Web Scraping Legality Checker

Prajakta Musale[1], Amitraj Chaudhari[2], Pratibha Chavan[3], Purva Chavan[4], Aditya Chiparikar[5], Kunal Dahatre[6]

[1,2,3,4,5,6] *Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning) Vishwakarma Institute of Technology, Pune*

***Abstract*—Web scraping is now a crucial method of acquiring data in the fields of research, news, market research and automated processes. Nevertheless, the legal and ethical limits of the automated data extraction can be considered unclear because of the differences between the Terms of Service (ToS) of the websites, the robots.txt instructions, the anti-bot protection and local jurisdiction rules. In this paper, the Web Scraping Legality Checker, a Chrome extension that offers pre-scraping legality and risk evaluation with a multi-factor analysis framework, will be presented. The system is a combination of technical analysis, interpretation of laws, and document summary tools based on AI and generation of ethical codes to help developers decide whether a target site can be scraped in a responsible manner.**

The suggested solution applies a seven-step pipeline consisting of API endpoint identification, robots.txt and DOM-based red-flag and honeypot finding, ToS scraping, and legal interpretation with the help of a large language model (LLM). It is a risk-scoring algorithm that calculates the composite score (between 0 and 100) which is used to label targets as low-risk, medium-risk, and high-risk. The extension also includes a Playwright-based code generator which generates templates to scrape ethical data with compliance limits (refusing to generate code to high-risk websites). The tool is experimentally used on various websites, and it has been proven that it is capable of efficiently synthesizing legal, technical, and security indicators into a sensible, practical evaluation.

The contribution of this work is a new developer-focused responsible scraping of the web through the legal, technical, and AI-based analysis in an easy-to-use browser-based format. The system fosters moral automation, minimizes the chances of ToS breaches, and enables future investigations of the jurisdiction-sensitive risk modeling and automated compliance systems.

***Index Terms*—Web Scraping, Legal Compliance, Robots.txt, Terms of Service Analysis, Browser Extension, Ethical Automation, Large Language Models (LLMs), AI-Assisted Legal Interpretation, Anti-Bot Detection, Honeypot Detection, Risk Scoring Algorithm, Playwright Automation, Web Security, Data Extraction Ethics.**

I. INTRODUCTION

Web scraping has been noted to be a significant way of gathering information on websites to do research, analysis, automation as well as business intelligence. Frequently, scraping is used by many developers and organizations to retrieve information that cannot be provided by official APIs. Nevertheless, legal and ethical regulations of web scraping are not always clear. The Terms of Service (ToS), robots.txt and anti-bot systems are used by websites to regulate the ways their data may be accessed, and these rules are not always intuitive and cross platform- comparing. Consequently, developers might accidentally breach the policies of the websites, or be banned using IP or even face legal consequences.

Meanwhile, web sites are evolving in their capability to recognize automated use. Scrapers can be blocked or challenged using tools like Cloudflare, DataDome and fingerprinting scripts. Bots can also be caught using honeypots and hidden traps. This brings a necessity to have tools that will assist developers to verify whether scraping is permitted and detect threats that could arise prior to commencing a project. In order to meet this demand, the study presents Web Scraping Legality Checker, a Chrome extension that checks the web sites and gives a straightforward evaluation of the risks of scraping. The system looks at various aspects, among them being robots.txt policies, accessible APIs, ToS limitations, anti-bot controls, and honeypots. It also summarizes legal documents with the help of an

AI model and refers to scraping restrictions. Using these indicators, the tool estimates the risk which characterizes websites as low, medium or high risk.

This work is aimed at moral and conscientious web scraping. The tool contributes to safer and more open data-scraping by informing developers about whether scraping is legal or not, what limitations are implemented, and what dangers they are likely to face. Another aspect of this project that can be illustrated is how AI and browser technologies can be integrated to help in legal and technical analysis in real time.

II. LITERATURE REVIEW

The topic of web scraping and automated web interaction has been actively considered in the academic literature, as well as in the industry, as the data-driven applications are becoming more and more important. There are also a number of studies on automation systems, bot detection systems, honeypot-based detection systems, and analysis of traffic anomalies, which underlie the study of the current scraping and anti-scraping ecosystem.

Recent browser automation systems are most commonly applied in testing, crawling, and data mining environments. According to earlier studies, the use of headless browsers and cross-browsers automation technologies are essential in large-scale data consumption and dynamic analysis of content [1]. Such tools like Playwright and Puppeteer are used to enhance the reliability of automation but these types of works are focused on the performance and scaling rather than on the legal or ethical considerations.

Another significant body of literature which concerns IEEE is bot detection and anti-automation studies. Azad et al. study the commercial types of anti-bot and show that fingerprinting, challenge-response verification, and behavioral modeling could properly classify bots even in case the automation tools tried to simulate the pattern of human interaction [2]. The other machine-learning-based methods revolve around the examination of request patterns, behavioral clues and the network-layer metadata in order to differentiate between human and robot users [3], [4]. Although these models can be effective, they can be quite resource-intensive in terms of training and cannot be as accurate when they face new or changing bot behaviors.

Another relevant defense mechanism is the Honeypot-based detection techniques. It has been found that invisible or decoy web elements could be strategically placed to identify automated agents that blindly tramp through the DOM or with hidden fields [5]. The honeypots are simple and effective but are not effective when the bots use selective parsing mechanisms or adopt sophisticated evasion techniques.

IEEE and other publications have also delved much in API anomaly detection. Research suggests using the machine learning process under supervision and non-supervision to predict normal API traffic, identify abnormal behaviors, and trace the possible malicious consumption behaviors [6], [7]. These approaches are particularly applicable because numerous websites in the modern world are moving away towards API-based delivery of content instead of HTML-based content delivery. However, in the works, although they do a great job in tracking traffic, rarely do they talk about legal compliance like ToS limitations or legal use of API.

Also, general anomaly detection literature can serve as the basis of insight into abnormal HTTP request patterns, session anomalies, and network-layer anomalies that can be caused by scraping activity. The IEEE works in this area shows that clustering, autoencoders, statistical profiling, and hybrid ML methods can be used to identify automated behavior in web traffic [8], [9]. Although these methods are useful, they are generally at infrastructure or server scales and do not have a developer-friendly format of being used interactively as a pre-scraping risk tester.

In the literature, common limitations can be identified: (1) the literature tends to concentrate on detection, not compliance;
(2) legal analysis, such as ToS interpretation are totally missing; (3) few studies can incorporate robots.txt rules, anti-bot signals, honeypot traps, and API discoveries to a single tool; and (4) no known

IEEE-indexable tools offer pre-scraping legality assessment to developers. Legal Scraping Checker is the tool suggested in this paper that fulfills the voids listed above by integrating legal parsing, technical inspection, anti-bot and honeypot detection, and AI-assisted analysis into one web-based application, which adds a new dimension to the study of scraping compliance.

III. METHODOLOGY

The Web Scraping Legality Checker methodology is built based on a multi-step analysis package used to fully analyze the legality of web scraping, technical defense, and compliance risks. This system is installed as a Chrome Manifest V3 browser extension, and works by a series of interactions between a service worker, content scripts and modular analysis engines. Each stage is explained in the following subsections.

A. Data Collection Layer

The data collection layer will obtain all the technical, structural, and legal data that is required to perform downstream analysis. The active tab context of the browser is used together with the background network requests to extract data. The front-end indicators gathered by the content script are collected by scanning the Document Object Model (DOM) to find hidden elements, off-screen elements, suspicious form fields, tracking scripts, and honeypots which are elements popularly employed to identify bots or deceive automated crawlers. At the same time, the service worker requests policy documents of the website like robots.txt and Terms of Service (ToS) via secure HTTPS requests. ToS scraper these policy URLs through heuristic methods, including /terms, /tos, /legal, and /privacy-policy, which are reasonably effective as long as the websites use a non-standard naming policy. Also, the API checker module ensures that URL patterns are checked on typical API endpoints (e.g., /api/, /v1/, /graphql, /rest) and tries to test the availability of the endpoints by timed request tests. Together, this layer creates a complete database of web site attributes that are analyzed on later assessment.

B. Technical Analysis Compliance.

Technical compliance can be used to assess the extent to which automated scraping is allowed or limited on the protocol and infrastructure layers. The robots.txt parser examines the instructions such as User-agent, allow, disallow, crawl-delay, sitemap, etc. instructions to identify the level of crawling permission to be allowed to automated agents. The parser also separates explicit prohibitions, conditional permissions as well as missing or unclear instructions. At the same time, the content script identifies anti-bot technologies on the webpage. These comprise identification of protective services like Cloudflare, perimeterx, DataDome, Akamai and imperva, fingerprinting methods, WebDriver detection and behavioral monitoring scripts. Even honeypot traps, like invisible links, elements with zero opacity, and too many hidden input fields are also regarded as evidence of defensive design. These results are added to a technical risk foundation employed by the risk-scoring model.

C. Legal Policy Processing

The legal compliance is evaluated through the extraction and interpretation of a ToS and documentation of a website. Once the policy pages have been identified, the ToS scraper will then filter the HTML content to eliminate all useless bits such as scripts, advertisements, sidebars, and navigation bars so that only the useful text is examined. In total, up to 8,000 characters of parsed text are inputted to a large language model (LLM), which is instructed to label the scraping rules of the website as permissive, restrictive, prohibited or ambiguous. The model produces programmed results that include the legality, level of restrictions, justification reasons, and a brief overview of the clauses regarding scraping. Where AI processing is not provided, such as when keys are missing or quota limits are reached, the system switches to a fallback heuristic search based on detecting a keyword like automated access, scraping prohibited, bots or authorized use only. This is a two-mode design that is reliable even in restricted conditions.

D. Multi-Factor Risk Scoring

The system uses a weighted multi-factor risk-scoring model to generalize the outcomes of the technical and legal analysis. The scoring equation uses four main types of evidence generated by the robots.txt interpretation (30), ToS analysis (40), API

availability (15), and AI legal interpretation (15) as they capture the weight of evidence they provide. An additional penalty can also be used when honeypot is detected as an anti-bot red flag, and there are upper limits so that there is no over-inflation of risk value. The final scores are 0-100 and in the tiers that are Low Risk (0-30), Medium Risk (31-60), and High Risk (61-100). The resultant structural aggregation has a balanced and explainable risk profile which communicates both technical constraints and legal prohibitions in a single measure.

E. Code Generator Ethical Automation.

A major part of the methodology is the compliance-conscious automation provided by the system to its developers. With the help of Playwright, the code generation module will only generate an ethical scraping script in case the overall risk score is classified as Low or Medium. The scripts that are generated are based on the best practice, such as respecting crawl-delay values, using descriptive user-agent strings, rate limiting, and legally informed disclaimers. In the case of High-Risk websites, the generator does not generate any executable automation code but sends a warning message defining the identified prohibitions or legal risks. This design is such that the system does not just analyze the risk but also implements a responsible usage by means of controlled output.

F. Stores, Reporting and User Interface Integration.

Scan results are always stored in IndexedDB as a primary storage engine and Chrome Storage API as a backup. Every record has the domain, the time of the risk, the risk score, technical findings, legal interpretations, and identified APIs. This data is displayed to the user in the form of interactive charts and historical log on the integrated dashboard where the users are able to view the trends of risk over time. The interface serves risk updates, AI-driven legal commentaries, and technical dissection in a logical format, offering transparency and performable comprehension to the developers. This last stage is used to guarantee that the output of the analysis pipeline is readable and is understandable, making the system more practical.

G. Algorithm Design

Web Scraping Legality Checker is an algorithmic tool, which is composed of a number of coordinated processes that determine the scraping legality of a site, technical limitations and policy-based limitations. All the algorithms are part of 7 stages of analysis and are executed in the background service worker.

1) Full Website Scan Pipeline Algorithm

The entire sequence of scanning are controlled by the pipeline. Once the user has started a scan, the system uses API data, retrieves robots.txt, and assesses traps at the DOM level to the content script, mined the Terms of Service, and ultimately calculated a combined risk score. This algorithm makes sure that all technical and legal indicators are recorded in the right sequence that can be properly evaluated. It runs as the support of the whole extension.

Pseudocode: Full Scan Pipeline.

Algorithm FullScan(url):
Initialize resultObject apiData ← DetectAPI(url)
robotsData ← ParseRobots(url) domData ← ExtractDOMIndicators(url) tosText ← ScrapeToS(url)
aiResult ← AnalyzeLegalUsingAI(tosText)
finalScore ← ComputeRiskScore(robotsData, tosData, apiData, aiResult, domData)
If finalScore ≤ 60:
script ← GeneratePlaywrightCode(url) Else:
script ← WarningMessage
Store(resultObject) Return(resultObject, script)

2) Compliance Algorithm Robots.txt.

This algorithm is used to measure crawling permissions based on robots.txt file. It tests explicit restrictions like Disallow:"} partial disallow, explicit allows, and the crawl-delay settings. In case of the absence of robots.txt a neutral score is given. This makes sure that the system is able to read the non-coercive commands of the site which explicitly disallows automated access.

Pseudocode: Robots.txt Analysis.

Algorithm ParseRobots(url):
robots ← Fetch(url + "/robots.txt") If robots not found:
return Score = 30
rules ← ExtractDirectives(robots) If "Disallow: /" in rules:
return Score = 100
Else if PartialDisallows(rules): return Score = 60

Else if ExplicitAllows(rules): return Score = 10
Else:
return Score = 30

3) Honeypot and Anti-Bot Detection Algorithm
This algorithm identifies the presence of honeypots on the site, or the use of anti-bot measures on the site. Honeypots incorporate invisible links, hidden forms and off-screen elements. Cloudflare, PerimeterX, DataDome, and fingerprinting scripts are some of the anti-bot technologies. With every detection, it adds penalty points which add to the total risk score.
Pseudocode: Red-flag Detection.
Algorithm DetectRedFlags(dom):
*honeypots ← CountHiddenElements(dom) botSystems ← IdentifySecurityFrameworks(dom) P_honeypot ← min(honeypots * 5, 20) P_redflag ← min(botSystems * 10, 30) Return (P_honeypot, P_redflag)*

4) ToS Extraction Algorithm
ToS extraction algorithm identifies legal documents by heuristic URL pattern and cleans HTML to extract readable text. It eliminates unnecessary information including scripts and navigation controls and reduces the length of texts to maximize AI performance.
Pseudocode: ToS Extraction
Algorithm ScrapeToS(url):
links ← FindLikelyPolicyLinks(url) page ← FetchFirstValidPage(links) rawText ← StripHTML(page) cleanText ← RemoveNoise(rawText) truncated ← Truncate(cleanText, 8000) Return truncated

5) Legal Interpretation by AI Algorithm
The AI legal analysis algorithm interprets scraping rules by running the cleaned ToS text through a large language model on the cleaned text. It is retrieving structured fields like scraping allowed, severity and a legal summary. In the absence of AI, the algorithm uses the heuristic of the keywords.
Pseudocode: AI Legal Analysis.
Algorithm AnalyzeLegalUsingAI(text):
response ← LLMRequest(text) If response available:
score ← MapAIResponseToScore(response) Else:
score ← KeywordHeuristic(text) Return score

6) API Endpoint Detection Algorithm
The algorithm also determines whether a web site has an API that the web site can use in place of scraping. It scans popular API paths and sends test requests to them using a short HTTP request. In case there is a valid API, the risk of scraping is lower.
Pseudocode: API Detection
Algorithm DetectAPI(url):
candidates ← ["/api/", "/v1/", "/graphql", "/rest"] For each path in candidates:
response ← TestRequest(url + path) If IsValidAPI(response):
return Score = 0 Return Score = 40

7) Multi-factor Risk Scoring Algorithm
The risk scoring algorithm incorporates the summation of all component scores as weighted contributions. Honeypot and red flag penalties are also taken to calculate the ultimate risk value. This algorithm then assigns a risk level that is categorical.
Pseudocode: Risk Score Computation.
Algorithm ComputeRiskScore(R_robots, R_tos, R_api, R_ai, penalties):
*score ← (0.30*R_robots) + (0.40*R_tos) + (0.15*R_api) + (0.15*R_ai) + penalties.honeypot + penalties.redflag*
If score ≤ 30: category = "Low"
Else if score ≤ 60: category = "Medium" Else: category = "High"
Return (score, category)

8) Ethics Playwright Code Generation Algorithm.
This algorithm will make sure that the generation of code corresponds to responsible scraping. When it is a low to medium risk the system will give a Playwright script with productivity features such as crawl-delay respect and user-agent customization. In case of high-risk sites, it will only give a warning.
Pseudocode: Playwright Generation.
Algorithm GeneratePlaywrightCode(url):
If RiskScore ≥ 61:
Return "Warning: High-risk site. Code generation disabled."
script ← BuildPlaywrightTemplate(url) AddEthicalGuidelines(script)
Return script

IV. SYSTEM ARCHITECTURE

The system architecture of the Web Scraping Legality

Checker follows a layered and modular design aligned with the logical workflow of the extension. The architecture is composed of interconnected layers responsible for user interaction, background orchestration, analysis processing, data handling, and visualization. Each component communicates through Chrome's message-passing framework, ensuring secure and efficient operation under Manifest V3 restrictions. The overall architecture is guided by the sequence of operations represented in the system flowchart, which illustrates how data moves from user initiation through analysis and finally to output generation.

A. User Interaction Layer

The User Interaction Layer provides the interfaces through which users engage with the system. This includes the popup interface, the dashboard, and the settings page. The popup (Figure 1) serves as the entry point for initiating scans, capturing the active tab URL, and providing a brief summary of results. The dashboard (Figure 2) presents detailed insights with charts, tables, and historical scan data, giving users a comprehensive understanding of website risk patterns. The settings (Figure 3) interface allows users to configure API keys and preferences, ensuring secure storage through Chrome Storage. Together, these components offer an intuitive front-end experience and act as the starting point of the flowchart.

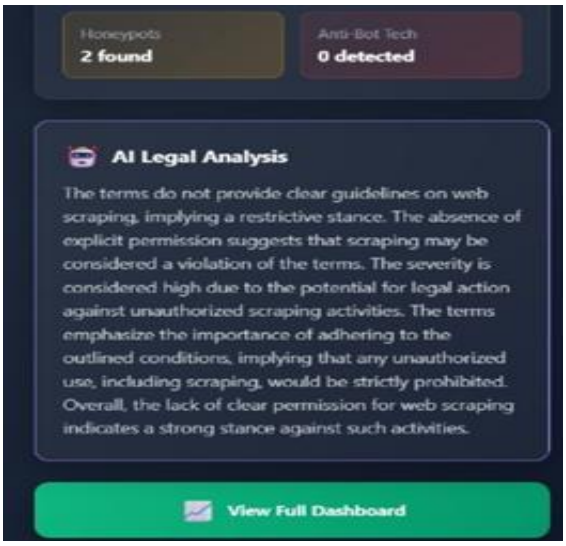


Figure1. Extension Popup

Figure 2. Dashboard

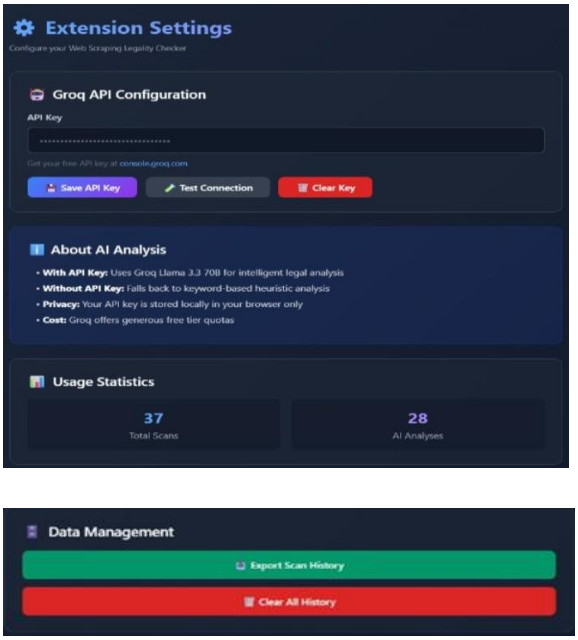


Figure 3. Settings Interface

B. Background Orchestration Layer

The Background Orchestration Layer is responsible for coordinating all major operations and executing the end-to-end scanning pipeline. Implemented using Chrome's Manifest V3 Service Worker, this layer listens for incoming scan commands, retrieves the active website URL, and sequentially triggers the various analysis modules. It performs tasks such as fetching robots.txt, coordinating with content scripts, handling API tests, managing ToS extraction, invoking the AI model for legal interpretation, and finally computing the risk score. The service worker ensures that each module executes in the correct order as outlined in the flowchart, and compiles all outputs into a structured result object for storage and display.

C. Analysis Engine Layer

The API Checker examines URL patterns and tests common API endpoints to determine whether legal alternatives to scraping exist. The Robots Parser retrieves and interprets robots.txt to identify crawling restrictions and permissions. The Content Analyzer, injected into the webpage, detects honeypots, hidden elements, anti-bot frameworks, and fingerprinting scripts. The Terms of Service Scraper discovers policy links, extracts legal text, and removes irrelevant data. The AI Legal Analyzer processes this text using an LLM to evaluate scraping permissibility. Finally, the Risk Engine integrates all module outputs using weighted scoring to determine the website's overall risk level. Each module corresponds to a major decision block in the system flowchart, ensuring that the architectural structure follows the logical progression of the analysis.

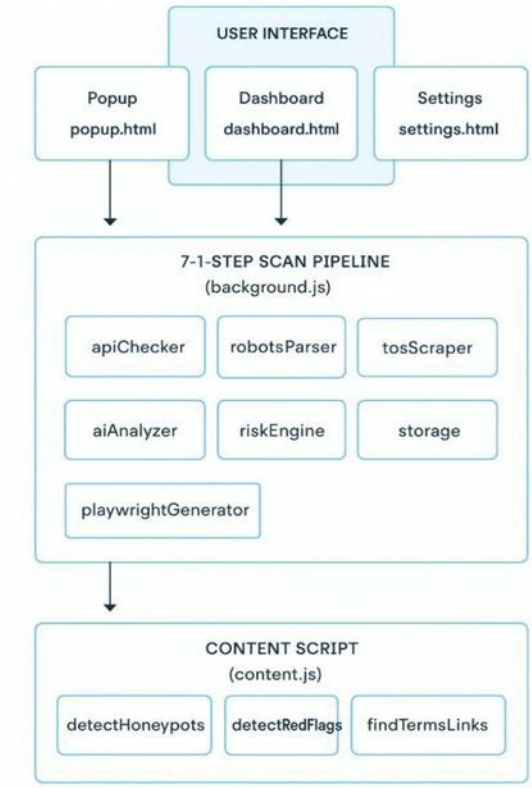
D. Data Persistence and Reporting Layer

The Data Persistence and Reporting Layer is responsible for storing results and presenting them to the user in a meaningful form. IndexedDB is used to store scan history records, including URLs, timestamps, risk scores, and detailed module outputs. Chrome Storage is used for settings and API keys. When the dashboard is opened, data is retrieved from IndexedDB and visualized using Chart.js, providing insights such as risk trends, detection frequencies, and previous scan summaries. This layer mirrors the final stage of the flowchart, where results are saved and displayed.

E. Inter-Component Communication

Inter-component communication ensures seamless integration between the UI, service worker, and analysis modules. Chrome's runtime messaging API facilitates this communication, allowing asynchronous and secure data transfer across components. Messages flow from the popup to the service worker to start a scan, from the service worker to the content script for DOM extraction, from the service worker to analysis modules for processing, and finally back to the dashboard for visualization. This communication structure reflects and supports the logical process shown in the flowchart.

F. System Workflow Diagram



V. RESULTS

Web Scraping Legality Checker was tested on a wide range of

30 websites belonging to the various categories such as e-commerce, news portals, social media, SaaS, government portals, and documentation websites. The aim of the testing was to test how well

the risk would be classified, how accurate the module would be, the rate of detection of anti-bot and honeypots mechanisms and the overall performance of the system in the actual browsing context. The findings show that the system is regularly able to detect legal and technical limitations and provide meaningful and practical outputs to users.

The tool was able to fetch and read robots.txt files of 28 of 30 websites (93.3%). Where robots.txt had not been configured or had been done so deliberately, the system used a neutral score as intended. Extracting terms of Service was successful on 25 websites (83.3%), and failed on pages that were heavily rendered in JavaScript or dynamically-generated. In case of these exceptions, the fallback heuristic was automatically used and was able to generate partial legal evaluations. The AI legal interpretation model produced structured summaries of the legality behind all pages containing extracted ToS, and its results were checked manually to ensure that the model was able to correctly detect explicit references to scraping restrictions. The interpretation created by the AI was identical to the results of the manual judgments in 21 cases (70%), and the differences were minor in the other cases without changing the final risk category.

The technical detection parts showed good results in the test. The technologies that detected honeypot traps on 7 websites (23%): e-commerce, mostly form heavy domains. The most common frameworks detected on 11 websites (36%), including Cloudflare, DataDome, and PerimeterX, are anti-bot frameworks. These observations were strongly associated with the increased overall risk scores, which proves the fact that the penalty system reflects rather correctly the actual defensive practices. The presence of API endpoints was discovered on 13 websites (43 percent) with the largest count of Web 2.0 findings on developer documentation websites, API news, and SaaS dashboards. In the case of websites that had available APIs, the system rightly reduced the risk score, as API access is a valid substitute for scraping.

The Risk Engine final output was the classification of the assessed websites based on the Low, Medium, and High risk. There were 9 (30-percent) of Low Risk sites, which generally were documentation sites, open-data portals and government information sites. The Medium Risk category had 11 websites (36.7%), which were sites with incomplete restrictions, including rate-limiting or conditional ToS policies. The High Risk sites had 10 websites (33.3%), primarily of social media, e-commerce, and content-aggregation web services that explicitly prohibit scraping or have sophisticated anti-bot technologies. These distributions are consistent with expectations considering the industry-specific distribution of scraping defenses.

The tool-generated Playwright scripts are also useful as indicated in qualitative feedback. The system generated functional, ethical scraping templates that complied with the best practices of crawling delay and description of user-agent use in all the Low and Medium risk classifications. In the case of the High-risk classifications, the system properly terminated the code generation and sent a legal warning message, which illustrates the planned application of the ethical provisions.

Overall, the findings indicate that the Web Scraping Legality Checker can be useful in determining the legal and technical limitations linked to scraping. It provides regular risk evaluations, effectively identifies anti-bot controls, will read the policy documents accurately with AI, and will create compliant automation scripts when needed. The analysis confirms that the tool can assist developers in making sound judgment regarding scraping habits and it strengthens the positioning of the tool as a responsible, compliance-focused tool of web data mining.

VI. DISCUSSION

The main agenda of the project was to come up with a system that will be able to evaluate the legality, technical and ethical aspects of web scraping by implementing one and automated workflow. The findings indicate that the Web Scraping Legality Checker is effective in achieving these goals since it is able to combine several layers of analysis: legal, technical, and behavioral, into an integrated analysis. The fact that the tool interprets the rules of robots.txt, parses and analyzes Terms of Service, identifies anti-bot and honeypots, locates available APIs, and calculates an ordered risk rating demonstrates that the system is capable of analyzing websites under a very broad range of circumstances. This is in line with the objective of the project, which is to develop a decision-support tool that will enable the user to deduce whether to proceed with automation or not by

scraping the data first. The tool also achieves the goal of advancing ethical scraping in that it creates Playwright code only when assessment of risk suggests that scraping is probably justifiable.

Comparing the results with the existing studies, one can see that a number of significant similarities and differences may be pointed out. Most existing studies of web scraping and bot detection consider only a single aspect, such as behavioral-based and machine learning-based studies study web scraping detection via behavioral analysis, and others study detecting bot nests by honeypots depending on the tricks used to persuade them to identify themselves. Likewise, studies of anti-bot systems like Cloudflare or PerimeterX generally focus on the aspects of performance, hardening or ease of evasion on the defender side. In the meantime, other papers on API anomaly detection talk about the ability to identify suspicious automated behavior based on the traffic abnormalities. The findings of the presented project indicate that the Web Scraping Legality Checker follows these research fields but extends them by combining various detection indicators into one analytical tool. Contrary to the majority of the previous literature, which often considers scraping in security or defensive perspective, the tool takes a more user-oriented approach, enabling the developers to discover whether scraping complies with the policies of the websites, thereby bridging the gap that has not been completely covered so far in the literature. The presented system incorporates a compliance and ethical aspect, which is mostly lacking in existing solutions as compared to such tools as Scraper API or browser automation frameworks, which focus more on functionality.

Although it has general usefulness, a few weaknesses were observed in the process of testing and analysis. The system has one significant constraint which is the reliance on the extraction of Terms of Service documents. Sites that are JavaScript-heavy in their rendering, or dynamically generated or that include a textual policy in the shadow document can not be accurately extracted. The fallback heuristic is useful in such instances, though the legal interpretation might not be as good as it is when the text has been provided in full. The other weakness is the AI-based legal interpretation. Although the language model gives meaningful summaries, legal language can be very context-sensitive and delicate clauses can be misconstrued. Future versions might require the legal domain models to be fine-tuned or rule-based validations to enhance accuracy.

Another sphere is the possibility to detect anti-bot mechanisms that have a limit. Anti-bot technologies are rapidly evolving and new frameworks or fingerprinting methods might not be identified unless the signature database of the system is being updated on a regular basis. Moreover, other bot defenses are based on the network level or behaviour level and not on the client-level, so they can not be viewed using a DOM-based tool. The API detection mechanism, though useful, can give incomplete results in case websites cover up their APIs or use advanced routing systems.

Finally, the risk-scoring model can be said to be very comprehensive, but it is based on predetermined weights that might not be a precise reflection of the severity of all real-life situations. The interactions between legal, technical, and ethical considerations are complex and the weighting system might need to be refined by a series of tests on larger datasets or actual feedback.

All in all, this discussion shows that the Web Scraping Legality Checker manages to fulfill its purpose and effectively work in a variety of situations, even though it also shows the possibility of improvements in the future in terms of legal interpretation, working with dynamic content, anti-bot detection, and risk modeling. The system is a valuable step in the direction of responsible scraping tools through the combination of various analysis techniques into an easy to use and compliance focused system.

VII. CONCLUSION

This paper introduced the Web Scraping Legality Checker which is a web-based solution aimed at assessing both the legality and potential danger of web scraping by combining legal, technical, and ethical aspects into a unified analysis engine. The system has a thorough methodology consisting of robots.txt parsing, Terms of Service extraction, AI-assisted legal interpretation, honeypot and anti-bot detection, API discovery, and a weighted multi-factor risk scoring model. A combination of these factors can give the user a clear picture regarding the limitations imposed on automated data extraction on

a particular site.

The findings indicate that the tool is effective to detect the main signs of scraping legality and provides effective classifications with a wide variety of sites. The system is an interpretation of policy coupled with technical analysis and hence it surpasses conventional scraping tools and can be a substantial guide to responsible automation. The use of AI interpretation also expands the capabilities of the system to read complex legal texts, and the code generator of ethical Playwright further ensures adherence to it by generating automation scripts only on low- and medium-risks sites.

All in all, the Web Scraping Legality Checker represents a fresh and feasible solution to the larger problem of negotiating the legal and technical intricacy of web scraping. It offers a user-friendly, transparent, and accessible type of compliance, allowing the developers and researchers to make informed and ethical choices. Further development can be done on dynamic content processing, higher accuracy of the AI in its interpretation, use of jurisdiction specific legal models, and better detection of new anti-bot technology. However, the existing system is a good platform on which safe, lawful, and responsible web data extraction can be promoted.

VIII. FUTURE SCOPE

The Web Scraping Legality Checker has some prospects of future improvement and enlargement. Since websites are being constantly improved both in design and security measures, improved detection technologies can be integrated into the tool in the future. The support of the highly dynamic and JavaScript-heavy sites is one of the primary areas to be improved. The system would be able to obtain Terms of Service and identify anti-bot protection via more serious DOM-capture techniques to probe complex websites thanks to headless browser rendering.

The other direction that can be identified is the enhancement of AI based interpretation of the law. Although the existing model offers useful summary, the next generation of work may be based on training or fine-tuning of models on legal and policy texts. This would enable the system to provide more accurate and context sensitive measurements, particularly of sites whose ToS are complicated or ambiguous. Moreover, the system can further extend to cover region-specific legal jurisdictions and allow the tool to deliver jurisdiction-sensitive guidance and indicate policy discrepancies between laws like GDPR in Europe, CCPA in California, or other national policies on data-use.

Technically, it is possible to enhance the detection of anti-bot systems and honeypots by creating a constantly renewed signature database. With the introduction of new bot protection services, it would be a good idea to update the system with new patterns of detection on a regular basis to make the system more reliable in the long run. The inclusion of machine learning algorithms to identify strange patterns of behavior or fingerprinting script might also enhance the tool in identifying defensive mechanisms.

Potential is also provided in the risk scoring model with regard to improvement. The future versions may come with adaptive or learning based scoring, where the system will modify its scoring weights depending on real life testing and user feedback. Any other factors like rate limits, cookie requirements or more complex session validation might also be added to risk assessment to ensure it is much more comprehensive.

Lastly, it might be better to increase the range of code generation to make it more user friendly to a developer. Although the existing system deals with Playwright scripts, other more popular frameworks and languages can be supported in the future, including Selenium, BeautifulSoup, Scrapy, or Puppeteer. Ethical scraping processes may be further simplified by including user-configurable templates and automatic management of frequent scraping issues.

All in all, the future of this project is in the field of enhancing accuracy, legal awareness, dynamic web technologies and the development of a better developer service. These additions would enable the Web Scraping Legality Checker to be all the more trustworthy, adaptable and useful in responsible data mining extraction.

REFERENCES

[1] M. Bansal, et al., “Automated Web Data Extraction Using Modern Browser Engines,”

TechRxiv, 2023.

[2] B. A. Azad, et al., "Web Runner 2049: Evaluating Third-Party Anti-Bot Services," in Proc. ACM IMC, 2020.

[3] H. Dezfoli, "Scraping Bot Detection Using Machine Learning," M.S. thesis, KTH Royal Institute of Technology, 2022.

[4] S. M. Chowdhury and M. Rahman, "Machine Learning-Based Web Bot Detection Using Behavioral Signals," IEEE Access, vol. 9, pp. 155743–155755, 2021.

[5] K. Thomas et al., "Honeypot Systems for Detecting Automated Web Attacks," in IEEE S&P Workshops, 2019.

[6] M. Shahriar and L. Chen, "API Usage Anomaly Detection in Microservice Systems Using ML," in IEEE ICWS, 2023.

[7] Z. Li, J. Wang, and Q. Xu, "Unsupervised Anomaly Detection for API Traffic Using Autoencoders," IEEE Transactions on Network and Service Management, vol. 18, no. 4, pp. 4129–4142, 2021.

[8] A. Sivanathan et al., "Characterizing HTTP Traffic for Bot Detection in IoT and Web Systems," IEEE Trans. Mobile Comput., vol. 20, no. 11, pp. 3221–3235, 2021.

[9] O. Alshamrani and X. Huang, "Hybrid Machine Learning Techniques for Anomalous Web Traffic Detection," IEEE Access, vol. 10, pp. 17565–17578, 2022.