A web application access for Jobs Observer program and verification

Friday 27th December, 2019 - 19:29

Stetsenko Olexandr
University of Luxembourg
Email: olexandr.stetsenko.001@student.uni.lu

Nicolas Guelfi
University of Luxembourg
Email: nicolas.guelfi@uni.lu

1. Introduction

The Bachelor Semester Project is focus to develop and to deploy a web application, called Jobs Observer, on an AWS cloud server. The web application to develop is based on an existed local software which purpose is to create statistics based on the actual jobs in Computer Science domain.
As, the project is in the domain of Software Engineering, a small research in the field of software testing is performed. The research consists to demonstrate the correctness of a program from the execution and behavior points of view.

2. Project description

The Bachelor Semester Project is realized in different domains of Computer Science such as Software Engineering, Web development and Software testing. 7
During this project several deliverables have to be produced which can be divided into scientific and technical deliverables.

2.0.1. Scientific deliverables. The scientific deliverables are the demonstrations of program correctness using logical reasoning and natural deduction. There are two demonstrations in total.
The first demonstration demonstrates the correctness of a program based on the correct execution. In other words, the result of a program has to be correct from the implementation point of view.
The second demonstration demonstrate the correctness of a program based on requirements from the behavior point of view.

2.0.2. Technical deliverable. The technical deliverable is an operational web application deployed on an AWS server. The web application, also called Jobs Observer, is composed of two parts which are the front-end and the back-end.
The front-end is the Graphical User Interface which allows the interaction between the user and the web application.
Back-end deliverable consists to develop a web-server by using Django python framework. This web-sever is used for creating statistics based on the data of jobs from the database. It is also used for deploying the Jobs Observer on an AWS server.

3. Pre-requisites

3.1. Natural Deduction

Natural Deduction[1] is an intuitional method which allows to develop a proof based on hypothesis. The proof is developed based on smaller proofs which are combine at the end based on the rules for providing a justification to an assumption or hypothesis. Each proof by deduction is a sequence of lines where each line of justification is based on any previous line.

3.2. Unit testing

Unit testing[2][3] is used for testing a program or some parts of the program. In unit testing the tester knows in advance what is the implementation of the program to test and can develop tests based on this knowledge. A unit test is successful when the output of the program is equal to the output of the test.

4. Production

In this Bachelor Semester Project, a web application and two demonstrations (ref. 6.1 and 6.2) of correctness of a program are developed.

4.1. First demonstration

The first demonstration is used for demonstrating the correctness of a program from the execution point of view.
Firstly, a program block is declared with the inputs, outputs and a function which is the implementation of the code. A program block is a part of the program to test which can be considerate as a function. A program is composed of multiple program block. Then, one or multiple tests are created for testing this program block.
As, the goal is to demonstrate the correctness of entire program, a set of tests is created and is composed of arrays of tests for each program block.
At the end, the correctness of the entire program can be tested by applying the set of tests on the program. As, all the tests can be applied on each program block, a validation

method is introduced. It consists to look at implementation of the program block and the test and if the both are the same then a test can be applied. For showing that the program block is successful with respect to a test, the output of the program block has to be the same as the output of the test. The entire program is correct if and only if all the tests are successful.

4.2. Second demonstration

The second demonstration is based on the requirements which allow to show correctness of a program from the behavior point of view. Additionally, the program has to be correct from the execution point of view.
Similarly, to the previously demonstration, a program composed of multiple program blocks and a set of tests composed of multiple arrays of tests are created.
Then, the creation of a requirement is performed by putting together two functions which are used for validating the inputs and the outputs of a program block or of a test. Then, an array of requirement is defined for each program block to test. All the requirements are assembled together into a set which is composed of arrays of requirements.
For showing that a requirement is satisfied, the inputs and outputs of a program block or of a test has to belongs to the requirement and the applied test has to be successful. For showing that the entire program is correct, all the requirements have to be satisfied and all the tests must be successful.

4.3. Web application

The developed web application is used for creating statistics based on the jobs in Computer Science domain. This web application is composed of front-end and back-end.
The front-end is the GUI which is used for interaction between the user and the application. It is composed of several HTML pages (ref. 6.3) which are the followed: "Home", "About Software", "Plots" and "Statistic". "Home" page is used as the principal view from where other pages can be accessed. "About Software" page gives a small description about the software. "Plots" is composed of a list of possible statistics which can be choose and displayed. The last "Statistic" page is the result of a statistic selected from the "Plots" page.
The back-end is created by using a Django framework which allows the creation of web applications. For this web application, the back-end is used for deploying the web application on an AWS server and generating statistics when they are requested from the front-end "Plots" page.

5. Conclusion

During this project, we described how the correctness of a program can be demonstrate by using the logical reasoning with the natural deduction. Moreover, we show that it exits different types of correctness of a program, in our case execution and behavior correctness.
Additionally, we saw some basis for developing a web application by using a Django framework and to deploy it on an AWS server. The web application developed uses a local software that generate statistics which allow to show the jobs in Computer Science.

References

[1] "3. Natural Deduction for Propositional Logic." 3. Natural Deduction for Propositional Logic - Logic and Proof 0.1 Documentation, https://leanprover.github.io/logic_and_proof/natural_deduction_for_propositional_logic.

[2] "Unit Testing." Software Testing Fundamentals, 3 Mar. 2018, http://softwaretestingfundamentals.com/unit-testing/.

[3] "Unit Testing Tutorial: What Is, Types, Tools, EXAMPLE." Guru99, https://www.guru99.com/unit-testing-guide.html.

6. Appendix

6.1. Demonstration of correctness of a program based on unit testing

6.1.1 Assumptions

Let P be a program to test. Assume that the program is syntactically valid.

6.1.2 Demonstrate that a program P is correct for a set of tests

6.1.2.1 Declaration of a program block

$Let\ inputs_{fpb}\ \equiv \{v \mid v\ is\ a\ value\}$

$Let\ func_{fpb}: inputs_{fpb} \ \rightarrow\ output_{fpb}, where\ func_{fpb}\ is\ a\ function$

$Let\ \ pb_i\ \equiv\ (func_{fpb}, inputs_{fpb}, output_{fpb})$

6.1.2.2 Declaration of all program blocks

$Let\ Blocks(P)\ \equiv \{\ pb_i \mid pb_i\ is\ a\ program\ block\ of\ P\}$

6.1.2.3 Declaration of a test

$Let\ inputs_{ft}\ \equiv \{v \mid v\ is\ a\ value\}$

$Let\ func_{ft}: inputs_{ft}\ \rightarrow\ output_{ft}, where\ func_{ft}\ is\ a\ function$

$Let\ \ t_j\ \equiv\ (func_{ft}, inputs_{ft}, output_{ft})$

6.1.2.4 Declaration of all test sets for P

$Let\ TA_{pb_i}\ \equiv\ [t_j \mid t_j\ is\ a\ test]$

$Let\ TS_P\ \equiv\ \bigcup_{pb_i \in Blocks(P)} TA_{pb_i}$

6.1.2.5 Show the validity and success of a test on a program block

$$success(t_j, pb_i)\ \equiv \begin{cases} 1, if\ [t_j]_{output_{ft}} = [pb_i]_{output_{fpb}} \\ \quad 0,\ otherwise \end{cases}$$

$$valid(t_j, pb_i)\ \equiv \begin{cases} 1, if\ [t_j]_{func_{ft}} = [pb_i]_{func_{fpb}}\ and\ [t_j]_{inputs_{ft}} = [pb_i]_{inputs_{fpb}} \\ \quad 0,\ otherwise \end{cases}$$

$valid(t_j, pb_i)\ \rightarrow\ success(t_j, pb_i)\ =\ 1$

6.1.2.6 Demonstrate that the whole program is correct

$correctBlock(TA_{pb_i}, pb_i)\ \equiv\ \forall t_j \in\ TA_{pb_i} \mid valid(t_j, pb_i)$

$correctP\big(TS_P, Blocks(P)\big) \equiv$

$\forall pb_i \in\ Blocks(P), \forall TA_{pb_i} \in TS_P \mid correctBlock(TA_{pb_i}, pb_i)$

6.2. Demonstration of correctness of a program from requirements point of view.

6.2.1 Considerate the previous parts 1.2.1 to 1.2.4 include as the first part of this proof.

6.2.2 Declaration of requirements

$Let\ condition\ \equiv\ [v \mid v\ is\ a\ value]$

$Let\ conditionsPre \equiv \{condition \mid condition\ is\ a\ set\ of\ values\}$

$Let\ conditionsPost \equiv \{condition \mid condition\ is\ a\ set\ of\ values\}$

$$Let\ pre-condition([t_j]_{inputs_{ft}}) \equiv \begin{cases} 1, \forall input \in [t_j]_{inputs_{ft}}, \exists\ condition\ \in\ conditionsPre \mid input\ \in\ condition \\ 0, otherwise \end{cases}$$

$$Let\ post-condition([t_j]_{output_{ft}}) \equiv \begin{cases} 1, \forall output \in [t_j]_{output_{ft}}, \exists\ condition\ \in\ conditionsPost \mid output\ \in\ condition \\ 0, otherwise \end{cases}$$

$Let\ r_i \equiv\ (pre-condition([t_j]_{inputs_{ft}}), post-condition([t_j]_{output_{ft}}))$

$Let\ RA_{pb_i}\ \equiv\ [r_i \mid r_i\ is\ a\ requirement]$

$Let\ RS_P\ \equiv\ \cup_{pb_i \in Blocks(P)} RS_{pb_i}$

6.2.3 Test satisfaction of a requirement

$$inputsEquality(t_j, pb_i)\ \equiv \begin{cases} 1, if\ [t_j]_{inputs_{ft}} = [pb_i]_{inputs_{fpb}} \\ 0, otherwise \end{cases}$$

$$outputsEquality(t_j, pb_i)\ \equiv \begin{cases} 1, if\ [t_j]_{output_{ft}} = [pb_i]_{output_{fpb}} \\ 0, otherwise \end{cases}$$

$$conditionInput(t_j, pb_i, r_i)\ \equiv \begin{cases} 1, if\ inputsEquality(t_j, pb_i) | [r_i]_{pre-condition(\ [pb_i]_{inputs_{fpb}})} \\ 0, otherwise \end{cases}$$

$$conditionOutput\ (t_j, pb_i, r_i) \equiv \begin{cases} 1, if\ outputsEquality(t_j, pb_i)\ | [r_i]_{post-condition([pb_i]_{output_{fpb}})} \\ 0, otherwise \end{cases}$$

$$satsify(t_j, pb_i, r_i)\ \equiv \begin{cases} 1, if\ conditionInput(t_j, pb_i, r_i)\ and\ conditionOutput\ (t_j, pb_i, r_i)\ and\ valid(t_j, pb_i) \\ 0, otherwise \end{cases}$$

6.2.4 A specific requirement is satisfied for a program block

$$Let\ satisfactionR(TS_{pb_i}, pb_i, r_i)\ \equiv \begin{cases} 1, if\ \forall t_j\ \in\ TS_{pb_i} |\ satsify(t_j, pb_i, r_i)\ \rightarrow\ (\exists t_j\ \in\ TS_{pb_i} | satsify(t_j, pb_i, r_i)\ =\ 1) \\ 0, otherwise \end{cases}$$

6.2.5 All requirements are satisfied for a program block

$$Let\ satisfactionSetR(TS_{pb_i}, pb_i, RS_{pb_i})\ \equiv \begin{cases} 1, if\ \forall t_j\ \in\ TS_{pb_i}, \forall r_i \in RS_{pb_i} |\ satisfactionR(TS_{pb_i}, pb_i, r_i)\ =\ 1 \\ 0, otherwise \end{cases}$$

6.2.6 All requirements are satisfied for a program

$Let\ satisfactionALL\ \equiv \forall pb_i \in\ Blocks(P), \forall TS_{pb_i} \in TS_P, RS_{pb_i} \in$
$RS_P \mid satisfactionSetR(TS_{pb_i}, pb_i, RS_{pb_i})\ =\ 1$

6.3 Web application Front-end

The Jobs Observer web application can be found on: http://ec2-18-215-248-225.compute-1.amazonaws.com/

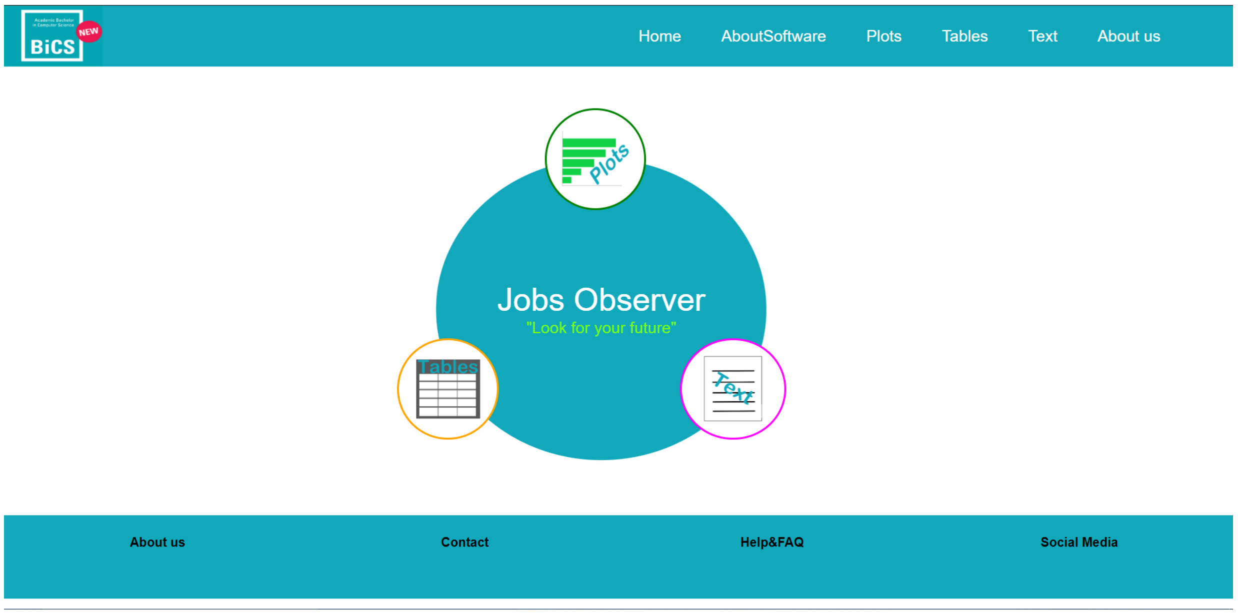
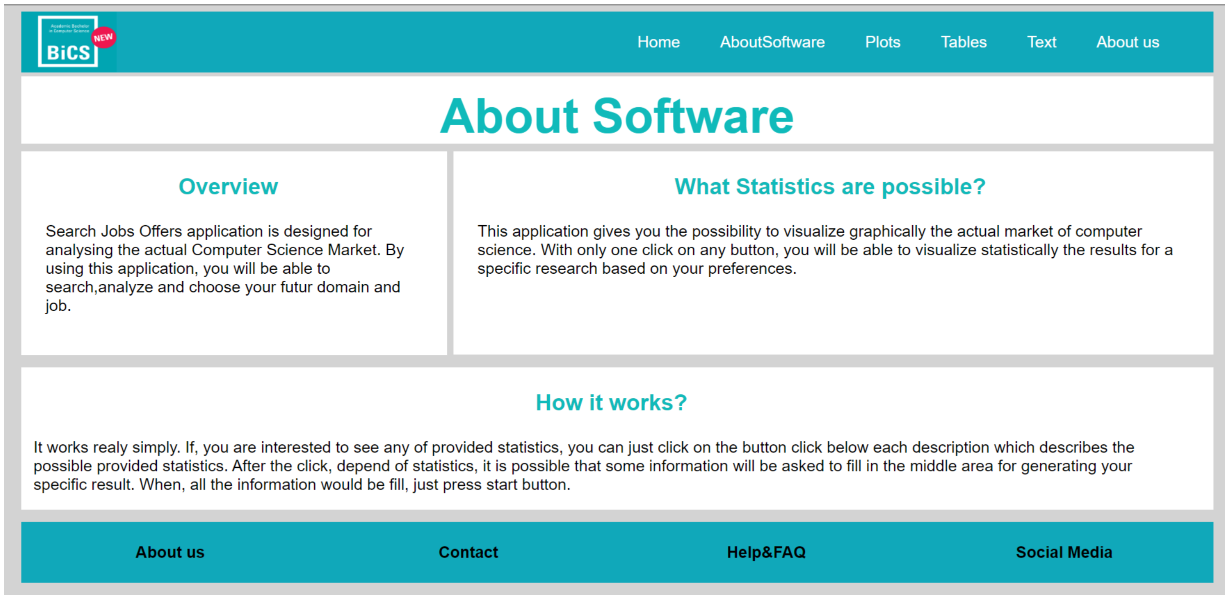


Figure 1. Home view of Jobs Observer web application



About Software

Overview

Search Jobs Offers application is designed for analysing the actual Computer Science Market. By using this application, you will be able to search,analyze and choose your futur domain and job.

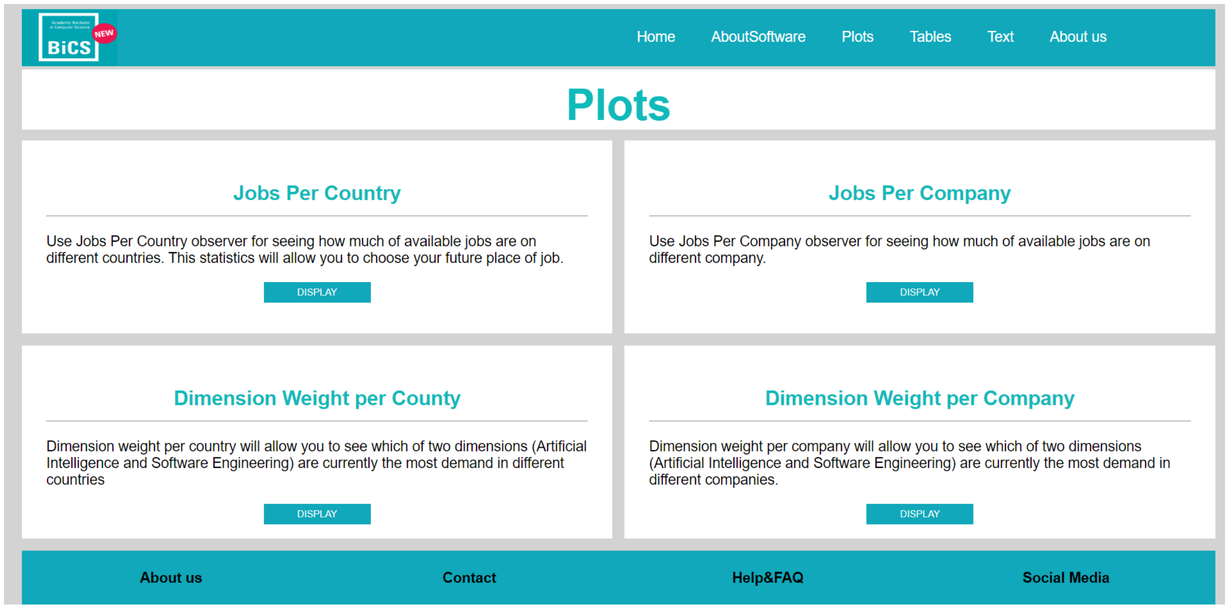
What Statistics are possible?

This application gives you the possibility to visualize graphically the actual market of computer science. With only one click on any button, you will be able to visualize statistically the results for a specific research based on your preferences.

How it works?

It works realy simply. If, you are interested to see any of provided statistics, you can just click on the button click below each description which describes the possible provided statistics. After the click, depend of statistics, it is possible that some information will be asked to fill in the middle area for generating your specific result. When, all the information would be fill, just press start button.

Figure 2. About Software page of Jobs Observer web application



BiCS NEW

Home AboutSoftware Plots Tables Text About us

Plots

Jobs Per Country

Use Jobs Per Country observer for seeing how much of available jobs are on different countries. This statistics will allow you to choose your future place of job.

DISPLAY

Jobs Per Company

Use Jobs Per Company observer for seeing how much of available jobs are on different company.

DISPLAY

Dimension Weight per County

Dimension weight per country will allow you to see which of two dimensions (Artificial Intelligence and Software Engineering) are currently the most demand in different countries

DISPLAY

Dimension Weight per Company

Dimension weight per company will allow you to see which of two dimensions (Artificial Intelligence and Software Engineering) are currently the most demand in different companies.

DISPLAY

About us Contact Help&FAQ Social Media

Figure 3. Plots page of Jobs Observer web application

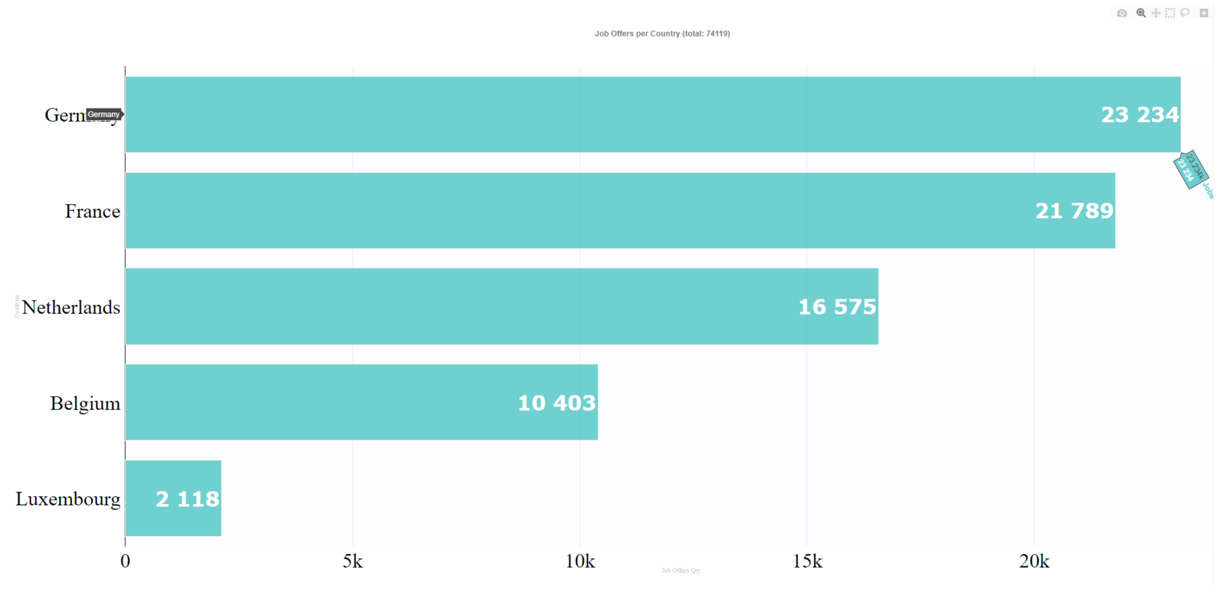


Figure 4. A statistic based on jobs per country