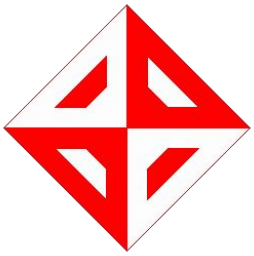


METU - Department of Computer Engineering

SOFTWARE REQUIREMENTS SPECIFICATION

WebCat

13th January 2016

Group Name :

Overcode

Group Members:

Mert Basmacı - 1819119

Onur Ozan Yüksel - 1881663

İzzet Barış Öztürk - 1881796

Özge Donmaz - 1819234

Table of Contents

1. Introduction

This document is a software requirement specification for a nonstop working web site crawler. In this document, foremost we are going to define the problem and explain the system. Next in order, we are going to give an overall description. After these steps, we are going to state specific requirements. Finally, we are going to present data models and behavioral models with their descriptions consecutively.

1.1 Problem Definition

The Internet causes people to categorize less on websites and search more. When everybody find everything on Internet and networked computers determine what is important, people begins to require means to search websites and categorize them. These means are web crawlers and categorization engines. There are some open source web crawlers and closed source categorization engines. But there aren't any open source categorization engines and none of these products target Turkish websites. So this project will be the first website categorization project that will be open source and target Turkish websites. The program will only categorize Turkish websites and target audience of this project is mostly companies and families. The project can be used to prevent employees and family members from visiting unapproved websites or obtain list of the websites of specific category.

1.2 System Overview

The purpose of the project is to develop a web crawler program which will crawl the web and after websites will be categorized (chat, adult, e-commerce, blog etc.) using machine learning techniques. The system's components will crawl on websites and download their information. Then it will select useful information from the gathered information and turn it into a usable input for machine learning algorithms. After these steps algorithm will determine the category of the website, then obtained information about the website will be stored in database. When there is an inquiry about a website system will search it firstly in database and if it isn't in database, it will be categorized and be stored in database.

The major components of the system are Crawler4j and WEKA. Crawler4j is a web crawler and it crawls on the web nonstop and acquires information about the websites. WEKA is a machine

learning software and it is used to manage and train machine learning algorithms for categorization.

1.3 Definitions, acronyms, and abbreviations

IEEE	The Institute of Electrical and Electronics Engineers
DESC	Deacription
TCP	Transmission Control Protocol

1.4 Assumptions and Dependencies

- System is assumed to answer multiple queries simultaneously.
- Once the seed URLs are provided to the system, it will continue crawling and categorizing by itself.
- Semi-supervised learning algorithm will be planned to be use. This algorithm involves human interruption. If there is a case of miscategorization notification from outside, system is assumed to fix the erroneous data by itself.
- The respond time to the queries is considered to be small, if the system encounters a fresh URL by a user, it will stop the usual crawling process for a while, do the user's request first and respond to the user in maximum of 15 seconds.
- The components of the system is not platform dependent, however CentOS7 and PostgreSQL server is primarily expected to be provided by the users before the installation.
- Unless there is a corruption on servers, system is planned to provide services all the time.

2. Overall description

The aim of this section is to provide background information about specific requirements of the system and make them easier to understand. This section does not state specific requirements which are defined in detail in Section 3 of the SRS as stated in the IEEE Std. 830-1984.

This section includes three subsections, as follows:

- Product Functions
- Interfaces
- Constraints

2.1 Product functions

WebCat provides a Web Interface which is a free software and does not requires any registration or authorization process and also provides an API for the Labris Company. Web page users and API users who are employees of Labris Company are the actors of the system and they have some functionalities. These functionalities are described briefly in this section. The more detailed descriptions and state specific requirements are available in Section 3.

Any actor (Web Page User or API User) basically can:

- provide a URL.
- get category of a given URL.
- report miscategorization.
- get similar pages.
- extend similar page results.

Different from the Web Page User, API User (employee of Labris Company) can

- get all the pages of a given category or multiple categories

2.1.1 Use-case model survey

Each use case model is listed and briefly described in this subsection.

2.1.1.1 “Provide a URL” Use Case

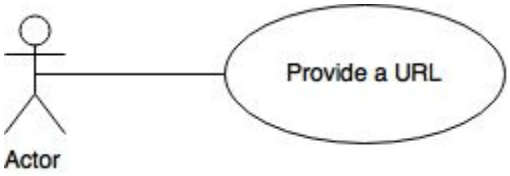


Figure : Use case 1

Use Case ID:	UC1
Use Case Name:	Provide a URL
Desc:	WebCat has a search box and any user may provide a Web URL there.
Actors:	Web Page Users, API Users

2.1.1.2 “Get Category of a Given URL” Use Case

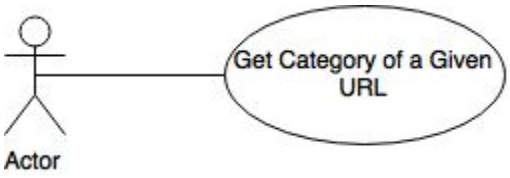


Figure : Use case 2

Use Case ID:	UC2
Use Case Name:	Get Category of a Given URL
Desc:	By clicking a button, user can get category of a given URL in the search box
Actors:	Web Page Users, API Users

2.1.1.3 “Report Miscategorization” Use Case

Figure : Use case 3

Use Case ID:	UC3
Use Case Name:	Report Miscategorization
Desc:	User can inform the system in case of returning unrelated category.
Actors:	Web Page Users, API Users

2.1.1.4 “Get Similar Pages with Same Category” Use Case

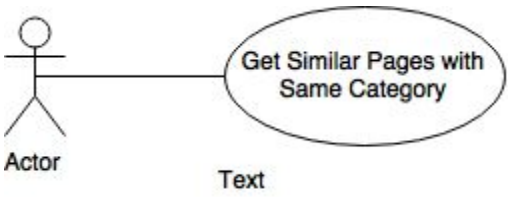


Figure : Use case 4

Use Case ID:	UC4
Use Case Name:	Get Similar Pages
Desc:	User can see other Web pages with the same category that he/she search for.
Actors:	Web Page Users, API Users

2.1.1.5 “Extend Similar Page Results” Use Case

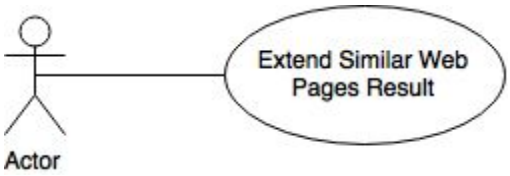


Figure : Use case 5

Use Case ID:	UC5
Use Case Name:	Extend Similar Page Results
Desc:	The system normally shows a fixed number of possible category alternatives. If the user wants to see more, they are able to extend the results.
Actors:	Web Page Users, API Users

2.1.1.6 “Get All the Pages of a Given Category or Multiple Categories” Use Case

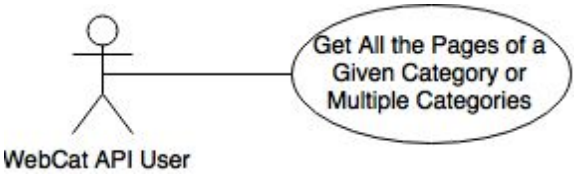


Figure : Use case 6

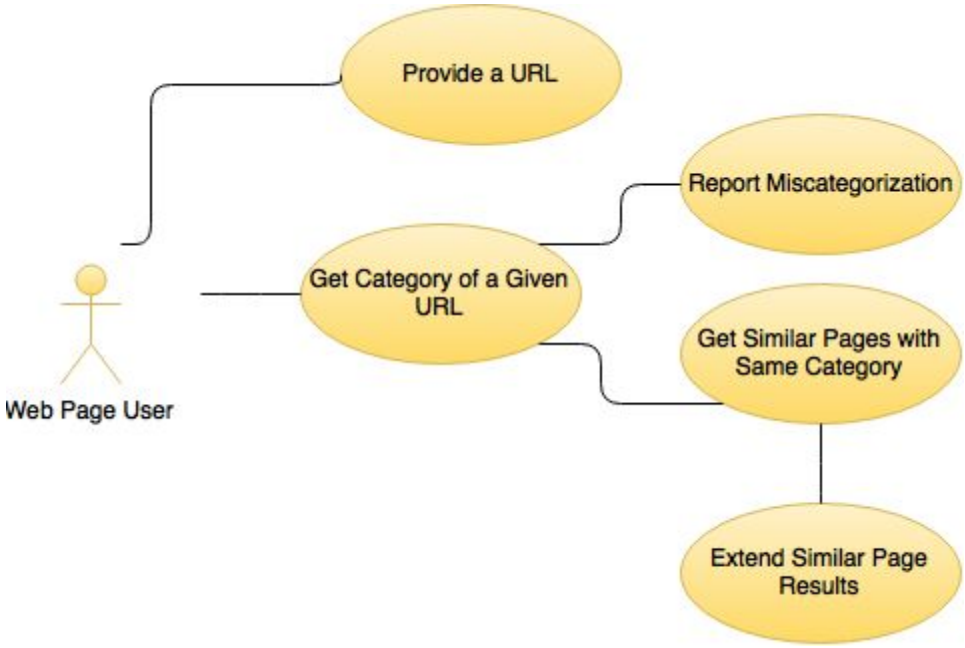
Use Case ID:	UC6
Use Case Name:	Get All the Pages of Given Category or Multiple Categories
Desc:	Different from the Web Page, WebCat API has other functionality that an API user (employee of Labris Company) can select one or more categories, then clicking a get URLs button all the URLs related with these categories are returned to the user, without any mixing (if at least 2 categories are selected). This functionality is a requirement wanted by Labris Company.

Actors:	API Users

2.1.2 Actor survey

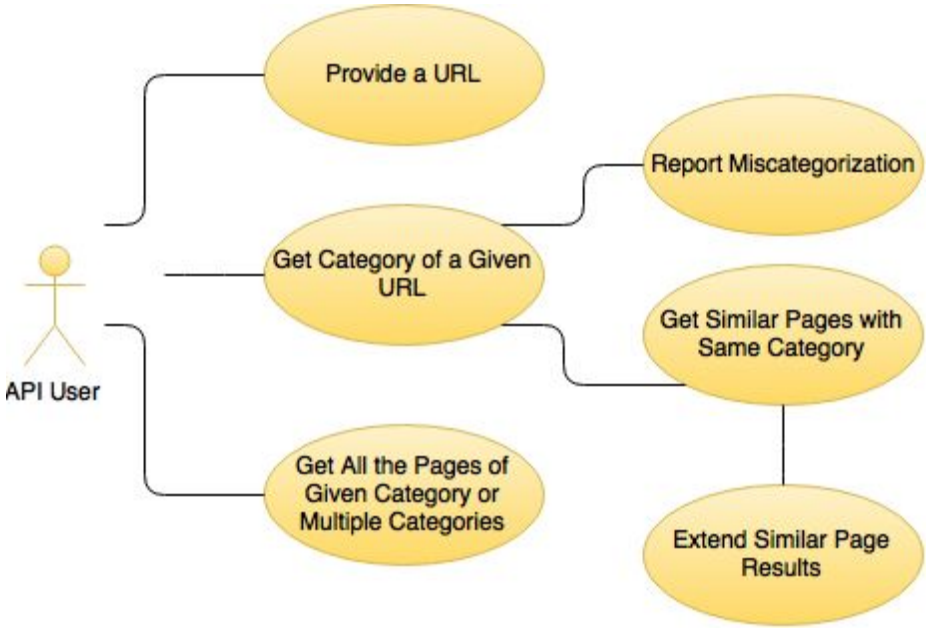
Any Webcat user, who has ability of accessing internet and a web browser is a Web Page User and any Labris Company employee who uses WebCat API is an API User. Web Page Users and WebCat API Users are the actors mentioned in the use-case model survey. They all have similar functionalities except API users (Labris employees) can get all the pages of a given category or multiple categories.

2.1.2.1 Web Page User



Actor ID:	AC1
Functionalities:	UC1, UC2, UC3, UC4, UC5
Actor Name:	Web Page User
Desc:	Any Webcat user, who has ability of accessing internet and a web browser is a Web Page User. They have functionalities described with the use cases UC1, UC2, UC3, UC4, UC5 which are stated in use-case model survey

2.1.2.2 API User (employee of Labris Company)

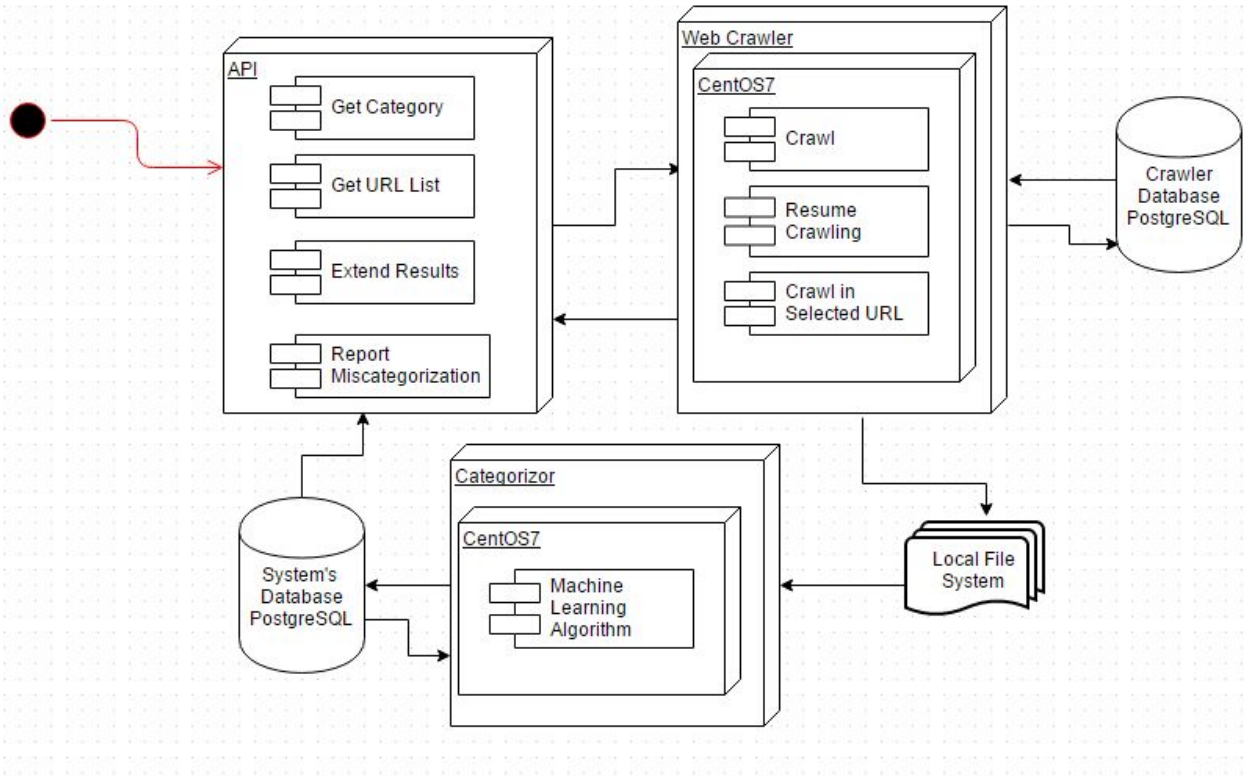


Actor ID:	AC2
Functionalities	UC1, UC2, UC3, UC4, UC5, UC6
Actor Name:	API User
Desc:	Any Labris Company employee who uses WebCat API is an API User. Different from the Web Page User, it has a functionality described with the use case UC6.

2.2 Interfaces

2.2.1 System Interfaces

The system has four main interfaces namely Web-Crawler, Categorizer, Database and API, that work together simultaneously. The system architecture is depicted below:



2.2.1.1 Web-Crawler

This is the part where the system crawls through entire Internet, jumps from the link to link and extracts the text content of the page to a '.txt' file. Once the seed URLs provided to the crawler, it continues to crawl till it reaches to the dead end. Only on the Turkish web pages, will the operations be made.

2.2.1.2 Database

Database is where all the core information for the system's processes are stored and fetched from. System uses 2 different database table. One of them consists of "URL", "Category", "Insert Date", "Update Date" and "Revision" blocks, the other consists of "Visited URLs" and "Unvisited URLs" blocks. Word List and the Word-Category pairs shall not be stored in the database due to performance concerns.

2.2.1.3 Web Categorizer

Web Categorizer is where machine learning algorithms are implemented. The main purpose of this component classify URLs under the correct categories. To predict categories in a

maximum rate of success, the categorizer should be well trained, that is, the categorized shall be fed with diversive content of inputs till the deployment date, in order to achieve reliability. After deployment, web categorizer is able to fix the erroneous matches with the help of the interventions from outside world (reporting miscategorization).

2.2.1.4 API

API is the user interface for the company from which we take the project. API enables system's users to give commands to the system which is useful for their work, such as nonstop crawling, data extraction, bag of words, formatting input, saving, updating database, extracting from database. The ones, other than the API users, have the limited permissions for operations. they can only get related categories of a corresponding web page and extract the results.

2.2.2 User Interfaces

The user interface will be accessed through the web browsers. It will have search engine GUI look, that is, there is a search bar in the middle of the page where users write or paste the URLs. Users can get the web page category after writing and confirming to start. When system returns the category, users are able to extend the results. The user interface will have also some other facilities that involve users to contribute reliability such as reporting miscategorization, or suggesting a new category. The use case diagram of the system is depicted below:

2.2.3 Hardware Interfaces

The system is divided into three main components working in parallel- namely crawler, database and categorizer- planned to work on server computers distinctively, and those computers run CentOS7. During the development procedure, our computers are being used as servers. However, when the project is deployed, hardware is provided by the company that orders the project.

2.2.4 Software Interfaces

2.2.4.1 Crawler4j

Crawler4j is a web-crawler that is found suitable due to its speed and its ease of adjustment and reusability for the project. It has some facilities that we took the advantage of such as:

Crawl Depth, where you can limit the depth of the crawling process in order not to have the crawler stick to one page.

Maximum Number of Pages to Crawl facilitates to end the process in a reasonable amount of time. It avoids the crawler to crawl entire Internet.

Politeness Delay is for avoiding excessive load put on the visited sites' servers during crawling process.

Number of Processors indicates number of crawlers, which are processing at the same time with same frontier. They will check each other and do not work on the same link.

We made several modifications on the crawler like the output format (.txt), language, visiting pages, visiting and add-remove operations on subpages, frontier URL operations (jumping next URL from the frontier URLs).

2.2.4.2 WEKA

WEKA is a machine learning software that is used for the classification process. It contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. WEKA is chosen to be used for the project because it fits the programming language that we use, it has free availability and portable. It also returns reliable data and users can work on the software easily even if they have big data to deal with.

2.2.5 Communications Interfaces

The communication between the components of the system is either wired or wireless. The system is capable of both, as soon as the network safety is provided. When the system is deployed, the wired connection is more likely to be used. TCP is used as a protocol, in order to set reliable connection.

2.3 Constraints

- Since the working environment that the project is going to be deployed is based on Linux kernel, the server computers shall run Linux operating systems (primarily CentOS7).
- Only the Turkish websites will be categorized, the system shall not respond any other query different than Turkish.
- Since there many operations run simultaneously on different big data, the hardware shall be capable of those kinds of operations in order to make the system reliable and available.
- The web-crawler's memory and the disk usage is limited upto 20% and 50% respectively for the API users.
- The memory and the disk usage of the database is limited upto 50 and 80% respectively for the API users.
- The connection between the components shall not be interrupted.

- The connection inside and outside of the system shall be safe.
- The machine learning aspect of the system works on the files with .arff format, so the crawler's .txt output shall be converted into .arff

3. Specific Requirements

This section contains all software requirements to a level of detail sufficient to design a system, for which background information is provided in the previous sections.

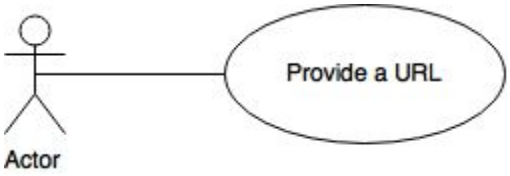
3.1 Functional Requirements

3.1.1 "Provide a URL" Functionality

3.1.1.1 "Provide a URL" Background

There is no big change if the user just enters a URL to the search bar, only the "Get Category of a Given URL" button isunlocked.

3.1.1.2 "Provide a URL" Use Case



Use Case ID:	UC1
Use Case Name:	Provide a URL
Desc:	WebCat has a search box and any user may provide a Web URL there.
Actors:	Web Page Users, API Users
Preconditions	User shall have the Internet connection and a working web browser.
Trigger	Filling the search bar with a URL
Basic Flow	Step 1: User enters the website Step 2: User writes the URL to the search bar

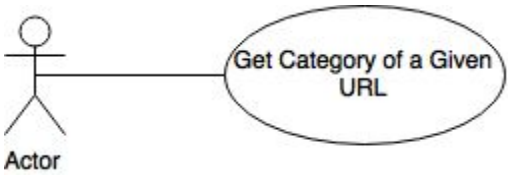
Alternate Flow	
Exception Flow	If the search bar isn't filled, the system shall not let the user proceed to the next step.
Post Condition	"Get Category of a Given URL" button is unlocked.

3.1.2 "Get Category of a Given URL" Functionality

3.1.2.1 "Get Category of a Given URL" Background

If the search bar is filled and the "Get Category of a Given URL" button is clicked, system takes the URL as an input. First, the URL is queried in the web-crawlers database if it's already been crawled and the date of revision is still up to date, the possible categories are fetch directly through the system's database. If the URL hasn't been visited before, the crawler stops its ongoing process and starts crawling corresponding URL. Its text content fetched first and is processed with some algorithms namely bag of words and directed to the categorization tool. The possible categories returned to the user and the crawler resumes its previous crawling process.

3.1.2.2 "Get Category of a Given URL" Use Case



Use Case ID:	UC2
Use Case Name:	Get Category of a Given URL
Desc:	By clicking a button, user can get category of a given URL in the search box
Actors:	Web Page Users, API Users
Preconditions	User shall have the Internet connection, a working web browser. URL shall also be provided.

Trigger	The process starts when the "Get Category of a Given URL" button is clicked.
Basic Flow	Step 1: User enters the website Step 2: User writes the URL to the search bar Step 3: User clicks "Get Results" button Step 4: Category of the URL (and the probabilities of other related categories) returned to the user.
Alternate Flow	
Exception Flow	If the content of the web page (of the given URL) is blank or incomprehensible to the system, such as language, or the system is unable to find the category, "Unknown Category" is returned to the user as a category.
Post Condition	Results are returned to the user. Users are able to use the "Extend" and “Report” option if needed.

3.1.3 “Report Miscategorization” Functionality

3.1.3.1 “Report Miscategorization” Background

Since the machine learning component may not always give the correct results, there may occur some fallacies in the results. In these cases human intervention is needed. When the miscategorization and the correct alternative is supported to the system. “URL”, “Old Category”, “New Category” are taken by the system as an input. According to the previously crawling data if its OK to change the category (because there may be some abuse) system updates its machine learning vector and returns an acknowledgement.

3.1.3.2 “Report Miscategorization” Use Case

Use Case ID:	UC3

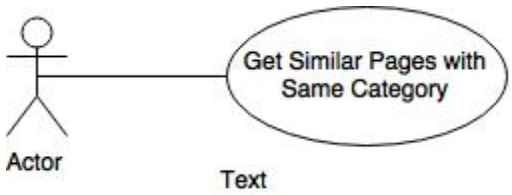
Use Case Name:	Report Miscategorization
Desc:	User can inform the system in case of returning unrelated category.
Actors:	Web Page Users, API Users
Preconditions	User shall have the Internet connection and a working web browser and a set of categories returned to the user previously.
Trigger	The process starts when the "Report Miscategorization" button is clicked.
Basic Flow	Step 1: User enters the website Step 2: User writes the URL to the search bar Step 3: User clicks "Get Results" button Step 4: Category of the URL (and the probabilities of other related categories) returned to the user. Step 5: User selects correct category alternative in the bar next to the button Step 6: User clicks "Report Miscategorization" button
Alternate Flow	
Exception Flow	
Post Condition	Acknowledgement message is returned to the user.

3.1.4 “Get Similar Pages with Same Category” Functionality

3.1.4.1 “Get Similar Pages with Same Category” Background

This a feature is only available for the company employees. When the "Get Similar Pages with Same Category" button is clicked, the system selects and returns all the URLs of the stated block name (category).

3.1.4.2 “Get Similar Pages with Same Category” Use Case



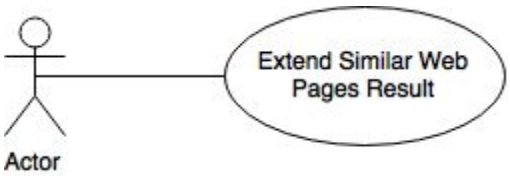
Use Case ID:	UC4
Use Case Name:	Get Similar Pages
Desc:	User can see other Web pages with the same category that he/she search for.
Actors:	Web Page Users, API Users
Preconditions	User shall have the Internet connection and a working web browser and, most importantly, permission.
Trigger	The process starts when the "Get Similar Pages with Same Category" button is clicked.
Basic Flow	Step 1: User enters the website Step 2: User selects a category of which the URLs are wanted to be fetched. Step 3: User clicks the "Get Similar Pages with Same Category" button Step 4:Similar URLs are returned to the user.
Alternate Flow	-
Exception Flow	If the user is unauthorized to do this operation, s/he gets the error message and is directed to the main page.
Post Condition	Results are returned.

3.1.5 “Extend Similar Page Results” Functionality

3.1.5.1 “Extend Similar Page Results” Background

Normally five of the possible category alternatives are printed on the screen but in the background there is a huge vector array. In the need of extending possible categories, the rest of the vector is returned to the user in stated amount. However, its undetermined wheter all the possible categories are stored in the database. It may need to re-crawled to URL again and all the alternatives are fetched from the local file system. In that case, it a bit slower than the previous alternative but it shall end at most 15 seconds.

3.1.5.2 “Extend Similar Page Results” Use Case



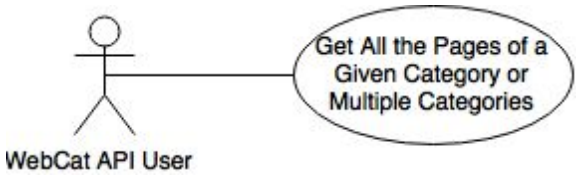
Use Case ID:	UC5
Use Case Name:	Extend Similar Page Results
Desc:	The system normally shows a fixed number of possible category alternatives. If the user wants to see more, they are able to extend the results.
Actors:	Web Page Users, API Users
Preconditions	User shall have the Internet connection and a working web browser and a set of categories returned to the user previously.
Trigger	The process starts when the "Extend Similar Page Results" button is clicked.
Basic Flow	Step 1: User enters the website Step 2: User writes the URL to the search bar Step 3: User clicks "Get Category of a Given URL" button Step 4: Category of the URL (and the probabilities of other related categories) returned to the user. Step 5: User clicks "Extend Similar Page Results" button Step 6: Additional categories in the selected amount are returned.
Alternate Flow	-
Exception Flow	If the selected number is too large, the system will return the maximum number of possible alternatives.
Post Condition	Results are returned to the user.

3.1.6 “Get All the Pages of a Given Category or Multiple Categories” Functionality

3.1.6.1 “Get All the Pages of a Given Category or Multiple Categories” Background

The process is the same as 3.1.4 individually, as stated above. But the whole process is done iteratively.

3.1.6.2 “Get All the Pages of a Given Category or Multiple Categories” Use Case



Use Case ID:	UC6
Use Case Name:	Get All the Pages of Given Category or Multiple Categories
Desc:	Different from the Web Page, WebCat API has other functionality that an API user (employee of Labris Company) can select one or more categories, then clicking a get URLs button all the URLs related with these categories are returned to the user, without any mixing (if at least 2 categories are selected). This functionality is a requirement wanted by Labris Company.
Actors:	API Users
Preconditions	User shall have the Internet connection and a working web browser and a set of categories returned to the user previously.
Trigger	The process starts when the "Get All the Pages of Given Category or Multiple Categories" button is clicked.
Basic Flow	Step 1: User enters the website Step 2: User writes the URL to the search bar

	Step 3: User clicks "Get Category of a Given URL" button Step 4: Category of the URL (and the probabilities of other related categories) returned to the user. Step 5: User selects the number of possible categories to be returned Step 6: User clicks "Get All the Pages of Given Category or Multiple Categories" button Step 7: Additional categories in the selected amount are returned without any mess (if there are multiple categories, the URLs appear on the screen separated by each other by their categories).
Alternate Flow	-
Exception Flow	-
Post Condition	-

3.2 Nonfunctional Requirements

The purpose of this section is to inform readers about the nonfunctional requirements of the system. All the requirements of the four main components of the system is stated under the categories of “Usability”, “Reliability”, “Performance” and “Supportability”.

3.2.1 Usability

- For the user who are not permitted to use the API, the user interface shall be quite simple to use. Anyone who is an average web-browser literate person can easily use the system.
- For the API users, there are several commands that the system take from the API. Learning these commands and the the issue of what button is what may take a time, but it won’t pass more than an hour.

3.2.2 Reliability

- System returns five possible category alternatives, those five categories shall converge to the highest rate of correction, and shall be in order from the highest possible answer to the lowest one.
- Availability:
  - The system shall accessible 99% of the time.
  - In case of a corruption or any other case that may lead the system to be closed, the system shall be accessible in maximum of 8 hours. (It is due to the system will run on servers when deployed.)

- TCP is used during the communication between components in order to prevent any harm happened on the data.

3.2.3 Performance

- System shall respond to 10000 request per second.
- If the system encounters a URL that hasn't been visited before, it shall respond to the user by returning the correct categories in at most 15 seconds.
- If the system has crawled the web page before and found it's category, it shall check whether the categorization out of date and return the result in seconds.
- In order to meet performance targets; response time, workload, scalability and platform issues shall be addressed precisely.
- Memory usage is determined for the API users as 80% memory- 50% disk for crawler side and vice versa for the database side.
- If more than 1 crawlers is running, they should not visit same links again.

3.2.4 Supportability

The system is going to be transferred to the API user, after the deployment process done. In order to make it easy to support and maintain, it is taken care of using understandable naming conventions, making the code clean and suitable for debugging.

4 Data Model and Description

This section provides classes of the components which contains data and their relationships.

4.1 Data Description

Data objects and software classes that manipulates data are described in this section.

4.1.1 Data Objects

4.1.1.1 Categories Class

Diagram:

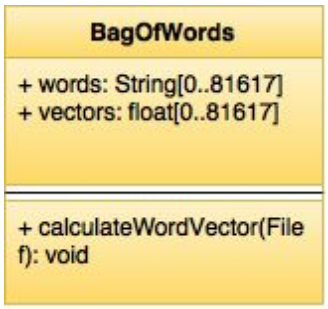
Categories: Class that contains category names.

categories: String Array

Description: Categories Class contains all the category names as a string array. Web crawler class creates datasets for machine learning algorithms using that class. It creates folders with these category names and stores web page contents in this folders. Then, this contents are converted to the ARFF format and html content will be saved as bag of words and its vector.

4.1.1.2 BagOfWords Class

Diagram:



BagOfWords: Class that contains information about a category

words: String Array.

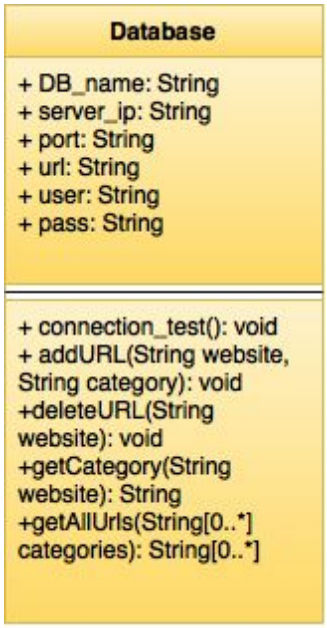
Desc: All the Turkish Language words.

vectors: Float Array.

Desc: All categories have a bagOfWords Class and word counts of a category is stored in this vectors array

4.1.1.3 Database Class

Diagram:



Database Class: Class that contains database connection information. It has also some methods to manipulate data.

DB_Name: String

Desc: Name of a database

server_ip: String

Desc: Server IP for database connection

port: String

Desc: Port number for database connection

url: String

Desc: Url for database connection includes server ip and port.

user: String

Desc: Username to initialize database connection

pass: String

Desc: Password to initialize database connection. Our database connection is secured by a password.

connection_test(): void

Desc: This method tries to initialize database connection. If database connection can not be set all the other process are stopped by error handling. If connection is set, then all the data manipulation is ready to be done.

addURL(String Website, String Category): void

Desc: After machine learning algorithm determines the category of a web page, we only store the URL's name(base64 unique string of the URL) and its category in our database. This adding operation is made by that method.

deleteURL(String Website): void

Desc: Any URL which is no longer available on the internet can be removed from our database using that method (First convert the website URL to base64 unique string, then remove from database).

getCategory(String Website): String

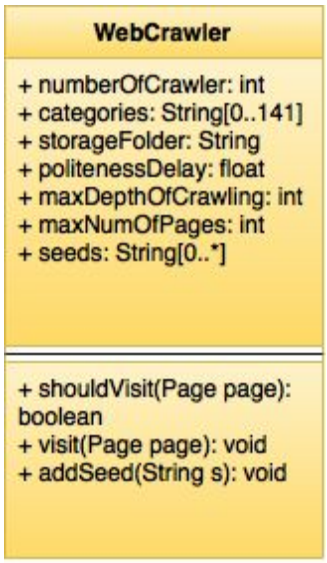
Desc: If a user asks a URL that we already categorized in our database, then we instantly return the category of it using that method.

getAllURLs(String category): String Array

Desc: For API users(Labris Company employees), they can fetch all the URLs for a given category or multiple categories.

4.1.1.4 WebCrawler Class

Diagram:



numberOfCrawlers: integer

Desc: Number of processes that we run while crawling process.

categories: String Array

Desc: Crawler creates folders for each category to keep a bag of words file for each of them.

storageFolder: String

Desc: Since frontier reaches to large sizes, it is keep in a folder. This describes Frontier's path.

politenessDelay: float

Desc: Waiting time required to jump next URL from frontier.

maxDepthOfCrawling: integer

Desc: Crawling depth property.

maxNumOfPages: integer

Desc: When crawler reaches the maximum number of pages value, crawling process will stop.

seeds: String Array

Desc: String Array which contains initial seeds.

shouldVisit(Page p): boolean

Desc: Method that determines whether a given web page p will be visited or not. If more than 1 crawlers is running, they should not visit same links again.

visit(Page p): void

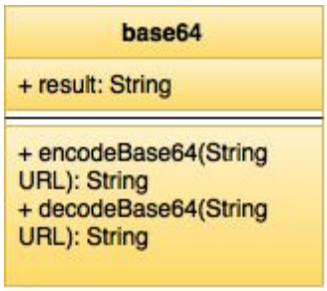
Desc: If shouldVisit method returns true, content of a web page p is download.

addSeed(String S): void

Desc: In order to supply initial URL list this method is used

4.1.1.5 Base64 Class

Diagram:



Base64: Class that manipulates URL. Makes encoding and decoding translations.

result: String

Desc: Keeps the result of encoded or decoded URL.

encodeBase64(String URL): String

Desc: Encode the given URL to base64 unique string.

decodeBase64(String URL): String

Desc: Decode the given base64 string to URL.

4.1.1.6 TextToARFF Class

Diagram:

textToARFF: Class that manipulates web page content. Text content of a web page is translated to ARFF Format

arff: BagOfWords Class

Desc: Contains the result of the text to arff translation.

translate(String textContent): BagOfWords Class

Desc: Translates the web page content to the arff format using all the Turkish words and counting their vectors.

4.1.2 Data dictionary

4.1.2.1 Categorization

- **based64URL:** unique String (base64 translation of a string ensures uniqueness)
- **category:** String category
- **insertDate:** Date date
- **updateDate:** Date date
- **revision:** integer

5 References

[1] IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

[2] Appendix C of Don Widrig, Dean Leffingwell, “Managing Software Requirements: A Unified Approach,” Addison-Wesley Professional, Release Date: October 1999, ISBN: 0201615932.