Δημοκρίτειο Πανεπιστήμιο Θράκης
ΕΙΔΙΚΟΣ ΛΟΓΑΡΙΑΣΜΟΣ ΚΟΝΔΥΛΙΩΝ ΕΡΕΥΝΑΣ

Democritus University of Thrace
SPECIAL ACCOUNT FOR RESEARCH FUNDS

Interreg
Greece-Bulgaria
eHealth Monitoring
European Regional Development Fund

EUROPEAN UNION

D4.3.2 APPLICATION/ SOFTWARE DEVELOPMENT

(Quality Assurance Manager)

«Testing report of the Healthcare Monitoring System»

Reporting period: 06/12/2018 – 05/05/2019

WP 4 Joint Monitoring System

project

IMPROVING HEALTHCARE ACCESS THROUGH A PERSONAL HEALTH MONITORING SYSTEM

Soultana – Anna Toumpalidou

November 2019

The project is implemented in the framework of INTERREG V-A "Greece-Bulgaria 2014-2020" CooperationProgramme and is co-funded by the European Regional Development Fund (ERDF) and by national funds of the countries participating in the Programme

http://www.ehealthmonitoring.eu/

Summary

The current report of the deliverable «Testing report of the Healthcare Monitoring System» is part of the project APPLICATION/ SOFTWARE DEVELOPMENT in the frame of WP 4 Joint Monitoring System of the overall project IMPROVING HEALTHCARE ACCESS TROUGH A PERSONAL HEALTH MONITORING SYSTEM, according to the contract (14/09/2018, Ref. No: 44956) that is being implemented under the INTERREG V-A Greece – Boulgaria 2014-2020 Programme.

The deliverable describes the methods, tools and the results of the testing procedure of the eHealth platform (and its subsystems) along with the necessary activities for the software updates and the Version Control that have been undertaken during 06/12/2018 – 05/05/2019.

Table of Contents

List of Figures

1 Testing

1.1 Methodology

Testing is aimed at extracting information about the quality of the product or service being tested. The tests also provide an objective and independent view of the software, allowing the company to understand implementation problems early and prevent risks. Testing techniques usually involve running a program or application to detect software bugs. Software testing examines the system with respect to the following properties:

- meets functional and non-functional requirements
- correctly responds to all possible inputs
- performs its functions in an acceptable time
- is reasonably easy to use
- can be installed and run in the intended environment
- achieves the result that its users want

Since the number of possible tests, even for simple systems, is practically infinite, the software control process is based on a selection strategy for specific tests, which will be feasible based on time and resources. Of course, the testing process is repetitive, as resolving an error can lead to other, more profound problems or even to create new ones.

The methodology focuses on two distinct types of testing a) black-box testing and b) instrumented testing as presented in the following figures.

TC			
	Client (web)		Checked
1	**Login**		
	Step 1:	Log in in Bioassist with a client account.	OK
	Step 2:	Homepage is displayed with Contacts, Account and Admin Panel options.	OK
	Step 3:	Get offline button is also displayed along with the Log out button.	OK
	Step 4:	The photo of the client is displayed on the left side of the page with the personal information he has provided.	OK
2	**Contacts**		
	Step 1:	Log in in Bioassist with a client account.	OK
	Step 2:	From the Homepage click on the Contacts menu.	OK
	Step 3:	The contacts page is displayed.	OK
	Step 4:	All contacts are divided into two categories, online and offline.	OK
	Step 5:	Counters next to each category are displayed correctly.	OK
	Step 6:	A phone icon is displayed next to each name.	OK
	Step 7:	If the contact is online the phone is green whereas if the contact is offline the phone is grey.	OK
3	**Contacts (edit)**		
	Step 1:	Log in in Bioassist with a client account and go to contact's page.	OK
	Step 2:	Click on a contact.	OK
	Step 3:	A drop down list is displayed with all the main information of the contact.Relationship,Phone, Address and the last called date.	OK
	Step 4:	An Edit and a Delete button are also displayed.	OK
	Step 5:	Click on edit.	OK
	Step 6:	The Edit contact fields are displayed ready to be editted.	OK
	Step 7:	Edit all editable fields and click Save.	OK
	Step 8:	The contact is displayed with the just inserted data.	OK
4	**Contacts (edit-erase all)**		
	Step 1:	Log in in Bioassist with a client's account and go to contact's page.	OK
	Step 2:	Click on a contact and then select to edit it.	OK
	Step 3:	Edit Contact's detail form is displayed.	OK
	Step 4:	Erase all fields and click Save.	OK
	Step 5:	An appropriate message is displayed that the user should select a valid first name.	OK

Figure 1: Test Cases List

The tests on the platform were performed by using the following technologies and tools:

- Mocha.js - https://mochajs.org
- Chai.js – https://www.chaijs.com
- Expect.js - https://github.com/Automattic/expect.js

```
const chai = require('chai');
const expect = require('chai').expect;

chai.use(require('chai-http'));

const app = require('../app.js');

describe('ha-api', function () {
    describe('Server Status', function () {
        it('returns the status', function (done) {
            chai.request(app) Agent
                .get( url: '/status') SuperAgentRequest
                .end( callback: function (err, res) {
                    expect(res).to.have.status( code: 200);
                    expect(res.body.status).to.equal( value: "SUCCESS");
                    expect(res.body.code).to.equal( value: 0);
                    done();
                });
        });

        it('Unknown page', function (done) {
            chai.request(app) Agent
                .get( url: '/unknown-page') SuperAgentRequest
                .end( callback: function (err, res) {
                    expect(res).to.have.status( code: 401);
                    expect(res.body.status).to.equal( value: "ERROR");
                    expect(res.body.code).to.equal( value: -900);
                    done();
                });
        });
    });
});
```

Figure 2: Sample Test Cases for REST APIs

```
$ npm install --no-optional
npm WARN deprecated circular-json@0.5.9: CircularJSON is in maintenance only, flatted is its successor.
npm notice created a lockfile as package-lock.json. You should commit this file.
added 380 packages from 830 contributors and audited 863 packages in 41.877s
found 0 vulnerabilities

$ npm test

> ha-api@18.1.0 test /builds/bioassist/heartaround/ha-api/app
> mocha --exit ./test/*

[2019-02-14T15:25:53.384] [INFO] default - DB connection: root@mysql:3306/heartaround
[2019-02-14T15:25:53.413] [INFO] default - Heartaround [ha-api - (dev)]
[2019-02-14T15:25:53.415] [INFO] default - Allowed HTTP Methods:  GET,PUT,POST,DELETE,OPTIONS
[2019-02-14T15:25:54.314] [INFO] default - AZURE: Initializing connection to Azure Blob Storage...


  ha-api
[2019-02-14T15:25:55.745] [INFO] default - REDIS: Connected to redis:6379
    Server Status
[2019-02-14T15:25:55.762] [INFO] default - REDIS: Ready
node_redis: Warning: Redis server does not require a password, but a password was supplied.
[2019-02-14T15:25:55.923] [INFO] default - ::ffff:127.0.0.1 - api-open "GET /status HTTP/1.1" 200 OK 113 23ms "-" node-superagent/3.8.3
      ✓ returns the status (104ms)
{
    "appversion": "ha-api - (dev)",
    "appcomments": "N/A",
    "code": -900,
    "status": "ERROR",
    "reason": "NOT-AUTHENTICATED",
    "message": "Login required",
    "locale": "en",
    "dev_errors": []
}
[2019-02-14T15:25:55.947] [INFO] default - ::ffff:127.0.0.1 - api-open "GET /unknown-page HTTP/1.1" 401 Unauthorized 166 3ms "-" node-superagent/3.8.3
      ✓ Unknown page


  2 passing (305ms)

Job succeeded
```

Figure 3: Test Cases Results for REST APIs

In addition, the Selenium tool was used for automated testing:

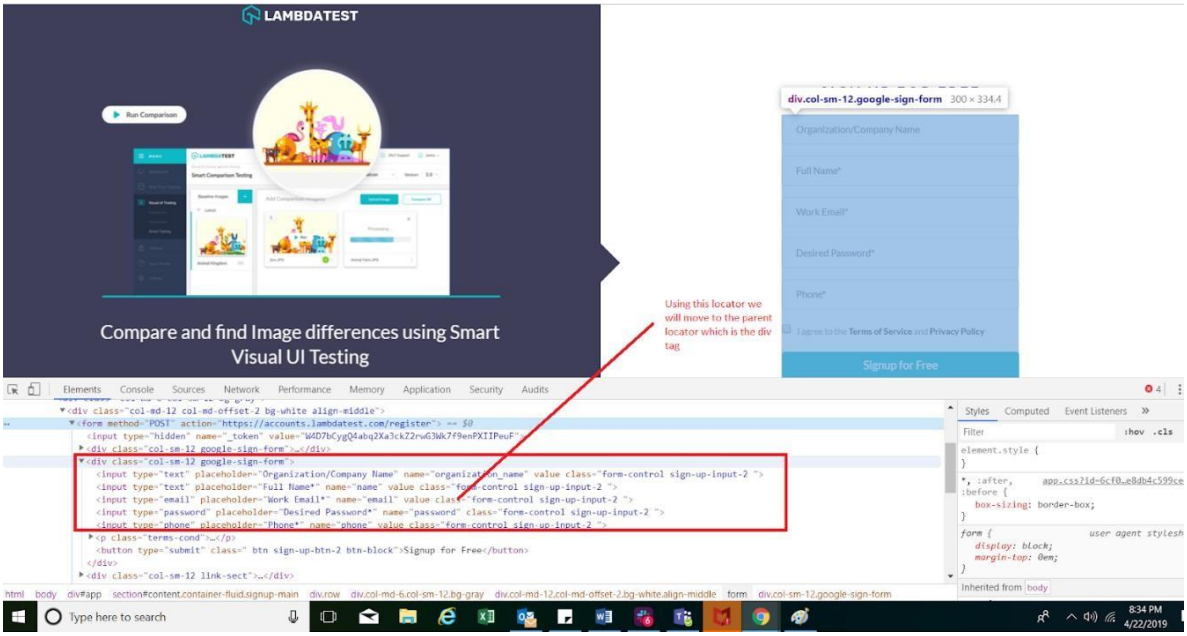

Figure 4: Automated testing with Selenium

In parallel, in the IDEs that was used for the development of the Android Smartphone application, several test have been designed and performed following the official guidelines from android.com

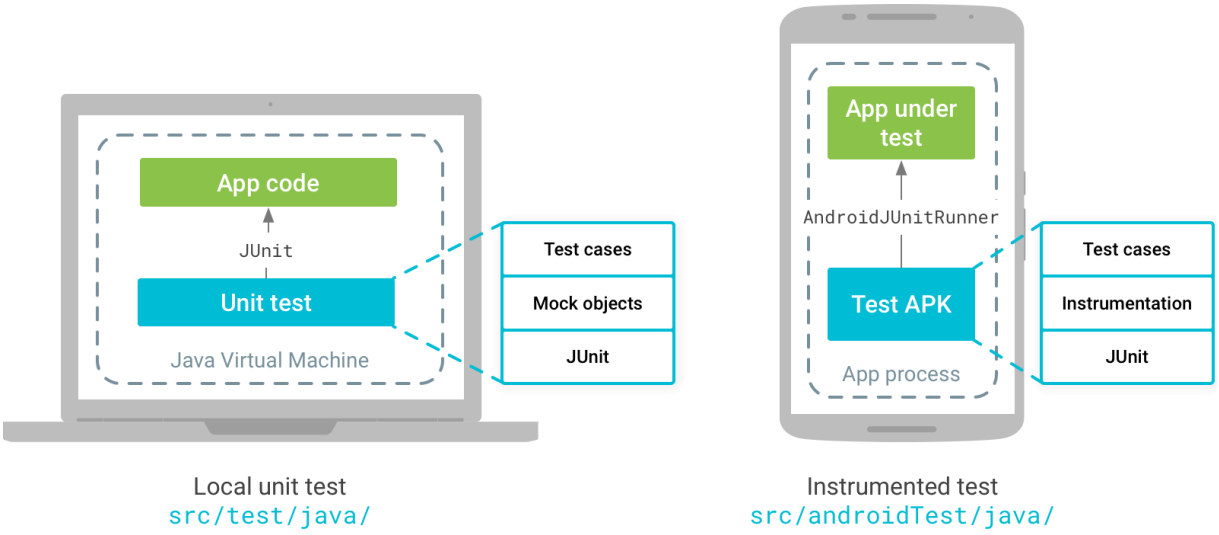

Figure 5: Unit και Instrumented Test in Android