# Language Detection Web Application – Duck Soft Works § Co.

## US 2 – As a user I want to create a text language analysis task, passing a URL (which points to a .txt file), a category and a timeout.

## 1. Requirements Engineering

### 1.1. Customer Specifications and Clarifications

**Q:** What does the client mean when he says "to obtain information about the language in which the text was written"?
**R:** It is the purpose of the task launched by the user. The user wants to identify the language of a text.

**Q:** Is there a minimum or maximum size of texts to be analyzed?

**R:** Does not exist a limit.

**Q:** How many languages should be covered, and if so, which ones?

**R:** At least 3 languages must be covered: Portuguese, Spanish and English

**Q:** Is there any maximum limit for the Task Timeout?

**R:** This question has already been asked. Since you are being so insistent, consider the maximum limit of 5 minutes.

**And from the specifications:**

The text to be validated must be stored in a text file and must be identified by the URL that allows it to be located, and by one of the existing categories in the information system.

### 1.2. Acceptance Criteria

- To create task analysis it is necessary to be logged in with an user with user privileges

- The analysis of the text must be performed asynchronously so that the user is not left waiting for the analysis to be completed

- For language identification services, client suggests the use of dictionaries from the Aspell project and the Lucene project and the use of the concepts of similarity between documents.

- Inserted URL must have a .txt extension

## 1.3. Found out Dependencies

- To create a task, it is necessary to insert a category previously created by the system administrator.

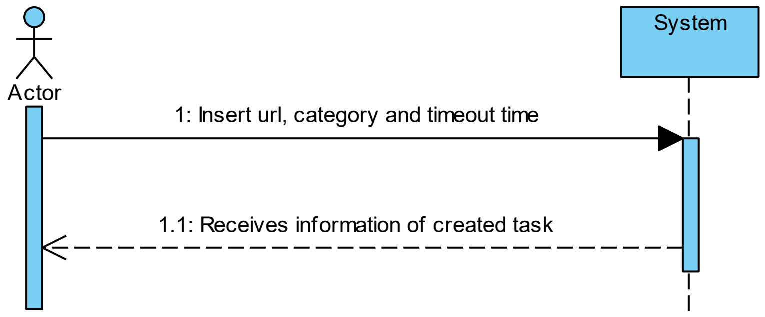
To be able to create a task, a user must provide a valid text URL that is not contained in the BlackList. For that reason, the URL must always be compared to all the elements in that list to make sure it has permission to continue to the language detection analysis.

It is also always necessary for the user to send a time limit (timeout) for the analysis of the language. This time should be between 1 and 5 minutes.

## 1.4. Input and Output Data

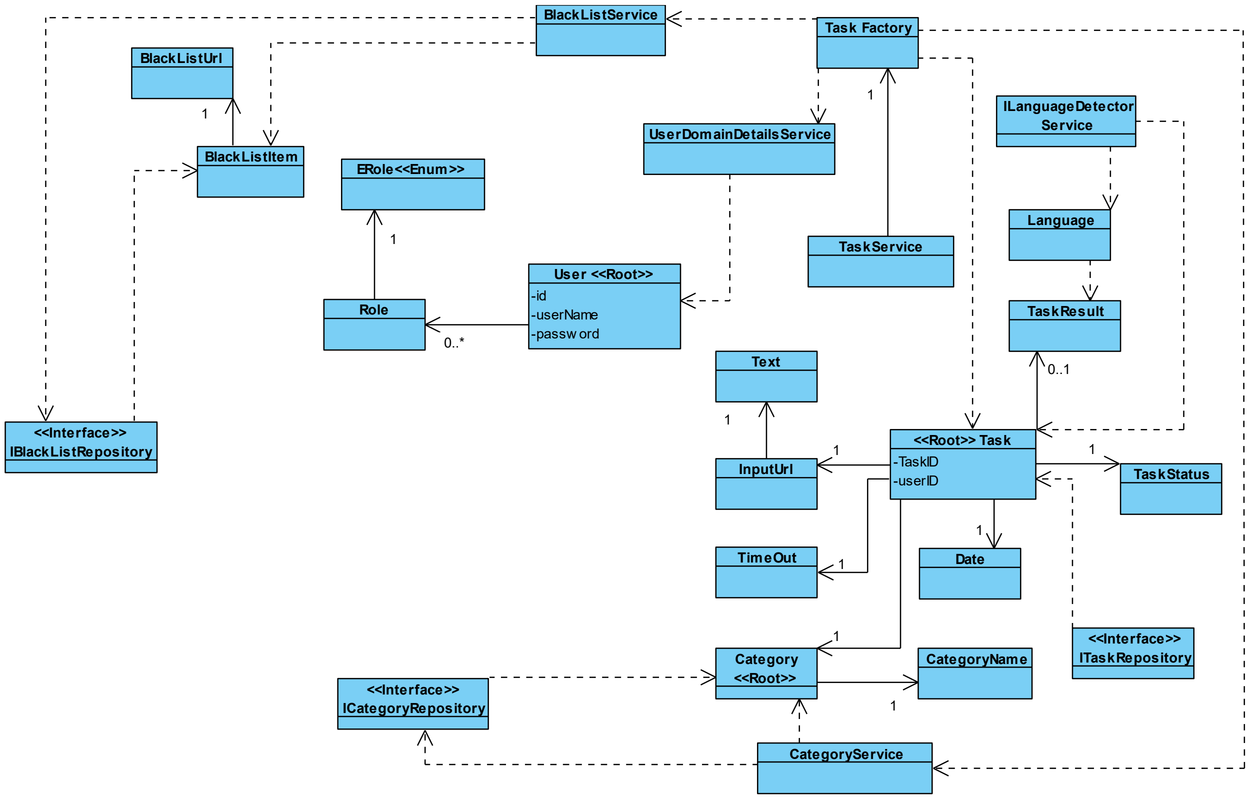
To be able to create an analysis task, it is necessary to introduce a URL link with a .txt extension that points to a text file, a time limit, and a previously registered category.

## 1.5. System Sequence Diagram (SSD)

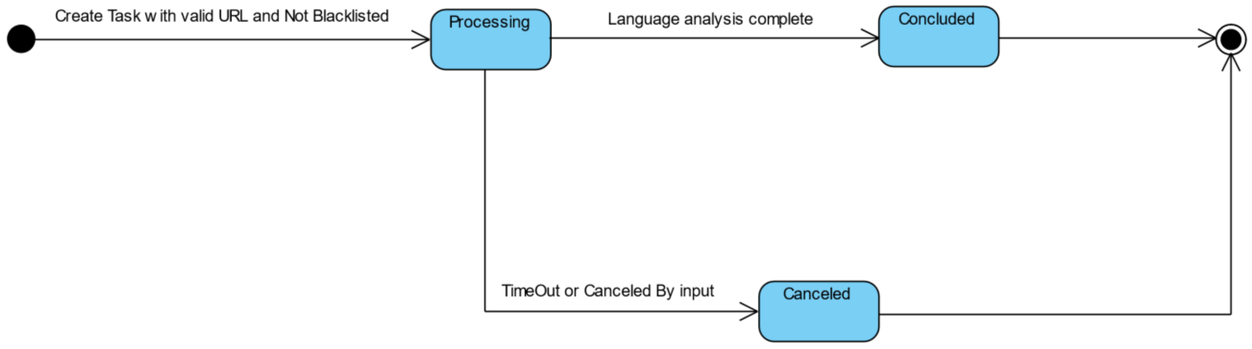

Actor
System
1: Insert url, category and timeout time
1.1: Receives information of created task

## 2. OO Analysis

### 2.1. Relevant Domain Model Excerpt



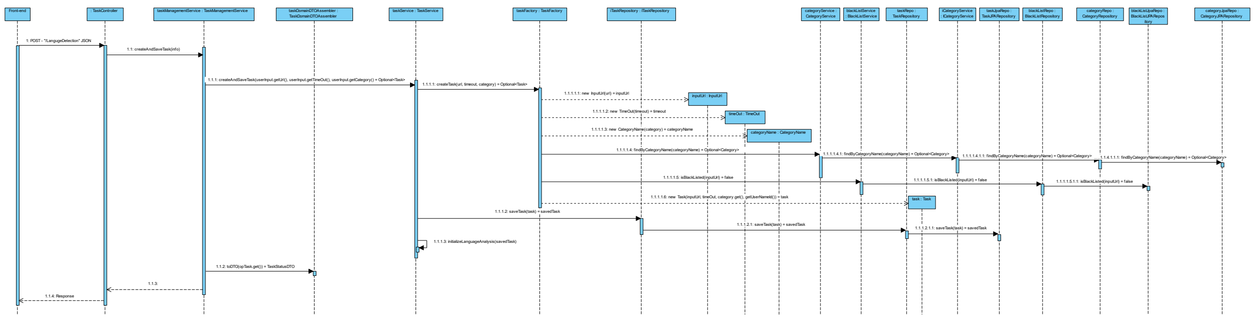
### 2.2. Other Remarks


Create Task with valid URL and Not Blacklisted
Processing
Language analysis complete
Concluded
TimeOut or Canceled By input
Canceled

This diagram represents the change of task status.

## 3. Design - User Story Realization

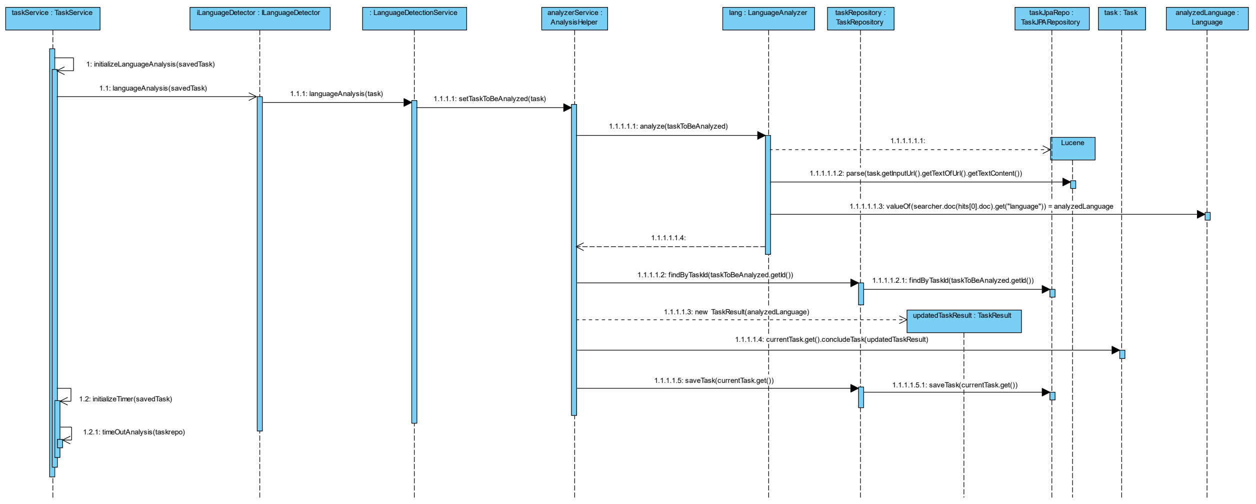
### 3.1. Sequence Diagram (SD)



This diagram represents task creation without asynchronous process of language identification.

The asynchronous analysis begins at initializeLanguageAnalysis(savedTask).

Below it's possible to check the asynchronous process of language identification.


taskService : TaskService
iLanguageDetector : ILanguageDetector
: LanguageDetectionService
analyzerService : AnalysisHelper
lang : LanguageAnalyzer
taskRepository : TaskRepository
taskJpaRepo : TaskJPARepository
task : Task
analyzedLanguage : Language
1: initializeLanguageAnalysis(savedTask)
1.1: languageAnalysis(savedTask)
1.1.1: languageAnalysis(task)
1.1.1.1: setTaskToBeAnalyzed(task)
1.1.1.1.1: analyze(taskToBeAnalyzed)
1.1.1.1.1.1:
Lucene
1.1.1.1.1.2: parse(task.getInputUrl().getTextOfUrl().getTextContent())
1.1.1.1.1.3: valueOf(searcher.doc(hits[0].doc).get("language")) = analyzedLanguage
1.1.1.1.1.4:
1.1.1.1.2: findByTaskId(taskToBeAnalyzed.getId())
1.1.1.1.2.1: findByTaskId(taskToBeAnalyzed.getId())
1.1.1.1.3: new TaskResult(analyzedLanguage)
updatedTaskResult : TaskResult
1.1.1.1.4: currentTask.get().concludeTask(updatedTaskResult)
1.1.1.1.5: saveTask(currentTask.get())
1.1.1.1.5.1: saveTask(currentTask.get())
1.2: initializeTimer(savedTask)
1.2.1: timeOutAnalysis(taskrepo)

### 3.3. Class Diagram (CD)

In this US, as it depends of almost every class on our project, it's possible to check our general class diagram in our CD folder.

## 4. Tests

Here is one example of createAndsaveTask method, but it's possible to see more on our project. On this example, as our Task constructor is protected, we used reflection to be able to use his constructor.

```
@Test
void ensureCreateAndSaveTaskReturnsCreatedTask() throws
NoSuchMethodException, InvocationTargetException, IllegalAccessException,
MalformedURLException, InstantiationException {
    //arrange
    String url = "https://www.w3.org/TR/PNG/iso_8859-1.txt";
    int inputTimeOut = 1;
    String inputCategory = "Sports";
    Category category = new Category(inputCategory);
    TimeOut timeOut = new TimeOut(inputTimeOut);
    InputUrl inputUrl = new InputUrl(url);

    Method isBeingAnalyzed =
TaskService.class.getDeclaredMethod("isBeingAnalyzed", String.class);
    isBeingAnalyzed.setAccessible(true);

    Constructor<Task> taskConstructor =
Task.class.getDeclaredConstructor(InputUrl.class,TimeOut.class,Category.class
,Long.class);
    taskConstructor.setAccessible(true);
    Task task = taskConstructor.newInstance(inputUrl,timeOut,category,1L);
    Optional<Task> createdTaskOp = Optional.of(task);


    try(MockedStatic<UserDetailsDomainService> utils =
Mockito.mockStatic(UserDetailsDomainService.class)){
        utils.when(()->
iTaskRepository.existsByUrlAndIsProcessing(any(InputUrl.class),
any(Long.class))).thenReturn(false);
        Mockito.when(taskFactory.createTask(url, inputTimeOut,
inputCategory)).thenReturn(createdTaskOp);

Mockito.when(iTaskRepository.saveTask(createdTaskOp.get())).thenReturn(task);

        //act
        Optional<Task> savedTask = taskService.createAndSaveTask(url,
inputTimeOut, inputCategory);

        //assert
        assertEquals(task,savedTask.get());
    }
}
```

## 5. Construction (Implementation)

The construction of this user story depends on several factors and therefore, it is the user story with more implementation details needed.

We also encountered some difficulties, namely the asynchronous detection of the language of a text, which made it impossible to interrupt a thread that was analyzing the text.

Also, an interface was implemented that aims to encapsulate the analysis of the language of a text. In this way, it is possible in the future to change to any other form of language analysis without having to modify the business model.

Therefore, to ensure resource prevention, we decided to implement some details:

- Limitation of 5 million characters of text to be parsed.

- Cleaning of a text with non-Latin characters, only with numbers, characters or only spaces.

- Limitation of entering the same text constantly while that text is still being processed

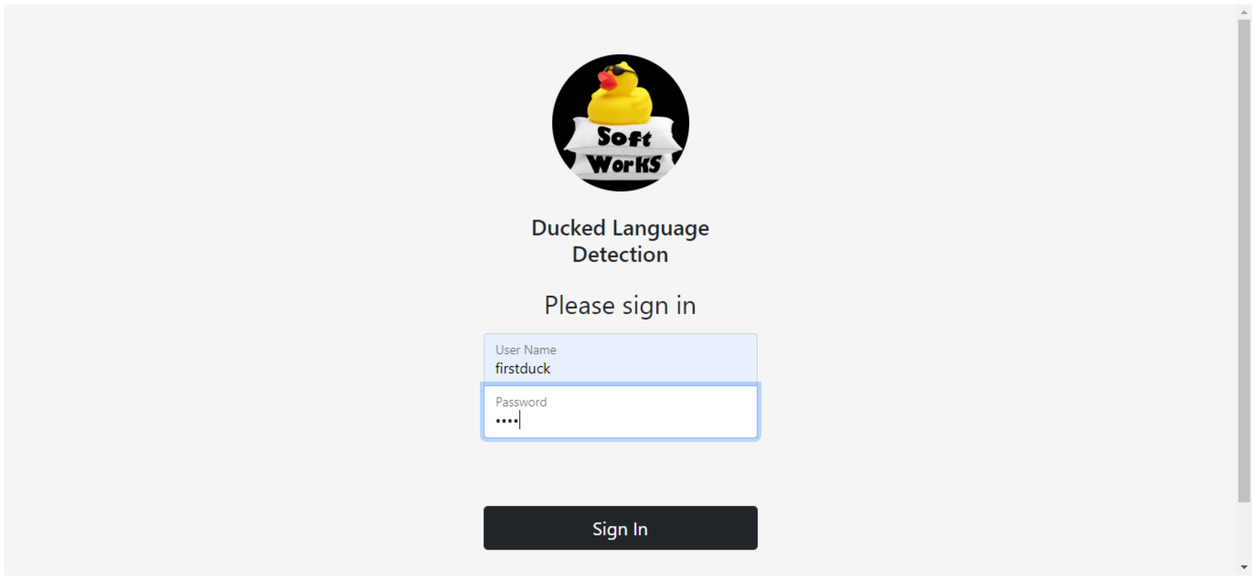
All these limitations were made having attention to the preservation of resources and also program inconsistencies.

You can get more details about these implementation decisions in our report.

## 6. Integration and Demo

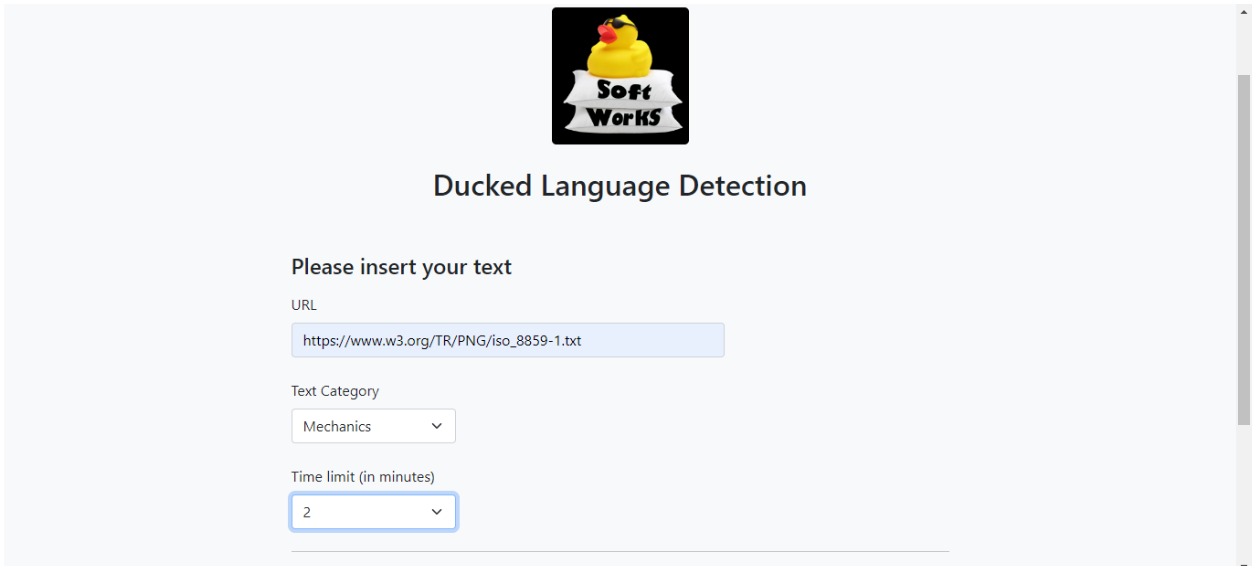
For this analysis task creation, it is then necessary to be logged in with a user with user permissions, to select/enter a previously created category, a timeout and a URL with .txt extension.

We made an integration with one frontend application that can be used in conjunction with the backend application.



firstduck

••••



Ducked Language Detection

Please insert your text

URL

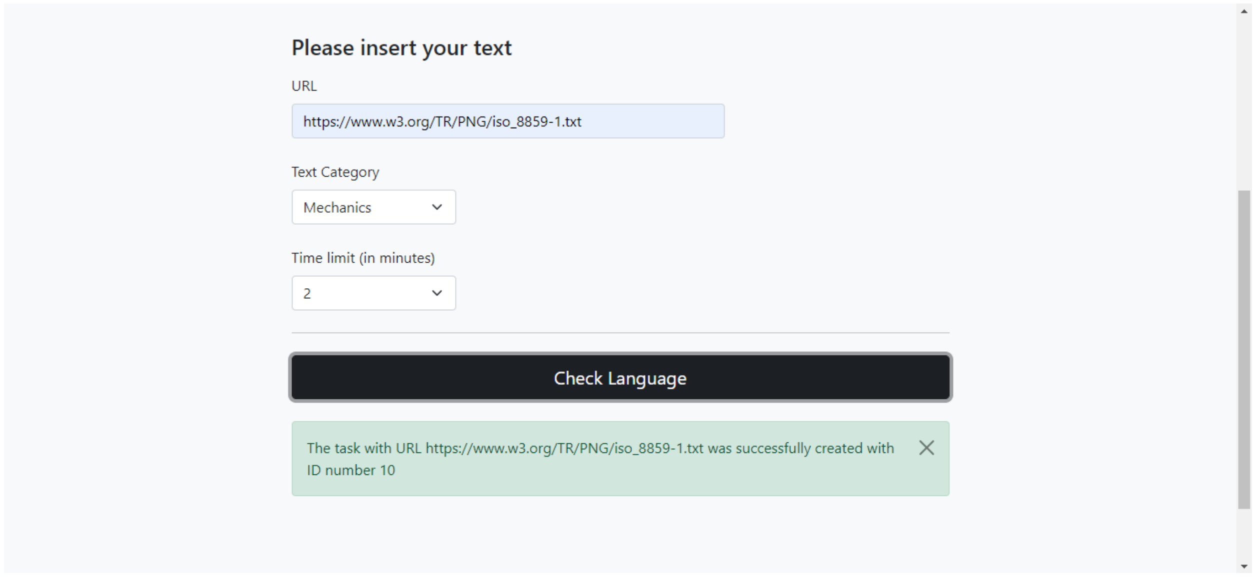
https://www.w3.org/TR/PNG/iso_8859-1.txt

Text Category

Mechanics

Time limit (in minutes)

2



Please insert your text

URL

https://www.w3.org/TR/PNG/iso_8859-1.txt

Text Category

Mechanics

Time limit (in minutes)

2

The task with URL https://www.w3.org/TR/PNG/iso_8859-1.txt was successfully created with ID number 10

## 7. Observations

It is necessary to review in the future the implementation of better methods for asynchronous processes in order to improve the performance and reliability of the program.