Project Name:
Software Requirements Specification
Version 1.0

Lucky 7
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA USA

October 10, 2018

Contents

Credits

Name	Date	Role	Version
Adam Beymer	October 10, 2018	Lead Author of Introduction	1.0
Poojitha Karumanchi	October 10, 2018	Co-Author of Description	1.0
Katie Mei	October 10, 2018	Co-Author of Description	1.0
Sam Koski	October 10, 2018	Lead Editor	1.0
Kyle Maxwell	October 10, 2018	Lead Author of System Features and Other Requirements	1.0
Max Loumena	October 10, 2018	Lead Author of External Interface Requirements	1.0
Jacob Territo	October 10, 2018	Lead Author of Nonfunctional Requirements	1.0

User Persona Authors

Name	User Persona	Version
Max Loumena	Bradley Broseph	1.0
Poojitha Karumanchi	Jordan Salins	1.0
Katie Mei	Riley Taylor	1.0
Jacob Territo	Dana S. Ience	1.0
Adam Beymer	Oscar Wilder	1.0
Sam Koski	Angela Stewart	1.0
Kyle Maxwell	Elliot Alderson	1.0

Use Case Authors

Name	Requirement Identifier	Version
Adam Beymer	Use Case 1, Cure Cancer	1.0
Max Loumena	Use Case 2, Split Categories	1.0
Jacob Territo	Use Case 3, Importing New Data-Set	1.0

Requirements Authors

Name	Requirement Identifier	Version
Adam Beymer	fREQ-1	1.0
Max Loumena	fREQ-2	1.0
Katie Mei	fREQ-3	1.0
Kyle Maxwell	fREQ-4	1.0
Jacob Territo	fREQ-5	1.0
Sam Koski	fREQ-8*1.0	
Poojitha Karumanchi	nfREQ-1	1.0
Max Loumena	nfREQ-2	1.0
Jacob Territo	nfREQ-3	1.0
Sam Koski	nfREQ-4	1.0

Revision History

Name	Date	Reason for Changes	Version

1 Introduction

1.1 Purpose

Lucky 7 is proud to introduce the Data Classifier, a cloud based tool that uses machine learning to classify data categories from thousands of data-centers into a single ontology. Lucky 7 has worked closely with MarkLogic, a database company based in San Francisco, to produce the data science tool. The Data Classifier will be the brains behind MarkLogic's Data Discovery Workbench, a tool to help data scientists see trends in massive data lakes. The Data Classifier is particularly useful when unifying data silos that have data-types that are each labeled with a slight variation on the same idea; take 'phone number' and 'contact number' for example. The following document presents the features, requirements and users of the Data Classifier.

1.2 Document Conventions

The authors for each section are labelled in the Credits section below. The name will correspond to a section in this document that that person wrote.

1.3 Intended Audience and Reading Suggestions

This document is intended as a point of reference for the Lucky 7 team. As the Lucky 7 team develops the Data Classifier they will need a point of reference for each of the requirements that they have previously set.

In addition this document is for the MarkLogic team who will be adopting the data classifier. It is Important the customer, MarkLogic, be informed of the specification of the Data Classifier throughout its development.

Lastly, this document is intended for any users of the Data Classifier. In order to know what the Data Classifier is capable of, it is recommended that users of the tool read this document.

1.4 Project Scope

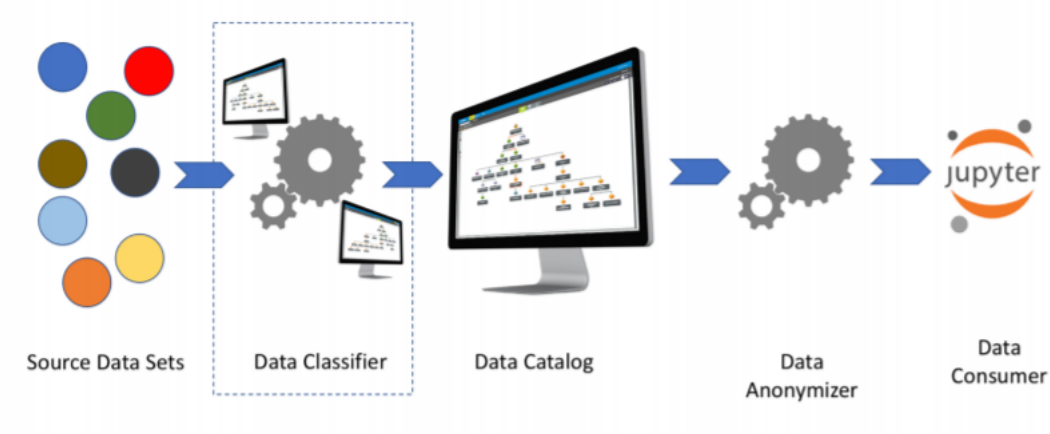
The Data Classifier will be used to classify data categories from thousands of data silos into one central location. For more information, see the vision and scope document.

2 Overall Description

2.1 Product Perspective

MarkLogic, a company that prioritizes the ability to manage customer data is interested in creating a Data Discovery Workbench, which provides a broad collection of tools and

technologies to help aid data classification. For the Data Discovery Workbench, there is a large focus on the Data Classifier which will use Machine learning techniques to identify classes of information contained in the data sets. Companies will be able to provide their source data sets and feed it into the Data Classifier that will classify the information, catalog it, and anonymize the data if necessary.



2.2 Product Features

Highlight features of the Data Discovery Workbench, as previously mentioned include the ability classify, catalog, and anonymize data sets. In more specificity, data classification is the main feature which involves processing data sets, sorting, and classifying the information based off category and inputs. Following the classification, cataloging will display data classifications to the user providing them an additional feature to edit or add to the data set manually. The final highlight would be anonymization, the ability to process data sets so that no users will be able to see sensitive information, this way it respects private information of customers and protects the company security as well.

2.3 User Personas

2.3.1 Bradley Broseph

Bradley is a data scientist employed by a California University. He is currently 22 years old and has just began his research for his masters thesis. He has worked with large sets of data before, and has strong opinions on data organization.

For his thesis, Bradley is attempting to analyze employee metrics for various financial institutions. He began this by collecting employee data from every institution that would hand over their data to him. When collecting this data, he realized that while all his data was in CSV files, none of them had a consistent format.

Bradley was rather annoyed by this lack of consistency. He doesn't want to manually create a standard format, and then put the data from every institution into that format.

He comes to MarkLogic, looking for a tool that can automatically sort and categorize his data.

2.3.2 Jordan Salins

Jordan Salins is a 24 year old data analyst at a sports data start up. He currently lives in San Jose, California with a few of his friends from college. He graduated from UC Berkeley with a degree in electrical engineering and computer science. Jordan has a huge world map in his room and wishes to travel the world someday. Jordan loves to learn and strongly believes that anyone is capable of learning as long as the subject is presented in a certain way. Jordan is a visual learner.

He is having a hard time at work as he is not able to understand all the data being thrown at him. He is looking for a data visualization tool that can help him and the rest of the company find trends. MarkLogic is the perfect solution for this problem as they own a tool that can sort and categorize the data.

2.3.3 Riley Taylor

Riley is a business analyst employed by Chameleon Auto Insurance Company. She is 30 years old, a mom working part time, and does not have a lot of computer science knowledge.

For her job, Riley was given the project of integrating the customer data from the recent acquisition of Geico to Chamelon's customer database. Geico, has millions of customers and trying to merge the data together was proving to be difficult for someone with little data science knowledge.

Riley looks and finds MarkLogic, a company that can assist with sorting and organizing all the customer data she needs to process.

2.3.4 Dana S. Ience

Dana is an manager at Hostess Brands. She is currently 37 years old and has a degree in psychology. Her main interests are cooking and knitting, and she has little computer skills.

At Hostess they've started using a new system for recording employee performance. The new system allows Dana to look over records from every single branch. However, each branch produces different pastries and records their data differently. Some include 'Twinky Errors' while others have something like 'HoHos Dropped'.

Dana doesn't think she'll be able to take advantage of all this new data. She doesn't have the time nor the computer skills to sift through and combine all of these records. What she needs is a computer application that is easy to work with and simplifies her job.

2.3.5 Oscar Wilder

Oscar is a young and aspiring data scientist. He graduated college 3 years ago from Yale University. He is 25 years of age and lives in San Francisco. He has worked for a medical research company called lab rats for the last 4 years. Oscar is very single and probably will be for a long time. Oscar lives with his mother and enjoys baking cakes with her. Oscar can usually be seen wearing jeans that are too big yet and yet still too short at the same time. Oscar likes to spend his weekends alone with his pet lizard while he sits in the basements and plays video games from his childhood. Oscar has had some recent struggles with his work. As his medical device company adds more and more branches, Oscar is having trouble seeing the origins of data when he is analyzing it. Oscar needs a data visualization tool.

2.3.6 Angela Stewart

Angela is a 29 year old doctor working at Johns Hopkins. She lives with her dogs and plays football on the weekend. Angela and her fellow doctor colleagues are having troubles with their patient data. The data isn't organized and makes it hard to find and use

patient information. During a friendly football game with doctors from another hospital she mentions the trouble her workplace is having with patient data. Her opponents mention a tool they have been using to organize their data, a data visualization tool from MarkLogic. The tool is exactly what Angela and her hospital needs to solve their data problems.

2.3.7 Elliot Alderson

Elliot is a 28 year old Cyber Security Engineer at Ecorp in New York City. He was not formally taught, but is a venerable security expert. Elliot devotes his spare time to grey-hat vigilante hacking. As a cyber security engineer at Ecorp, Elliot deals with a lot of data that could potentially be compromised. Elliot has a definite need for a tool to quickly understand the massive amounts of data that could be compromised as Ecorp constantly takes on new clients. Elliot decides to use the Data Discovery Workbench to help him understand the data he is working with.

2.4 User Classes and Characteristics

User Class	**Description**
Developers	Design and build the application.
Customer(MarkLogic)	Communicates to developers what is needed in the application. Discusses and prioritizes the most important features

2.5 Operating Environment

The Data Discovery Workbench will be web based on the MarkLogic site and integrated with the existing MarkLogic software. It will be hosted on MarkLogic servers, but will be operated on the client side following the initial set up by MarkLogic.

2.6 Design and Implementation Constraints

There will be limitations for the developers due to the sheer amount of data that will need to be processed in order to train a model. Many choose to use AWS because of the available credits provided as well as the ease of storing and processing the information. Further, there are hardware constraints when training a neural network model. It requires a computer with a lot of cores, as the processor needs more threads to perform repetitive actions over and over again.

2.7 User Documentation

With the delivery of the data classifier production-ready, the source code will be provided open-source on a GitHub account as well as user documentation guides. There is the possibility of tutorials being provided after the completion of the entire Data Discovery Workbench on the MarkLogic site. Further, this document along with the Vision and Scope document for the data classifier will be made available on GitHub as well.

2.8 Assumptions and Dependencies

The project could be affected if some assumptions we are making are incorrect, are not shared, or changed. The machine learning aspect of this project is to match similar categories and provide categorical recommendations. We are to implement a user interface of our choice rather than strictly follow MarkLogic's current design.

We are dependent on the requirements made by the stakeholders. We hope to communicate with them each week and receive feedback as we make progress.

2.9 Business Rules

The final deliverable is released under an open source license. Students will be able to refer to their participation on this project in the future.

3 Use Cases

3.1 Use Case 1: Cure Cancer

Oscar Wilder has been working with a medical research company called Lab Rats Lab Rats has several different research facilities across the country Each facility has sloppy data keepers that dont usually follow the unified model Oscar has a csv file from each facility Oscar loads the csv files into MarkLogics data workbench The data workbench displays the data in an easily understandable fashion Oscar notices a trend in the data that was previously unnoticeable Oscar tells his boss about the trend Oscar invents a new medicine Trillions (yes, trillions) of people buy the new product The human race is immune to cancer Oscar makes billions All thanks to MarkLogics data workbench

3.2 Use Case 2:

Use Case ID:	2
Use Case Name:	Split Categorization
Actors:	Reviewer

Description:	After our system sorts through the data, the user reviews the generated categorizations and notices that one groups columns incorrectly. He splits the generated category into two separate categories.
Preconditions:	1. Reviewer is currently using our GUI. 2. Reviewer has already gone through Use Case 1 at some point.
Postconditions:	1. Category specified by the user is split into two separate categories. 2. Items in the previous category are automatically placed into the two new categories.
Normal Flow:	2.0 Split Categorization 1. User uses the system to categorize his data. 2. System categorizes and displays data on the GUI. 3. User reviews the data and notices that one of the categories is incorrect. 4. User selects the category. 5. System displays a list of options for manipulating said category. 6. User chooses to split the category from the list of options. 7. System automatically finds the two categories that the selected category will be split into. 8. System displays the new categories for confirmation. 9. User confirms the new categories.
Alternative Flows:	2.1 Incorrect split of category (branch after step 8) 1. User indicates that the split categories are incorrect. 2. System asks for user input on what the new categories should be. 3. User specifies the two new categories. 4. System creates new categories based on user input. 5. Return to step 8.
Exceptions:	2.0.E.1 Data is formatted in a non-CSV file (at step 1) 1. System informs User that their data is incorrectly formatted.

	2. System terminates use case. 3. User inputs CSV files into the system.
Includes:	None
Priority:	Medium
Frequency of Use:	Every user should never have to use this, but exact frequency will be determined by accuracy of our machine learning program
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	Assume that most users will need to use this at least once, but not more than a few times.
Notes and Issues:	1. When the user splits the categorization, what should happen if our program incorrectly splits the items in the previous categorization into the new categorizations?

3.3 Use Case 3: Importing New Data-Set

Use Case ID:	3
Use Case Name:	Importing New Data-Set
Actors:	User
Description:	Our user enters the application to upload a new data-set consisting of multiple csv files.
Preconditions:	1. User has opened our program. 2. User has a multiple csv files that make up a data set in a 'data' folder.
Post-conditions:	1. System has successfully imported the data.
Normal Flow:	3.0 Importing New Data-Set

	1. User selects the new data-set button. 2. System prompts them with a file selection window. 3. User selects the 'data' folder that contains the multi-file data-set. 4. System prompts with import options such as TBD. 5. User selects their options. 6. System displays progress bar as it imports the data. 7. System completes import. 8. System prompts user with the option to automatically classify their data set. 9. User declines and import is concluded.
Alternative Flows:	3.1 User accepts automatic classification (after 3.0.8) 1. User indicates that they would like the system to automatically classify their data-set. 2. System processes data-set and classifies their data. 3. Use case 2, split categorization begins.
Exceptions:	TBD
Includes:	None
Priority:	Medium
Frequency of Use:	Every single user will follow this use case atleast once, but probably more. It is necessary to upload data for this tool to be useful.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	TBD
Notes and Issues:	1. TBD

4 System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

4.1 System Feature 1: Initial Unsupervised Classification

4.1.1 Description and Priority

The Data Classifier will use unsupervised machine learning techniques to identify classes of information contained in selected data sets fed as input to the Data Classifier. This is of High priority and provides the main strength of the project.

4.1.2 Stimulus/Response Sequences

The user will link the source for their data sets, then select the data sets that they wish to analyze.

4.1.3 Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

1. fREQ-3: The system should complete categorization within a reasonable timeframe(Currently unknown, estimated at <2 seconds).
2. fREQ-4: The system can glean at least one suggestion for data classification based on the patterns it identifies in the data.
3. fREQ-8: The system shall use machine learning to categorize data.

4.2 System Feature 2: User Classifications

4.2.1 Description and Priority

The Data Classifier will allow users to browse, edit, and add to the classifications that were performed automatically. A user may remove an erroneous classification and add a new classification for a data element that was not automatically classified. This is of high priority.

4.2.2 Stimulus/Response Sequences

After the classification, the user is presented with the options for editing the classifications.

4.2.3 Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

1. fREQ-5: The system should allow users to manually reclassify data after it has been automatically classified by the system.
2. fREQ-5: The system should allow users to create new classification categories for their data.
3. fREQ-5: The system should remember the user's edits and learn from them in order to improve automatic classifications.

4.3 System Feature 3: Interpretation of Classifications

4.3.1 Description and Priority

Allow users to see predefined and learned categories of data elements that will be recognized by the Data Classifier. This is of medium priority since it is necessary to present the results to the user.

4.3.2 Stimulus/Response Sequences

After iteratively editing the classifications, the user can initialize the visualization of the classifications. The system then transitions to the presentation of the ontology.

4.3.3 Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

1. fREQ-1: The system should be able to recognize categories that exist in a csv file display them in the GUI.

5 External Interface Requirements

5.1 User Interfaces

Our primary user interface will be a browser GUI. They will be given a prompt to input data, which they will use to select files to put into our program. If any of the files are not CSV files, an error will be displayed and the system will ask for CSV files. After the files are input into the system, you will be taken to a waiting page while our system finds categories within the data.

Afterwards, the majority of the screen will be taken up by a display of the categories that our software has found in the data, with some example line-item data below the categories so that the user knows what has been placed into the categories. The user will be given a minimal amount buttons of buttons for interaction, as we will want the entire screen to be able to be interacted with. They will have a button to confirm the categories, and prompts when hovering over categories. The prompts will give the user the option to manipulate the data, with the main options being splitting the category, combining the category with another category, and renaming the category.

(nfREQ-4)The user should be able to know how to use the tool without needing a ton of explanation.

5.2 Hardware Interfaces

This software should work in Chrome on any desktop or laptop. We have no current plans to extend this to mobile devices or other browsers, but it should be usable on Firefox and IE as well.

5.3 Software Interfaces

We plan on using the MarkLogic database in order to store, order, and retrieve data sent into the system.

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for

example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

5.4 Communications Interfaces

The browser GUI will be delivered to the client through through HTTPS.

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

6 Other Nonfunctional Requirements

6.1 Performance Requirements

(nfREQ-1)The user should be see the origin of categories presented to them.

6.2 Safety Requirements

(nfREQ-2)The system should not directly modify the data in any way, which makes the in-line data completely immutable and secure. (nfREQ-3)The system should preserve the origins and original categorization of all data that has been inputted into the system.

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

6.3 Security Requirements

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

6.4 Software Quality Attributes

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

7 Other Requirements

Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.

A Glossary

Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.

B Analysis Models

Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.

C Issues List

This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.