### CONTEXT BASED SEARCH ENGINE (BATCH\_A2)

## Contents:

1. [Requirements](#_Requirement_in_detail)
2. [Architecture diagram.](#_ARCHITECTURE_DIAGRAM)
3. [Sequence diagrams.](#_SEQUENCE_DIAGRAM)
4. [Class diagram.](#_CLASS_DIAGRAM)
5. [Data Format](#_DATA_FORMATS)

## Objective : - To Crawl the web efficiently and extract the information from the web pages effectively and ranking the content accordingly.

## Requirement Analysis

R1 [Starting of Context-Retrieval System and Connecting to the A2.Master.](#_R1 Starting of Context-Retrieval System and Connecting to the A2.Master.)

R2 [Start the Worker Threads to Crawl the Web Page of Obtained Link](#_R2 Start the Worker Threads to Crawl).

R3 [Filtering of context.](#_R3 Filtering of Context)

R4 [Assigning Rank to the context.](#_R4 Assignment of Ranks)

R5 [Frequency of Crawling.](#_R5 Frequency of Crawling)

R6 [Writing status of parsed link.](#_R6 Write status of parsed link)

R7 [Update links status to A2.Master.](#_R7 Update link status to A2.Master)

R8 Data-Structure to store link information.

R9 Implementation of Heartbeat probing system to check the status.

## Requirement in detail

### R1 Starting of Context-Retrieval System and Connecting to the A2.Master.

A. A standalone application called fireup is used to start the Context-Retrieval System and helps to connect to A2.Master.

B. Helps to Boot A2.Master and configures it to given IP Address and PORT.

C. Sending CONNECT request to A2.Master.

### R2 Start the Worker Threads to Crawl

1. A cross-platform thread library to support both windows & Linux platform.
2. Crawling the web and assigning name to it which is further used by parser.

### R3 Filtering of Context

1. Removal of Ads.
2. Spam detection.
3. Avoiding Redirection Loops.
4. Extraction of Links.
5. Mapping extracted links to [robot.txt](https://stackoverflow.com/robots.txt)to check this web Page is allowed to parse or not.

### R4 Assignment of Ranks

1. Ranking algorithm is required to assign rank to each web pages.
2. Avoiding loop in links in order to compute ranks
3. Considering words frequency in order to increase the relevance.

### R5 Frequency of Crawling

1. Determining when to crawl the web pages based on their frequency of updation.

B. Need to store last modified date and Last crawled date.

C. Construction of Probabilistic model and judging whether to crawl or not.

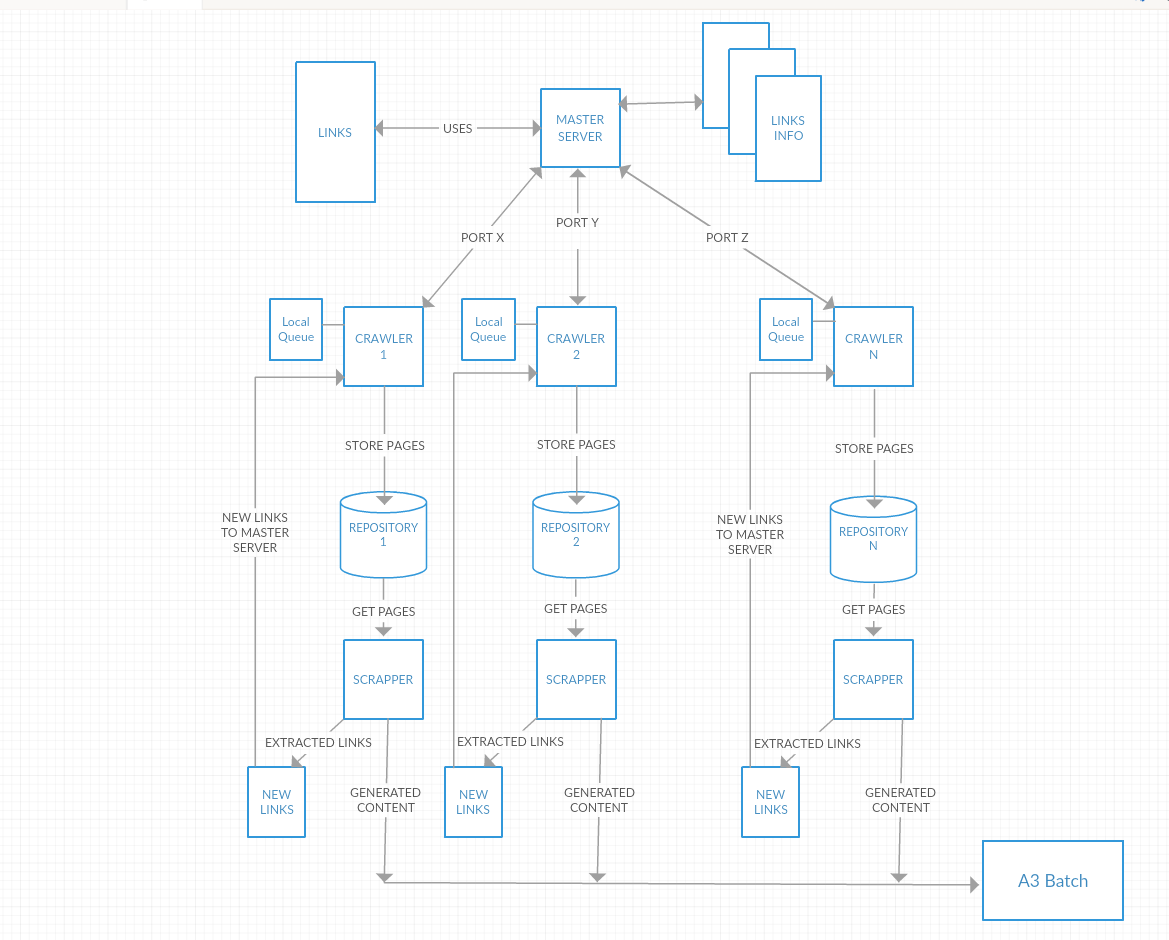
### R6 Write status of parsed link

1. Helps to design probabilistic model

### R7 Update link status to A2.Master

1. To know which link got updated and which is not .
2. To update the rank of a link.
3. To update the output links from the web pages.

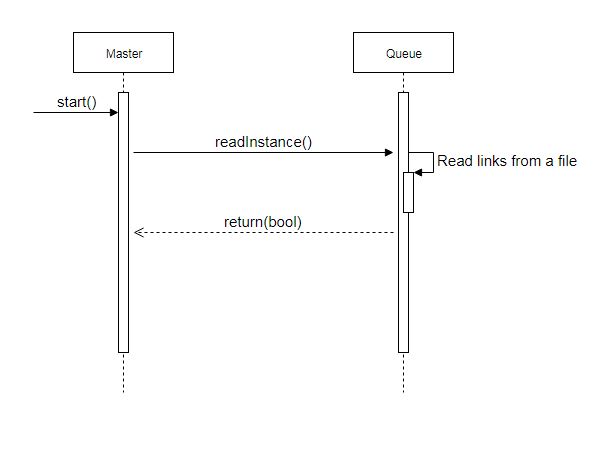
## ARCHITECTURE DIAGRAM



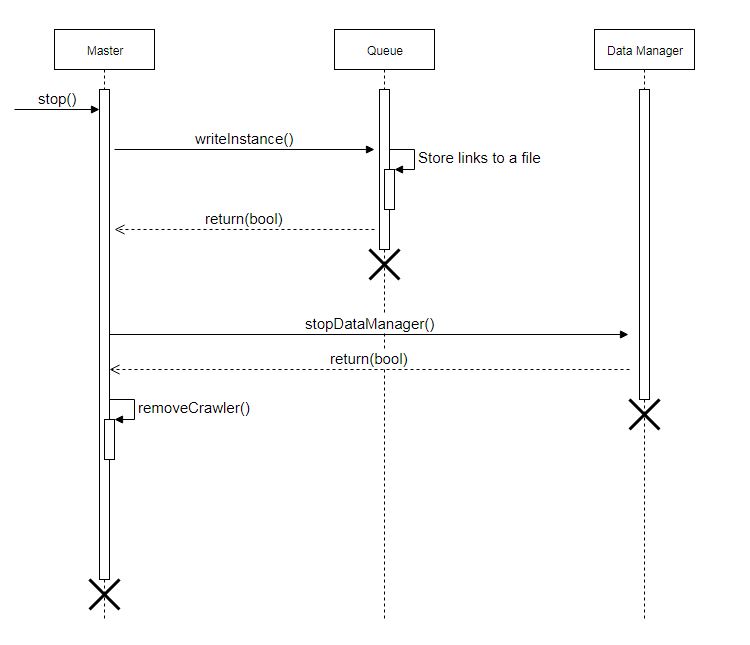
## SEQUENCE DIAGRAM

1. Starting Master.
2. Stop Master.
3. Add links to Queue.
4. Get links from Queue.
5. Add new Links to Data Storage.
6. Fetch links from Data Storage.
7. Crawler connecting to Master
8. Crawler Crawling the Page
9. Parser refining the context.

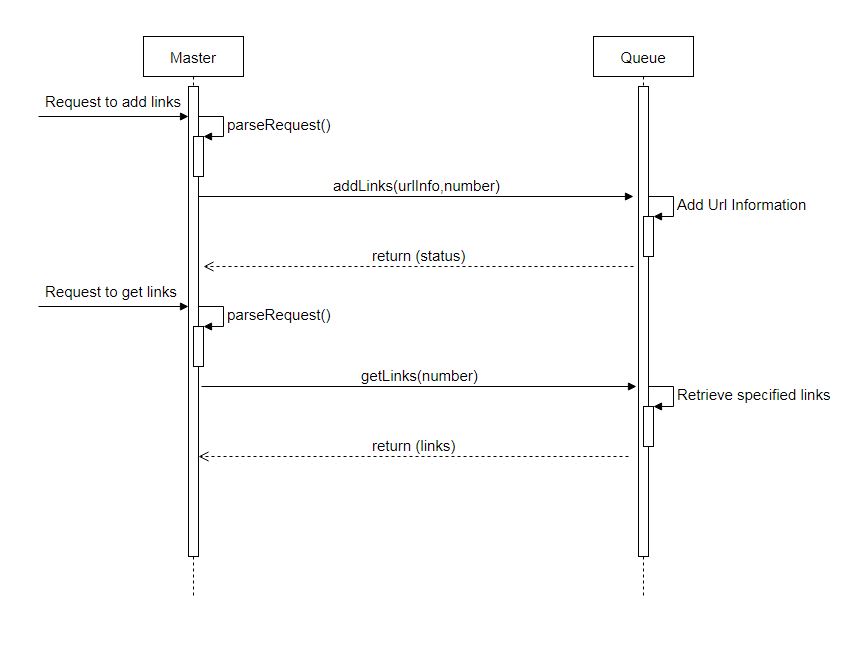
Starting Master



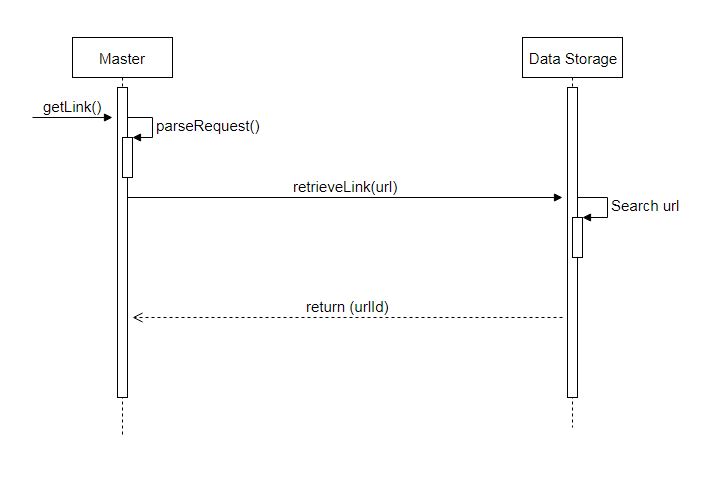
Stop Master



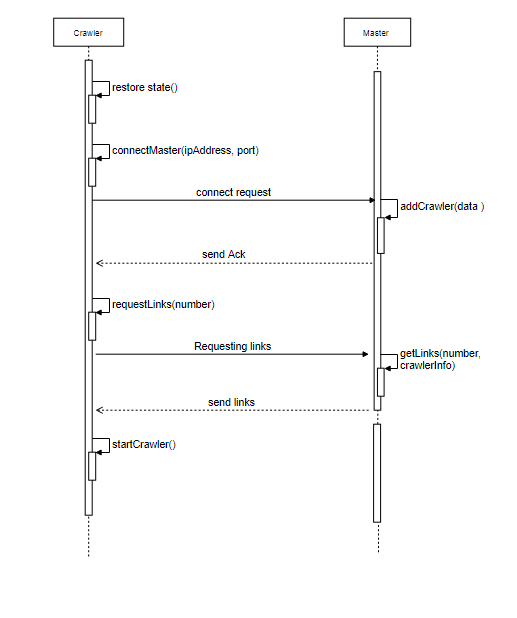
Add links to queue



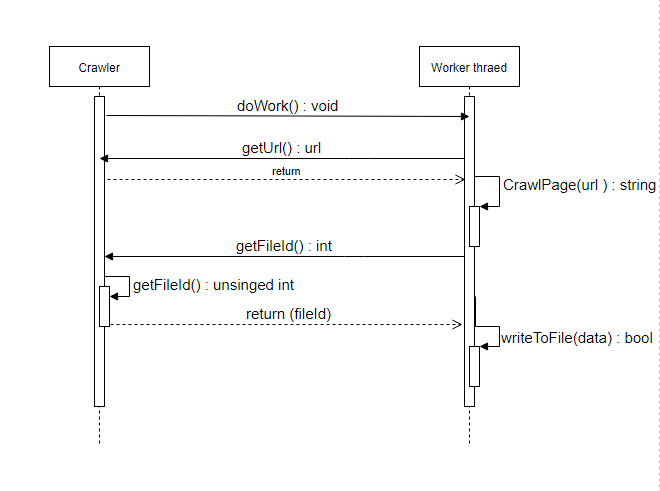
Retrieve links from queue



Crawler Connect to Master

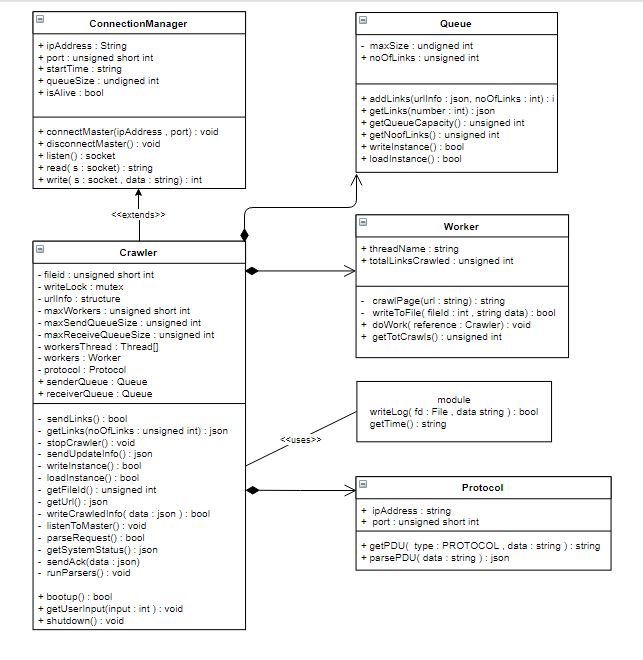


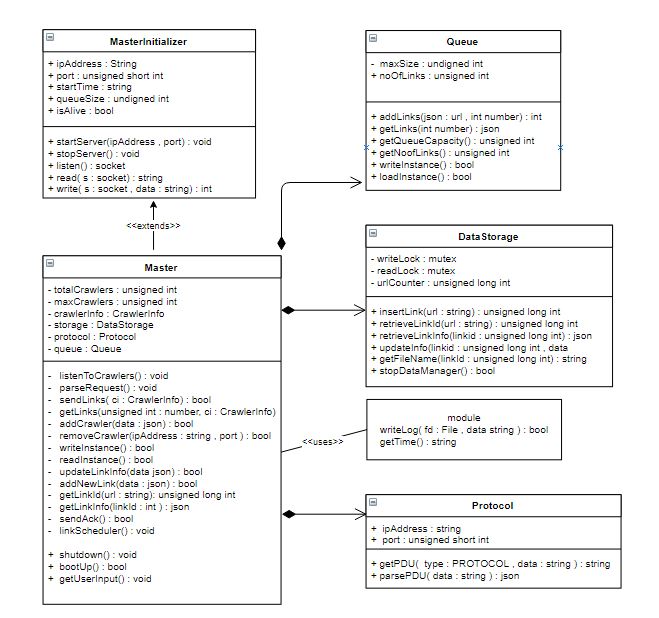
Get Html Content



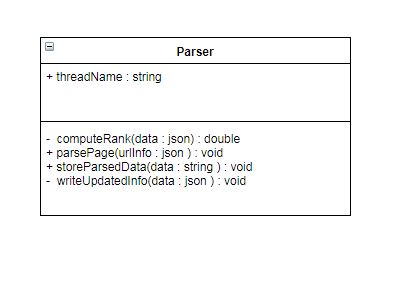
## CLASS DIAGRAM

1. Crawler class diagram



2. Master Class Diagram

3. Parser Class Diagram



## DATA FORMATS

1. Add links to queue

{

No\_of\_Links : 25,

Data[0] : {

url\_id : 3,

url : [www.google.com](http://www.google.com),

depth : 7,

hash\_value : 25343

},…

}

Response:

{

Status : SUCCESS/FAILURE

}

2. Fetch Links from queue

{

No\_of\_Links : 25,

Data[0] : {

url\_id : 3,

url : [www.google.com](http://www.google.com),

depth : 7,

hash\_value : 25343

},…

Status: OK/LESS/EMPTY

}

#defines

OK 1

LESS 2

EMPTY 3

DUPLICATE 4

3. Connect request from crawler to master

Connect request

{

Type : CONNECT/RECONNECT,

Ip address : 192.168.3.3,

Queue size : 3000,

Time : 31/07/2017/8/23/23

}

Response

{

Type: : OK,

Port : 8123,

}

Response

{

Type : DUPLICATE/LIMITEXCEED

Port : : -1

}

4. Crawler Disconnect from Master

{

Type : DISCONNECT

Ip address : 192.168.3.3,

Port : 3000

}

Response

{

Type : DISCONNECT

Ip address : 192.168.3.3,

Port : 3000

Status : OK

}

5. Master Polling the crawlers

Poll connection

{

Type : ISALIVE

}

Response

{

Type : ISALIVE

Ip address : 192.168.3.3,

Port : 3000

Status : YES

}

1. Master requesting new links to crawlers

Request Links from crawlers

{

Type : REQUEST\_LINKS\_MASTER,

No : 212,

Ip address : 192.168.3.3,

Port : 3000

}

Response

{

Type : REQUEST\_LINKS\_MASTER,

No : 212/100,

Ip address : 192.168.3.3,

Port : 3000,

Status : OK/LESS/ZERO

Data[ No ] : {

URL : [www.google.com](http://www.google.com),

Depth : depth-1

},

{

URL : [www.stackoverflow.com](http://www.stackoverflow.com),

Depth : depth-1

}

}

1. Sending links to crawler on request by master

Send links to crawler

{

Type : REQUEST\_LINKS\_CRAWLER,

No : 212,

Ip address : 192.168.3.3,

Port : 3000,

Data[ No ] : {

Urlid : 123,

URL : [www.google.com](http://www.google.com),

Depth : 7,

Hashvalue : 65245

},

{

Urlid : 313,

URL : [www.stackoverflow.com](http://www.stackoverflow.com),

Depth : 17,

Hashvalue : 655945

}

}

Response

{

Type : SENDLINKS,

Status : OK

}

1. Global file format

Global file format

{

Urlid : 213,

url : [www.google.com](http://www.google.com),

thread : 1,

depth : 6,

hashvalue : 65123,

modified : 0/1

filename: 123.html,

time : dd/mm/yy/h/m/s,

status : OK/FAILED

}

1. Sending new links to master by crawler

Global file format

{

Urlid : 213,

url : [www.google.com](http://www.google.com),

thread : 1,

depth : 6,

hashvalue : 65123,

modified : 0/1

filename: 123.html,

time : dd/mm/yy/h/m/s,

status : OK/FAILED

}