**Requirements specification**

***Automatically add missing data***

*8vance Matching Technologies BV*

*Venlo*

|  |
| --- |
| **Date : 25-02-2016** |
| **Version : 0.1** |
| **Status : Concept** |
| **Document name : Data exploration v0\_1** |
| **Author : Tim Hermens** |

**Version**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **Status** |
| 0.1 | 25-02-2016 | Tim Hermens | First concept. | Concept |
|  |  |  |  |  |

**Distribution**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Recipient** |
|  |  |  |

**Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date of approval** | **Name** | **Signature** |
|  |  |  |  |

**Glossary**

**KSC**

Knowledge, Competence, Skills. This is the data in a profile providing information about the person's competence and skills.

**Table of contents**

1. Introduction 5

1.1 Purpose 5

1.2 Goal of this document 5

1.3 Intended audience and reading suggestions 5

1.4 Project scope 5

1.5 References 6

2. Overall description 7

2.1 Product perspective 7

2.2 Product functions 7

2.3 User classes and characteristics 7

2.4 Operating environment 7

2.5 Design and implementation constraints 7

2.6 User documentation 8

2.7 Assumptions and dependencies 8

3. Requirements regarding the server-side algorithm functionality 9

3.1 Overall product description 10

3.2 Features 10

4. Requirements regarding the stand-alone application 11

4.1 Overall product description 12

4.2 Features 12

5. Non-functional requirements 13

5.1 Performance requirements 13

5.2 Safety requirements 13

5.3 Security requirements 13

5.4 Software quality attributes 13

5.5 Business rules 13

6. User interface wireframes 14

# Introduction

## Purpose

The company 8vance Matching Technologies BV uses scrape techniques to collect a large amount of data in a short time span. The problem is that this data often misses interesting information - a complete list of skills of persons in particular (this missing data is called [KSC](#KSC)). The [KSC](#KSC) data gives an overview what a person is capable of.

The company wants to have a product that can solve this problem (this product will be called product B1 from now on). Product B1 should be able to automatically add the missing [KSC](#KSC) data for all the profiles and must be integrated in the company's server. Product B1 is basically a piece of algorithm functionality for the server that will add data to the profiles, meaning it won't have any interaction with users.

The company also wants to have a product that can be used to analyze algorithms on correct input and output, and performance (this product will be called product B2 from now on). Product B2 is a general tool that should be able to analyze various different algorithms. There're three users for product B2:

* Programmers (algorithm developers). They'll want to use the product to test the algorithm on correct input and output, and performance. If the analysis overall negative, it's a sign the algorithm needs to be improved.
* Data scientists. They'll want to use the product to look for correct and incorrect predictions of the algorithm to find possible flaws they need to fix in their models.
* Other users. The previous two users are the main users of the product. There're also other users who don't want to use the product with a problem-solution mindset, but who're just interested what predictions the algorithm comes up with and how well it performs.

To summarize, the company wants to have the following two products:

* Product B1: Server-side algorithm functionality. An algorithm that supplements the missing [KSC](#KSC) data for profiles that can be integrated into the server as a compiled Python library.
* Product B2: Algorithm analysis tool. A stand-alone application to test various algorithms on correct input, output, and performance.

## Goal of this document

All of the requirements of these two products are discussed within this document. Every requirement in this document have a certain urgency and importance factor.

The urgency factor indicates how early the requirement needs to be implemented. Requirements with a high urgency need to be implemented first before everything else. Requirements with a medium urgency should be implemented as soon as no high urgency requirements are left. Requirements with a low urgency can be implemented as soon as no medium urgency requirements are left.

The importance factor indicates which requirements have higher priority than others. Requirements with a high priority indication need to be implemented no matter what, as soon as possible (this doesn't mean they have to be rushed!). Requirements with a medium priority also need to be implemented, but not necessarily straight away. Requirements with a low priority are things that would be nice to have (like an enhancement of the user experience/interface, additional features, etc.). This roughly translates to the MoSCoW notation as follows: high priority = must have, medium priority = should have, low priority = could have.

(a summary introducing the upcoming chapters)

## Intended audience and reading suggestions

(...)

## Project scope

(...)

In product B1, only the profiles of LinkedIn are supported to predict the missing [KSC](#KSC) data for. Whenever *profiles* are mentioned within this document, only the *LinkedIn profiles* are meant (read reference [R.1](#R1) why this decision was made).

## References

|  |  |
| --- | --- |
| **Reference code** | **Reference** |
| R.1 | Research document |

# Overall description

## Product perspective

Product B1, the server-side algorithm functionality, serves as an extension for the current profile matching algorithm, which aims to improve the overall profile matching results by adding missing [KSC](#KSC) data to the profiles. This product will be integrated in the server's Django framework.

Product B2, the algorithm analysis tool, is a new self-contained product. This product should work without a internet connection and doesn't have any interconnection with other existing products or external databases.

## Product functions

In this section, the major functions the product must perform or must let the user perform are briefly summarized. Details will be provided in chapter 3 and 4 for the different products separately.

### Product B1 - Server-side algorithm functionality

* The algorithm is able to predict missing skills for a selection of profiles.
* The algorithm can calculate a certainty score for every predicted skill per profile.

### Product B2 - Algorithm analysis tool

* The user can import an algorithm he wants to analyze.
* The user can provide a data source for the algorithm to use for the analysis.
* The user can save and load analysis results.
* The application can write returned values from the algorithm to a external target.
* The application can show analysis results of the algorithm such as the predictions, certainty scores of predictions, execution time, memory cost, and more.

## User classes and characteristics

Product B1 is a piece of algorithm functionality for the server that will add data to the profiles. It won't have any interaction with users.

Product B2 will mainly be used by the following users:

* Programmers (algorithm developers). They'll want to use the product to test algorithms on correct input and output, and performance. If the analysis is overall negative, it's a sign the algorithm needs to be improved. It's anticipated these users will use all of the product's functionality frequently.
* Data scientists. They'll want to use the product mainly to look for correct and incorrect predictions of the algorithm to find possible flaws they need to fix in their models. It's anticipated these users will use the product frequently.
* Other users. The previous two users are the main users of the product. There're also other users who don't want to use the product with a problem-solution mindset, but who're just interested what predictions the algorithm comes up with and how well it performs. It's anticipated these users only want to use the product's functionalities that allows them to see this information, occasionally.

## Operating environment

Product B1 will be integrated in the Django framework, which runs on a UNIX-like server.

Product B2 is a stand-alone tool which has to be able to run on any operation system, assuming the user has installed the required libraries used by the product. This product must be able to run on systems without any network connection.

## Design and implementation constraints

* The documentation of both products have to be written in English.

### Product B1 - Server-side algorithm functionality

* The algorithm has to work with huge amounts of data which can't all be read into memory. There has to be a solution to split and read the data in smaller parts.
* The algorithm has to be able to be integrated in the server's Django framework.
* The algorithm has to be able to predict missing skills of one profile in 1 second or less.
* Due to a limited amount of time and experience in data science, the algorithm will be written in Python 2.7 instead of possible better-performing alternatives. Python is also an ideal choice because Django is a Python framework.

### Product B2 - Algorithm analysis tool

* The tool has to function without an internet connection.
* The tool must be able to run on any operating system.

## User documentation

There'll be additional documentation provided for product B2. This documentation will contain information for all the users how the product can be used. So basically, the documentation will be a user manual.

## Assumptions and dependencies

### Product B1 - Server-side algorithm functionality

* The algorithm won't communicate with the server's Apache Cassandra database. All save and read operations are done outside of the algorithm.
* The input data type for the algorithm must have a JSON format.
* The input data for the algorithm must contain all the necessary data fields for a typical LinkedIn profile.

### Product B2 - Algorithm analysis tool

TODO

# Requirements regarding the server-side algorithm functionality

(This whole chapter will be rewritten. Features and rules will be separated and mentioned in different sections in this chapter. Every feature will have a separate description / motivation, list of functions, applicable rules, urgency / importance indications, and scenarios describing user actions. Non-functional requirements will be discussed in a separate chapter. This chapter will start with an overall description of how the product will work.)

The following table contains the requirements for product A.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Requirement description** | **Urgency** | | | **Importance** | | |
| **High** | **Medium** | **Low** | **High** | **Medium** | **Low** |
|  | The algorithm is created in Python. |  |  |  |  |  |  |
|  | The algorithm can be integrated in the server's existing Django framework. |  |  |  |  |  |  |
|  | The input data type for the algorithm must have a JSON format. |  |  |  |  |  |  |
|  | The input data must contain all the necessary data fields (specified in the research document) of the profiles. |  |  |  |  |  |  |
|  | The algorithm can parse the profile data of a selection of profiles to the wanted syntax. |  |  |  |  |  |  |
|  | The algorithm can write the parsed profile data of a selection of profiles to a external target (database/file). |  |  |  |  |  |  |
|  | The algorithm can predict missing skills for a selection of profiles. |  |  |  |  |  |  |
|  | The algorithm can write the predicted missing skills of a selection of profiles to a external target (database/file). |  |  |  |  |  |  |
|  | The algorithm can calculate a certainty score for every predicted skill in a profile. |  |  |  |  |  |  |
|  | The algorithm can calculate a certainty score for every skill for a profile. |  |  |  |  |  |  |
|  | The algorithm can determine whether or not user-specified skills are correct. |  |  |  |  |  |  |
|  | The algorithm can flag and exclude the incorrect user-specified skills. |  |  |  |  |  |  |
|  | The algorithm can write the flagged incorrect user-specified skills to a external target (database/file). |  |  |  |  |  |  |
|  | The algorithm can calculate an certainty score for every determination of whether or not a user-specified skill is correct or incorrect. |  |  |  |  |  |  |
|  | The algorithm can predict missing skills for every profile in 1 second or less. |  |  |  |  |  |  |

## Overall product description

Product B can be seen as a algorithm package, containing various different functions that return different values. The package will have a function that does some pre-processing of the input data and the resulting data is returned. This data can then be used to call a function that will return predictions for the missing skills per profile. This function has two additional parameters which indicate if you:

1. want it to also return certainty scores per prediction (for every *predicted* skill);
2. want it to also return certainty scores of every skill in the system.

It's important to know that this product will not communicate with a external target. So reading any data from a external data source or writing any data to a external target has to be done outside this product.

## Features

### Feature 1

#### Description and priority

(...)

#### Functions

(...)

#### Rules

(...)

#### User actions

(...)

### Feature 2

#### Description and priority

(...)

#### Functions

(...)

#### Rules

(...)

#### User actions

(...)

# Requirements regarding the stand-alone application

(This whole chapter will be rewritten. Features and rules will be separated and mentioned in different sections in this chapter. Every feature will have a separate description / motivation, list of functions, applicable rules, urgency / importance indications, and scenarios describing user actions. Non-functional requirements will be discussed in a separate chapter. This chapter will start with an overall description of how the product will work.)

The following table contains the requirements for product B.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Requirement description** | **Urgency** | | | **Importance** | | |
| **High** | **Medium** | **Low** | **High** | **Medium** | **Low** |
|  | The user can import an algorithm he wants to analyze. |  |  |  |  |  |  |
|  | The user can select/specify a data source for the algorithm. (it's important that the structure of the data source is the same as the one used to create the algorithm) |  |  |  |  |  |  |
|  | The application runs locally (without any network connection). |  |  |  |  |  |  |
|  | The user can select/specify a dump target which is used to dump algorithm-specific data (for instance, for pre-processing). |  |  |  |  |  |  |
|  | The user can save the algorithm analysis results. |  |  |  |  |  |  |
|  | The user can load and view previous algorithm analysis results. |  |  |  |  |  |  |
|  | If an algorithm is selected, a data source is specified and no analysis is running (see b8), the user can run an analysis of the algorithm. |  |  |  |  |  |  |
|  | The user can kill the analysis process at any given time. |  |  |  |  |  |  |
|  | The user can see the analysis' progress. |  |  |  |  |  |  |
|  | The user can run one analysis at a time. (heavy cpu load) |  |  |  |  |  |  |
|  | The application should be horizontally scalable in such a way that more visuals can be included for the analysis. (GUI tab control so more viewpoints of the analysis can be supported?) |  |  |  |  |  |  |
|  | The result of the analysis shows the execution time of the algorithm. (ticking clock when analysis starts and ends?, optional?) |  |  |  |  |  |  |
|  | The result of the analysis shows the cpu cost per line in the algorithm. (optional?) |  |  |  |  |  |  |
|  | The result of the analysis shows the memory cost per line in the algorithm. (optional?) |  |  |  |  |  |  |
|  | The result of the analysis shows the predictions. (predicted skills per profile, in a simple table) |  |  |  |  |  |  |
|  | The result of the analysis shows the certainty score per prediction. (predicted skill per profile, all skills per profile, certainty bar graph per profile?, overall certainty bar graph?, can be sorted) |  |  |  |  |  |  |
|  | The result of the analysis shows information about the algorithm's code structure (lines of code, comments, statements, classes, cyclomatic complexity) (optional?) |  |  |  |  |  |  |
|  | When the analysis is done, the user can search for the analysis results for a particular profile. |  |  |  |  |  |  |
|  | When viewing the analysis results of a particular profile, the user can search for a particular prediction (predicted skill / owned skill). |  |  |  |  |  |  |
|  | When an analysis is running, the user can't do any actions apart from killing the analysis process. (to save cpu load) |  |  |  |  |  |  |

## Overall product description

(...)

## Features

### Feature 1

#### Description and priority

(...)

#### Functions

(...)

#### Rules

(...)

#### User actions

(...)

# Non-functional requirements

## Performance requirements

(...)

## Safety requirements

(...)

## Security requirements

(...)

## Software quality attributes

(...)

## Business rules

(...)

# User interface wireframes

(GUI sketches of product B2)