and Karma.

Jasmine is a testing framework that allows for test functions to be written in Javascript that supports Test Driven Development. Jasmine allows for functions to be written in a low-level, easy-to-understand format. Multiple Javascript files can be run by a single Jasmine HTML file.

Karma integrates with Angular and Jasmine in that it runs a separate browser with all Jasmine tests running. This browser automatically updates after changes are made in the development environment, re-running the Jasmine tests.

These two tools, when used with Angular make automated testing simple and efficient.

2.5 NodeJS Testing Procedure

The framework used for NodeJS testing is Mocha. Mocha allows test functions to be written in JavaScript and run using a command.

Similar to Karma, nodemon is a tool that ensures as changes are made to the code, the Mocha tests are run again automatically.

2.6 Python Testing Procedure

Python by default includes a test module that allows for test functions to be written in the same file as the implementation. This is sufficient for function testing.

However for test cases and more complex tests, "pytest" is a standard tool that is full-featured and simple to use. Test files are written and then run using a command. Note that this is process.

A manual process is used in the system for the reason that Python is used to develop the image processing and classification aspect of the system. Testing and running this process is resource-heavy, especially when training the model. For this reason, testing is done manually after a certain milestone is reached instead of automatically after code is changed.

3 Unit & End-to-end Tests

This section describes the general format of unit and e2e tests and test reports for the respective subsystems.

3.1 Android Unit Tests

This section will be completed at a later date.

3.2 Angular Unit Tests

These unit tests ensure that the functionality and UI workflow of the web app are working.

The Angular subsystem is made up of components which each serve a function or provide a feature. For example, the Home component serves as a landing page of the web app, allowing users to quickly access the app's main pages.

For each component, a set of test functions are written and specified in the component's spec file. For example, the home component's spec file is named: home.component.spec.ts.

When 'ng test' is run, Angular uses Karma to run all spec files (for all the components) and displays all the results in a new browser window. The page in this browser window is saved as an HTML page called "Karma.html" in the "tests" directory, located here:
`https://github.com/OrishaOrrie/SoftwareSharks/tree/web/Application%20Source/tests`

When 'ng e2e' in the ss-image-rec folder, Angular makes use of Jasmine to drive through all the unit tests and generate a report.The e2e test directory is located here:
`https://github.com/OrishaOrrie/SoftwareSharks/tree/master/Application%20Source/ss-imagerec-webapp/e2e`

3.2.1 Test Cases

The following use cases and descriptions of their respective test functions are shown below:

1. Use Case: Capture an Image.

   (a) Test Function: If the "Webcam" button is clicked, then check if the webcam is opened.

2. Use Case: Select an Image from Storage
   (a) Test Function: If the "Upload" button is clicked, then check if the file explorer opens.
   (b) Test Function: Check that an image file is selected after the file explorer closes.
3. Use Case: Send Request
   (a) Test Function: Once the image is uploaded, check that a loading bar is displayed indicating a wait for response.
   (b) Test Function: Once the image is uploaded, check that a "Upload Successful" message is displayed.
4. Use Case: Receive Response
   (a) Test Function: Check that once an HTTP OK Status response is received, a list of classification results are returned
5. Use Case: Weight Analysis
   (a) Test Function: When a field is touched or dirtied, check that the "Item Weight" text is updated dynamically.
   (b) Test Function: Check that only numerical values can be entered in each field.
   (c) Test Function: Use mock values (float, integer, large, small) to test if the "Item Weight" calculation is correct.
   (d) Test Function: Check that negative values are not submittable.
6. Use Case: Contact Company
   (a) Test Function: After the email field is touched or dirtied, check that the validation function is run.
   (b) Test Function: When the submit button is clicked, check if the email send function is run.
   (c) Test Function: When the email function is run, check if an Http request is sent.

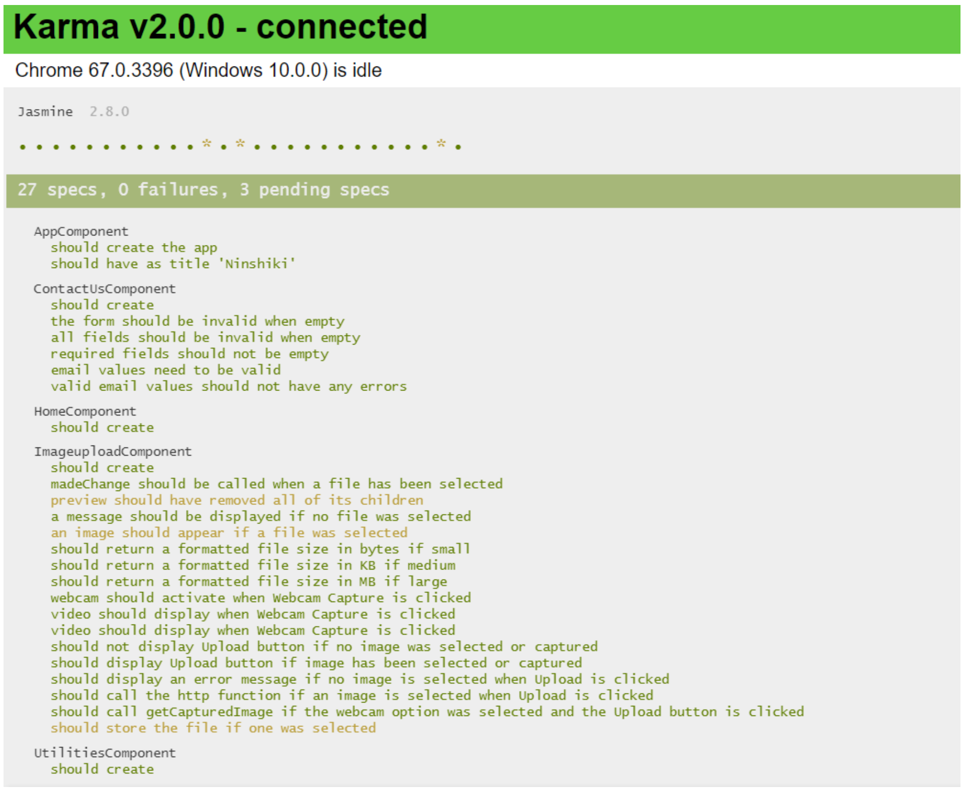
3.3 NodeJS Unit Tests

These unit tests ensure that the backend interface with the web app, the mobile app, and the Python image classification program as well as the implementation of the email service and image processing are working correctly.

Mocha test files are written testing the functionality of each written Javascript file used for the server. The command 'npm test' is used to run each test file, with the test results being displayed and saved in the same "tests" directory.

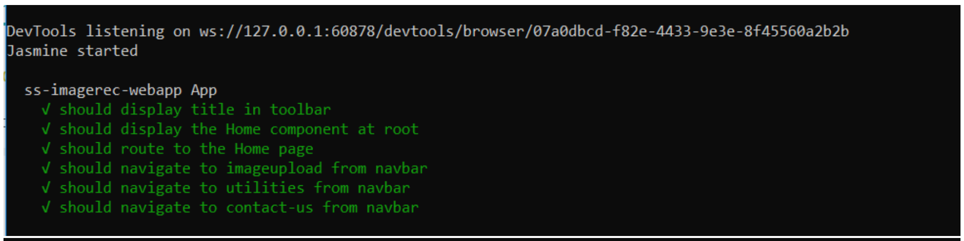
4 Result Outputs

4.1 Unit Tests: Karma

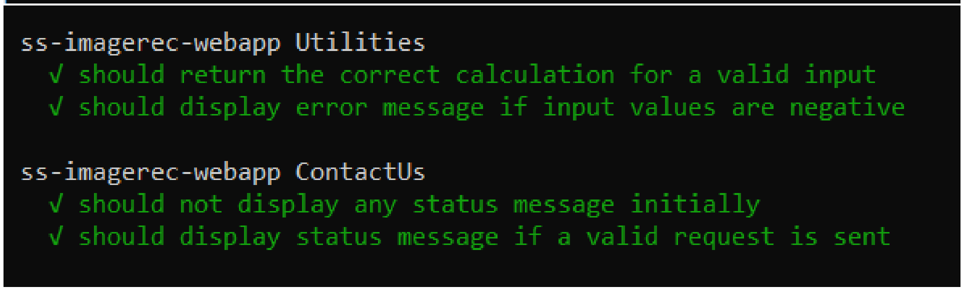


```
Karma v2.0.0 - connected
Chrome 67.0.3396 (Windows 10.0.0) is idle

Jasmine 2.8.0
•••••••••••*•*•••••••••••*•

27 specs, 0 failures, 3 pending specs

  AppComponent
    should create the app
    should have as title 'Ninshiki'
  ContactUsComponent
    should create
    the form should be invalid when empty
    all fields should be invalid when empty
    required fields should not be empty
    email values need to be valid
    valid email values should not have any errors
  HomeComponent
    should create
  ImageuploadComponent
    should create
    madeChange should be called when a file has been selected
    preview should have removed all of its children
    a message should be displayed if no file was selected
    an image should appear if a file was selected
    should return a formatted file size in bytes if small
    should return a formatted file size in KB if medium
    should return a formatted file size in MB if large
    webcam should activate when Webcam Capture is clicked
    video should display when Webcam Capture is clicked
    video should display when Webcam Capture is clicked
    should not display Upload button if no image was selected or captured
    should display Upload button if image has been selected or captured
    should display an error message if no image is selected when Upload is clicked
    should call the http function if an image is selected when Upload is clicked
    should call getCapturedImage if the webcam option was selected and the Upload button is clicked
    should store the file if one was selected
  UtilitiesComponent
    should create
```

4.2 End-to-end Tests: Jasmine

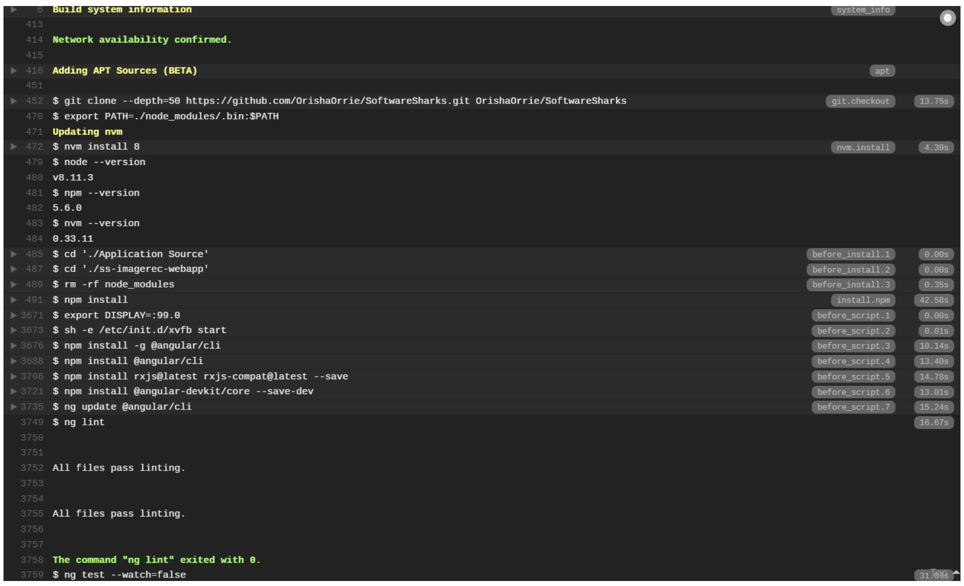


```
DevTools listening on ws://127.0.0.1:60878/devtools/browser/07a0dbcd-f82e-4433-9e3e-8f45560a2b2b
Jasmine started

  ss-imagerec-webapp App
    √ should display title in toolbar
    √ should display the Home component at root
    √ should route to the Home page
    √ should navigate to imageupload from navbar
    √ should navigate to utilities from navbar
    √ should navigate to contact-us from navbar
```

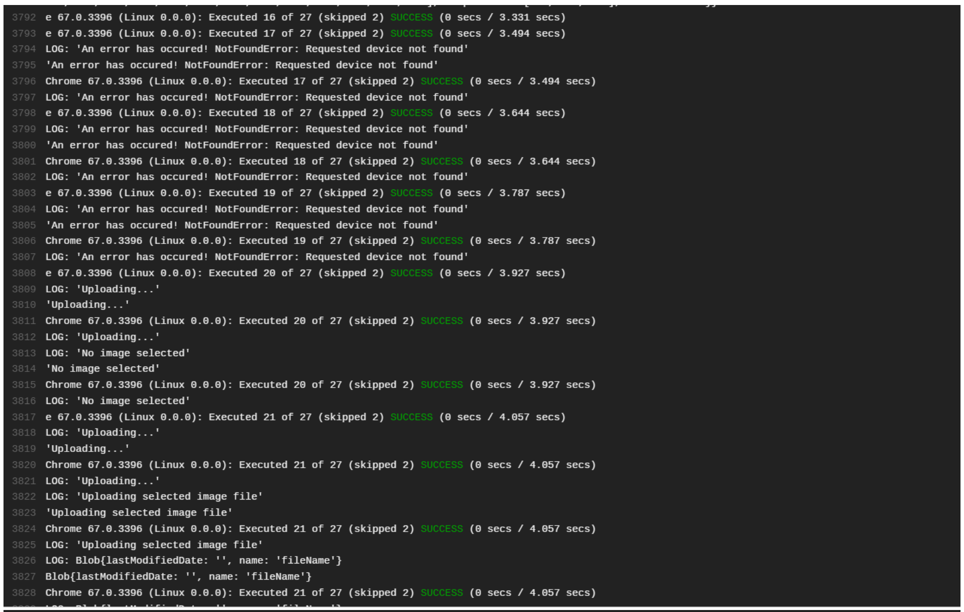


```
ss-imagerec-webapp Utilities
  √ should return the correct calculation for a valid input
  √ should display error message if input values are negative

ss-imagerec-webapp ContactUs
  √ should not display any status message initially
  √ should display status message if a valid request is sent
```

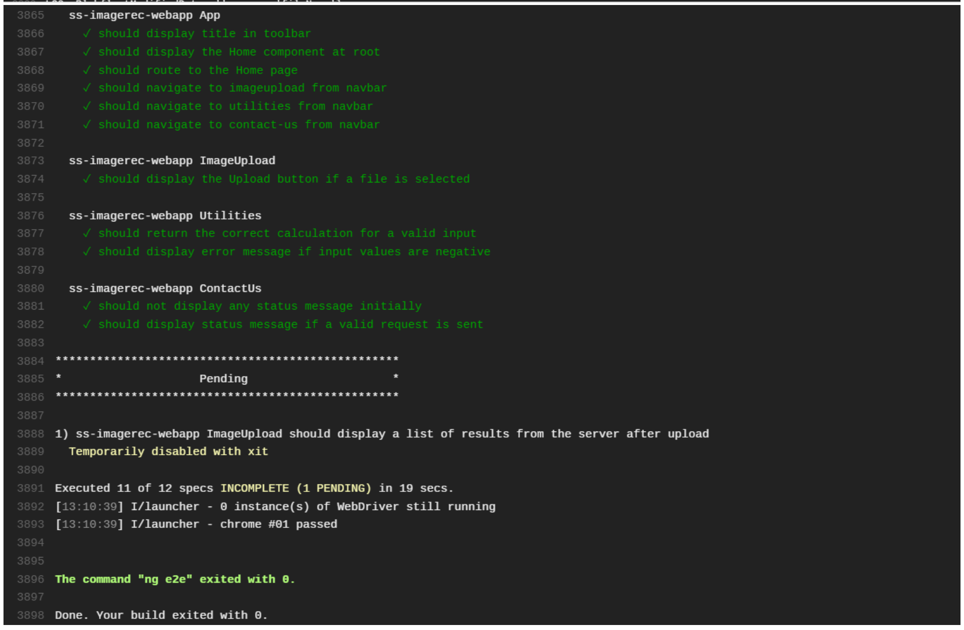
4.3 Continuous Integration: Travis CI



```
Build system information                                                         system_info

Network availability confirmed.

Adding APT Sources (BETA)                                                        apt

$ git clone --depth=50 https://github.com/OrishaOrrie/SoftwareSharks.git OrishaOrrie/SoftwareSharks   git.checkout   13.75s
$ export PATH=./node_modules/.bin:$PATH
Updating nvm
$ nvm install 8                                                                  nvm.install        4.39s
$ node --version
v8.11.3
$ npm --version
5.6.0
$ nvm --version
0.33.11
$ cd './Application Source'                                                      before_install.1   0.00s
$ cd './ss-imagerec-webapp'                                                      before_install.2   0.00s
$ rm -rf node_modules                                                            before_install.3   0.35s
$ npm install                                                                    install.npm        42.58s
$ export DISPLAY=:99.0                                                           before_script.1    0.00s
$ sh -e /etc/init.d/xvfb start                                                   before_script.2    0.01s
$ npm install -g @angular/cli                                                    before_script.3    10.14s
$ npm install @angular/cli                                                       before_script.4    13.40s
$ npm install rxjs@latest rxjs-compat@latest --save                              before_script.5    14.78s
$ npm install @angular-devkit/core --save-dev                                    before_script.6    13.01s
$ ng update @angular/cli                                                         before_script.7    15.24s
$ ng lint                                                                                           16.67s


All files pass linting.


All files pass linting.


The command "ng lint" exited with 0.
$ ng test --watch=false                                                                             31.07s
```

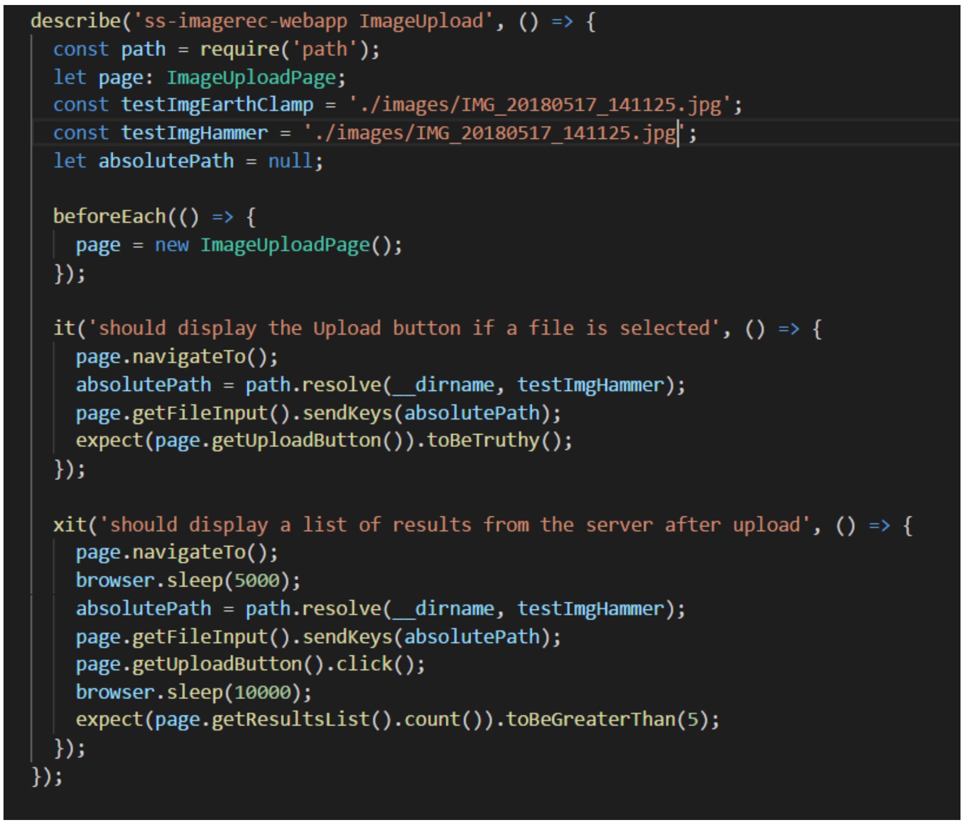


```
e 67.0.3396 (Linux 0.0.0): Executed 16 of 27 (skipped 2) SUCCESS (0 secs / 3.331 secs)
e 67.0.3396 (Linux 0.0.0): Executed 17 of 27 (skipped 2) SUCCESS (0 secs / 3.494 secs)
LOG: 'An error has occured! NotFoundError: Requested device not found'
'An error has occured! NotFoundError: Requested device not found'
Chrome 67.0.3396 (Linux 0.0.0): Executed 17 of 27 (skipped 2) SUCCESS (0 secs / 3.494 secs)
LOG: 'An error has occured! NotFoundError: Requested device not found'
e 67.0.3396 (Linux 0.0.0): Executed 18 of 27 (skipped 2) SUCCESS (0 secs / 3.644 secs)
LOG: 'An error has occured! NotFoundError: Requested device not found'
'An error has occured! NotFoundError: Requested device not found'
Chrome 67.0.3396 (Linux 0.0.0): Executed 18 of 27 (skipped 2) SUCCESS (0 secs / 3.644 secs)
LOG: 'An error has occured! NotFoundError: Requested device not found'
e 67.0.3396 (Linux 0.0.0): Executed 19 of 27 (skipped 2) SUCCESS (0 secs / 3.787 secs)
LOG: 'An error has occured! NotFoundError: Requested device not found'
'An error has occured! NotFoundError: Requested device not found'
Chrome 67.0.3396 (Linux 0.0.0): Executed 19 of 27 (skipped 2) SUCCESS (0 secs / 3.787 secs)
LOG: 'An error has occured! NotFoundError: Requested device not found'
e 67.0.3396 (Linux 0.0.0): Executed 20 of 27 (skipped 2) SUCCESS (0 secs / 3.927 secs)
LOG: 'Uploading...'
'Uploading...'
Chrome 67.0.3396 (Linux 0.0.0): Executed 20 of 27 (skipped 2) SUCCESS (0 secs / 3.927 secs)
LOG: 'Uploading...'
LOG: 'No image selected'
'No image selected'
Chrome 67.0.3396 (Linux 0.0.0): Executed 20 of 27 (skipped 2) SUCCESS (0 secs / 3.927 secs)
LOG: 'No image selected'
e 67.0.3396 (Linux 0.0.0): Executed 21 of 27 (skipped 2) SUCCESS (0 secs / 4.057 secs)
LOG: 'Uploading...'
'Uploading...'
Chrome 67.0.3396 (Linux 0.0.0): Executed 21 of 27 (skipped 2) SUCCESS (0 secs / 4.057 secs)
LOG: 'Uploading...'
LOG: 'Uploading selected image file'
'Uploading selected image file'
Chrome 67.0.3396 (Linux 0.0.0): Executed 21 of 27 (skipped 2) SUCCESS (0 secs / 4.057 secs)
LOG: 'Uploading selected image file'
LOG: Blob{lastModifiedDate: '', name: 'fileName'}
Blob{lastModifiedDate: '', name: 'fileName'}
Chrome 67.0.3396 (Linux 0.0.0): Executed 21 of 27 (skipped 2) SUCCESS (0 secs / 4.057 secs)
```

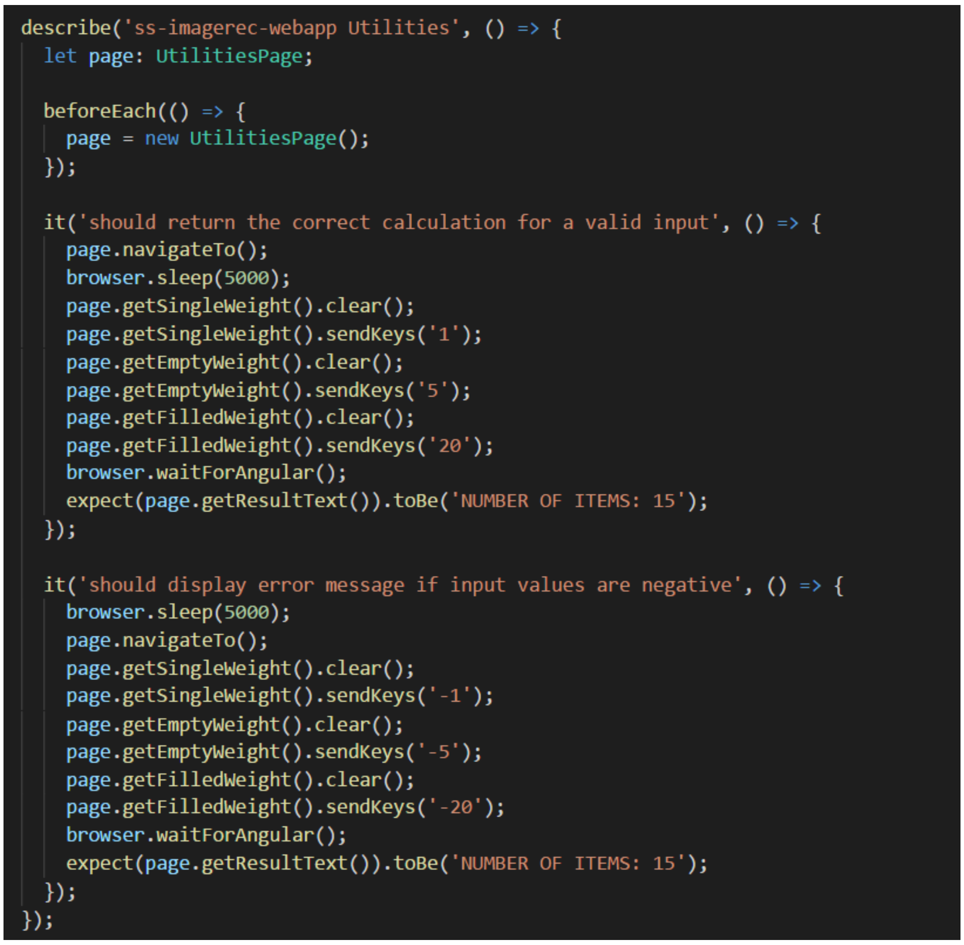


```
  ss-imagerec-webapp App
    ✓ should display title in toolbar
    ✓ should display the Home component at root
    ✓ should route to the Home page
    ✓ should navigate to imageupload from navbar
    ✓ should navigate to utilities from navbar
    ✓ should navigate to contact-us from navbar

  ss-imagerec-webapp ImageUpload
    ✓ should display the Upload button if a file is selected

  ss-imagerec-webapp Utilities
    ✓ should return the correct calculation for a valid input
    ✓ should display error message if input values are negative

  ss-imagerec-webapp ContactUs
    ✓ should not display any status message initially
    ✓ should display status message if a valid request is sent

**************************************************
*                    Pending                     *
**************************************************

1) ss-imagerec-webapp ImageUpload should display a list of results from the server after upload
  Temporarily disabled with xit

Executed 11 of 12 specs INCOMPLETE (1 PENDING) in 19 secs.
[13:10:39] I/launcher - 0 instance(s) of WebDriver still running
[13:10:39] I/launcher - chrome #01 passed


The command "ng e2e" exited with 0.

Done. Your build exited with 0.
```

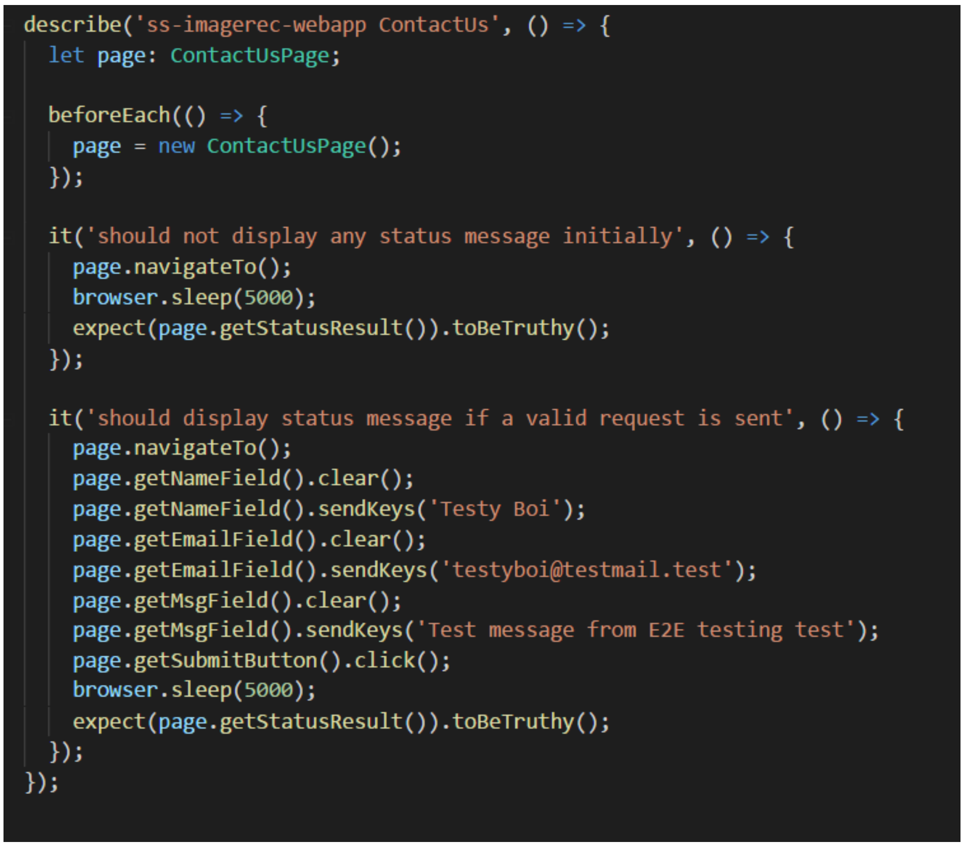
4.4 Test Case Examples



```
describe('ss-imagerec-webapp ImageUpload', () => {
  const path = require('path');
  let page: ImageUploadPage;
  const testImgEarthClamp = './images/IMG_20180517_141125.jpg';
  const testImgHammer = './images/IMG_20180517_141125.jpg';
  let absolutePath = null;

  beforeEach(() => {
    page = new ImageUploadPage();
  });

  it('should display the Upload button if a file is selected', () => {
    page.navigateTo();
    absolutePath = path.resolve(__dirname, testImgHammer);
    page.getFileInput().sendKeys(absolutePath);
    expect(page.getUploadButton()).toBeTruthy();
  });

  xit('should display a list of results from the server after upload', () => {
    page.navigateTo();
    browser.sleep(5000);
    absolutePath = path.resolve(__dirname, testImgHammer);
    page.getFileInput().sendKeys(absolutePath);
    page.getUploadButton().click();
    browser.sleep(10000);
    expect(page.getResultsList().count()).toBeGreaterThan(5);
  });
});
```



```
describe('ss-imagerec-webapp Utilities', () => {
  let page: UtilitiesPage;

  beforeEach(() => {
    page = new UtilitiesPage();
  });

  it('should return the correct calculation for a valid input', () => {
    page.navigateTo();
    browser.sleep(5000);
    page.getSingleWeight().clear();
    page.getSingleWeight().sendKeys('1');
    page.getEmptyWeight().clear();
    page.getEmptyWeight().sendKeys('5');
    page.getFilledWeight().clear();
    page.getFilledWeight().sendKeys('20');
    browser.waitForAngular();
    expect(page.getResultText()).toBe('NUMBER OF ITEMS: 15');
  });

  it('should display error message if input values are negative', () => {
    browser.sleep(5000);
    page.navigateTo();
    page.getSingleWeight().clear();
    page.getSingleWeight().sendKeys('-1');
    page.getEmptyWeight().clear();
    page.getEmptyWeight().sendKeys('-5');
    page.getFilledWeight().clear();
    page.getFilledWeight().sendKeys('-20');
    browser.waitForAngular();
    expect(page.getResultText()).toBe('NUMBER OF ITEMS: 15');
  });
});
```



```
describe('ss-imagerec-webapp ContactUs', () => {
  let page: ContactUsPage;

  beforeEach(() => {
    page = new ContactUsPage();
  });

  it('should not display any status message initially', () => {
    page.navigateTo();
    browser.sleep(5000);
    expect(page.getStatusResult()).toBeTruthy();
  });

  it('should display status message if a valid request is sent', () => {
    page.navigateTo();
    page.getNameField().clear();
    page.getNameField().sendKeys('Testy Boi');
    page.getEmailField().clear();
    page.getEmailField().sendKeys('testyboi@testmail.test');
    page.getMsgField().clear();
    page.getMsgField().sendKeys('Test message from E2E testing test');
    page.getSubmitButton().click();
    browser.sleep(5000);
    expect(page.getStatusResult()).toBeTruthy();
  });
});
```