

# PROJECT PROPOSAL

# 

# MULTI-THREADED WEB CRAWLER

# LAB: PROJECT MEMBERS:

# OPERATING SYSTEM Rizwan Nisar 20F-0309

# Haseeb Mushtaq 20F-0316

# Raffay Shahzad 20F-0152

# Haseeb Mushtaq 20F-0316

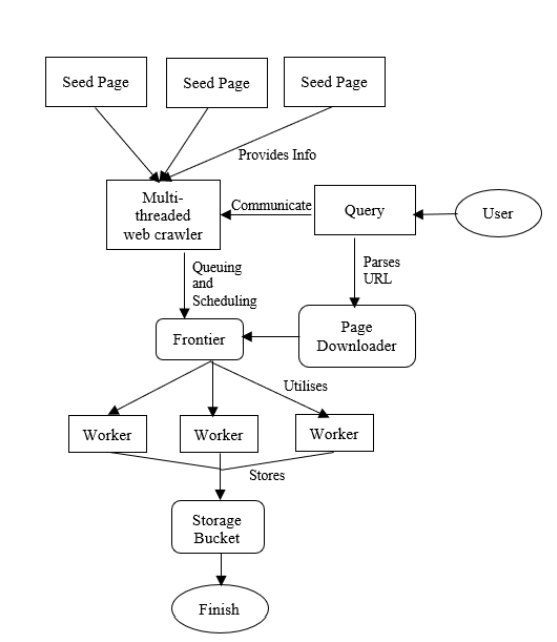
**Introduction:**

The goal of this project is to create a multi-threaded web crawler. A Web crawler is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing. Any search engine uses these indexes, web graphs, and an appropriate algorithm to rank the pages. The main focus of the project would be to implement a multi-threaded downloader that can download multi websites at the same time. The plan is to implement this in C++ language.

**Features:**

* Distributed
* Scalable
* Performance and efficiency
* Quality
* Freshness
* Extensible

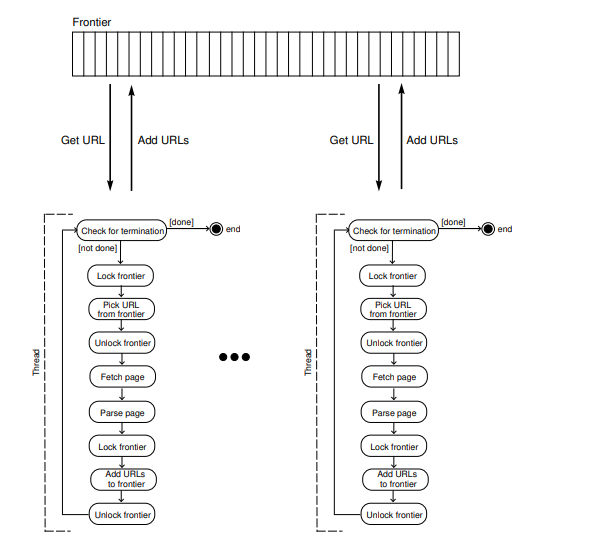
**Flowchart:**



**Working**

Note that each thread starts by locking the frontier to pick the next URL to crawl. After picking a URL it unlocks the frontier allowing other threads to access it. The frontier is again locked when new URLs are added to it. The locking steps are necessary in order to synchronize the use of the frontier that is now shared among many crawling loops (threads). Note that a typical crawler would also maintain a shared history data structure for a fast lookup of URLs that have been crawled. Hence, in addition to the frontier it would also need to synchronize access to the history. If a thread finds the frontier empty, it does not automatically mean that the crawler as a whole has reached a dead-end. It is possible that other threads are fetching pages and may add new URLs in the near future. When the thread wakes up, it checks again for URLs. A global monitor keeps track of the number of threads currently sleeping. Only when all the threads are in the sleep state does the crawling process stop.

**Methodology**



**Language**

* **C++**