National University of Computer and Emerging Sciences

FAST School of Computing, Karachi Campus

CS3001 - Computer Networks | Fall 2025

PROJECT PROPOSAL

Group Members:

Huzaifa Abdul Rehman (23K-0782)
Abdul Moiz Hossain (23K-0553)
Meeran Uz Zaman (23K-0026)

Section: 5-F

Date: October 26, 2025

Web Crawler with Data Extraction and Search Engine

1. PROJECT DESCRIPTION

Our project will implement a network-based Web Crawler and Search Engine System that systematically navigates websites, extracts structured data, and enables intelligent information retrieval using core Computer Networks protocols.

How It Will Work

The web crawler will operate as an autonomous agent traversing the web using breadth-first search. Starting from a seed URL, the crawler will:

- Establish TCP/IP connections to web servers using HTTP/HTTPS protocols
- Perform DNS resolution to translate domain names to IP addresses
- Send HTTP GET requests and process server responses (status codes 2xx, 3xx, 4xx, 5xx)
- Parse robots.txt files (RFC 9309) to ensure ethical crawling
- Parse HTML responses to extract hyperlinks and textual content
- Normalize URLs and maintain a FIFO queue for breadth-first traversal
- Detect duplicate pages using SHA-256 content hashing
- Process text through NLP (tokenization, stopword removal, Porter stemming)
- Build a searchable index using TF-IDF weighting and cosine similarity ranking
- Export data to CSV and Pickle formats

The system will handle network errors gracefully with timeout and retry mechanisms. Users can submit search queries which undergo text processing and cosine similarity calculations to return ranked results.

Functional Features

Our project will implement the following functional features:

Network & Protocol Features:

1. HTTP/HTTPS Protocol Implementation
2. TCP/IP Socket Communication
3. DNS Resolution Handler
4. HTTP Status Code Processor (2xx, 3xx, 4xx, 5xx)
5. HTTP Error Handler with retry mechanisms

6. Robots.txt Protocol Parser (RFC 9309)
7. Connection Timeout Manager

URL Management:

8. URL Validation System (RFC 3986)
9. URL Normalization Algorithm
10. URL Frontier Queue (FIFO for BFS)
11. Visited URL Tracker (hash-based)
12. Domain Scope Controller
13. Page Limit Enforcer

Content Processing:

14. HTML Parser (BeautifulSoup)
15. Link Discovery & Extraction Engine
16. Content Extractor
17. SHA-256 Content Hasher
18. Duplicate Detection System

Crawling Algorithms:

19. Breadth-First Search Crawler

Text Processing:

20. Text Tokenizer
21. Word Validator
22. Stopword Filter
23. Porter Stemmer (NLTK)

Search Engine:

24. Term Frequency Matrix Builder
25. TF-IDF Calculator
26. Cosine Similarity Engine
27. Query Processor
28. Relevance Ranking System
29. Thesaurus Query Expansion

Data Management:

30. CSV Data Exporter
31. Pickle Serializer
32. Statistics Reporter

Note: *User interface, database interaction, and unit testing are NOT functional features.*

2. PLAN OF WORK (Next 5 Weeks)

Week 1: Network Foundation & Protocol Setup

- Set up Python environment and libraries (urllib, BeautifulSoup, NLTK)
- Implement HTTP/HTTPS client with urllib
- Add DNS resolution and HTTP status code processing
- Develop Robots.txt parser (RFC 9309)
- Test basic page fetching

Deliverable: HTTP client with robots.txt compliance

Week 2: URL Management & Crawling Algorithm

- Implement URL validation (RFC 3986) and normalization
- Build URL frontier queue (FIFO) for BFS
- Develop visited URL tracker (hash-based)
- Implement breadth-first crawler algorithm
- Add domain scope and page limit controls
- Implement connection timeout and retry logic

Deliverable: Functional BFS crawler

Week 3: Content Extraction & Duplicate Detection

- Integrate BeautifulSoup for HTML parsing
- Implement link discovery and content extraction
- Add SHA-256 content hashing
- Build duplicate detection system
- Implement text tokenization, word validation, stopword filtering

Deliverable: Crawler with content extraction and duplicate detection

Week 4: Search Engine & Indexing

- Integrate NLTK Porter Stemmer
- Build term frequency matrix
- Implement TF-IDF weighting algorithm
- Create cosine similarity calculator
- Develop query processing pipeline with thesaurus expansion

Deliverable: Search engine with TF-IDF ranking

Week 5: Data Export & Testing

- Implement CSV export and Pickle serialization
- Create statistics reporting
- Test on multiple websites (books.toscrape.com, quotes.toscrape.com)
- Validate robots.txt compliance and error handling
- Performance benchmarking and documentation

Deliverable: Production-ready system

3. INDIVIDUAL MEMBER CONTRIBUTIONS

Member 1: Meeran Uz Zaman (23K-0026)

Functional Features:

1. Robots.txt Protocol Parser (RFC 9309) - Parse Allow/Disallow rules
2. HTTP Status Code Processor - Handle 2xx, 3xx, 4xx, 5xx responses
3. HTTP Error Handler - Network fault tolerance with try-catch
4. URL Validation System (RFC 3986) - Regex pattern validation
5. Domain Scope Controller - Keep crawler within seed domain
6. Page Limit Enforcer - Prevent infinite crawling
7. Stopword Filter - Remove common words from documents
8. Word Validator - Validate extracted words with regex
9. CSV Data Exporter - Export term-document matrix
10. Statistics Reporter - Track pages crawled, links found, duplicates

CN Contribution: Application layer protocols (RFC 9309, HTTP status codes), URL structure (RFC 3986), network error handling, and resource management.

Member 2: Huzaifa Abdul Rehman (23K-0782)

Functional Features:

1. Connection Timeout Manager - Set timeouts and retry mechanisms
2. HTML Parser (BeautifulSoup) - Parse HTML DOM structure
3. Link Discovery & Extraction - Extract hyperlinks from HTML
4. Content Extractor - Extract text from HTTP responses
5. Text Tokenizer - Parse text into words with regex
6. Visited URL Tracker - Hash-based visited URL storage
7. Porter Stemmer Integration (NLTK) - Normalize words to root forms
8. Term Frequency Matrix Builder - Construct document-term matrix
9. Query Processor - Parse and match search queries
10. Thesaurus Query Expansion - Expand queries with synonyms
11. Pickle Serializer - Index persistence and file I/O

CN Contribution: Content processing pipeline for network-retrieved data, HTML parsing from HTTP responses, and data persistence for network applications.

Member 3: Abdul Moiz Hossain (23K-0553)

Functional Features:

1. HTTP/HTTPS Protocol Implementation - Application layer GET requests
2. TCP/IP Socket Communication - Transport layer reliable connections
3. DNS Resolution Handler - Domain name to IP translation
4. Breadth-First Search Crawler - Complete BFS algorithm (main crawl method)
5. URL Frontier Queue Manager - FIFO queue for BFS traversal
6. URL Normalization Algorithm - Relative to absolute URL conversion
7. SHA-256 Content Hasher - Cryptographic hashing for duplicate detection
8. Duplicate Detection System - Hash comparison and deduplication
9. TF-IDF Calculator - Logarithmic term frequency × inverse document frequency
10. Cosine Similarity Engine - Vector space model for ranking
11. Relevance Ranking System - Sort and return top-K results

CN Contribution: Core networking stack (HTTP/HTTPS, TCP/IP, DNS), BFS graph traversal algorithm, URL parsing (RFC 3986), and search engine algorithms for network-crawled data.

4. REFERENCES

33. **RFC 9309 - Robots Exclusion Protocol**https://www.rfc-editor.org/rfc/rfc9309.html*Used for: Implementing Robots.txt parser with Allow/Disallow rules*
34. **RFC 3986 - Uniform Resource Identifier (URI) Syntax**https://tools.ietf.org/html/rfc3986*Used for: URL validation and normalization algorithms*
35. **Python urllib Documentation**https://docs.python.org/3/library/urllib.html*Used for: HTTP/HTTPS protocol implementation and DNS resolution*
36. **Beautiful Soup 4 Documentation**https://www.crummy.com/software/BeautifulSoup/bs4/doc/*Used for: HTML parsing and DOM manipulation*
37. **NLTK - Natural Language Toolkit**https://www.nltk.org/*Used for: Porter Stemmer implementation for word normalization*
38. **Salton & Buckley (1988) - Term Weighting in Information Retrieval***Information Processing & Management*, 24(5), 513-523*Used for: TF-IDF weighting algorithm implementation*

Date: October 26, 2025