Section 1: Introduction

* Discuss background information about the project.
  + The main goal of the project is to build a Web Crawler which will build a list of website links from a give input website.
* Give examples (use cases) to illustrate the expected end product
* A very brief summary of your work
  + a high-level description about the design
    - The user provides and input website. The crawler will start from the website. Recover its web page contents. Then it will find all the hyperlinks in the page. Once it has listed all the links in the page, it will pick up the first link and repeat the process.
* The functionalities
  + Provides the user with a input field to enter a website address.
  + Retrieves website content.
  + Retrieves all hyperlinks.
  + List the hyperlinks.

organization of this report

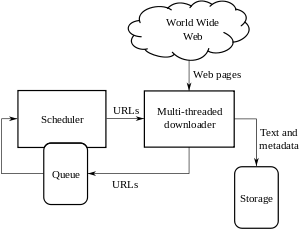
Section 2: Related Work

Following are the list of Web Crawlers.

* Bingbot is the name of Microsoft's Bing webcrawler. It replaced Msnbot.
* FAST Crawler is a distributed crawler, used by Fast Search & Transfer, and a general description of its architecture is available.
* Googlebot is described in some detail, but the reference is only about an early version of its architecture, which was based in C++ and Python. The crawler was integrated with the indexing process, because text parsing was done for full-text indexing and also for URL extraction. There is a URL server that sends lists of URLs to be fetched by several crawling processes. During parsing, the URLs found were passed to a URL server that checked if the URL have been previously seen. If not, the URL was added to the queue of the URL server.
* GM Crawl is a crawler highly scalable usable in SaaS mode
* PolyBot is a distributed crawler written in C++ and Python, which is composed of a "crawl manager", one or more "downloaders" and one or more "DNS resolvers". Collected URLs are added to a queue on disk, and processed later to search for seen URLs in batch mode. The politeness policy considers both third and second level domains (e.g.: www.example.com and www2.example.com are third level domains) because third level domains are usually hosted by the same Web server.
* RBSE was the first published web crawler. It was based on two programs: the first program, "spider" maintains a queue in a relational database, and the second program "mite", is a modified www ASCII browser that downloads the pages from the Web.
* Swiftbot is Swiftype's web crawler, designed specifically for indexing a single or small, defined group of web sites to create a highly customized search engine. It enables unique features such as real-time indexing that are unavailable to other enterprise search providers.

Section 3: System Overview

* An overview of the design (use a picture to illustrate the architecture)
  + Following is the basic architecture.



* a brief discussion of each component
  + SchedulerSends the URLs to the MTD.
  + Multi-threaded Downloader (MTD) download the text and metadata and stores in a database or a text file. It retrieves all the URLs from the website content and sends to the Queue.
* how components interact with each other.
  + The Scheduler sends a list of URLs to the MTD based on the threading process.
  + The MTD runs on multiple threads and sends a request to the WWW based on the URL sent by the Scheduler.
  + The MTD separates the Text, Metadata and the URLs in the website content. Text and Metadata are stored. The URLs are sent back to the Scheduler through a queue.

Section 4: Detailed Design

* the detailed design of each component
* list and discuss all challenges
  + The challenge is IPC.
* explain what you do to solve these challenges
  + We will try to solve by sending a JSON object between the processes.

Section 5: Implementation

* We will implement using Python

Section 6: Theoretical/Simulation Study

* what is the communication overhead?
  + The major overhead is the communication between the processes.
* what is the storage overhead?
  + The storage of the JSON data is the over head
* what is the computational overhead?
  + The computational overhead is minimal. There are only a few loops that the processes use.
* any other aspects that people may be interested in.
  + The usage of JSON data to communicate between the processes.

Section 7: Future Work

* what you want to do in the future? (e.g., additional functionalities)
  + Consolidate the URLs that are retrieved from the project and provide the highest used URL.
  + Also, provide a search functionality similar to Google search