Requirement Analysis Document

v1

1. Brief Description

1.1 Product Perspective

The program "Labeling Mechanism" classifies some groups of instances according to predetermined labels by human experts.

1.2 Product Functions

The product will simulate “Labeling Mechanism” by using the given information. The pre given input files (i.e input.json and config.json) will be parsed in order to be processed in our product. The Mechanism will label the instances that were given in the input file with one or more class labels. When the mechanism processes a task, the logging mechanism will give information about which user added which labels to the given instances and when it happened. After the execution of the labeling mechanism, all labeling that is made by the system will be written in an output file.

1.3 User Characteristics

Users can attach single or multiple labels to the instances whenever he/she/it wants via an input file.

1.4 General Constraints

Since the system is meant to be run by itself, the product can’t be interrupted in any process when the user gives the command to execute it.

1.5 Assumptions and Dependencies

The product can be run on any operating system since it is programmed in Java programming language. There is no dependency.

2. Glossary of Terms

assigned instance: the instance that is labelled by the mechanism.

dataset: the object representation of the input file.

instance: the data that can be processed by the mechanism.

label: tags that will be used to stamp data.

logger: logs the events of the program execution at any time automatically.

parser: decomposes the input file to the readable code and objects.

configset: decomposses the config file to the readable code and objects.

user: the agent that labels the data.

3. Functional Requirements

3.1 Inputs

Required Json files to store predetermined labels and instances that are user provided and get the information about users that will take place in attaching labels.

3.2 Processing

The data obtained from JSon files will be processed via user execution command to the labeling mechanism which will execute the whole process.

3.3 Outputs

When a user wants to add labels to instances, we will show all the details via logging the process. Ultimately, when all instances are undergoing the process, the program will write the values to an output json file.

4. Non-Functional Requirements

4.1 Performance and scalability

The system returns the results almost immediately and our modular software architecture allows us to scale and improve the program.So it should still be functional in higher workloads.

4.2 Portability and compatibility

Since the program is written in Java and Java runs on almost any device on the planet.

4.3 Reliability, availability, maintainability

As long as the configuration files are correct there is no risk of any failure.

4.4 Security

Since the system is %100 offline there are no security vulnerabilities.

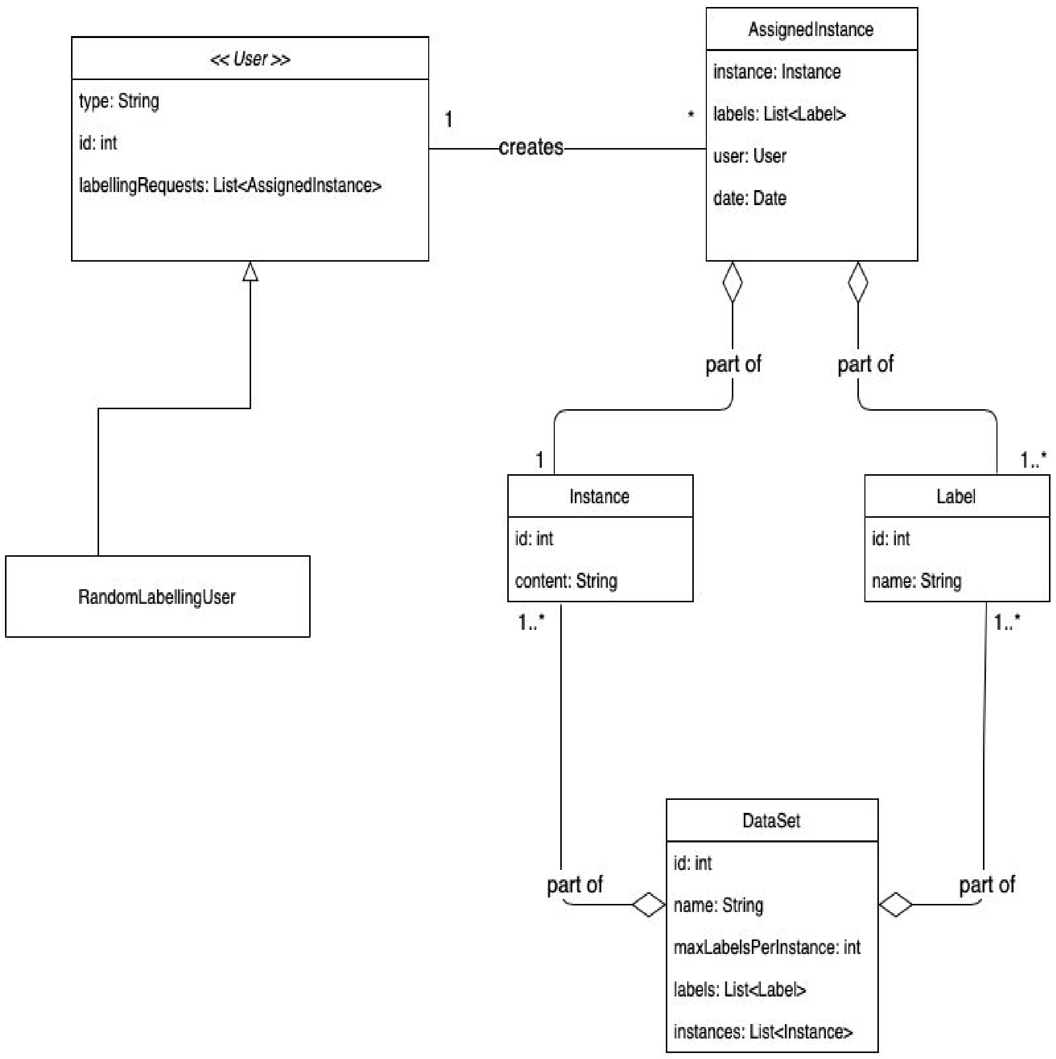
4.5 Localization

Since the system takes the labels and instances from the input file it can be configured to any localization the user desires.

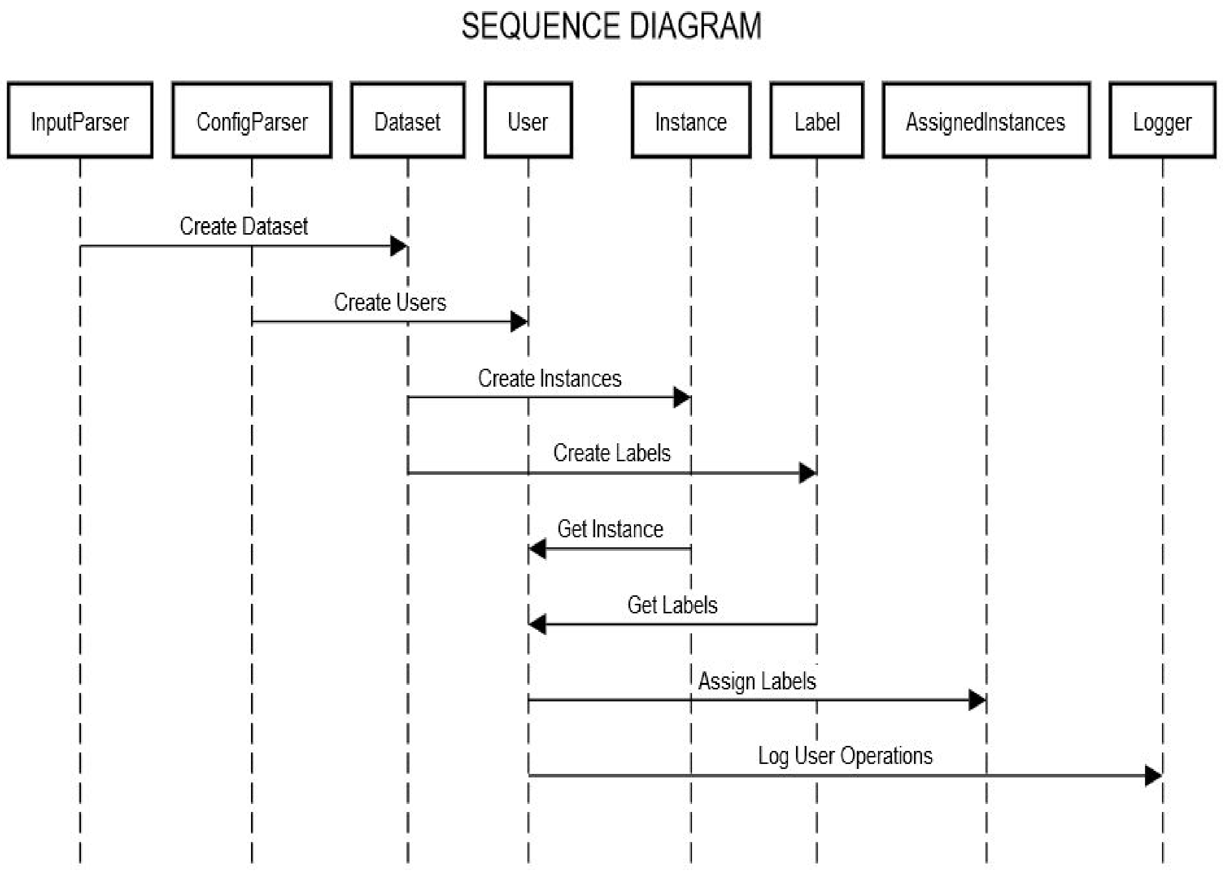
4.6 Usability

Since there is no interface that the user can interact with, utilization is very simple as long as the required input files are provided correctly.

Domain Model



System Sequence Diagram


SEQUENCE DIAGRAM
InputParser
ConfigParser
Dataset
User
Instance
Label
AssignedInstances
Logger
Create Dataset
Create Users
Create Instances
Create Labels
Get Instance
Get Labels
Assign Labels
Log User Operations