UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

Unit Test Plan and Report

Client: Gavin Potgieter

Team: CodeBlox

Lethabo Mogase (Bsc: Computer Science)

Lorenzo Spazzoli (Bsc: Computer Science)

Bilal Muhammad (BIS: Multimedia)

Dirk de Klerk (BIS: Multimedia)

September 28, 2016

Contents

1 Introduction

1.1 Purpose

The main objective of this system is to allow a delivery person into a demarcated area of your house when you are not there. You should be able to give access remotely and monitor the delivery person while you they are in the area.

This document will demonstrate how the functionality of this system was tested by team codeBlox

1.2 Scope

The scope of this document is structured as follows. The features that are considered for testing are listed in section *INSERT SECTION NUMBER*. Tests that have been identified from the requirements are discussed in detail in section *INSERT SECTION NUMBER* Furthermore, this document outlines the test environment and the risks involved in the testing approaches that will be followed. Assumptions and dependencies of this test plan will also be mentioned. Section*INSERT SECTION NUMBER* outlines, discusses and concludes on the results of the tests, respective

1.3 Test Environment

- Programming Language
  - node js
  - Angular js
  - Python
  - Java (android)
- Coding Environment
  - Node package envrionment (npm)
- Operating system
  - Linux
- Hardware
  - Hardware testing is done on a level of using a voltameter
  - Raspberry Pi 3

1.4 Assumption and Dependencies

- assume that user has android device and is able to use it
- assume that the Pi 3 and camera have been setup in the house

1.4.1 Dependencies

- angular
- bcrypt
- body-parser
- express
- jwt-simple
- mongojs
- mocha
- should
- supertest

all npm node-modules

2 Test Items

3 Functional Features to be Tested

- user registration
- notification when someone is at the gate
- open/close gate
- open camera

Feature ID	RDS source	Summary	Test Case ID
1	server.get('/')	Passed	001
2	server.post('/registration')	Passed	002
3	server.get('/returnUser/:email')	Passed	003
4	newUser.getActivationStatus()	Passed	004
5	newUser.getPin()	Passed	005
6	newUser.getStaus()	Passed	006

4 Test Cases

4.1 Test Case 1

Test case 1: connection to server
condition: open home page
objective: check if the Pi connects to the server
Input: web URL
Outcome: 200 status

4.2 Test Case 2

Test case 2: add user
condition: user should not exist
objective: check if user details get persisted to the database Input: first name, last name, id, email, password1, password2
Outcome: 200 status and user added to database

4.3 Test Case 3

Test case 3: getting user email
condition: user should exist
objective: check if user info can be accessed
Input: user name
Outcome: print user email

4.4 Test Case 4

Test case 4: user status
condition: user should exist and be active
objective: check if correct status returned is active
Input: end Activation Date
Outcome: active

condition2: user should exist and be inactive
objective: check if correct status returned is inactive
Input: end Activation Date
Outcome: inactive

4.5 Test Case 5

Test case 5: generating pin
condition: request pin
objective: generate pin for the back up system
input: getpin()
Outcome: 8-digit random pin

4.6 Test Case 6

Test case 6: pin status
condition: pin has been generated and un-used
objective: check if pin has not been used
input: getPinStatus()
Outcome: un-used

condition2: pin has been generated and used objective: check if pin has been used input: getPinStatus() Outcome: used

5 Item Pass Criteria

- The home raspberry pi should be running
- The raspberry pi should be connected to the internet
- The raspberry should have a connection to the server

6 Test Deliverables

7 Detailed Test Results

7.1 Overview of Test Results

All the tests that were carried out passed and meet the expected results. These tests were carried out using the Mocha framework along with an assertion library called Chai.

7.2 Functional Requirements Test Results

Two separate tests were carried out to ensure that everything works. The first set of tests were done over the internet to ensure connection to server and correct communication

with the server. The second set of tests, tested the functionality of the individual functions in the module locally. The tests are on Github at :https://github.com/billibongers/CodeBlox—Main-Project/tree/master/Code/Interfaces/tests and https://github.com/billibongers/CodeBlox—Main-Project/tree/master/Code/NodeJs/Services/personModule/test

7.2.1 Test Case 1 (4.1)

The website opened and the video stream started
Result
Pass

7.2.2 Test Case 2 (4.2)

New user was added to the system
server returned status code of 200
Result
Pass

7.2.3 Test Case 3 (4.3)

User email was returned
The data was returned in the correct format
Result
Pass

7.2.4 Test Case 4 (4.4)

condition 1 returned active and condition 2 returned inactive
Result
Pass

7.2.5 Test Case 5 (4.5)

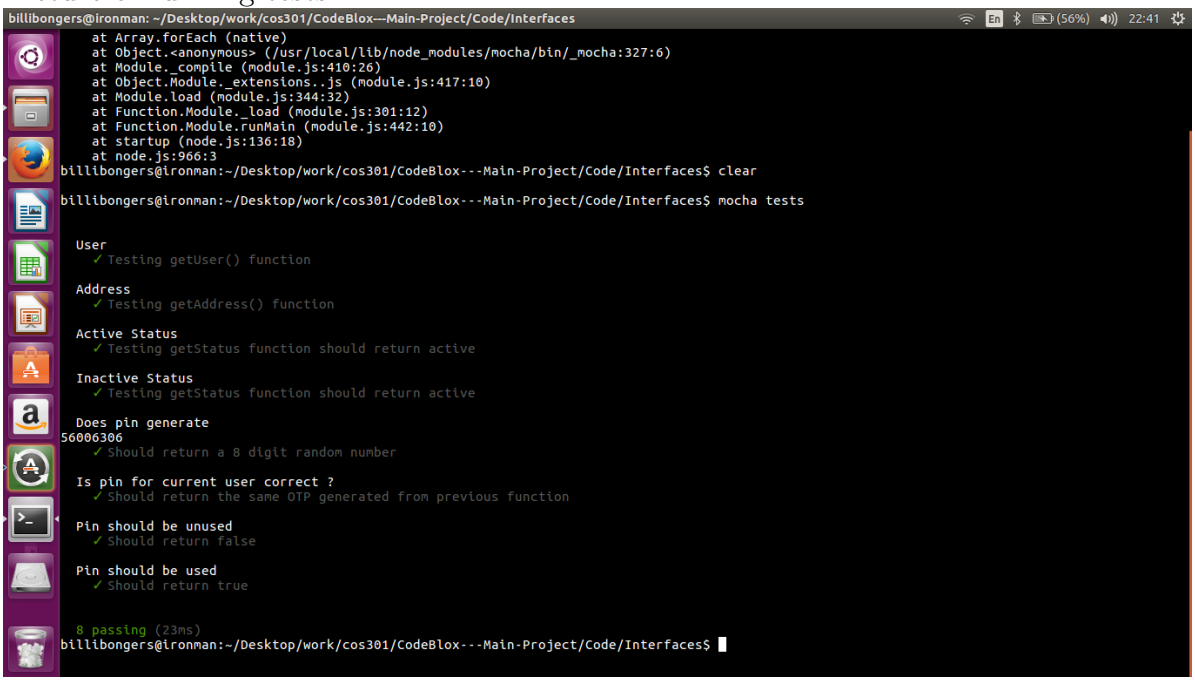
Random 8-digit pin was created the pin was returned in the correct format
pin was linked to correct user
Result
Pass

7.2.6 Test Case 6 (4.6)

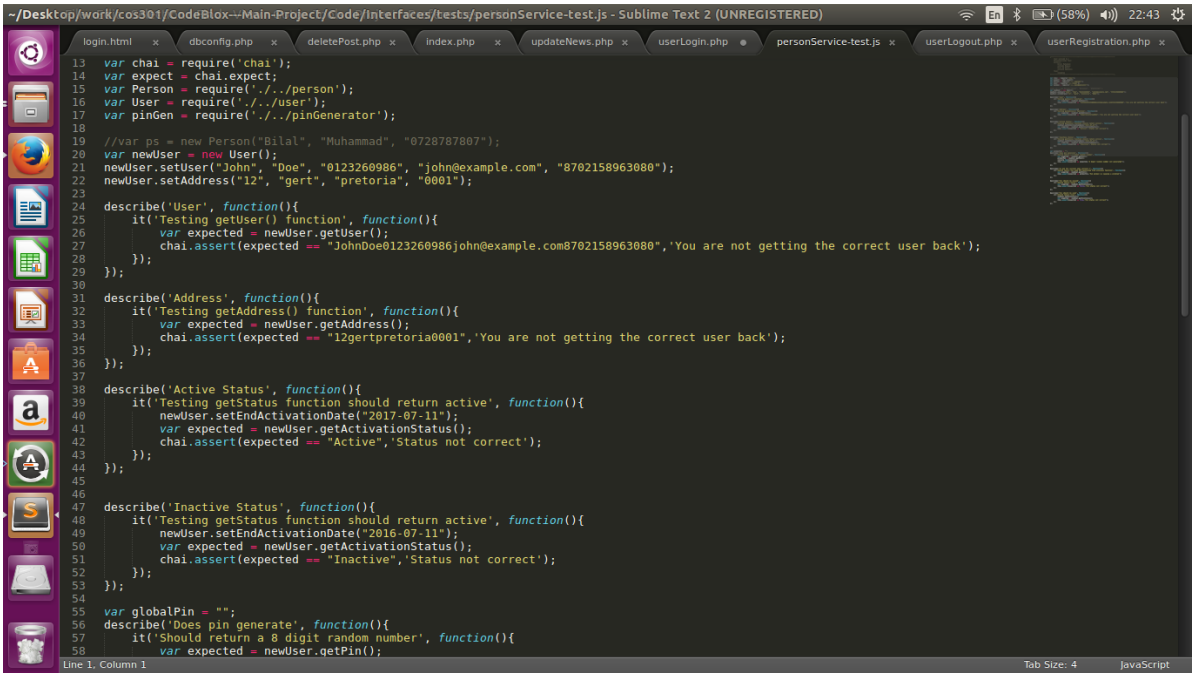
Condition 1 returned un-used status and condition 2 returned used status
Result
Pass

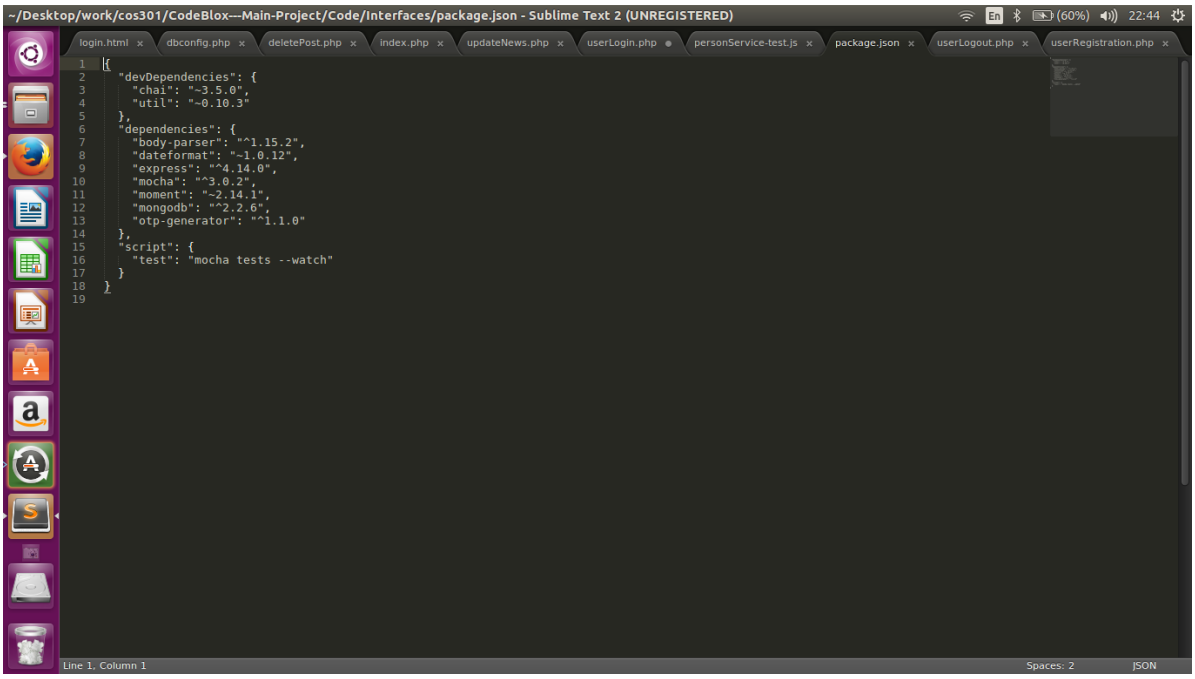
8 Other

Picture of running tests



picture of unit test code


~/Desktop/work/cos301/CodeBlox---Main-Project/Code/interfaces/tests/personService-test.js - Sublime Text 2 (UNREGISTERED)
login.html
dbconfig.php
deletePost.php
index.php
updateNews.php
userLogin.php
personService-test.js
userLogout.php
userRegistration.php
var chai = require('chai');
var expect = chai.expect;
var Person = require('./../person');
var User = require('./../user');
var pinGen = require('./../pinGenerator');
//var ps = new Person("Bilal", "Muhammad", "0728787807");
var newUser = new User();
newUser.setUser("John", "Doe", "0123260986", "john@example.com", "8702158963080");
newUser.setAddress("12", "gert", "pretoria", "0001");
describe('User', function(){
it('Testing getUser() function', function(){
var expected = newUser.getUser();
chai.assert(expected == "JohnDoe0123260986john@example.com8702158963080",'You are not getting the correct user back');
});
});
describe('Address', function(){
it('Testing getAddress() function', function(){
var expected = newUser.getAddress();
chai.assert(expected == "12gertpretoria0001",'You are not getting the correct user back');
});
});
describe('Active Status', function(){
it('Testing getStatus function should return active', function(){
newUser.setEndActivationDate("2017-07-11");
var expected = newUser.getActivationStatus();
chai.assert(expected == "Active",'Status not correct');
});
});
describe('Inactive Status', function(){
it('Testing getStatus function should return active', function(){
newUser.setEndActivationDate("2016-07-11");
var expected = newUser.getActivationStatus();
chai.assert(expected == "Inactive",'Status not correct');
});
});
var globalPin = "";
describe('Does pin generate', function(){
it('Should return a 8 digit random number', function(){
var expected = newUser.getPin();
Line 1, Column 1
Tab Size: 4
JavaScript


~/Desktop/work/cos301/CodeBlox---Main-Project/Code/interfaces/package.json - Sublime Text 2 (UNREGISTERED)
login.html
dbconfig.php
deletePost.php
index.php
updateNews.php
userLogin.php
personService-test.js
package.json
userLogout.php
userRegistration.php
{
"devDependencies": {
"chai": "~3.5.0",
"util": "~0.10.3"
},
"dependencies": {
"body-parser": "^1.15.2",
"dateformat": "~1.0.12",
"express": "^4.14.0",
"mocha": "^3.0.2",
"moment": "~2.14.1",
"mongodb": "^2.2.6",
"otp-generator": "^1.1.0"
},
"script": {
"test": "mocha tests --watch"
}
}
Line 1, Column 1
Spaces: 2
JSON

9 Conclusions and Recommendations

We have gained much insight in terms of testing and its uses. We switched to a testing strategy called Test Driven Development, hereto referred as TTD. With TTD, we are able to write tests without any code written to pass it. This enables the user to write unbiased code because we have no knowledge of the inner code.

We have also learnt to write tests that test a function in depth. This was achieved by using a special library called chai that is used in conjunction with mocha. It has a vast library of asserts that can be used.

Overall, we are happy with our progress as far as testing is concerned. After completion, if time allows, we wish to carry out a usability test.