Data Classifier: Software Architecture version 2.0

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

December 5, 2018

Contents

Credits

Name	**Date**	**Role**	**Version**
Joey Buelow	November 7, 2018	Document Owner	1.0
Kent Tran	November 7, 2018	Document Owner	1.0
Bonita Galvan	November 7, 2018	Document Owner	1.0
Diana Chiu	November 7, 2018	Document Owner	1.0
Victoria Law	November 7, 2018	Document Owner	1.0
Tanner Villarete	November 7, 2018	Document Owner	1.0

Revision History

Name	**Date**	**Reason for Changes**	**Version**
Joey Buelow	November 7, 2018	Introduction and Initial baseline	1.0
Kent Tran	November 7, 2018	Problem Description	1.0
Bonita Galvan	November 7, 2018	Editing and Image Formatting	1.0
Bonita Galvan	December 5, 2018	Updated Diagrams	2.0
Diana Chiu	December 5, 2018	Updated Diagrams	2.0
Victoria Law	December 5, 2018	Updated Diagrams	2.0
Kent Tran	December 5, 2018	Updated Diagrams	2.0

1 Introduction

This document provides a detailed description of the system architecture for the Data Classifier component of the Mark Logic Data Discovery Workbench. Contained in this document is an outline for the development of the aforementioned system. This includes an overall description of the system, including diagrams, definitions for the components of the system, design considerations, testing plan, and issues.

2 Problem Description

The system is a data classifier that uses machine learning to categorize input data and outputs the data formatted into the appropriate categories. Additional features of the system include editing the generated categories, marking specific data fields as sensitive, and a web-based user interface.

3 Solution

The solution is to create a user interface that allows a user to categorize data using machine learning to improve future use.

3.1 Overview

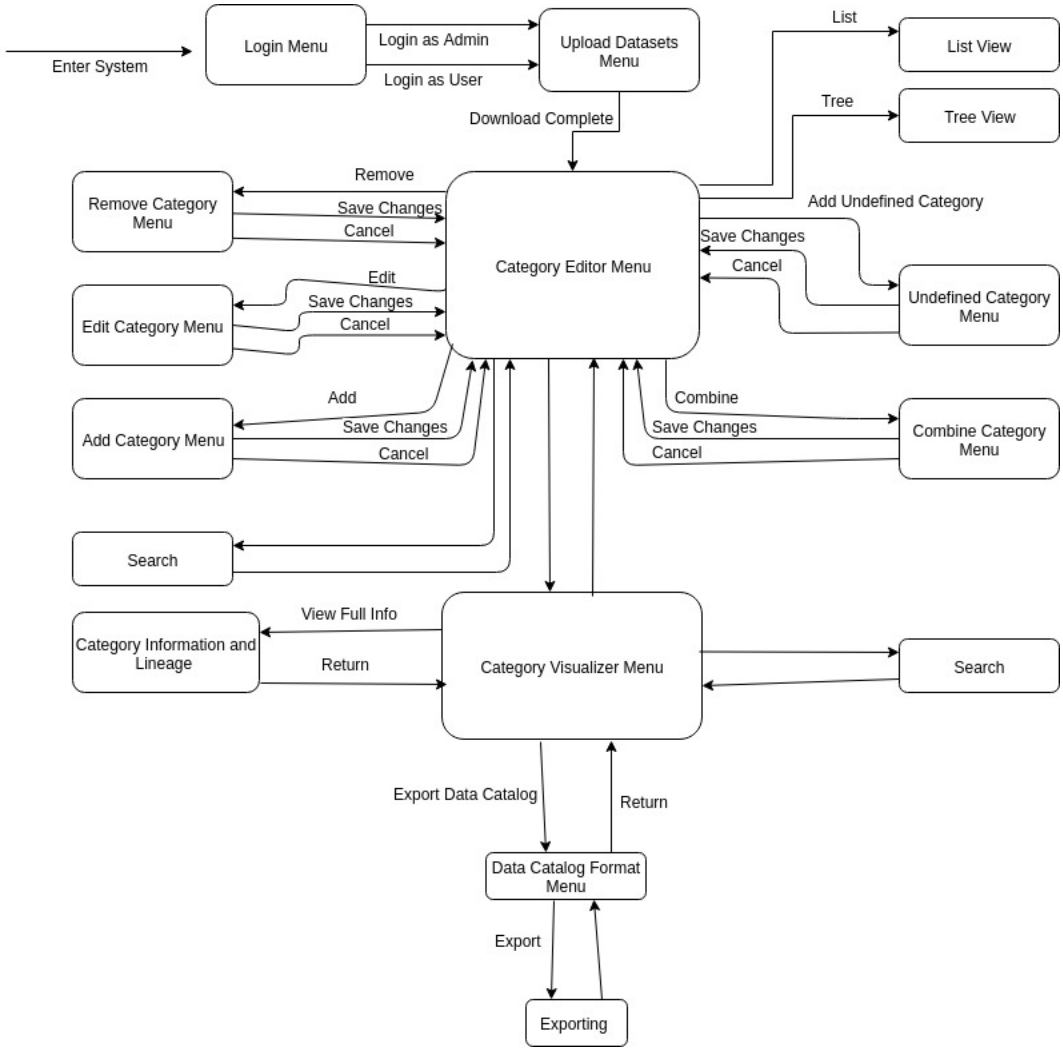
The current plan is to have a web-based user interface. It will connect to cloud computing resources such as AWS S3 storage and utilize machine learning libraries such as Python SciKit to execute the machine learning components.

3.2 Components

The system is primarily composed of three components: the front-end, the back-end, and the machine learning. Since the goal is to have a web-based user interface, React and Redux will likely be used to develop. Back end systems as well as the machine learning will be in Python to best incorporate the necessary external libraries.

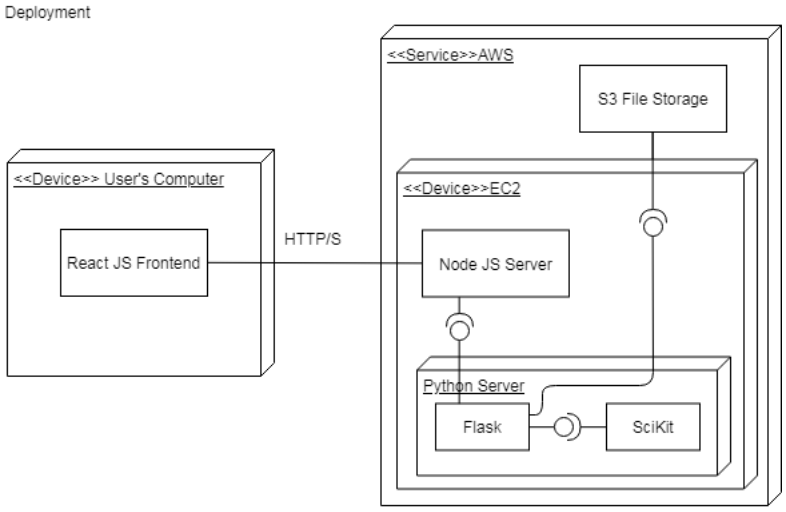
3.3 Design

Bonita Dialog Map: The diagram is a comprehensive overview of the programs UI. The user would start by entering the system onto a login page. From there they would be able to upload data sets that would then under go processing. Our program would continue to the category editor page where the user could edit, add, remove, combine, search for, and define undefined categories. After the user has completed editing they can move on to the final category visualizer in which they could search for specific categories and examine data lineage. Finally the user would be able to export the information as a data catalog.

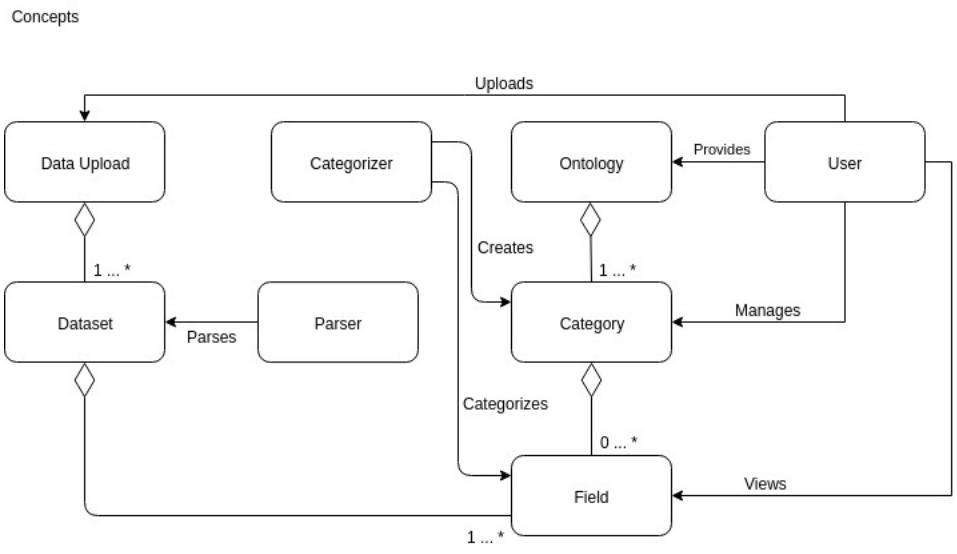


Diana Deployment Diagram: This diagram details the physical deployment of the related

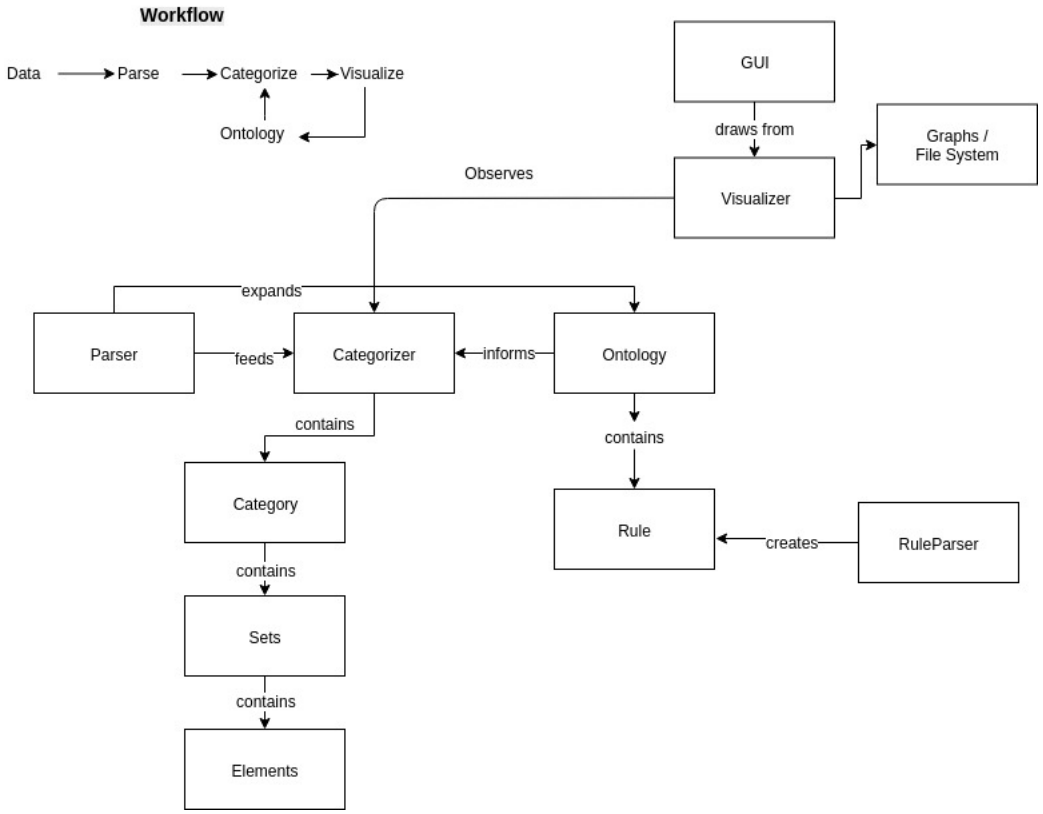
artifacts. The React JS Frontend will run on the user's computer and use http/https to connect to the Python/Flask API which will route via REST to the rest of the components.



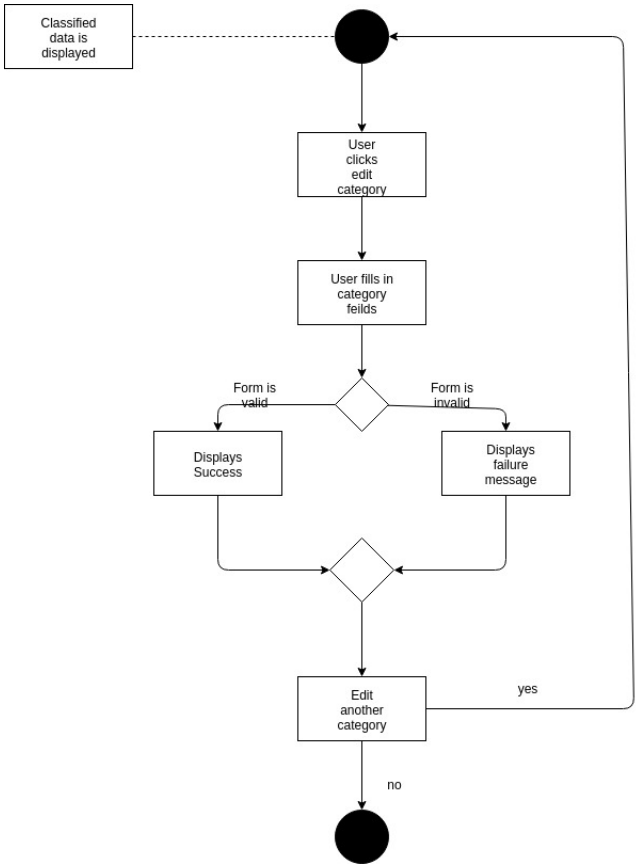
Diana Class Diagram: The class diagram maps concepts related to the Data Workbench instead of concrete classes.



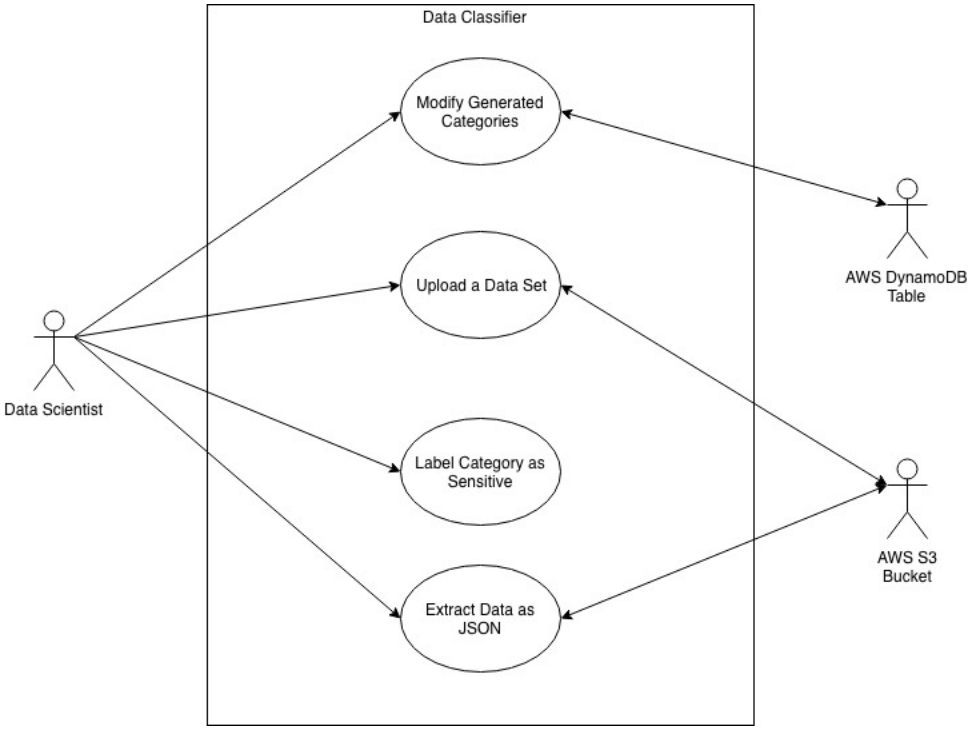
Joey Architecture Diagram: This diagram represents a high level, informal depiction of the major pieces the solution, demonstrating what components will interact with others and what the nature of that interaction is.



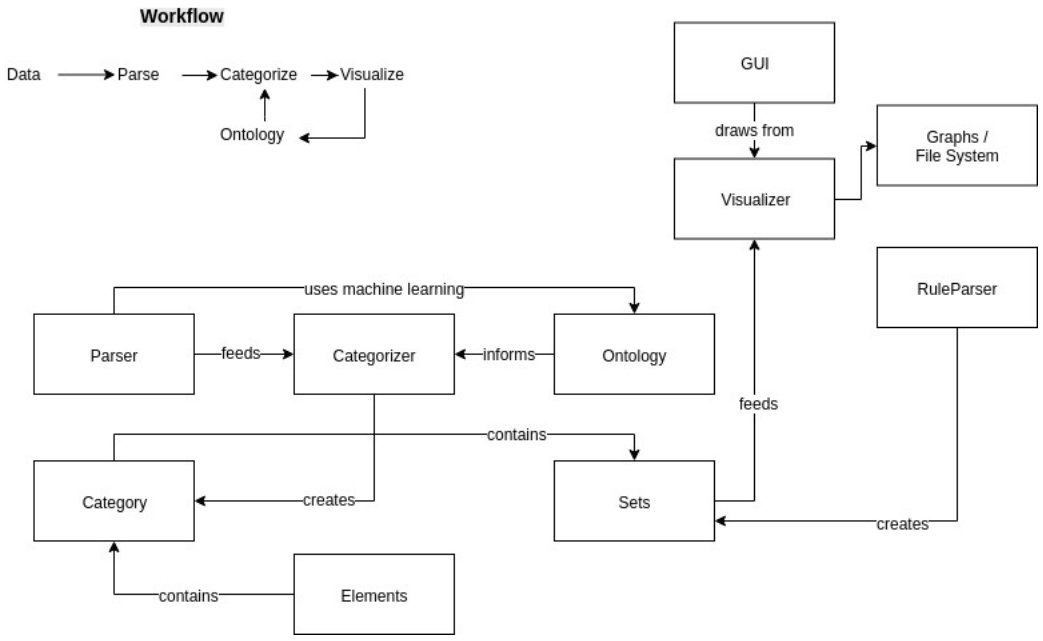
Joey Activity Diagram: This diagram describes the flow of the application as experienced by the user if said user wants to edit a category definition. The precondition for this activity is that classified data is being displayed by the visualizer. The user clicks "Edit Category". The system displays the category fields and allows the user to edit them. The user then commits the desired edits and clicks "save". The system then checks the form to determine if valid configuration was entered. If valid configuration was entered the user is displayed a success message, else the user is shown a failure message. The user is then presented with the option to edit another category or quit.



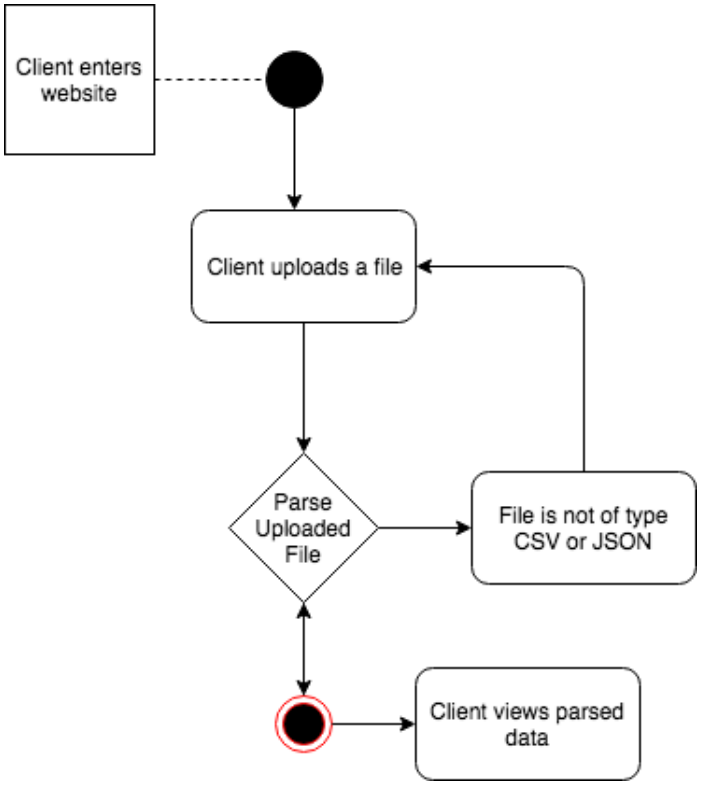
Kent Use Case Diagram: In this diagram, several use cases of that a data scientist may need are showcased. In addition, certain use cases' interactions with data stores are also provided.



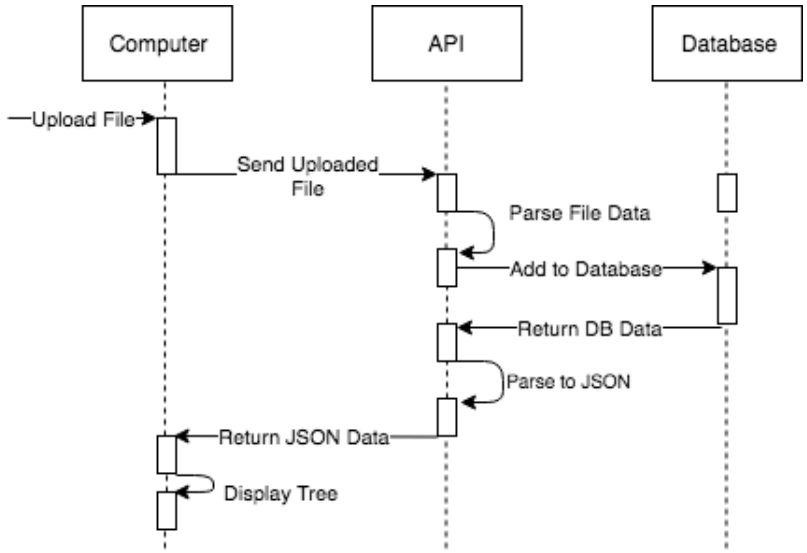
Kent Architecture Diagram: The smaller Workflow diagram in the upper left corner shows a high level overview of what the general flow of the system will look like. The larger architecture diagram goes into more detail with components within the class.



Tanner Activity Diagram: This activity diagrams outlines the flow of a user who is uploading a file to the data classifier. It showcases what would occur should the uploaded file be in an unsupported format and how the system would behave accordingly.

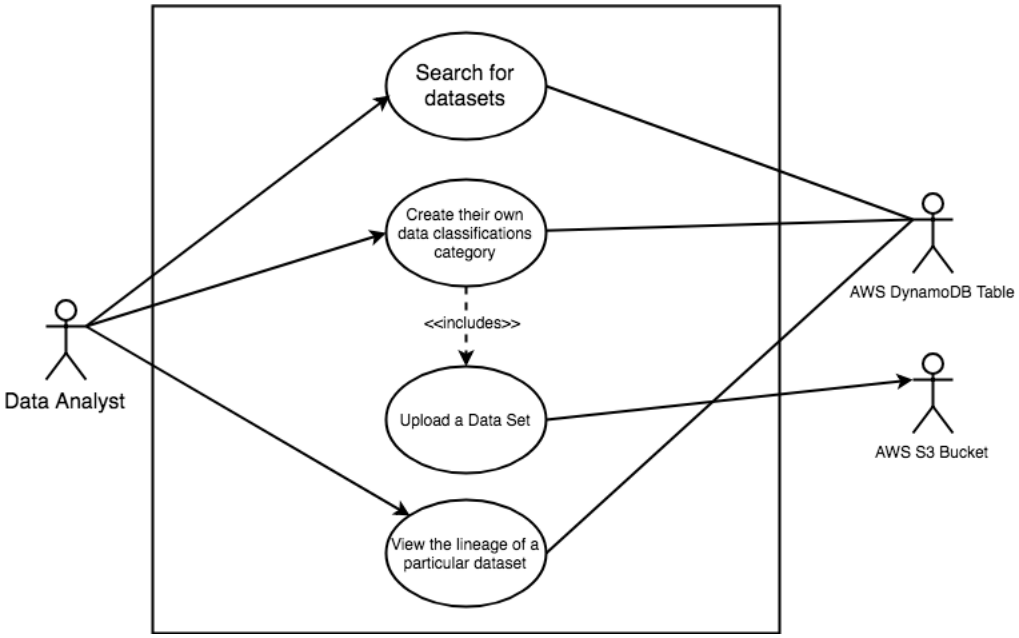


Tanner Sequence Diagram: In this sequence diagram, the series of calls are illustrated between the computer, API, and database. Data sent and returned are displayed on each of the arrows as is what call is to follow.



Victoria Activity Diagram: The activity diagram represents the flow of a user, which in this case is a data analyst, searching for certain data sets. Before being able to search for certain data sets, the system expects data sets are pre-loaded into the data classification model. From there the data analyst enters the search string and clicks the search button. It will either route to "Results Found" or "No Results Found". If results are found, then it will display the results in a list format, and there will be a search again button at the bottom of the page if the displayed results isn't what the user is looking for. If no results are found, it will ask the user if they would like to search again.

Victoria Use Case Diagram: This use case diagram represents possible uses a data analyst may be interested in doing. A data analyst may want to search for data sets in the AWS DynamoDB table, or view the lineage of a particular data set. In addition, it shows where the data sets are going, or where the data sets are retrieved from per use case.



4 Test

Although implementation is still in the early stages, the current testing plan is to provide a suite of unit tests as well as overarching integration tests. With the use of Travis as our continuous integration framework, the tests will be run each time a commit is done to ensure testing occurs often and incrementally.

5 Issues

As the first quarter comes to an end, no outstanding issues are present.

A Glossary

AWS	Amazon Web Service; cloud based web hosting
Category	A collection of like elements defined, discovered, and maintained by the Mark Logic Discovery Workbench.
Visualizer	Internal object that creates the model for displaying the categorized data.
Ontology	A set of rules that defines which forms of raw data qualify for which data categories