Data Classifier: Software Requirements Specification Version 2.2

Team Mark
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA USA

November 1, 2018

Contents

Credits

Name	Date	Role	Version
Geraldo Macias	October 9, 2018	Lead Author of Other Nonfucn-tional Requirements	1.0
Matt Yarmolich	October 9, 2018	Lead Author of System Features	1.0
Spencer Schurk	October 10, 2018	Lead Author of Introduction	1.0
Jake Veazey	October 10, 2018	Maintains Document and glossary. Assisted on front end System Features	1.0
Brad Foster	October 10, 2018	Lead Author of Overall Description	1.0

Revision History

Name	Date	Reason for Changes	Version
Geraldo Macias	October 8, 2018	Completion of Other Nonfunctional Requirements	1.0
Spencer Schurk	October 10, 2018	Addition of my User Persona, Use Case, Functional Requirement, and Non-Functinal Requirement	1.0
Jake Veazey	October 10, 2018	Addition of front end system reqs, Use Case, User Persona, clean up/add glossary	1.0
Spencer Schurk	October 10, 2018	Completion of Introduction	1.0
Brad Foster	October 10, 2018	Completion of Overall Description	1.0
Geraldo Macias	October 24, 2018	Section 6 verified, User Description update, Use Case 6 update	2.0
Spencer Schurk	October 24, 2018	Section 1 revisions	2.0
Jake Veazey	October 24, 2018	Clarification and additions to Overall Description, Glossary, and overall maintenance of the document	1.0
Spencer Schurk	October 30, 2018	Modification to use case 2	2.1
Geraldo Macias	December 10, 2018	Various fixes for 2.2	2.2

1 Introduction

1.1 Purpose

The purpose of this document is to present the functional and non-functional requirements of our Data Classification application version 1.0. In addition, it also features potential user personas, fully-dressed use cases of the product, and diagrams to complement text descriptions. This document serves as a source of reference and guidance during the development process. Please note, this document does not describe the architecture of this application.

1.2 Document Conventions

Requirements listed in later sections will adhere to the following convention: FR-n for functional requirements, and NR-m for non-functional requirements, where n and m is the requirement's numerical value. No two functional requirements shall have the same n-value, and likewise, no two non-functional requirements shall have the same m-value.

1.3 Intended Audience and Reading Suggestions

This document was intended for and targeted at the following parties.

1.3.1 Developers

Developers of this project will use this document as a reference during the development process. Specifically, the Use Cases and Functional Requirements should be referenced the most.
Suggested Reading Order

1. Overall Description
2. System Features
3. Functional Requirements
4. Non-Functional Requirements

1.3.2 Customer - Mark Logic

Our customers at Mark Logic will use this document to make sure we're on track with the project and have the correct vision. They will be able to verify our requirements are applicable to the product they want us to produce.
Suggested Reading Order

1. Overall Description
2. User Cases
3. External Interface Requirements
4. Non-Functional Requirements

1.3.3 Supervisor - Bruno Da Silva

Bruno Da Silva, the team's professor in CPE 402, will read this document to understand how we view the project. Additionally, he will use this document to verify the team will remain up to date with changes, and that the team will update this document.
Reading Suggestion

Bruno Da Silva should thoroughly read this document to ensure our team has an acceptable level of understanding of the class material.

1.4 Project Scope

For information regarding the scope and vision of this project, please reference the team's *Vision and Scope* document.

`https://github.com/rocketeer55/CSC402-GENERAL/blob/master/VisionAndScope.pdf`

2 Overall Description

2.1 Product Perspective

One of the biggest unsolved roadblocks for data scientists is data sanitization and grooming. Data scientists spend excessive amounts of time preparing data before even being able to do productive work. Eliminating this downtime would allow more room for data scientists to do the meaningful work that they actually want to do. Our application aims to do just this. Through the use of machine learning techniques and a robust web interface, our application will take various ungroomed data sets as input and will convert them into cohesive, organized documents as output. We want to make the process of pre-analysis data grooming as automated and painless as possible.
The classification service will be composed of two parts: a back end that contains machine learning and connection to databases and storage technology and a front end that links the back end to the user. The overall scope of the back end will be to take any number of inputted, ungroomed, files, classify using machine learning and algorithms, and use services and databases to store user inputted information and any output that relates to the user's input. The front end takes on the role of the link from user to backend and will, overall, allow user uploads of files and user downloads of the back ends output.

2.2 Product Features

The primary feature of the application is that it will be able to take data from various different "silos" and automatically make meaningful, organized categories out of them. The program will be robust enough to make sense out of data that is unconventionally or even improperly formatted. The application will also have a web interface that data scientists will be able to interact with.

2.3 User Personas

2.3.1 Frank McLaughlin

Author: Spencer Schurk
Age: 24
Job: Data Scientist
Bio:

Frank McLaughlin is a 24 year old data scientist working for a large e-Sports company. Frank has been interested in video games from a very young age, and this interest has helped shape his life and influence his friends. His parents never appreciated his video game obsession, but Frank didn't let that stop him.

Frank graduated from UC Irvine with a bachelor's degree in Computer Science. After graduating, Frank worked at a startup for a delivery app. He was assigned data analysis tasks, but they were very basic. Frank did not like this job at the startup very much, as he felt it was too fast paced with very little organization. Everbody was stepping on everyone else's feet. Eventually, it was time for Frank to find a new job.

Frank then found a job at a major e-Sports company. Here, he was also hired as a data analyst. He likes working at this new company a lot better, since they use better tools and he can be more efficient. However, the data sets Frank works with now are much larger. He hates manually searching through tables to find the correct data types for his tasks.

Luckily, Frank has heard that his e-Sports company will soon be using a new tool to automatically classify the data in these data sets. He believes this will instantly help him be more efficient, and get more data analysis done quicker. Now, Frank can go home sooner and play more video games.

2.3.2 Dave Fuego

Author: Matt Yarmolich
Age: 25
Job:Sports Data Analyst
Bio:

Brian Fuego is a 25 year old Sports Data Analyst working for the Seattle Seahawks. He has been an avid sports fan for his entire life and loves following the numbers and figuring out threats on the field for his team. His dad really wanted him to be a football player, but he was never good enough to play on any competitive team.

Thanks to his father?s drive, he has been able to push himself into fields that he believed would make his father proud. Thus he began a career as a data analyst for his dad (and his) favorite sports team. Unfortunately, Dave is getting sick of doing his job week by week and wants tools to help make his job go by faster so he has more time doing his actual passion, coding.

Through his relentless search for tools, Dave has found the data classifier with a GUI that helps make his job way easier. He likes that it uses the newest trends in computing (machine learning) and that it has GUI for ease of use to make his job go by faster. Dave plans on implementing this technology so that he has more time to do his passion and work on creating games written in OpenGl and c++ since that is his bread and butter and what he likes doing the most in this world.

2.3.3 Bread Feuyser

Author: Brad Foster
Age: 27
Job:Music Data Analyst
Bio:

Bread Feuyser is a 27 year old Music Data analyst working for a major record label. Bread has been a passionate music fan for his entire childhood, adolescence, and adult life. He found his way into the music data industry when he realized that he had an unusual ability to make sense out of seemingly meaningless and nonsensical data.

Bread is a Third Degree Supreme-Master class data analyst, but has trouble listening to music while grooming data during the pre-analysis phase. Any sort of mis-classification or other kind of organizational mistake completely throws off his rhythm. This consequently adversely affects his ability to ”connect” with the music he is listening to, which ultimately leads to drastic reductions in his productivity. Bread’s main strength is his ability to make wild and unconventional connections between musical data sets, and this ability is fueled by his unrivaled love for ”sick beats” and ”clean mixes” (sic).

Seeming as though The Hand of God reached down and personally delivered it to him, Bread one day stumbled upon a data classifier that had the perfect set of features to solve all of his data problems. It allowed him to feed it various disorganized data sets and turn them into a single, cohesive, analysis-ready database. It even had a web based interface with buttons that Bread could click in sync with the music he was listening to. The musical world will continue to be changed, though now at an even faster rate, by Bread’s groundbreaking discoveries.

2.3.4 Chad Chaderson

Author: Jake Veazey
Age: 19
Job:Data Company Intern
Bio:

Chad Chaderson is a Computer Science student, with a minor is data science, at Yale University and began working for a data analyst company the summer after his second year of school. Chad does not know much of what he wants to do with his degrees after college, so he is using the internship as a way to expand his horizons to learn and discover.

At this internship Chad has been asked to do a lot of chores that the rest of the employees don't want to do. One such task has been to take many files of data that has not been classified into its categories but had no way to do so in a fast and easy way. After a few weeks of manually sorting and classifying data from multiple files that we assigned to him, Chad got frustrated and was contemplating leaving the company and changing his area of concentration. Before he made the decision, Chad, spoke to his boss and learned that there are tools online that may be able to assist him in his work – this is when Chad found the Data Classifier tool and online GUI.

Chad found that the entire system worked well with his company's information, and because of his excellent work, he got an offer to come back the following summer.

2.3.5 Davide Falessi

Author: Landon Gerrits
Age: 37
Job:Project Manager
Bio:

Davide is 37 year old project manager from Italy. He works for a data analysis company, Matthew Reason. Davide is a good project manager, in fact he's highly technical. He began his career and a software engineer specifically working in data science. He now is a project manager for the data science division of his company. Davide interfaces often interfaces with the business of the company. The business often asks Davide to provide high level data sheets that show the final output of Davide's team's classification algorithms.

Last year, Matthew Reason had many clients from the sports industry, specifically soccer. Many major news networks contacted Matthew Reason to analyze and filter their data based around player statistics. Davide's team was tasked with creating a program that could rapidly classify and sort this player data.

Davide's team was underfunded to have enough man power to create this program. Davide's team was already working on 4 other important projects for the company. While researching this problem, he found a data classifier that did exactly this. Davide purchased this program and it was downloaded and installed in a matter of minutes. Davide read the documentation and imported a data set with player stats. He ran the program and moments later he received an output file that classified player names, countries, and many other valuable classifications. Davide showed this to the business and they were thrilled that he was able to get these data classifications with such a short turnaround.

2.3.6 Mantis Tobaggon

Author: Geraldo Macias
Age: 60
Job:Financial Investor
Bio:

Mantis was a failing financial investor who always seemed to be a little bit behind the trends on Wall Street. He tried using different brokering companies but their rates were beyond reasonable. After doing a little research Mantis started to see the buzz statement Data Science in multiple places. This looked like a lot of work to get into, but this was the answer Mantis has been looking for.

For one year Mantis spent time on Udemy taking various online classes learning how to maximize the effectiveness of Scikit learn.Because Mantis has not taken any formal computer science courses he doesn't really understand coding outside of data science.After taking his year of Udemy courses Mantis became a power house of an investor on Wall Street but he quickly became bored with the market. With laws changing about sports betting, Mantis saw this as an opportunity to move into the sports industry.

There is a lot of public available datasets of sports statistics but due to Mantis' inexperience in computer science, he is unable to manage and quickly organize all his data. This is becoming a major problem for Mantis and is now looking for a solution to manage all his data so he can spend more time analyzing data and making money.

2.4 User Classes and Characteristics

User Class	Description
Developer	The Developer will design, implement, test, and maintain the application.
Data Scientist with programming knowledge	A Data Scientist that will require advanced technical features from the application.
Data Scientist without programming knowledge	A Data Scientist that will require a simplistic interface and feature set from the application.

2.5 Operating Environment

The program will exist as a web based application with a React front end, Flask back end as well as an s3 database and a block storage system. The data classifier will be written in Python.

2.6 Design and Implementation Constraints

The back end/machine learning parts of the application will have to be written in Python coupled with the Scikit Learn. Our application also must implement a web based interface that will be written in JavaScript that will call APIs built in the back end.

2.7 User Documentation

During implementation, we will be monitoring issues through the use of JIRA. We will likely intend to include some simple documentation for using the user interface along with the application.

2.8 Assumptions and Dependencies

We are assuming that we will easily be able to integrate a database solution (i.e. MongoDB, DynamoDB) into our application during development. In addition, when using data with columns, our machine learning algorithm will use Linear SVC. If this process fails we will use Naive Bayes. For data without column names we will use K-Means, Spectral clustering, and GMM in that order. AWS has also been made available for us to use and can be nicely integrated with the back end to store files in S3 Buckets and other information in one of the offered databases. Finally, we are assuming that we will be able to use something like Google Firebase for our web app development.

3 Use Cases

3.1 Use Case 1: Select Proper Classification Category

This use case details the the path of the data analyst providing feedback to the machine learning algorithm written by Matt Yarmolich

Use Case ID:	1
Use Case Name:	Select Proper Classification Category
Created By:	Matt Yarmolich
Last Updated By:	Matt Yarmolich
Date Created:	October 9, 2018
Date Last Updated:	October 9, 2018
Actors:	Data Analyst
Description:	A Data Analyst accesses the data classification front end by feeding it information from an HTML-5 input tag and having the information parsed by the system. .
Preconditions:	1. the Data Analyst is logged into the system. 2. the Data Analyst is a registered user for the system. 3. the Data Analyst has a dataset that needs to be analyzed.
Postconditions:	1. the dataset has been correctly parsed by the data classifier 2. the classifier has outputted a variety of tags for verification by the Data Analyst. 3. the Data Analyst is capable of correctly identifying the dataset
Normal Flow:	1.0 Select the correct category from the data classifier

	1. The Data Analyst uploads the dataset to the front end of the system 2. System displays upload status 3. System displays upload complete and feeds the data to the back end 4. the back end outputs the results of the data in a web-parsable format (JSON) 5. The front end parses this output and displays it to the data analyst along with the columns in question 6. System displays upload complete and feeds the data to the backend 7. The machine learning classifier classifies the initial column data 8. The machine learning classifier the rest of the data based on past data columns and previous data fed through the system 9. the backend outputs the results of the data in a web-parsable format (JSON) 10. the Data Analyst reviews the data and selects the correct category based on their domain knowledge 11. The front end feeds their result back to the data classification back end 12. System displays the correct tagging of the data set
Alternative Flows:	1.1 No results from back end (branch after step) 1. The system outputs no previously learned categories 2. The Data Analyst inputs a new category for the data set 3. Return to step 8.
Alternative Flows:	1.2 The Data Analyst wants to rename a dataset 1. The Data Analyst clicks the button to edit the name of a data set 2. The Data Analyst inputs a new name for the data set 3. The Data Analyst clicks confirm 4. Return to step 8.
Exceptions:	1.0.E.1 System loses connection to front end(at step 1)

	1. System informs Data Analyst that connection has been lost to the server 2. Patron reloads page 3. System restarts use case.
Includes:	None
Priority:	High
Frequency of Use:	Approximately n uses per data set (depends on column inputted)
Business Rules:	TBD

3.2 Use Case 2: Upload Data Sets

This use case details the the user flow of uploading data sets to the data classifier using the web interface, written by Spencer Schurk.

Use Case ID:	2
Use Case Name:	Upload Data Sets
Created By:	Spencer Schurk
Last Upadted By:	Spencer Schurk
Date Created:	October 10, 2018
Date Last Updated:	October 10, 2018
Actors:	Data Analyst
Description:	After logging in, the system allows a data analyst to upload multiple data sets. After uploading data sets, the data analyst can begin the classification process.
Preconditions:	1. The data analyst is logged into the system. 2. The data analyst has not uploaded any data sets. 3. The data analyst has not began the classification process.
Postconditions:	1. All selected data sets have been uploaded. 2. The data analyst is now able to begin the classification process.

Normal Flow:	1.0 Select multiple files to upload. 1. The data analyst selects "Upload Files" button. 2. System opens local machine's file explorer / file selector. 3. Data analyst selects multiple files from their local machine. 4. Data analyst selects "Ok" or "Upload" on system file explorer. 5. System displays list of files selected and begins to upload files. 6. System allows user to edit name of file being uploaded. 7. System displays progress of each file being uploaded. 8. System displays "Upload successful" message after upload completes. 9. Data analyst selects "Begin Classification" button.
Alternate Flows:	1.1 Select files more than once. (branch after step 4) 1. While previously selected files are uploading, data analyst selects "Upload more files" 2. System opens local machine's file explorer / file selector. 3. Data analyst selects multiple files from their local machine. 4. Data analyst selects "Ok" or "Upload" on system file explorer. 5. System adds selected files to list of previously selected files that are uploading. 6. Flow continues on step 8 of normal flow

Exceptions:	1.0.E.1 Data analyst uploads unsupported file. (at step 4) 1. System displays error message, with file name present. 2. Data analyst selects ”OK” button. 3. System removes selected file from list of files to upload. 4. System continues to upload other selected files. 1.0.E.2 Upload fails for any reason. (at step 5) 1. System displays ”Upload failed” message 2. Data analyst selects ”Try again” or ”Cancel” button. 3. If ”Try again” is selected, system continues at step 5, with all progress at 0. 4. If ”Cancel” button selected, page is refreshed, and continues at step 1.
Includes:	None
Priority	High
Frequency of Use:	Once per data classification process.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	1. Data sets are locally stored on data analyst’s machine. 2. Data analyst has active internet connection capable of uploading files.
Notes and Issues:	1. What if the data analyst uploads the wrong file?

3.3 Use Case 3: Log In

This use case details the flow of logging into the application, written by Brad Foster.

Use Case ID:	3
Use Case Name:	Log In
Created By:	Brad Foster
Last Upadted By:	Brad Foster
Date Created:	October 10, 2018
Date Last Updated:	October 10, 2018
Actors:	Data Analyst

Description:	The user must log into their account before accessing the app's features.
Preconditions:	1. The data analyst is not logged into the system. 2. The data analyst has the application open.
Postconditions:	1. The data analyst is now logged in.
Normal Flow:	1.0 Log In Successfully. 1. The data analyst opens the application. 2. System prompts the user for their username and password. 3. Data analyst inputs their username and password. 4. System verifies their credentials. 5. System displays the application's main screen.
Alternate Flows:	1.1 Log in Failed (branch after step 3) 1. System detects that there is a mistmatch between username and password. 2. System displays an appropriate error. 3. System prompts the user for their username and password again.
Includes:	None
Priority	High
Frequency of Use:	Once per launch of the application.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	1. Account credentials can be securely stored.
Notes and Issues:	TBD

3.4 Use Case 4: Log Out

This use case details the flow of logging out of the application, written by Brad Foster.

Use Case ID:	4
Use Case Name:	Log Out
Created By:	Brad Foster
Last Upadted By:	Brad Foster
Date Created:	October 10, 2018
Date Last Updated:	October 10, 2018
Actors:	Data Analyst

Description:	The user must log out of their account once they are done using the application.
Preconditions:	1. The data analyst is logged into the system. 2. The data analyst has the application open.
Postconditions:	1. The data analyst is now logged out. 2. The System is now showing the log in screen.
Normal Flow:	1.0 Log Out Successfully. 1. The data analyst clicks the log out button. 2. The user's uploaded/categorized datasets are stored to a database. 3. System closes the main screen and displays the log in screen. 4. System displays a successful log out message.
Includes:	None
Priority	High
Frequency of Use:	Once per login session.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	TBD
Notes and Issues:	TBD

3.5 Use Case 4: Viewing the Classifier Logs

Use Case ID:	4
Use Case Name:	Upload Data Sets
Created By:	Jake Veazey
Last Updated By:	Jake Veazey
Date Created:	October 10, 2018
Date Last Updated:	November 1, 2018
Actors:	Data Analyst
Description:	After logging in, the system allows a data analyst to upload multiple data sets. After uploading data sets, the data analyst can begin the classification process. Each step of the classification process will be noted and time logged. The Logs page will provide original input and classified output for all data sets uploaded.

Preconditions:	1. The data analyst is logged into the system. 2. The data analyst has just uploaded a data set and identied the set with a name or ID 3. The data analyst has just begun the classification process.
Postconditions:	1. All selected data sets have been uploaded. 2. The classification process has been completed. 3. All uploaded data sets are displayed with outputs and timestamps in the logs.
Normal Flow:	1.0 Select multiple files to upload. 1. The data analyst uploads their data sets. 2. The data begins to be classified 3. As the data is going through classifications, each step is being logged to show user status 4. When the data is done being classified, the logs will have available the input, output, and status logs
Alternate Flows:	1.1 User uploads more than one file. (branch after step 4) 1. Each uploaded file will enter a queue 2. While queue is not empty, continue Normal Flow
Alternate Flows:	1.2 User replaces a file. (branch after step 1) 1. The previous file(s) will be deleted from the upload process and the new file(s) will take its place
Alternate Flows:	1.3 User cancels classification of file (branch after step 3) 1. The file(s) selected to stop will break out of the code they're being run through and display a "User Ended" message. Any other files not yet classified or stopped will continue classification.

Exceptions:	1.0.E.1 Data analyst uploads a supported file, but incompatible data. (at step 3) 1. While the data set(s) the user uploaded are being classified the process fails 2. This can be due to incompatible information or incorrect files 3. The logger will display the failed status and attempt to show the reason the classification failed.
Includes:	None
Priority	Medium
Frequency of Use:	Once per data classification process.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	1. Data sets are locally stored on data analyst's machine. 2. Data analyst has active internet connection capable of uploading files.
Notes and Issues:	1. Check for all errors at each step in the classification process

3.6 Use Case 5: Exporting data and logs

Use Case ID:	5
Use Case Name:	Export Data
Created By:	Landon Gerrits
Last Updated By:	Landon Gerrits
Date Created:	October 10, 2018
Date Last Updated:	October 10, 2018
Actors:	Data Analyst and Project Managers
Description:	After the system has run and classified its data input, it is ready to be exported. Data analysts and project managers can choose to export this data to specified file types.
Preconditions:	1. The user is logged into the system. 2. The user has uploaded a data set 3. The user has run the classification process. 4. The classification process has been completed.

Postconditions:	1. All selected data sets have been uploaded. 2. The classification process has been completed.
Normal Flow:	1.0 Select expected file type output. 1. The user can view the classifications in the program's GUI 2. The user can select the specified file output type: .JSON, .CSV, .XML 3. The specified file output type is exported.
Alternate Flows:	1.1 User chooses multiple export file types 1. User runs the program 2. User checks multiple file types for export 3. The system exports the selected file types.
Exceptions:	1.0.E.1 Data analyst selects .CSV 1. User selects .CSV file type for export (Comma Separated Value documents do not work well with grouped sets the JSON and XML can handle) 2. User selects a location to put the file 3. The system exports a .zip file that contains a folder of .CSV files for each classification
Includes:	None
Priority	Medium
Frequency of Use:	Once per data classification process.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	1. Data set is large enough to be classified (minimum several hundred entries)
Notes and Issues:	1. Supporting multiple export file types will be time consuming for developers. A single file type may be desirable if there are higher priorities for the project

3.7 Use Case 6: Account Creation

Use Case ID:	6
Use Case Name:	Account Creation
Created By:	Geraldo Macias
Last Updated By:	Geraldo Macias

Date Created:	October 10, 2018
Date Last Updated:	October 10, 2018
Actors:	Data Analyst, System Administrator
Description:	Before the system can start to perform analytics, an Admin must create an account for a data scientist. It is important that only the data analyst may view the data.Two types of accounts can be created, a data analyst, and a system administrator
Preconditions:	1. The system must be installed. 2. If a Data Analyst account is being created, a System Administrator must already exist.
Postconditions:	A System Administrator must create a new Data Analyst account before it may be accessed.
Normal Flow:	1.0 New account created. 1. A System Administrator has been established. 2. A System Administrator creates an data scientist account.. 3. Data Analyst now has access to companies data and our product.
Alternate Flows:	1.1 New account denied. (branch after step 2) 1. A System Administrator deletes an account. 2. A data scientist account is deleted.
Exceptions:	1.0.E.1 A System Administrator accidentally creates an account (at step 2). The account is deleted.
Includes:	None
Priority	Low
Frequency of Use:	Once per new data scientist.
Business Rules:	TBD
Special Requirements:	TBD
Assumptions:	The System Administrator has access to the system first.
Notes and Issues:	Ensure the customer hands the software to the System Administrator first

4 System Features

This system will be primarily used for data classification and adding meta-tags/titles to unidentified datasets inputted by a user from various formats including CSVs, JSON, and SQL. The main purpose of this product is to give Data Scientists, Data Engineers, and Data Analysis's insight on what kind of data they are collecting, and the ability to verify or modify any tags generated by the machine learning processes in our back end. This will be done through a React front end written for the web interfacing with most modern browsers with javascript enabled. This front end will be powered by a back end written in Python leveraging a machine learning library such as TensorFlow. This back end will take data in from a recognized data format, parse the data into a data structure for temporary storage, feed it into this library and spit out possible classification tags. These tags will then be sent to the front end for verification by the data analyst for verification/modification. This data analyst will be able to see the data in question in order to judge the classifiers accuracy and to teach it about new datasets/data types by piping these recommendations back to the machine learning algorithm.

4.1 Machine Learning Python Back End

4.1.1 Description and Priority

This feature of the product will allow data being fed into the product to produce meaningful data for output/viewing by the data analyst. The benefit of this algorithm is that it will provide data to the front end for easy viewing and provide an endpoint for where feedback from the data analysts. In terms of cost, this is probably going to be a fairly expensive feature to implement as its the main focus of this project and without it, the rest of the project is useless. However there are a ton of resources for implementing this feature minimizing the potential development time required. Finally, this system requirement is of High priority due to the fact it is the main feature that makes our product useful.

4.1.2 Stimulus/Response Sequences

The system shall take in a dataset from the front end and then feed this dataset into the machine learning algorithm column line by line. This step will be done by the data parser which will read in the columns line by line and separate them (detailed below). This data will then be compared to previous inputs taught by the algorithm by the Software Engineers developing the program. With these comparisons, the algorithm shall develop a result that fits the dataset, or output unknown if it doesn't have any idea what the dataset is of. It also will look to previous classifications contained by the dataset

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. REQ-1: The system shall be able to take in a specific data column and be able to identify it based on previous machine learning techniques used on the system and output a string of what it thinks it should output .
2. REQ-2: The system shall be able to learn new classifications given to it by the Data Analysts as feedback from the front end.
3. REQ-3: The system shall be able to learn categories over time as training is provided to it by the Software Engineers developing the system, and by data analysts feeding it new categories/training.

(Written by Matt Yarmolich)

4.2 Data Classification front end

4.2.1 Description and Priority

This feature set will allow the data analyst to interact with the output of the machine learning algorithm for further classification and viewing of what the algorithm outputs. This feature set is another high priority feature as it allows our algorithm to be interacted with by the client and provide meaningful responses to help further teach the algorithm about new datasets and expectations. In addition, the user will be given download options of their initial input file and the output of the classifier with logs completed with timestamps as the file(s) move through the classifier.

4.2.2 Stimulus/Response Sequences

Users will be able to navigate the site using the front end and its linkage to the system and back end to perform the tasks necessary

4.2.3 Functional Requirements

1. REQ-1: User Uploads data set(s) after pressing the upload button
    - Each data set uploaded is timestamped by the system
    - Each set is added to a FIFO queue for processing
    - First set is checked for compliance and then computed. Else fails

- Output is stored in User's storage (AWS S3)

2. REQ-2: User requests the Output or Input of the the classifier
   - Cloud storage is accessed by the back end and then provided to the user
   - Storage is only accessable by the service for security purposes

Written by Jake Veazey

4.3 Data Parser

4.3.1 Description and Priority

This feature set will allow the data analyst to submit data to the System and parse it into a computer readable interface and parse it into a data structure for storage and analysis.

4.3.2 Stimulus/Response Sequences

The data analyst will select a data set to upload to the server. The server will then accept this file and parse the file. During parsing, the computer will store each columns data in separate data structures for analysis.

4.3.3 Functional Requirements

1. REQ-1: The system shall accept data to be uploaded to the back end
2. REQ-2: The system shall parse the data, storing each row's data for each column in a separate dynamically growing
3. REQ-3: The system shall feed the data to the machine learning algorithm
4. REQ-4: The system shall be able to accept over 10 data sets to upload at a time. (Written by Spencer Schurk)

5 External Interface Requirements

5.1 User Interfaces

This system has a web application with an interact-able web GUI. The web interface for the initial release will scale on all desktop web browsers. A mobile friendly web interface may be available in future releases. Upon visiting the site, the user is presented with a login screen. After logging in, the user is given the option to import one or more data sets. This will be done by clicking a button called "Import Data Set". The user may select one or more data sets of the appropriate file type from their local machine.

There will be a button to confirm the file selections. There will be a button to confirm upload and a screen that shows the progress of the file upload.

The user will then be greeted with another screen explaining the data classification process. This screen has a diagram of the entire pipeline of the process: File Import - Data Classifier - Output to user - User edits - File Output

There is a button titled ”Begin Data Classification” that will have a progress indicator while the data is being processed.

The following screen will display the data set with it's classifications. The user will have the ability to edit the names of the classifications by clicking on each of the classifier fields.

5.2 Hardware Interfaces

For the prototype the system will run off a user's local machine. The computer must have at least 8gb of ram to run efficiently. In the future the program will be hosted on a website and all of the computation processing will be done on Mark Logic's server machines.

5.3 Software Interfaces

All software will be written in Docker containers. This allows fro greater scalabilitity and reliability. The Data Classifier will be written in Python with Scikit Learn, Pandas, and Numpy libraries.

5.4 Communications Interfaces

Communication will be handled with a React front end and a Flask back end.

6 Other Nonfunctional Requirements

6.1 Performance Requirements

The customer has specified that performance is not a concern.

1. The software will be usable by Data Analyst with and without programming assignments

2. The system shall be able to be simple to understand and usable by anyone within the company that wants to use it (Written by Matt Yarmolich)

6.2 Safety Requirements

During the classification stage, safety is not a concern. After the catalog phase we will begin redacting sensitive information.

6.3 Security Requirements

1. The software will have two factor authentication and will only allow verified users to access the systems databases.
2. The system shall not parse any uploaded data set until the user has selected ”Begin Classification.” (Written by Spencer Schurk)

6.4 Software Quality Attributes

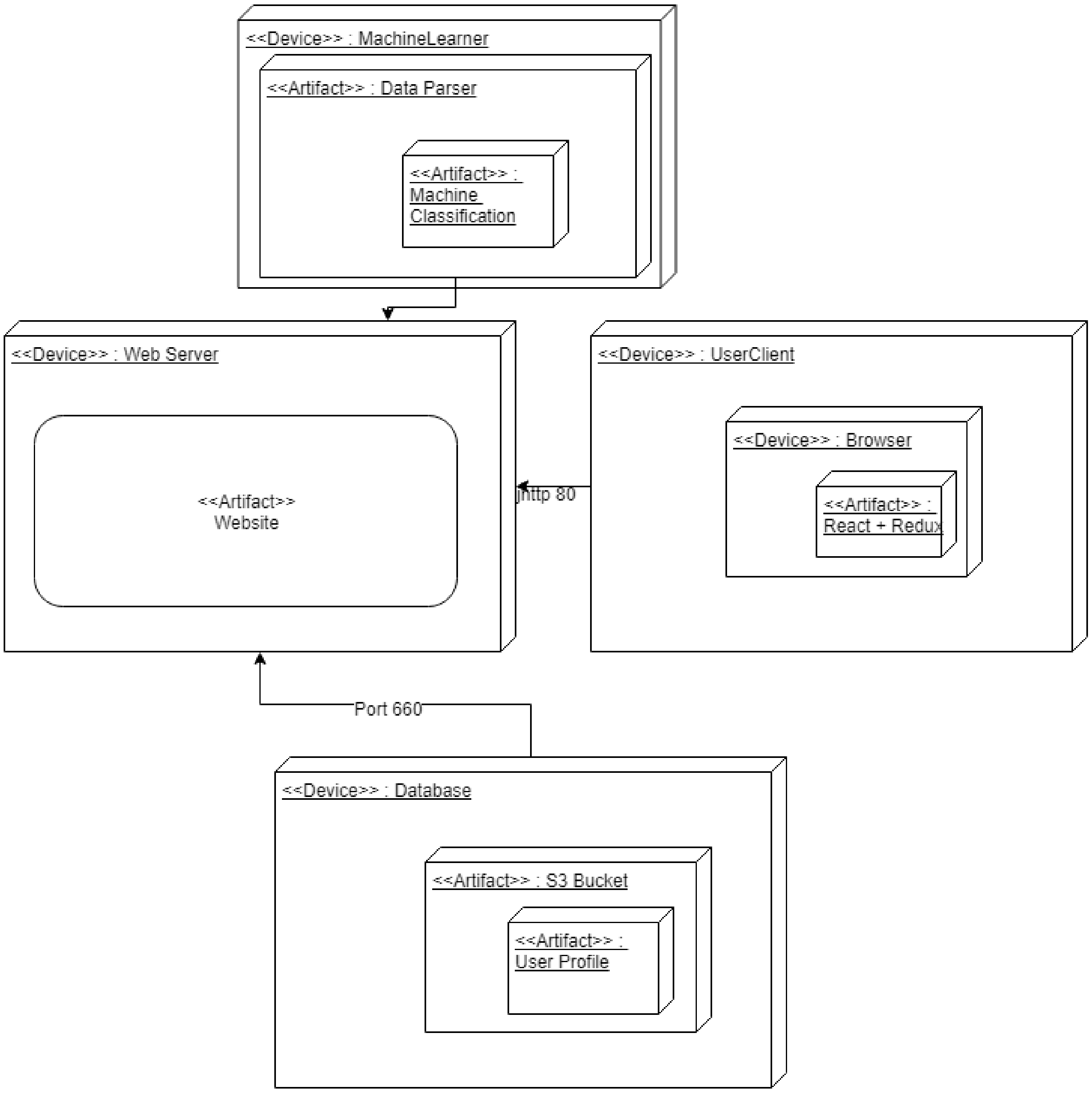
1. The system shall have a learning curve of 24 hours. This applies to all users ranging from a sports analyst to a data scientist.
2. The system shall be available 99.99% of the time.
3. The system shall be able to maintain one million datasets.
4. The system will be portable so different companies or entrepreneurs may use our solution.

7 Other Requirements

- Example Req
  - Example description
- Example Req
  - Example description

A Analysis Models

A.1 Deployment Diagram



(See section 2.5 for details.)

B Issues List

- Example Ticket 1
  - Example description
- Example Ticket 2
  - Example description

Glossary

AWS Amazon Web Services: a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow. 11

back end the database or logic of a system, often is not directly accessible by the user. 6, 11, 22, 24, 25, 26

front end a client which is usually a device or service that is used to communicate between the user and the system's database or logic (back end). 3, 6, 11, 22, 23, 24, 26

S3 Simple Storage System: Storage of files and information on the cloud. Part of AWS. 11