Data Classifier: Software Architecture Version 1.0

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

November 7, 2018

Contents

Credits

Name	Date	Role	Version
Matt Yamolich	November 7, 2018	Contributing Member	1.0
Spencer Schurk	November 7, 2018	Lead Author of Introduction, Upload Data Testing	1.0
Geraldo Macias	November 7, 2018	Lead Author of Overview and Co-Author of Component	1.0
Jake Veazey	November 7, 2018	Lead Author of Login/Signup and Co-Author of Component, Some Testing Documentation	1.0

Revision History

Name	Date	Reason for Changes	Version
Spencer Schurk	November 7, 2018	Completed Introduction, Upload Data Testing, and two diagrams	1.0
Jake Veazey	November 7, 2018	Updated the project problem	1.0

1 Introduction

The purpose of this document is to describe the architecture of our Data Classifier. This document will be used as a reference when development begins. This document may change over time as we learn more about specific technologies being used for this project. For more information regarding this project, please see the Vision and Scope document, as well as the SRS document.

2 Problem Description

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.

3 Solution

This project is meant as an interface between a database and the data being inputted to it by the user. The main goal of this project is to provide analytics for the user in order to better identify what type of data is being inputted. With this project, the user will be able to tell the categories of data being inputted, and how they potentially relate to each other. The classifier will allow the user to modify these relations and to interact with a visual model based on these interactions. As defined by the customer, the user will interact with a web based GUI, most likely based on React. This GUI will interface with the Python backend, which is another design restraint, and utilize Scikit, a python library, as our chosen machine classifier. This classifier will act as the main interface/buffer between the user and actually logging the data to the database.

3.1 Overview

The system will allow a user to:

1. Select proper classification category
2. Upload datasets from a Data Silo
3. Log in and log out
4. View the classifier logs

5. Export data logs

The System will allow the System Administrator to:

1. Create user account
2. Delete user account

3.1.1 Use Case Diagram

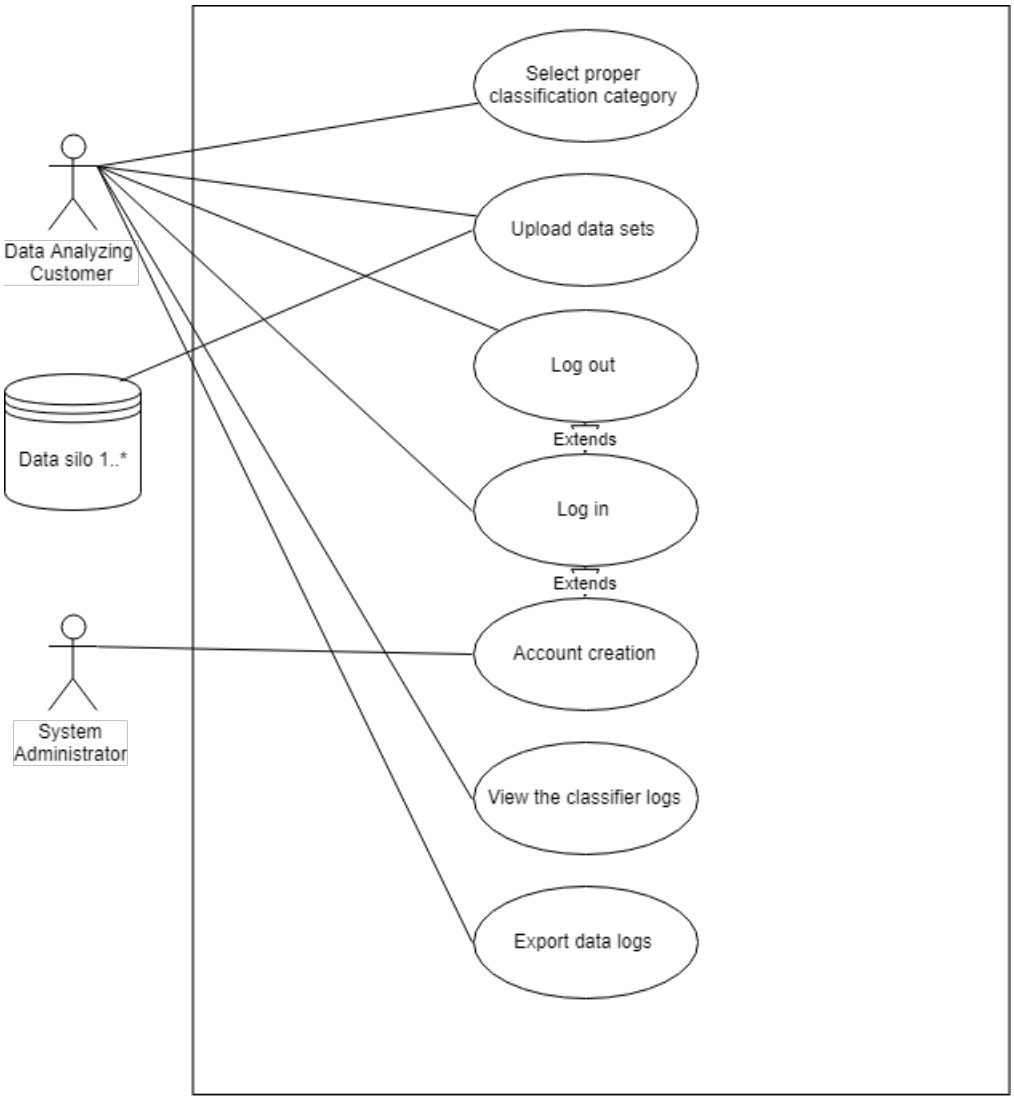

Figure 1: Use Case Diagram - Geraldo Macias

3.2 Components

The Account Management component will allow a System Administrator to create User accounts. A user cannot crate accounts by themselves and System Administrator cannot

view the data. The user account will store basic information and allow the user to log in and log out.

The Data Management component will allow a user to explore and select files that was been uploaded and/or processed and execute processes on them. The user will be able to send the file(s) through the Classifier Component and return a classified file(s). A user can then export any files, including files that have been classified or not classified.

The Classifier Component will receive files from the Data Management component and classify the data according to content and column name. If a dataset without column names is provided, the Classifier will attempt to classify the columns. If under a 80% confidence level, the classifier will ask the user to supply a proper name for the column.

3.2.1 Component Diagram

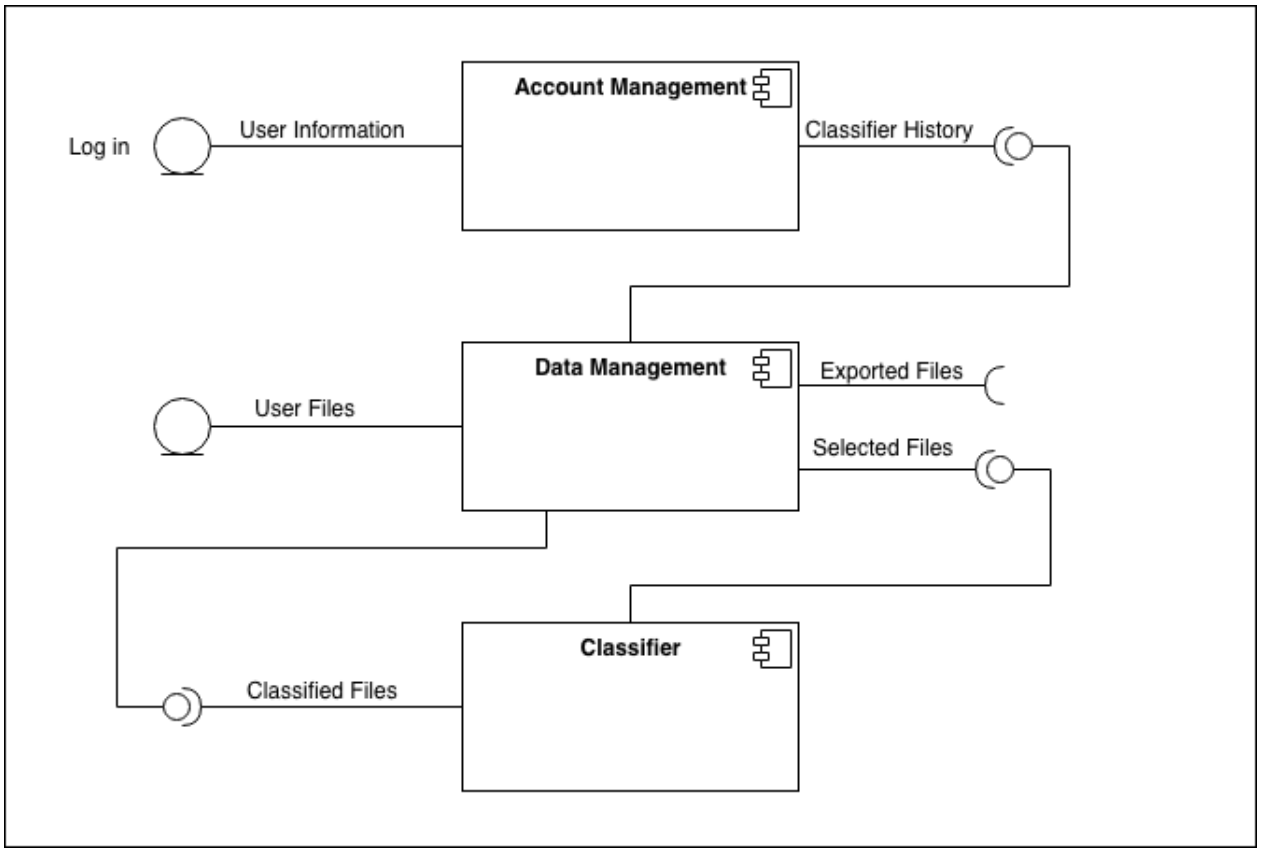

Figure 2: Component Diagram - Geraldo Macias & Jake Veazey

3.3 Design

[This is where the meat of the design lives. For each component, there should be a subsection describing the design of that component in as much detail as you want to provide in a high-level design. There should also be a subsection that describes how the

components work together. This section should include class, sequence, and collaboration diagrams as appropriate.]

TBD

3.3.1 Login & Create an Account State Diagram

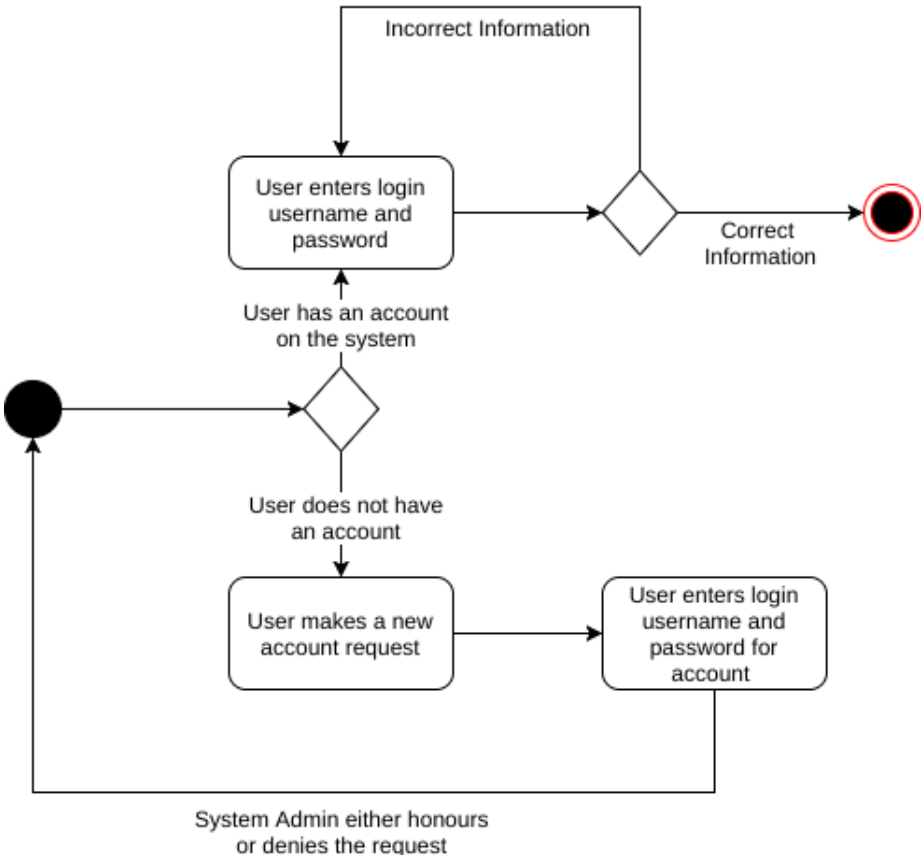

Figure 3: Login and Create Account State Diagram - Jake Veazey

Figure 2 shows the state diagrams of the user's first interaction with the service – to login to the system or ask the system admins to create an account. The user enters the service and enters their information (username and password) or asks for a new account. If the user already has an account, they can input their information. If it is correct, they gain access to the Data Management system for their account, where they can use the classifier or export. If it is not correct, they are prompted with the opening screen again to try information again.

3.3.2 Classification Structure

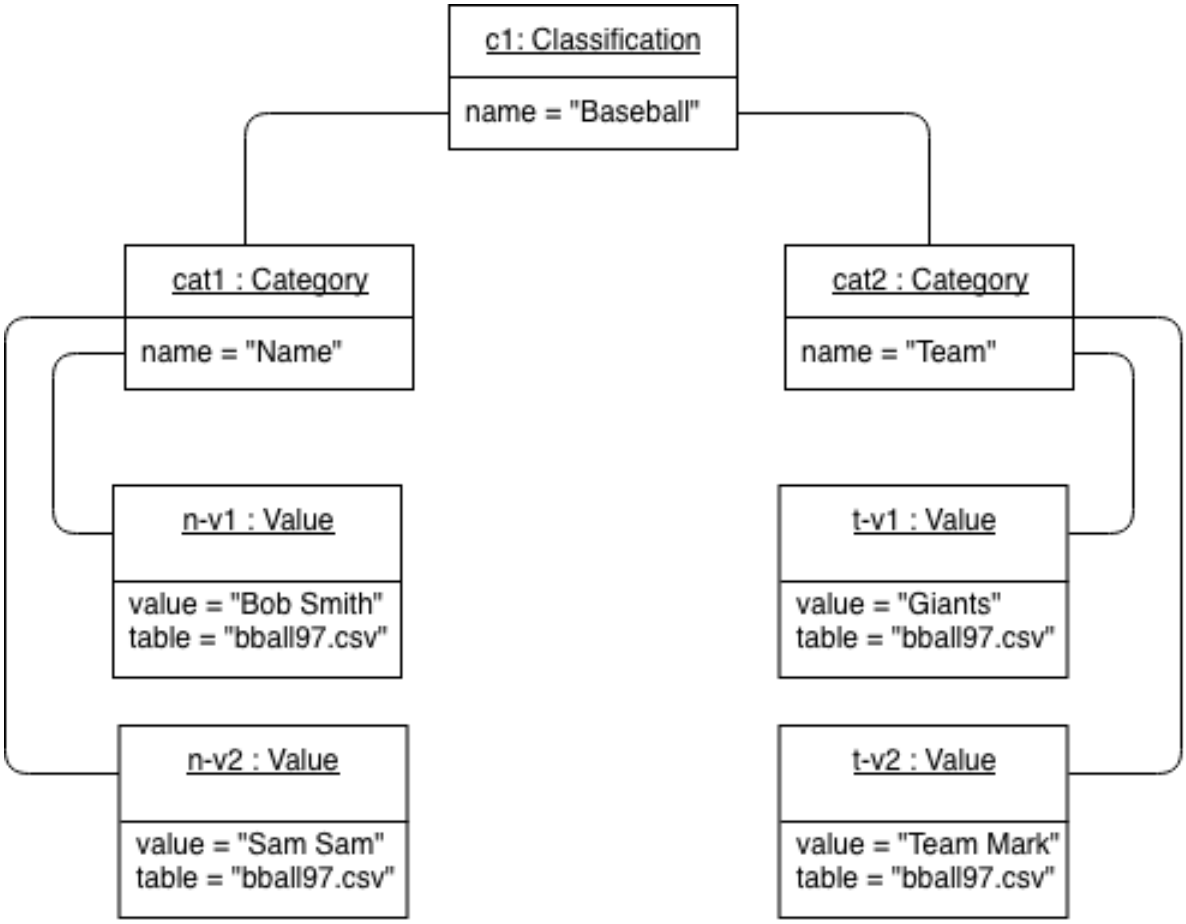

Figure 4: Data Classification Object Diagram - Spencer Schurk

Figure 3 depicts the architecture of our Classification object. Once data classification has completed on the provided files, the finalized result will be structured like this. A classification consists of one Classification object, which contains a String called name, and one to many Categories.

Categories are what the classified data is sorted into by the machine learning algorithms. A category consists of a String called name, and one to many values.

Values are the object representation of one value in a data set. The Value object contains the value, and a reference to the table in which it came from. More attributes may be stored in the Value object as necessary.

3.3.3 Uploading Files

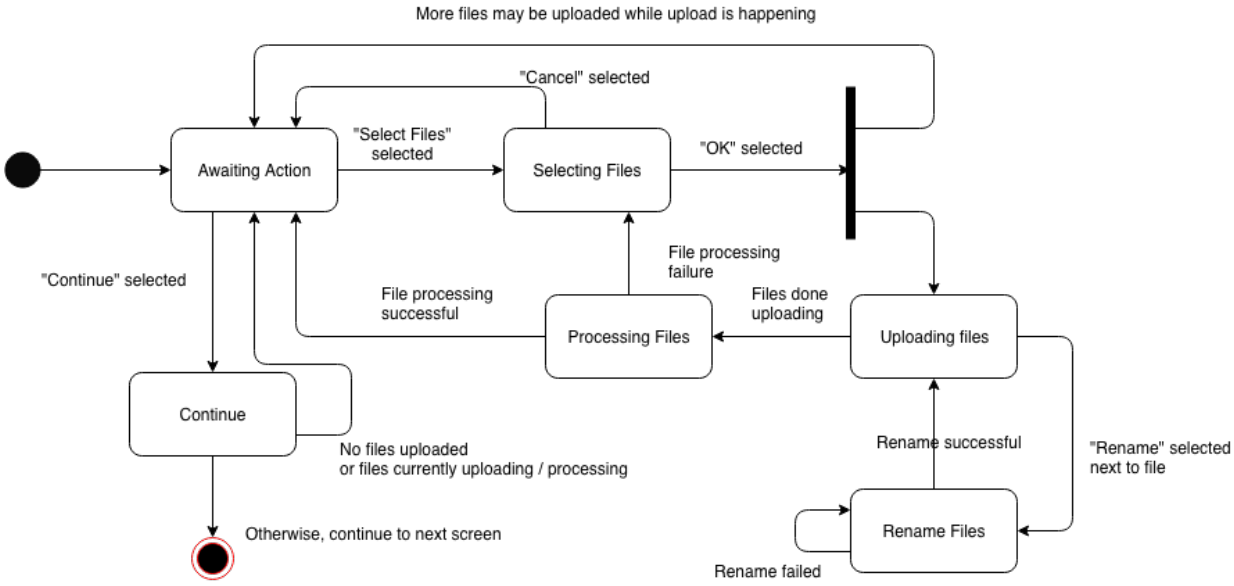

Figure 5: Upload Files State Diagram - Spencer Schurk

Figure 4 models the sequence of the data through the system, and the interaction between the user and the system. Users will begin their interaction with the UI by uploading files to the front-end. Then it will be processed into a temporary holding database for processing, fed to the machine learning algorithm, then the resulting data will be displayed by the UI.

Since this process can happen multiple times, there must be a cleanup for purging the temporary database so that unwanted/sensitive data is not stored permanently in an potentially unsecured/unknown entry.

This process will then repeat per data entry uploaded by the user and can be repeated as many times as the user desires.

3.3.4 Overview of Classes

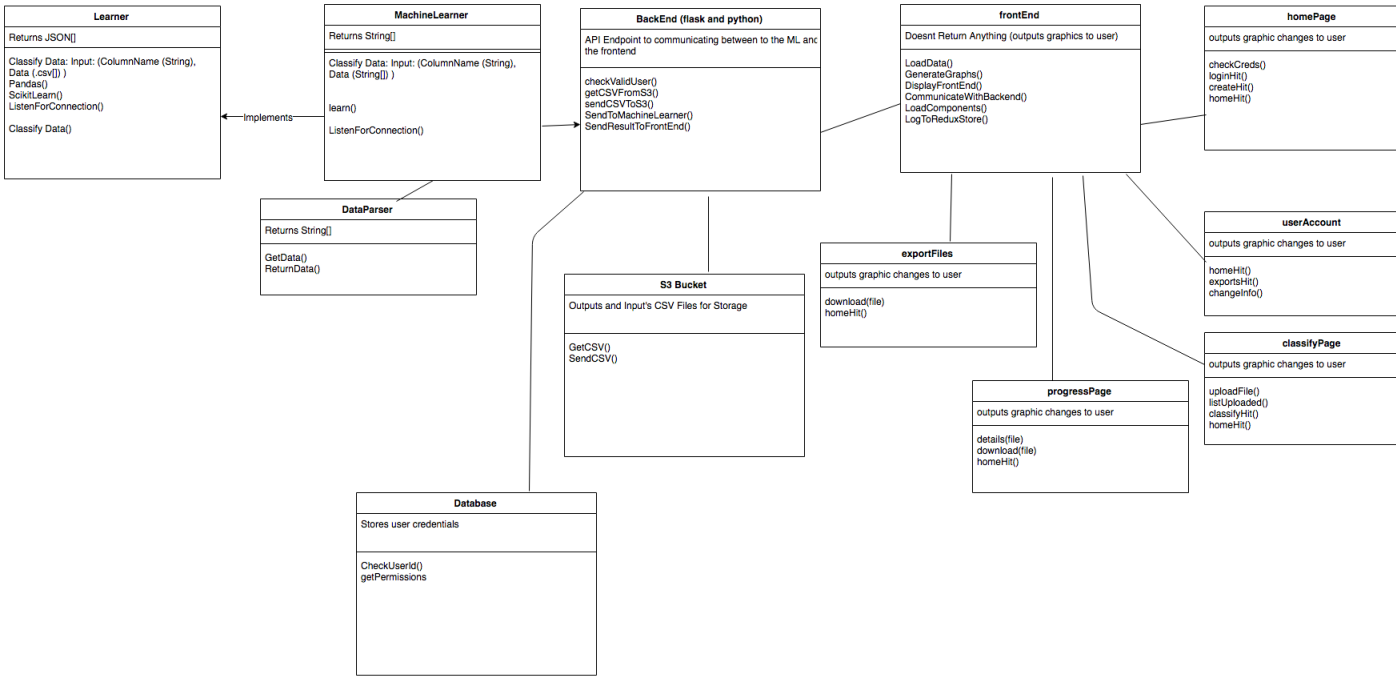

Figure 6: Overview of Classes - Matt Yarmolich

Figure 5 depicts a state diagram modeling the user flow when uploading files to be classified. Users will enter this page from the main menu, and leaving this page will take users to the start of the classification process. When leaving this state, all files that were in the process of being uploaded must be completed or canceled.

Since users may select more files to upload while some files are already in the process of uploading, a fork is used in this diagram. This means that while files are uploading, a user may select more files to upload, then begin uploading those files. Selecting "continue" will not continue to the next page unless all files have completely uploaded or have been canceled.

The processing files state may fail for numerous reasons, such as an incompatible file type, or a corrupt file. When this occurs, that file is deleted from the system, and users may select another file to upload, or cancel.

3.3.5 Flow of Data

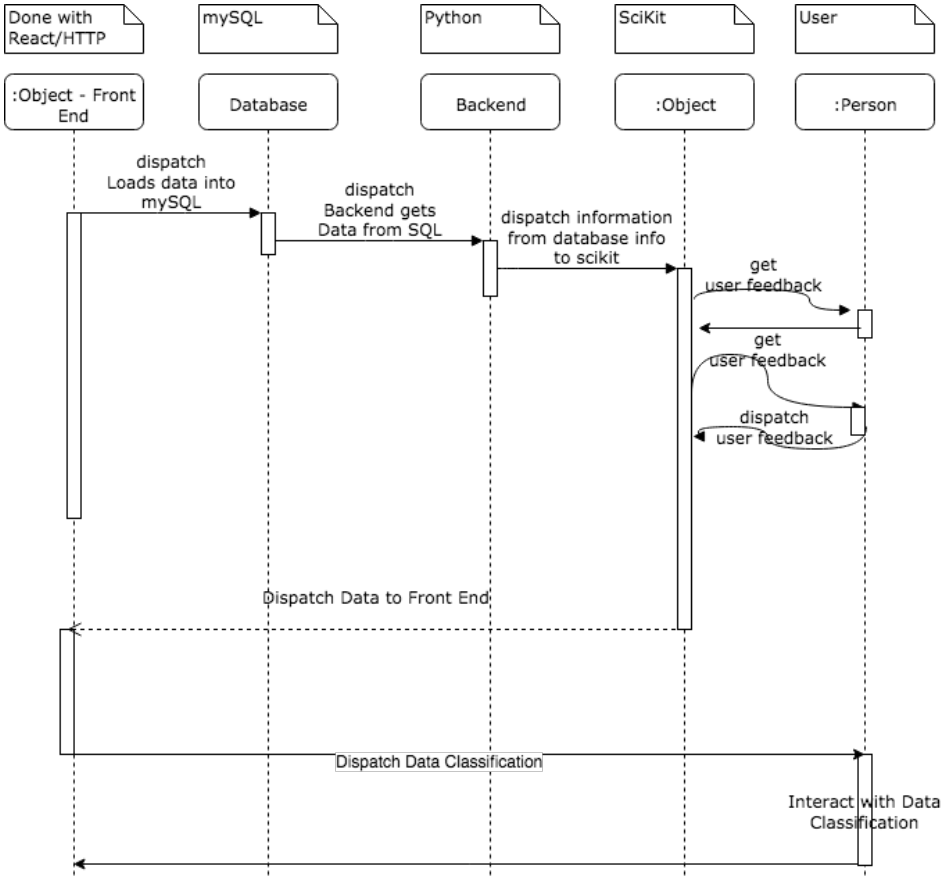

Figure 7: Sequence Diagram of Data - Matt Yarmolich

Figure 6 gives an overview of the classes that will make up the designed system. The main parts of this diagram pertain to the front end, back end, and the machine learning aspect of the project. This diagram will be used to model our architecture of our project and to determine how all the pieces of the system fit together.

All of these components will be designed to be reusable so that they can be strung together and be reused depending on the users use-case. Unfortunately, the languages we have picked do not lend themselves particularly well for object oriented programming (python is a scripted language and react is based on javascript) so we ultimately decided to split these into files.

3.3.6 Flow of Data

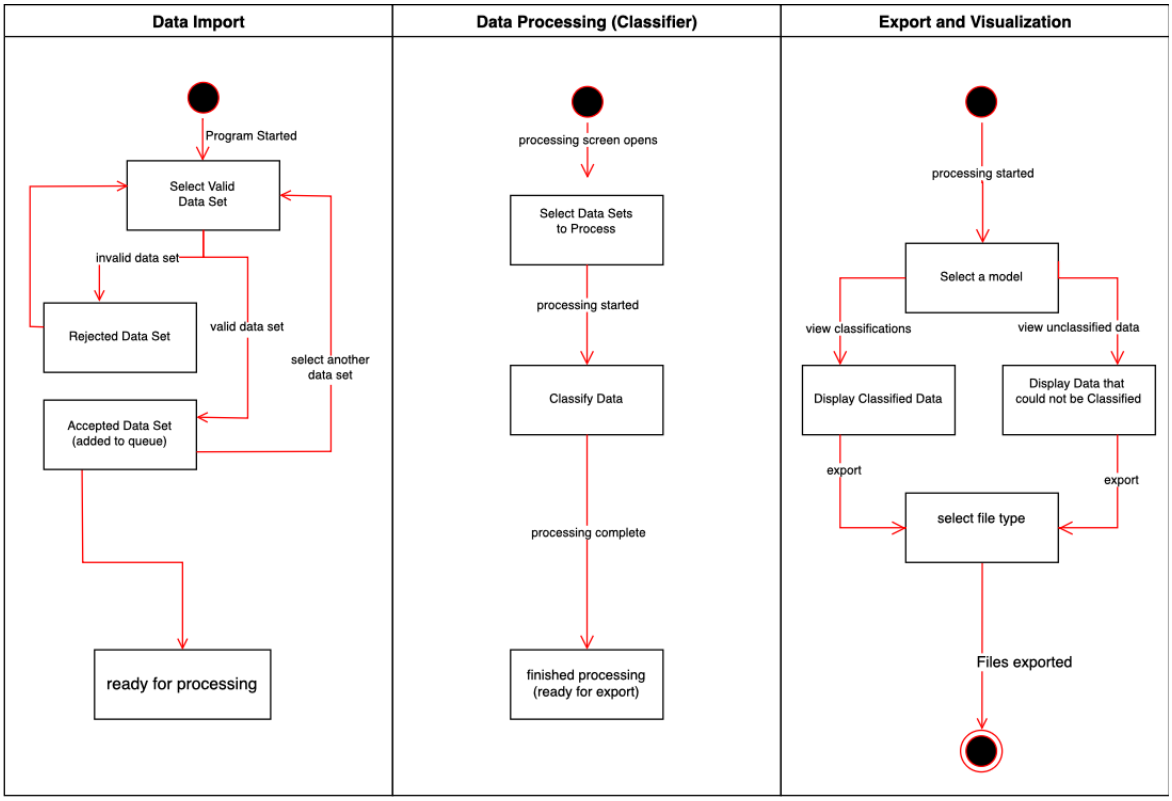

Figure 8: State Diagram of System Overview - Landon Gerrits

Figure 7 Gives an overview of the main flow of our system. The diagram is grouped into 3 sections, each showing each the critical states at each phase of the system. Our other diagrams go into further detail of how each phases will behave (i.e. See Figure 3 for a more detailed breakdown of file upload)

Our machine learning model is still mostly a "black box" at the moment. We do not have an architecture for how we will process our input data. For now, this component exists as "classify data" in the center section of the diagram.

3.3.7 Edit Data Classifications

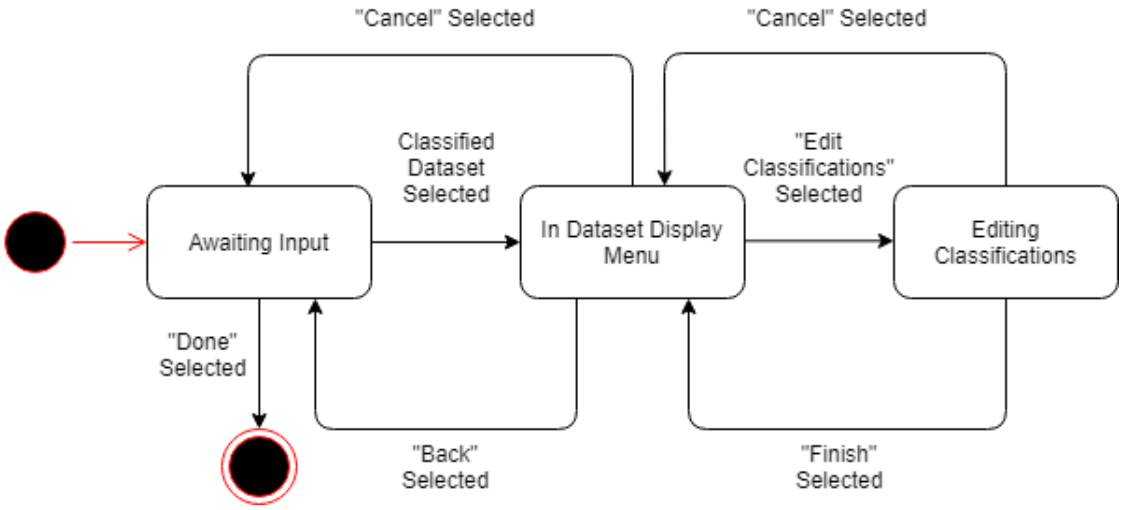

Figure 9: State Diagram for Editing Data Classifications - Brad Foster

Figure 8 shows an overview of the activity flow for editing data classifactions. The user first selects a dataset to edit. After that, the user may click an "Edit Classifications" button which will allow them to edit the names (and possibly other settings) of the classifications. After the user is finished, they may click the "done" (or "back") button to navigate backwards through the activity flow.

3.3.8 Deployment Diagram

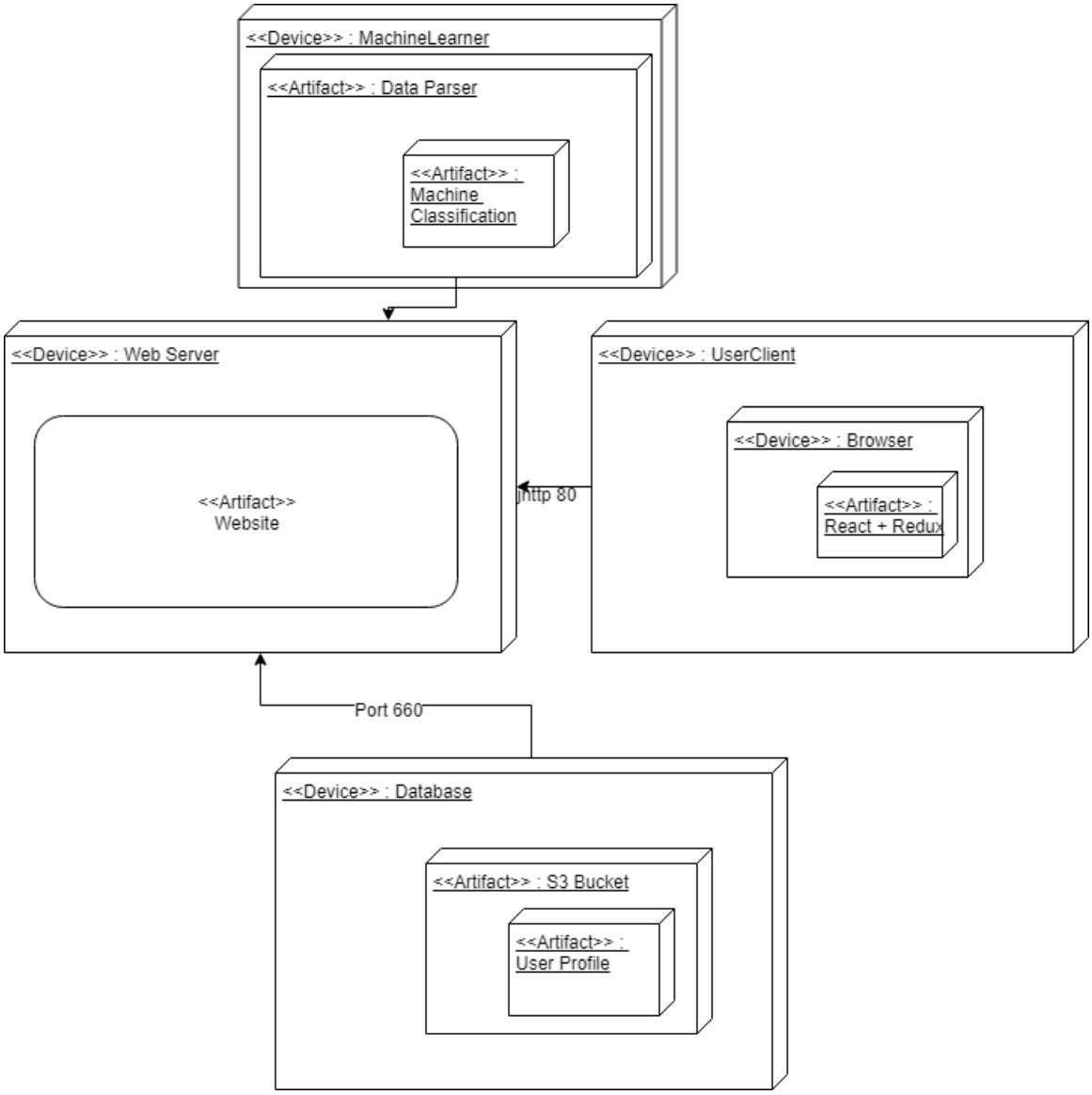

Figure 10: State Diagram for Editing Data Classifications - Brad Foster

Figure 9 shows the projected architecture for deploying the Data Classifier Software. The design is expected to make use of a web server and a database for the back-end. The user interface will be web based, and the Data Classifier Software will make use of a machine learning component.

3.3.9 Visualization and Export

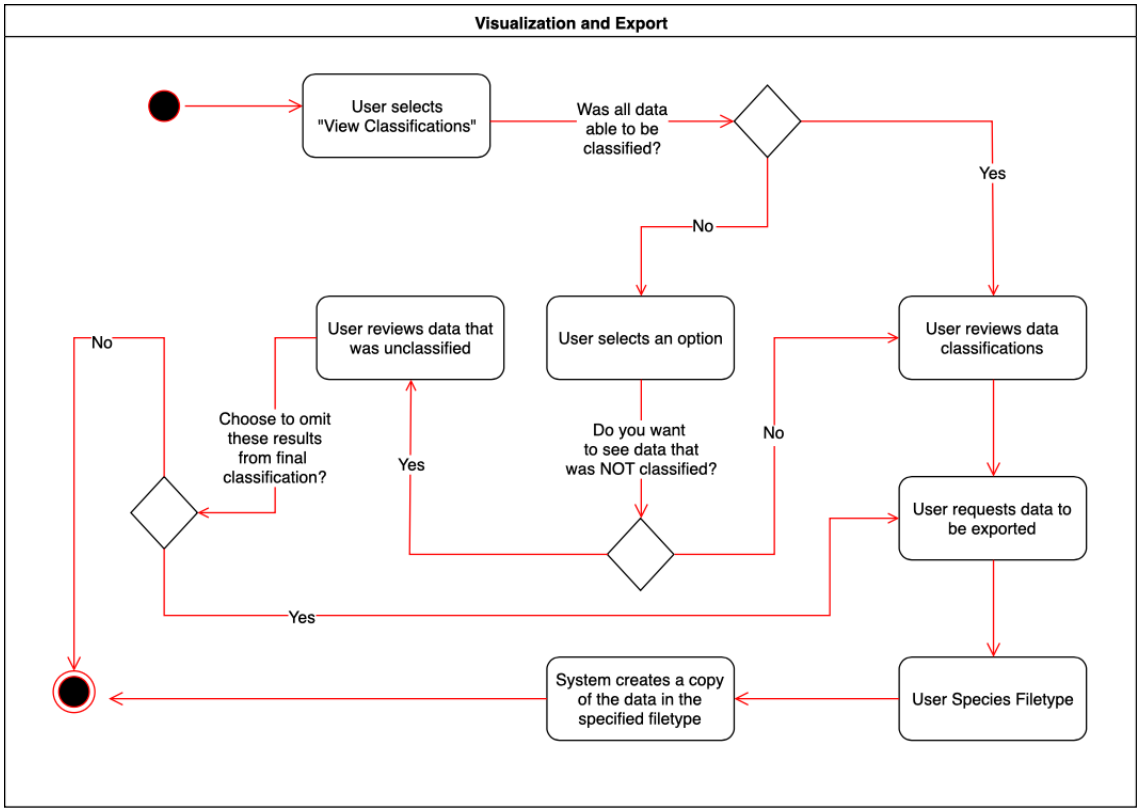

Figure 11: Activity Diagram for Data Visualization and Export - Landon Gerrits

Figure 10 shows the interaction between the user and the system when choosing to view and export data that has been classified (or data that could not be classified). The user can choose to review the data classifications. At this time, we do not know how we will present this data to the user. One proposed solution was to use Google's Firebase platform to parse our classified JSON file. After viewing the data, the user can select a file type for export.

In the case where the system cannot classify a subset of the data, the user has the option to omit this data from the export, meaning that there will be some amount of data found in the input data set that will NOT be in the exported (classified) data set. If the user chooses to NOT omit this unclassified data, then then the system will return to the original data import screen.

4 Test

Since our solution includes multiple components written in different languages, testing should almost exclusively be written separately for each individual component, with system-wide acceptance testing completed at the end. Described below are testing methods for each component.

4.1 Login & Signup

In order to test the UI for the login, an automated testing tool can be used, but manual testing will be just as effective and easier to implement as no additional code will be required for testing. The back end will be tested by providing the login API correct and incorrect information to be queried from the database. The results of an approved and logged in acoount or an error will tell us whether testing was successful. Signing up for an account will be similar. The tester will provide a login and a password that will be passed to an API. The API will notify the admin of a request. Testing will be looking for the correct information passed in.

4.2 Data Upload

Testing should be performed with all supported data types. Testing should also be performed with unsupported data types to ensure errors are being caught gracefully. Corrupted or improperly-formatted files should be tested as well, since these will also produce errors. This component should also be tested with bulk-uploading, as users may want to upload a large amount of data files at a time.

Once files have successfully been uploaded, their processed output should be tested for consistency with the source file. Ensure that no values are missing, and the data follows the same structure as the source file.

Additionally, all other functionality of the Upload Files screen should be functionally tested. Ensure that all buttons work in all states, and users do not get trapped in any states due to unhandled exceptions.

5 Issues

[Any issues or open questions should be described here.]

A Glossary

Define all the terms necessary to properly interpret the software architecture, 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 software architecture.

B Issues List

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