



Sheng, S.W. & Ishak, W. H. W. (2026). Development and User Acceptance Evaluation of a Desktop Web Crawler for Keyword-based Information Retrieval. *Journal of Digital System Development*, 4 (2), 34-51. <https://doi.org/10.32890/jdsd2026.4.2.3>

Development and User Acceptance Evaluation of a Desktop Web Crawler for Keyword-Based Information Retrieval

¹Saw Ming Sheng & ²Wan Hussain Wan Ishak

^{1,2}School of Computing, Universiti Utara Malaysia, Kedah, Malaysia

²Corresponding author: hussain@uum.edu.my

Received: 12/1/2026

Revised: 6/8/2026

Accepted: 14/10/2026

Published: 28/10/2026

ABSTRACT

The growing volume of online information makes it difficult for users to find relevant content without manually examining multiple webpages. Although web crawlers can automate information retrieval, many existing tools require programming knowledge or complex configuration. This study developed and evaluated a Windows-based desktop web crawler for keyword-based information retrieval. The prototype followed a six-phase Agile process: planning, design, development, testing, deployment, and review. The team used Java and JavaFX to develop the desktop application and its graphical user interface, Selenium WebDriver to retrieve dynamically rendered webpages, and Java Database Connectivity to connect the application to a MySQL database. The prototype integrates input validation, webpage retrieval, domain-link extraction, HTML content cleaning, rule-based keyword matching, URL filtering, result categorisation, database storage, and retrieval-history searching. We assessed the completed prototype through 11 functional test cases and a user acceptance evaluation involving 30 undergraduate students from Universiti Utara Malaysia. Participants tested the prototype using different publicly accessible URLs and subsequently completed a questionnaire measuring their perceptions of input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction. Ten functional test cases achieved a 100.0% pass rate, while link extraction achieved a 96.7% pass rate because one webpage required human verification. The user acceptance evaluation produced mean scores ranging from 4.00 to 4.70 on a five-point Likert scale. Input validation received the highest mean score ($M = 4.70$), followed by result-display accuracy ($M = 4.43$) and crawling effectiveness ($M = 4.13$). Reliability and stability, as well as overall satisfaction, each obtained a mean score of 4.00. The findings indicate that the prototype's principal functions operated consistently and received generally favourable evaluations under the tested conditions. Further comparative testing is required to determine whether the crawler reduces search time, manual effort, cognitive load, or information overload.

Keywords: Desktop Web Crawler, Functional Testing, Information Retrieval, Keyword Matching, User Acceptance Evaluation, Web Automation.

INTRODUCTION

The rapid growth of digital information on the World Wide Web (WWW) has transformed how people search for and access information. Search engines provide convenient access to extensive online resources, but the volume of available content can make it difficult for users to identify information that is relevant to their needs. Users may need to open, examine, and compare numerous webpages within a limited period, particularly when conducting academic or professional searches. This process increases the effort required to assess relevance and may contribute to information overload (Bawden & Robinson, 2020).

Information overload occurs when the amount of information presented exceeds an individual's capacity to process it effectively. It can increase task completion time, reduce productivity, and affect decision-making quality. Under time constraints, users may evaluate information superficially, delay decisions, or select less appropriate information. Prolonged exposure to excessive information may also contribute to stress, fatigue, anxiety, and a perceived loss of control (Bawden & Robinson, 2020). Therefore, tools that assist users in collecting, filtering, and organising online information may help reduce the burden associated with manual web searching.

A web crawler, also known as a spider or web robot, is an automated program that systematically navigates webpages and collects information for retrieval, indexing, or analysis. A crawler generally begins with one or more seed URLs, retrieves the corresponding webpages, extracts their content and hyperlinks, and follows selected links according to predefined rules. Crawlers can then process, index, or store relevant information for later retrieval (Al Galib et al., 2024). By automating these processes, web crawlers can reduce the need for users to visit and examine each webpage individually.

Despite their usefulness, web crawlers face technical and usability challenges. Modern websites often rely on JavaScript, dynamically generated content, interactive components, and access-control mechanisms that traditional HTML-based crawling tools cannot always handle effectively. Although browser-based technologies can improve dynamic content retrieval, developing and configuring crawlers may still require programming knowledge and considerable technical effort (Chang, 2022; Truong et al., 2025). Recent research on end-user web scraping also indicates that users need sufficient control and transparency when defining what information to extract, how to process different sources, and how to organise the resulting data (Tacuri et al., 2025). These requirements can make existing tools difficult for individuals with limited technical experience to configure and use.

Recent systems have introduced graphical interfaces and browser-based rendering to make web extraction more accessible. However, many existing solutions primarily support large-scale crawling, web archiving, multisource data integration, or technically configurable extraction workflows. A need therefore remains for a lightweight desktop tool that integrates dynamic webpage access, domain-based link extraction, keyword filtering, cleaned-text generation, structured result presentation, and persistent search history within a simplified interface. Such integration may be particularly useful for students and other users who need to collect and review information from a specified website without writing crawler scripts.

Accordingly, this study aims to develop and evaluate a Windows-based web crawler for automated, keyword-based information retrieval. The system accepts a user-defined keyword and seed URL, retrieves accessible webpages within the specified domain, converts relevant HTML content into cleaned text, performs rule-based keyword matching, excludes selected non-HTML resources, stores the retrieved records in a database, and presents the results through a graphical user interface. Specifically, the study has three objectives: (1) to design and develop a desktop web crawler that integrates webpage retrieval, keyword matching, link extraction, data storage, and result presentation; (2) to evaluate whether the

system's principal functions operate as intended under the specified testing conditions; and (3) to assess users' perceptions of the system's input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction. The system is intended to reduce repetitive manual searching; however, the present evaluation focuses on functional performance and user perceptions rather than directly measuring reductions in search time, cognitive load, or information overload.

LITERATURE REVIEW

Information retrieval refers to the processes and systems used to identify information that is relevant to a user's needs from a large collection of documents. Chowdhury (2004) described an information retrieval system as comprising a document collection, indexing mechanisms, a search interface, and a retrieval process that compares user queries with stored information. Such a system's primary purpose is to reduce the effort required to locate relevant information by organising, matching, and presenting documents according to users' information needs. In web-based environments, however, information retrieval is more challenging because the World Wide Web contains a large and continuously changing collection of structured, semi-structured, and unstructured information. Effective web-based information retrieval therefore requires mechanisms that can discover webpages, extract their contents, determine their relevance, and organise the retrieved information.

The effectiveness of information retrieval depends not only on the retrieval system but also on users' knowledge and search behaviour. Yamin et al. (2013) examined user knowledge in terms of topic knowledge and understanding of the search system, while search behaviour was evaluated through the breadth and depth of search queries. Their findings showed that user knowledge influenced how users conducted information searches. Users who understand both the subject being investigated and the search system are generally better positioned to formulate appropriate queries and determine how extensively to explore the available information. This finding matters for developing automated retrieval tools because users do not have equal levels of subject knowledge, search experience, or technical ability. A retrieval tool for general or academic users should therefore simplify specifying search requirements and examining retrieved results.

Web crawlers are important components of web-based information retrieval systems because they automatically discover and collect online content for indexing, filtering, analysis, or storage. A conventional crawling process begins with one or more seed URLs. The crawler retrieves the corresponding webpages, extracts their content and hyperlinks, adds eligible links to a URL queue, and repeats the process until a predetermined condition is satisfied, such as a page limit, domain boundary, crawling depth, or time constraint (Al Galib et al., 2024). Duplicate detection and URL-filtering rules may also prevent repeated retrieval and reduce unnecessary use of processing, storage, and network resources.

Crawler architectures have evolved from centralised designs to distributed and hybrid approaches that allocate crawling tasks across multiple crawler instances. Distributed architectures can improve scalability and coverage when crawlers must retrieve large collections of webpages. Researchers have also developed different crawler types for different purposes. Universal crawlers aim for broad web coverage, incremental crawlers revisit webpages to keep collections up to date, and deep-web crawlers retrieve information accessible only through forms or interactive interfaces. Focused crawlers, in contrast, selectively retrieve webpages that are relevant to a predefined topic or information requirement.

Focused crawling is particularly relevant to keyword-based information retrieval because it limits retrieval to webpages that match the topic or information requirement. The relevance of a webpage may be determined using keyword matching, semantic similarity, hyperlink structures, machine-learning classifiers, or a combination of these techniques. Kumar and Aggarwal (2021) demonstrated that learning-based focused crawling could prioritise URLs and reduce unnecessary bandwidth, storage, and processing consumption. Huang et al. (2024) later proposed a more advanced focused crawler that combined a semantic graph with a genetic algorithm to interpret topic terms and optimise the prioritisation of unvisited hyperlinks. These studies indicate that semantic and learning-based techniques can improve retrieval relevance. However, these techniques may also increase computational and implementation complexity. For a lightweight desktop application intended for routine end-user searches, rule-based keyword matching offers a simpler, more transparent way to identify content that matches a user-defined term.

The dynamic and continuously changing nature of the Web presents several challenges to automated crawling. New webpages are constantly created, while existing content may be updated, relocated, or removed. Crawlers must therefore balance retrieval coverage and frequency against processing time, bandwidth consumption, storage requirements, and server load. Excessive or repeated crawling may consume unnecessary resources and place additional demands on target servers, whereas limited crawling may produce incomplete or outdated results. Modern crawler designs consequently require clear rules for link selection, duplicate detection, crawl boundaries, request frequency, and termination conditions.

Modern webpages also increasingly depend on JavaScript, asynchronous requests, interactive components, and dynamically generated content. Traditional command-line tools and static HTML parsers, such as Wget and cURL, generally retrieve the initial page response but may not execute JavaScript or reproduce browser-based interactions. As a result, the retrieved content may differ from what users see in a conventional browser (Ahmad et al., 2020). Browser-automation technologies such as Selenium WebDriver can partially address this limitation by loading webpages within an actual browser environment and allowing dynamically rendered content to become available before extraction.

Nevertheless, browser-based crawling does not guarantee access to every webpage. Websites may require authentication, CAPTCHA verification, rate limits, session controls, or automated-traffic detection. Ahmad et al. (2020) found that different crawler designs can elicit different responses from web servers, affecting the consistency and completeness of the retrieved information. Truong et al. (2025) similarly identified the increasing complexity of dynamic websites as a challenge for conventional scraping tools and demonstrated the value of combining browser-based rendering with a graphical interface. Restricted webpages should therefore be treated as retrieval limitations rather than situations in which the crawler should circumvent website protection mechanisms. Responsible crawler implementation should also consider domain restrictions, request frequency, website policies, robots.txt directives, and applicable access permissions.

Another important consideration is the technical knowledge required to operate existing web crawling tools. Many crawling libraries and frameworks are intended primarily for software developers. Their use may require knowledge of programming languages, software installation, HTML structure, browser drivers, selectors, crawling rules, and error handling. Chang (2022) found considerable variation among crawler tools in terms of documentation, configuration requirements, learning costs, and support for dynamic content. These requirements can prevent users with limited programming knowledge from applying web crawling technology to routine information-retrieval tasks.

Recent studies have attempted to make web extraction more accessible through graphical interfaces and end-user configuration mechanisms. Tacuri et al. (2025) emphasised that end users need control and transparency when specifying what information to extract, how to process sources, and how to organise

the resulting data. Their study evaluated an end-user scraping approach using task-performance and usability measures, demonstrating that technically functional extraction mechanisms must also be understandable and manageable from the user's perspective. Similarly, the graphical interface developed by Truong et al. (2025) demonstrates growing interest in making dynamic web extraction accessible to non-specialists.

These developments show that graphical web extraction tools for non-technical users already exist. Therefore, the research gap is not simply the absence of a user-friendly crawler. Instead, the gap concerns the need for a lightweight desktop application that integrates several task-oriented functions within a single workflow. These functions include accepting a user-defined seed URL and keyword, accessing dynamically rendered webpages, limiting extracted links according to predefined rules, converting HTML content into readable text, identifying keyword matches, storing retrieval history, and presenting structured results through a simplified interface. Such integration may help users examine relevant information on a specified website without developing or configuring crawler scripts.

An interactive system's suitability depends not only on its technical performance but also on the quality of the experience it provides to its intended users. A system may perform its programmed functions correctly while still presenting difficulties related to navigation, feedback, responsiveness, learnability, or user control. Therefore, technical testing should be complemented by evaluating how users interact with and perceive the system.

Researchers can evaluate user experience through several approaches, including task-based observation, interviews, expert evaluation, and structured questionnaires. One example of a standardised questionnaire is the Website Analysis and Measurement Inventory (WAMMI), which was developed to evaluate users' experiences with websites and web applications. WAMMI evaluates five dimensions: attractiveness, controllability, efficiency, helpfulness, and learnability. Wan Ishak et al. (2025) applied WAMMI to web applications and found that user satisfaction varied across these dimensions. Their study showed that systematic user evaluation can identify strengths and weaknesses not evident from functional testing alone, including issues related to visual design, system responsiveness, user guidance, and ease of learning.

The literature shows that web crawlers are key mechanisms for acquiring online content for information retrieval. It also identifies continuing challenges involving dynamic content, retrieval relevance, resource consumption, access restrictions, technical complexity, user knowledge, and usability. Research on focused crawling has primarily concentrated on improving relevance prediction, crawling efficiency, scalability, and algorithmic performance, while recent end-user extraction tools have focused on web archiving, configurable scraping, or multisource data integration. An opportunity remains to develop and evaluate a lightweight Windows-based crawler that combines keyword-based retrieval, browser-rendered webpage access, content cleaning, link filtering, database storage, retrieval-history searching, and structured result presentation.

Accordingly, this study develops a desktop web crawler for users with limited web-crawling and programming experience and evaluates both its functionality and user acceptance. Functional testing is used to determine whether the principal features operate as expected under the specified testing conditions. The user acceptance evaluation examines participants' perceptions of input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction. The study contributes a task-oriented prototype and initial empirical evidence concerning how intended users interact with and perceive its principal functions. However, the evaluation does not directly establish that the system reduces information overload, cognitive effort, or search time; these outcomes would require a comparative study involving objective performance measures.

METHODOLOGY

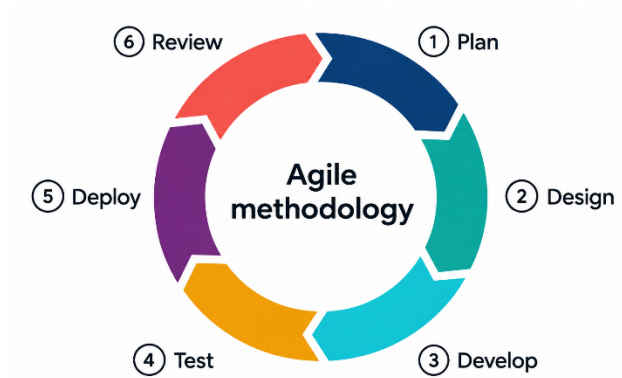
This study adopted a design-and-evaluation approach to develop and assess a desktop web crawler for keyword-based information retrieval. The methodology consisted of two related components. The first component involved developing the desktop application using a six-phase Agile process. The second involved evaluating the completed prototype through functional testing and a user acceptance evaluation involving 30 undergraduate students from Universiti Utara Malaysia (UUM). This approach was aligned with the three research objectives: (1) to design and develop the desktop web crawler; (2) to determine whether its principal functions operated as intended under the specified testing conditions; and (3) to assess users' perceptions of its input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction.

Agile was selected as the development methodology because it supports iterative development, incremental feature implementation, and continuous refinement. Rather than developing the entire system in a single sequence, the application was produced through repeated cycles in which requirements, interface designs, and system functions could be reviewed and improved. Alsaqqa et al. (2020) described Agile as an iterative and incremental software development approach that enables development teams to respond to changing requirements and deliver functional components progressively. In this study, this flexibility allowed modifications in response to supervisory consultations, functional testing, debugging, and emerging implementation requirements.

As illustrated in Figure 1, the development process comprised six phases: planning, design, development, testing, deployment, and review. Although these phases are presented sequentially, revisions during testing and review could send the project back to an earlier phase. This iterative process allowed the team to correct identified problems and progressively refine system functions.

Figure 1

Six-phase Agile development process for the desktop web crawler



During the planning phase, the team identified the problem context, system scope, intended users, and development objectives. The main problem was the effort required to manually open and examine multiple webpages when searching for information related to a particular keyword. The project therefore focused on developing a desktop application that could begin from a user-specified URL, retrieve accessible webpages, identify keyword matches, and store the results for later examination. The team gathered functional and non-functional requirements through discussions with the project supervisor, reviews of related literature and existing crawler technologies, and preliminary analysis of the proposed workflow.

Figure 2

Main interface of the desktop web crawler

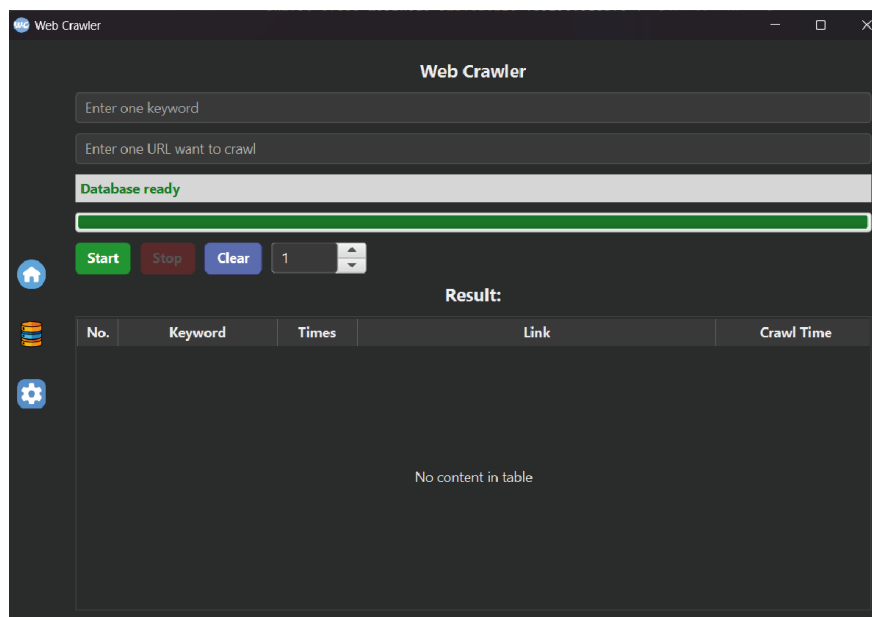


Figure 2 presents the main interface of the desktop web crawler. Users enter a keyword, seed URL, and page limit before starting the crawling process. The interface provides database status and progress indicators, together with Start, Stop, and Clear controls. The interface displays results in a table with the keyword, occurrence frequency, webpage link, and crawl time. Navigation icons provide access to the main page, stored crawling records, and application settings.

The functional requirements included URL and keyword input, input validation, webpage retrieval, domain-link extraction, keyword matching, excluding selected non-HTML resources, result categorisation, database storage, retrieval-history searching, month-based filtering, stopping and clearing crawling activities, and automatically shutting down the computer after a crawling task completes. The non-functional requirements emphasised interface simplicity, understandable feedback, system responsiveness, reliability, and accessibility for users with limited crawler-programming experience.

During the design phase, the team prepared the application architecture, crawling workflow, database structure, and graphical user interface based on the identified requirements. The team used low-fidelity interface prototypes to plan the main crawling page, the database page for viewing and searching previous crawling records, and the settings page for supporting functions. The crawling workflow was designed to begin with a seed URL provided by the user. The application then accessed the webpage, identified eligible links, retrieved accessible content, converted relevant HTML into readable text, matched the cleaned text against the user-defined keyword, and stored the resulting records in the database.

The interface let users initiate, monitor, stop, and clear a crawling task without writing code. The system incorporated input-validation rules to reduce errors caused by missing or invalid URLs, keywords, or page-limit values. URL-filtering rules also excluded file types outside the application's scope, including selected image and document files. The database design supported storing retrieved URLs, webpage information, matched content, crawl dates, and other information needed to display and search previous results.

During development, the application was implemented as a Windows-based desktop program using Java. JavaFX built the graphical user interface, while Selenium WebDriver controlled the browser and retrieved content from webpages, including content rendered dynamically through JavaScript. The application used Java Database Connectivity (JDBC) to connect to a MySQL database and store and retrieve crawling records.

The functions were implemented incrementally. Initial development focused on accepting and validating user inputs. Subsequent iterations introduced browser-based webpage retrieval, hyperlink extraction, keyword matching, text cleaning, URL filtering, result categorisation, database storage, history searching, month filtering, crawling controls, and the automatic-shutdown option. We tested each function during development, and we corrected identified errors through debugging and repeated testing. Feedback from consultations with the project supervisor also helped refine the system workflow and interface.

We then subjected the completed prototype to functional testing and user acceptance evaluation. Functional testing addressed the second research objective by determining whether the system's principal functions operated as intended under the specified testing conditions. Eleven functional test cases were evaluated: input validation, invalid URL handling, starting a crawling task, stopping a crawling task, clearing crawling inputs and outputs, link extraction, keyword matching, displaying and saving records, searching stored records, filtering records by month, and automatically shutting down the computer after task completion.

Functional testing can determine whether features such as input validation, link extraction, keyword matching, and database storage operate as expected. However, user acceptance evaluation is required to determine whether intended users perceive these functions as effective, reliable, understandable, and satisfactory. This matters because previous research shows that users' topic knowledge and understanding of search systems influence their searching behaviour (Yamin et al., 2013). Users with different levels of software and searching experience may therefore interact with the crawler differently.

For each test case, we compared the observed system output with the expected outcome and classified it as a pass or a fail. A pass indicated that the function produced the expected result during the participant's test, whereas a fail indicated that the function could not be completed or did not produce the expected

result. We calculated the pass rate for each function by dividing the number of successful tests by the total number of participant tests and multiplying by 100. This procedure provided a descriptive indication of functional consistency across the 30 participant sessions. It did not serve as a technical benchmark for crawling speed, precision, recall, or large-scale performance.

The user acceptance evaluation involved 30 undergraduate students from Universiti Utara Malaysia (UUM). We recruited participants through convenience sampling based on their availability and willingness to participate. Eligible participants were current undergraduate students with general experience using computer applications and web browsers, and who could operate the prototype on a Windows-based laptop. We did not require previous experience with web crawler software because the application was intended for users without specialist web-crawling or programming knowledge. Participants' general software experience ranged from beginner to advanced, and most reported no prior use of web crawler software. This variation allowed us to examine prototype acceptance among users with different levels of general software experience.

This study did not administer the complete standardised WAMMI instrument. Instead, we designed a task-based user acceptance evaluation specifically for the desktop web crawler's functions. The evaluation examined participants' perceptions of input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction. These aspects reflected the prototype's principal functions and expected user experience.

The evaluation was conducted in a controlled academic setting. Participants used laptop computers connected through Wi-Fi. The computers primarily operated on Windows 11, while one participant used Windows 10. Using broadly comparable devices, browsers, and network conditions helped reduce major environmental differences during testing, although variations in individual computer performance and network connectivity could not be eliminated completely.

During each evaluation session, participants entered the required crawling inputs, including a seed URL, keyword, and page limit. The URLs were not drawn from a predefined list; instead, participants selected different publicly accessible URLs during their respective sessions. This approach allowed the prototype to be tested across webpages with varied structures and content. Participants then executed the assigned functions, observed the retrieved results, and examined the records stored in the crawling history.

Participants' interactions with the prototype formed the basis for recording functional outcomes. For each test case, we compared the observed system behaviour with the expected functional outcome and classified it as a pass or a fail. A pass indicated that the function was completed and produced the expected result, whereas a fail indicated that the function could not be completed or did not produce the expected result. Because different URLs were used, webpage structure, dynamically generated content, and website access-control mechanisms may have influenced the functional outcomes.

After completing the assigned activities, participants provided feedback through a Google Forms questionnaire. The questionnaire examined five aspects of user acceptance: input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction. Participants rated each aspect on a five-point Likert scale from 1 (strongly disagree) to 5 (strongly agree), with higher scores indicating more favourable perceptions of the prototype.

Input validation concerned participants' perceptions of how appropriately the application handled the URL, keyword, and page-limit information users entered. Crawling effectiveness represented their perceptions of the crawler's ability to perform the requested retrieval task. Reliability and stability concerned whether the application operated consistently without freezing or becoming unresponsive. Result-display accuracy reflected whether participants found the displayed information clear and consistent with the completed

crawling task. Overall satisfaction represented participants' general evaluation of their experience with the prototype.

We analysed the collected data using descriptive statistics. We summarised functional outcomes using the number of passes, number of failures, and pass rate for each test case. We summarised participant characteristics and questionnaire responses using frequencies and percentages. We calculated mean scores and standard deviations for the five user-acceptance items to describe central tendency and variation in responses. The analysis was descriptive because the study was intended to provide an initial evaluation of a functional prototype rather than test causal relationships or statistically significant differences between participant groups.

During the deployment phase, the completed prototype was installed and executed on the Windows-based laptops used in the evaluation. This phase verified that the application could operate outside the immediate development environment under ordinary academic testing conditions. The team recorded any technical problems encountered during installation, execution, or participant testing for later review.

In the final review phase, we examined functional outcomes and participant feedback to identify successful features, recurring problems, and opportunities for improvement. We analysed functional failures according to the conditions under which they occurred. For example, we recorded a webpage requiring human verification as a limitation of automated link extraction rather than a function that should circumvent the website's access-control mechanisms. Participant ratings also identified aspects of the interface and system behaviour that require further refinement.

FINDINGS

The findings are presented in three parts: the participant profile and testing environment, the functional test outcomes, and the user acceptance evaluation. Functional testing examined whether the principal system functions produced their expected outcomes, while the user acceptance evaluation examined participants' perceptions of input validation, crawling effectiveness, reliability and stability, result-display accuracy, and overall satisfaction.

Overall, the functional results indicated that the principal features operated consistently under the specified testing conditions. Ten of the eleven test cases achieved a 100.0% pass rate, while link extraction achieved a 96.7% pass rate. The questionnaire results were also generally favourable. For every evaluated aspect, most participants selected ratings of 4 or 5 on the five-point scale. The mean scores ranged from 4.00 to 4.70, indicating positive user perceptions of the prototype. These findings provide evidence of the system's functional suitability and user acceptance under the tested conditions. However, they do not directly demonstrate reductions in search time, cognitive load, or information overload.

Table 1*Participant profile and testing environment*

Attribute	Category	Frequency, <i>n</i>	Percentage (%)
Academic level	Undergraduate	30	100.0
Device	Laptop	30	100.0
Network type	Wi-Fi	30	100.0
Browser version	Google Chrome 138	30	100.0

Table 1 presents the participant profile and testing environment. All 30 participants were undergraduate students from Universiti Utara Malaysia (UUM). Each participant used a Wi-Fi-connected laptop running Google Chrome version 138 to support the browser-automation process. These broadly comparable testing conditions helped reduce major environmental differences during the evaluation. Nevertheless, differences in individual laptop specifications and network performance may still have influenced the participants' experiences.

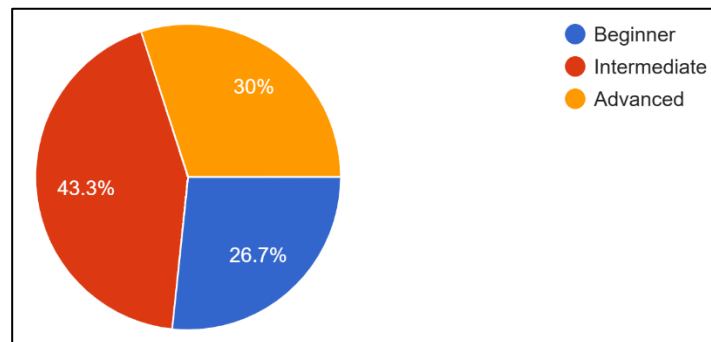
Figure 3*Participants' experience using software tools*

Figure 3 presents the participants' general experience with software tools. Of the 30 participants, 13 (43.3%) described their experience as intermediate, nine (30.0%) as advanced, and eight (26.7%) as beginner. Thus, 22 participants (73.3%) reported at least an intermediate level of software experience. The sample nevertheless included users with different levels of general software experience, allowing both less-experienced and more-experienced users to evaluate the prototype.

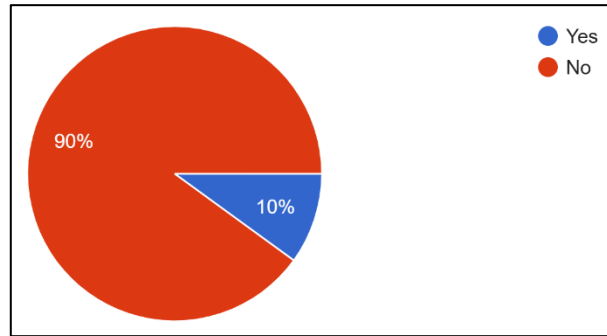
Figure 4*Participants' previous experience with web crawler software*

Figure 4 presents the participants' previous experience with web crawler software. Twenty-seven participants (90.0%) reported that they had not previously used a web crawler, while three participants (10.0%) reported having previous experience. Therefore, most participants were unfamiliar with web crawler software before the evaluation. This participant profile was appropriate for examining whether the desktop application could be operated by users without specialist crawling experience.

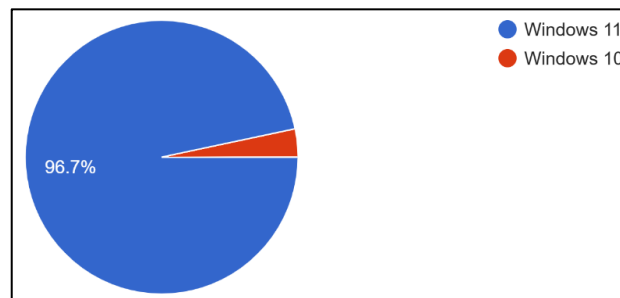
Figure 5*Operating systems used during the evaluation*

Figure 5 shows the operating systems used on the participants' laptops. Twenty-nine participants (96.7%) used Windows 11, while one participant (3.3%) used Windows 10. Thus, all participants evaluated the prototype in a Windows environment, consistent with the system's intended desktop platform. The findings should therefore be interpreted as representing the prototype's operation on Windows 10 and Windows 11 rather than on other operating systems. Table 2 summarises the functional test outcomes. For each test case, we compared the observed result with the expected outcome and classified it as a pass or a fail.

Table 2*Functional test outcomes*

Test case	Pass, <i>n</i>	Fail, <i>n</i>	Pass rate (%)
Input validation for URL, keyword, and page limit	30	0	100.0
Invalid URL handling	30	0	100.0
Starting a crawling task	30	0	100.0
Stopping a crawling task	30	0	100.0
Clearing crawling inputs and outputs	30	0	100.0

Link extraction	29	1	96.7
Keyword matching	30	0	100.0
Displaying and saving records	30	0	100.0
Searching stored records	30	0	100.0
Filtering records by month	30	0	100.0
Automatic shutdown	30	0	100.0

Input validation, invalid URL handling, crawling controls, keyword matching, record display and storage, history searching, month filtering, and automatic shutdown passed in all 30 participant sessions. Link extraction passed in 29 sessions and failed in one, for a 96.7% pass rate. The failure occurred because the target webpage required human verification and could not be accessed through the automated crawling process. This outcome represents a limitation imposed by the website's access-control mechanism rather than a function that the crawler should attempt to bypass.

The results indicate that the prototype's principal features operated as expected in most participant sessions. However, the pass rates reflect outcomes under the selected URLs, devices, browser versions, and network conditions. They should not be interpreted as measures of retrieval precision, recall, crawling speed, or performance across all websites.

The user acceptance questionnaire examined five aspects of the participants' experience. Table 3 summarises the frequency and percentage of favourable ratings (scores of 4 or 5), along with the mean and standard deviation for each item.

Table 3

Summary of user acceptance evaluation

Evaluation item	Ratings of 4 or 5, <i>n</i>	Ratings of 4 or 5 (%)	Mean	Standard deviation
Input validation	29	96.7	4.70	0.65
Crawling effectiveness	26	86.7	4.13	0.73
Reliability and stability	22	73.3	4.00	0.83
Result-display accuracy	26	86.7	4.43	0.73
Overall satisfaction	26	86.7	4.00	0.64

The mean scores ranged from 4.00 to 4.70. Input validation received the highest mean score, followed by result-display accuracy and crawling effectiveness. Reliability and stability and overall satisfaction both had a mean score of 4.00. Reliability and stability had the largest standard deviation, indicating greater variation in participants' ratings for these aspects.

Figure 6

Participant ratings of input validation

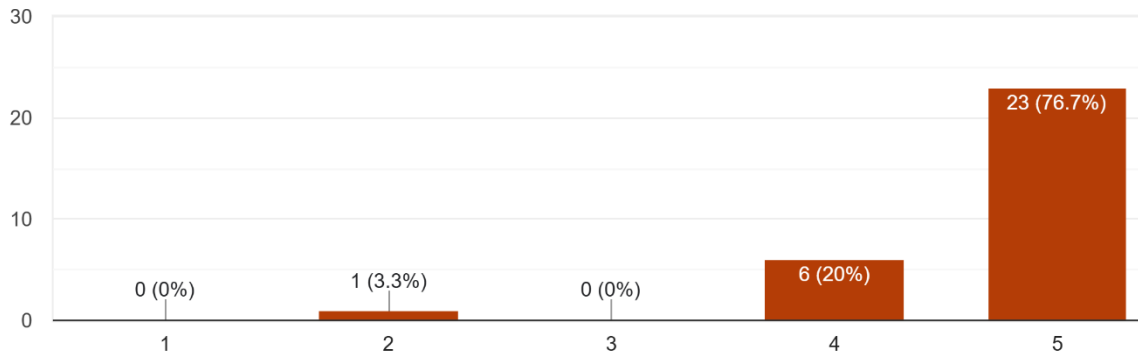


Figure 6 presents the participant ratings of the input-validation function. Twenty-three participants (76.7%) assigned a score of 5, while six participants (20.0%) assigned a score of 4. One participant (3.3%) assigned a score of 2, and no participant selected scores of 1 or 3. The item achieved a mean score of 4.70 (SD = 0.65), with 96.7% of participants providing favourable ratings of 4 or 5. This was the prototype's highest-rated aspect.

Figure 7

Participant ratings of crawling effectiveness

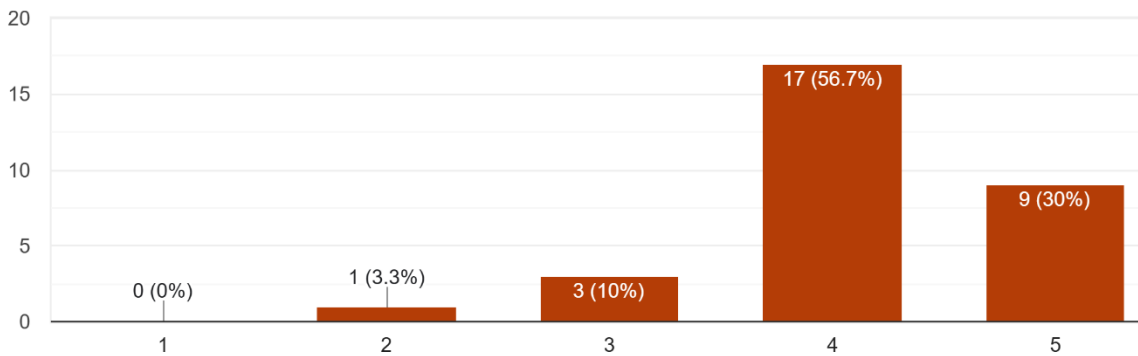


Figure 7 presents the ratings of crawling effectiveness. Seventeen participants (56.7%) assigned a score of 4, nine (30.0%) assigned a score of 5, three (10.0%) assigned a score of 3, and one (3.3%) assigned a score of 2. No participant selected a score of 1. Crawling effectiveness obtained a mean score of 4.13 (SD = 0.73), and 86.7% of participants provided favourable ratings. These findings indicate that most participants perceived the crawler as effective under the tested conditions, although four participants provided neutral or lower ratings.

Figure 8

Participant ratings of system reliability and stability

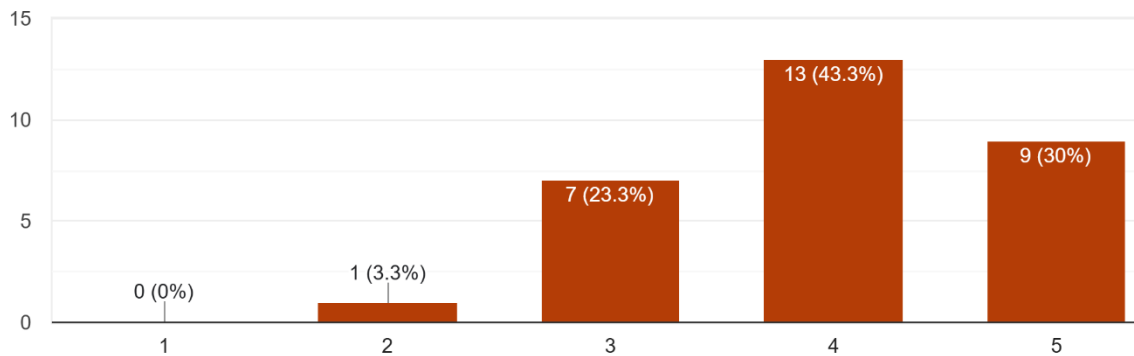


Figure 8 presents the ratings of system reliability and stability. Thirteen participants (43.3%) assigned a score of 4, nine (30.0%) assigned a score of 5, seven (23.3%) assigned a score of 3, and one (3.3%) assigned a score of 2. No participant selected a score of 1. Reliability and stability had a mean score of 4.00 ($SD = 0.83$), with 73.3% of participants rating them 4 or 5. Although most responses were favourable, this item had the lowest proportion of favourable ratings and the greatest variation. This suggests that system stability should receive particular attention in subsequent development and testing.

Figure 9

Participant ratings of result-display accuracy

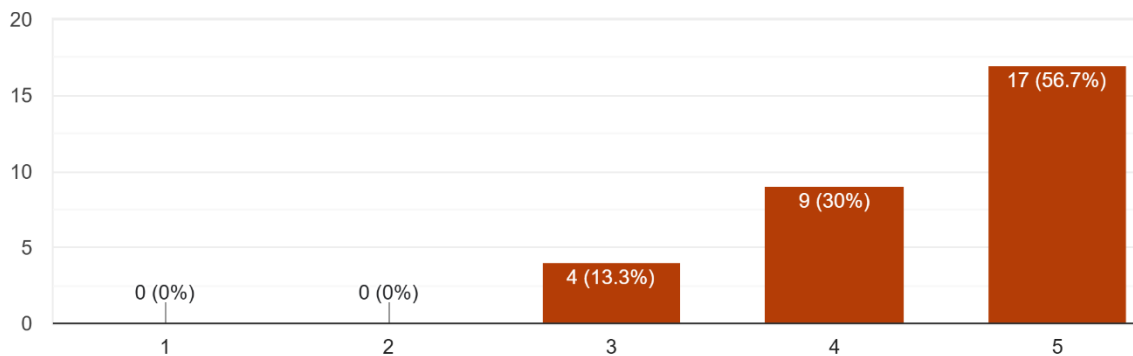


Figure 9 presents participants' perceptions of result-display accuracy. Seventeen participants (56.7%) assigned a score of 5, nine (30.0%) assigned a score of 4, and four (13.3%) assigned a score of 3. No participant selected scores of 1 or 2. The item achieved a mean score of 4.43 ($SD = 0.73$), with 86.7% of participants providing favourable ratings. These results indicate that most participants perceived the displayed information as clear and consistent with the completed crawling task. Since this measure was based on participant perceptions rather than comparison with a verified reference dataset, it should be interpreted as perceived result-display accuracy.

Figure 10

Participant ratings of overall satisfaction

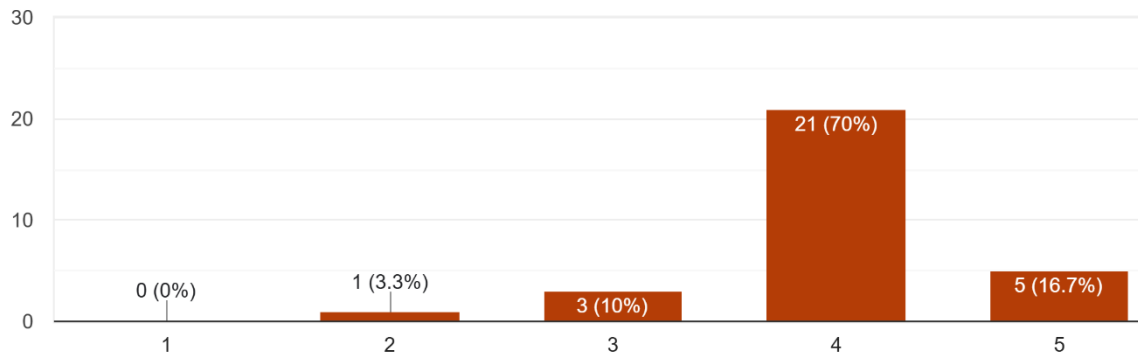


Figure 10 presents the overall satisfaction ratings. Twenty-one participants (70.0%) assigned a score of 4, five (16.7%) assigned a score of 5, three (10.0%) assigned a score of 3, and one (3.3%) assigned a score of 2. No participant selected a score of 1. Overall satisfaction obtained a mean score of 4.00 (SD = 0.64), and 86.7% of participants provided favourable ratings. These findings indicate generally positive acceptance, although they also suggest further improvement may be needed before evaluating the prototype with a larger, more diverse group of users.

Collectively, the functional testing and user acceptance evaluation addressed the second and third research objectives. The functional results showed that the principal features operated as expected in nearly all participant sessions, while the questionnaire results demonstrated generally favourable perceptions across all five evaluated aspects. Nevertheless, the findings are limited to a sample of 30 undergraduate students and a controlled Windows-based testing environment. Additional comparative testing is required to establish whether the crawler improves search efficiency, reduces manual effort, or decreases perceived information overload.

CONCLUSION

This study developed and evaluated a Windows-based desktop web crawler for keyword-based information retrieval. The completed prototype integrated URL and keyword validation, browser-based webpage retrieval, domain-link extraction, HTML content cleaning, rule-based keyword matching, result categorisation, database storage, history searching, month filtering, crawling controls, and structured result presentation within a graphical user interface. These functions addressed the first research objective by providing an integrated desktop application intended to support users with limited web-crawling experience.

The functional testing addressed the second research objective by examining whether the principal system functions operated as intended under the specified testing conditions. Ten of the eleven test cases achieved a 100.0% pass rate. Link extraction achieved a 96.7% pass rate because one webpage required human verification and could not be accessed through the automated process. These findings indicate that the prototype's principal functions operated consistently across most of the participant sessions. However, the results apply only to the selected webpages, devices, browser version, network conditions, and testing environment. They should not be interpreted as evidence of performance across all types of websites.

The user acceptance evaluation addressed the third research objective by examining participants' perceptions of five aspects of the prototype. Input validation received the highest mean score (4.70),

followed by result-display accuracy (4.43) and crawling effectiveness (4.13). Reliability and stability, along with overall satisfaction, each achieved a mean score of 4.00. Most participants rated every evaluated aspect 4 or 5, indicating generally favourable acceptance of the prototype. Nevertheless, reliability and stability showed the greatest variation in responses, suggesting these aspects should receive particular attention during further development.

Overall, the findings demonstrate that the prototype is functionally suitable and generally accepted by the participants under the tested conditions. The crawler may reduce the need to open and inspect webpages individually by automatically retrieving, filtering, storing, and presenting keyword-related content. However, the present study did not directly measure search time, manual effort, cognitive load, information overload, or productivity. Therefore, we cannot draw definitive conclusions about the system's effect on these outcomes.

The study was limited by its relatively small sample of 30 undergraduate students, the participants' limited previous experience with web crawlers, and the controlled Windows-based testing environment. Website access restrictions also affected the crawler, including pages requiring human verification. Such controls should be treated as legitimate access limitations and should not be bypassed. In addition, the current evaluation relied primarily on functional outcomes and self-reported user perceptions rather than comparative performance measures.

Future research should evaluate the crawler using a larger and more diverse sample, different types of websites, and a wider range of computing and network conditions. Comparative experiments could measure task-completion time, number of webpages manually inspected, retrieval precision and recall, task accuracy, perceived cognitive load, and user satisfaction when using the crawler compared with conventional manual searching. Future development could also introduce completion and interruption notifications, more informative error messages, configurable crawling limits, and improved monitoring of lengthy crawling tasks. Integration with subscription-based databases should be implemented only through official application programming interfaces or other authorised access mechanisms. These improvements and further evaluations would provide stronger evidence of the crawler's usefulness, usability, and suitability for longer-term academic or institutional application.

REFERENCES

- Ahmad, S. S., Dar, M. D., Zaffar, M. F., Vallina-Rodriguez, N., & Nithyanand, R. (2020). Apophanies or epiphanies? How crawlers impact our understanding of the web. In *Proceedings of the Web Conference 2020* (pp. 271–280). Association for Computing Machinery. <https://doi.org/10.1145/3366423.3380113>
- Al Galib, A., Mehedi, M. H. K., Rhythm, E. R., & Rasel, A. A. (2024). Large-scale web crawling and distributed search engines: Techniques, challenges, current trends, and future prospects. In *Communications in Computer and Information Science* (Vol. 2001, pp. 17–29). Springer. https://doi.org/10.1007/978-981-99-9589-9_2
- Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, 14(11), 246–270. <https://doi.org/10.3991/ijim.v14i11.13269>
- Bawden, D., & Robinson, L. (2020). Information overload: An introduction. In *Oxford research encyclopedia of politics*. Oxford University Press. <https://doi.org/10.1093/acrefore/9780190228637.013.1360>
- Chang, Z. (2022). A survey of modern crawler methods. In *Proceedings of the 6th International Conference on Control Engineering and Artificial Intelligence* (pp. 21–28). Association for

- Computing Machinery. <https://doi.org/10.1145/3522749.3523076>
- Chowdhury, G. G. (2004). *Introduction to modern information retrieval* (2nd ed.). Facet Publishing.
- Huang, W., Li, X., Zhou, X., Qi, D., Xi, J., Liu, W., & Zhao, F. (2024). A semantic and optimized focused crawler based on semantic graph and genetic algorithm. *Symmetry*, 16(11), Article 1439. <https://doi.org/10.3390/sym16111439>
- Kumar, N., & Aggarwal, D. (2021). Learning-based focused web crawler. *IETE Journal of Research*, 69(4), 2037–2045. <https://doi.org/10.1080/03772063.2021.1885312>
- Tacuri, A., Firmenich, S., Fernández, A., Riva, F., Urbietta, M., & Rossi, G. (2025). Web scraping by end users. *IEEE Access*, 13, 205027–205044. <https://doi.org/10.1109/ACCESS.2025.3636662>
- Truong, B.-V., Nguyen, L. T. T., Pham, P., & Vo, B. (2025). HTMLDownloader: An open-source tool for dynamic web scraping and archiving using WebView2. *SoftwareX*, 32, Article 102373. <https://doi.org/10.1016/j.softx.2025.102373>
- Wan Ishak, W. H., Yamin, F. M., Razali, M. N., & Manaf, M. S. (2025). Evaluating user experience in web applications using WAMMI. *Borneo International Journal*, 8(2), 1–5. <https://majmuah.com/journal/index.php/bij/article/view/675>
- Yamin, F. M., Ramayah, T., & Ishak, W. H. W. (2013). Information searching: The impact of user knowledge on user search behavior. *Journal of Information & Knowledge Management*, 12(3), Article 1350023. <https://doi.org/10.1142/S0219649213500238>