Exhibit A

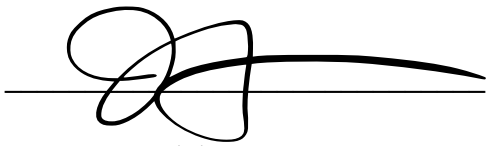
CONTAINS HIGHLY CONFIDENTIAL AEO MATERIAL - SUBJECT TO PROTECTIVE ORDER

UNITED STATES DISTRICT COURT
SOUTHERN DISTRICT OF NEW YORK

IN RE: OPENAI, INC. COPYRIGHT INFRINGEMENT LITIGATION	25-md-03143 (SHS) (OTW)

EXPERT REPORT OF DR. NICK FEAMSTER

APRIL 24, 2026



Nick Feamster

website. When a server receives a connection request, it can check whether the originating IP address falls within a known, published range.

93. Website administrators can use IP address ranges to filter incoming traffic by, for example, restricting requests from certain IP ranges.[57] This is known as *ingress filtering*. As I discuss further in **Section VII.C**, OpenAI publishes its IP egress ranges so that websites can identify and manage incoming traffic associated with its automated systems.

v. Rate Limiting

94. Rate limiting imposes a limit on the number of requests allowed from a client to a server within a period of time.

95. Website owners typically implement rate limiting to optimize website performance (which can degrade under excessive incoming network traffic) and ensure security (such as mitigating the effects of Denial of Service attacks, where a server is overwhelmed with network traffic or requests, rendering it unavailable). When a server detects that a client has exceeded the allowed request rate, it responds with an HTTP `429 "Too Many Requests"` status code, which signals the client to slow down.[58]

96. Responsible network operators implement rate limiting to pace requests to avoid overburdening servers and respect the computational resources of website owners.[59]

B. Web Crawling and Scraping

97. This section provides a detailed explanation of web crawling and web scraping and their foundational roles in the modern Internet. Together, crawling and scraping underpin the operation of core Internet services, including search engines, GenAI systems, and e-commerce platforms.[60]

---

[57] *See What is Egress Filtering?,* Twingate (Oct. 2, 2024), https://www.twingate.com/blog/glossary/egress-filtering.
[58] *See* Mark Nottingham and Roy T. Fielding, *RFC 6585 Additional HTTP Status Codes*, Internet Engineering Task Force (Apr. 2012), https://datatracker.ietf.org/doc/html/rfc6585.
[59] *See Best practices for ethical web crawlers*, AWS (Jan. 6, 2025), https://docs.aws.amazon.com/prescriptive-guidance/latest/web-crawling-system-esg-data/best-practices.html.
[60] *See, e.g.*, Syed Suleman Ahmad et al., *Apophanies or Epiphanies? How Crawlers Impact Our Understanding of the Web*, Proceedings of The Web Conference (Apr. 20, 2020), https://doi.org/10.1145/3366423.3380113 ("Web

i. Web Crawling

98. *Web crawling* is the process of systematically navigating the web to discover and retrieve webpage content. It is typically carried out by specialized software programs known as *web crawlers* or *web spiders*, which conduct crawling in an automated and systematic fashion.[61] These programs are commonly developed and operated by *search engine* companies (such as Google), academic researchers, and any other organization that collects web content at scale. As I mentioned in **Section I.G**, I designed the first web crawler and indexer for the company LookSmart, an early search engine company.

99. A *search engine* is a software system that allows users to find relevant information on the web by entering keywords or phrases into a search interface. In response to a query, the search engine identifies webpages with relevant content and presents the results in an ordered list. To do so efficiently, a search engine must first discover webpages, process their contents, and store information about them in a structured format. This discovery and collection process is accomplished through web crawling.

100. A web crawler begins with a collection of web addresses known as the *URL frontier*.[62] The initial set of URLs from which the crawler starts, defined by the organization operating the crawler, is referred to as the *seed set*.[63] The web crawler proceeds by selecting a URL from the frontier and fetching the corresponding webpage – that is, sending an HTTP `GET` request to retrieve it from the server on which it is hosted. Once the page is fetched, the crawler extracts its content, including text and links.

101. The text content of the page is passed to a *text indexer*, a component of the search engine responsible for organizing and storing the page's contents in a searchable format.[64] At the same time, each extracted link represents a new potential URL to be visited. As a page is fetched, its corresponding URL is removed from the frontier, and newly discovered

crawlers and scrapers have been a core component of the toolkits used by the Internet measurement, security & privacy, information retrieval, and networking communities.").

[61] *See generally What is a Web Crawler?*, Akamai, https://www.akamai.com/glossary/what-is-a-web-crawler.

[62] *See* Christopher D. Manning et al., *An Introduction to Information Retrieval* at 444 (Cambridge University Press, 2009).

[63] *See id.*

[64] *See id.*; *see also What is full-text search?,* Google Cloud, https://cloud.google.com/discover/what-is-full-text-search (last updated Jan. 14, 2026).

URLs are added to the frontier. In this way, the URL frontier functions as a queue of webpages waiting to be visited.

102. Some website owners may disallow crawling on some or all of their webpages. These preferences are reflected in the website's `robots.txt` file, which I explain in detail in **Section V.C**. When the crawler needs to determine whether it is allowed to fetch a URL from a given site or domain, it retrieves that site's `robots.txt` file, parses the rules applicable to its user agent, and caches those rules. Subsequent fetch decisions are then checked against this cached set of rules rather than by retrieving `robots.txt` for every page request. These `robots.txt` policies are commonly cached for up to 24 hours, subject to HTTP caching directives and operational considerations.[65] Through this process, the crawler systematically expands the search index's coverage of the web.

103. Using Google Search as an illustrative example, the process operates as follows.[66] First, Google engages its crawler `Googlebot` in what it refers to as "URL discovery," identifying new and updated webpages. Google discovers webpages through several mechanisms, including following hyperlinks found in previously crawled webpages and processing sitemaps. Sitemaps are files provided by website owners that list the pages and content available on a site, along with additional information about their structure and relationships.[67] This information helps Google identify relevant URLs and crawl the site more efficiently. See **Figure 7** below for a simple visualization of an example sitemap.

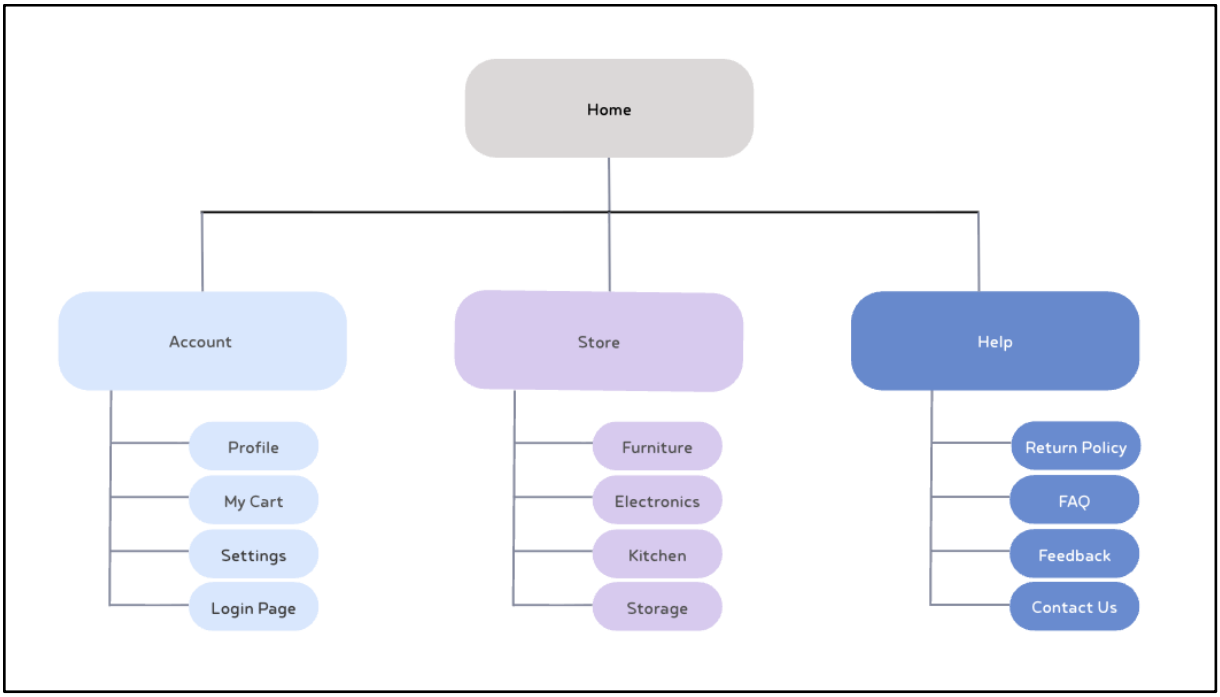
[65] *See, e.g., How Google Interprets the robots.txt Specification — Caching*, Google Developers, https://developers.google.com/crawling/docs/robots-txt/robots-txt-spec#caching (last updated Apr. 14, 2026) (explaining that Google generally caches robots.txt files for up to 24 hours, subject to cache-control headers and operational considerations); *see also Overview of OpenAI Crawlers*, OpenAI Developers, https://developers.openai.com/api/docs/bots/ (describing OpenAI's use of web crawlers and how robots.txt settings are interpreted, including caching robots.txt files for up to 24 hours).

[66] *See, e.g., In-depth guide to how Google Search works*, Google Search Central, https://developers.google.com/search/docs/fundamentals/how-search-works (last updated Dec. 18, 2025).

[67] *See, e.g., Learn about sitemaps*, Google Search Central, https://developers.google.com/search/docs/crawling-indexing/sitemaps/overview (last updated Dec. 10, 2025).

CONTAINS HIGHLY CONFIDENTIAL AEO MATERIAL - SUBJECT TO PROTECTIVE ORDER

Figure 7: A Sample Website Sitemap



104. Once the discovery process is complete, `Googlebot` determines which webpages should be added to the URL frontier. `Googlebot` then examines each discovered website's `robots.txt` policies and ensures that each discovered site's access rules are complied with.

105. Once the relevant URLs have been identified, `Googlebot` proceeds with the crawling phase, sending HTTP requests to retrieve the content of those URLs. As part of this process, `Googlebot` does not merely retrieve the static HTML file of a webpage. Instead, it uses a headless Chromium instance, which is a headless version of the Chrome browser to render the page and execute any associated JavaScript files to capture content that is dynamically generated.[68]

106. After crawling is complete, Google proceeds to the indexing phase. During indexing, Google's search engine analyzes the page's content, parses the structured and unstructured information (such as text, metadata, and links), and stores that information

[68] *See* Lizzi Harvey and Martin Splitt, *Making JavaScript and Google Search work together*, Web.dev , https://web.dev/blog/javascript-and-google-search-io-2019 (last updated June 5, 2019).

in its search index. The indexing process enables subsequent ranking and retrieval of results when users submit search queries. These same processes underpin all web crawling applications.

ii. Web Scraping

107. *Web scraping* is another process on the modern web and is defined as the automated extraction of web-based data.[69] Web scraping is typically carried out by automated programs called *web scrapers*, which extract and download specific content from webpages and store it for later use. A web scraper generally operates by following a pre-defined set of target URLs, or in some cases, by automatically following links found within retrieved webpages.

108. To retrieve a webpage, the scraper sends an HTTP `GET` request to the server hosting the page and downloads the content from the server in the corresponding HTTP response. It then parses the received files and extracts only the specific elements it was designed to capture, such as, for example, contact information from a directory website, or article titles from a digital newspaper.

iii. Understanding the Roles of Web Crawlers and Scrapers

109. It is important to distinguish between web crawlers and web scrapers, as the two perform related but distinct functions. A web crawler focuses on discovering and retrieving webpages by systematically following links across the web, often for the purpose of building a search index. A web scraper, in contrast, focuses on extracting specific data from those pages. In simplified terms, a crawler determines *which pages to visit* by indexing them, while a scraper determines *what information to extract* from the webpages. In practice, a single system may combine both functions.

[69] *See, e.g.,* Tim Keary, *Web Scraping,* Techopedia (June 20, 2024), https://www.techopedia.com/definition/5212/web-scraping ("web scraping is the process of automatically extracting data from a web page. A web scraper extracts a page's HTML code, plus the data stored in its underlying database, and exports it to a third party."); *see also What is a bot?*, AWS, https://aws.amazon.com/what-is/bot/ ("Scrapers, or web scraping crawlers, scan and download specific content on the internet.").

110. Both crawlers and scrapers are examples of what are commonly referred to as *robots,* or simply *bots*. A bot is a software program that performs tasks over a network in an automated fashion. Bots operate by following predefined instructions and communicating using standard network protocols. Once deployed, they can execute tasks continuously and at scale. Similarly, many AI chatbots and LLM-based systems are supported by AI bots, which enable them to interpret natural language, learn from data, and functionally perform the requests of their users.[70]

iv. Industry Norms for Web Crawling and Scraping

111. There are established technical practices designed to promote transparency and prevent undue burden on websites. These industry norms reflect widely accepted practices of responsible automated access to publicly available web content.[71]

112. I discuss five such industry norms: (1) the use of lightweight HTTP `GET` requests rather than resource-intensive headless browsers whenever possible; (2) the declaration of an accurate and descriptive user agent token; (3) the public documentation of that user agent along with opt-out instructions; (4) the publication of IP egress ranges to enable verification of bot identity; and (5) the implementation of rate limits to avoid overburdening source websites. Together, these industry norms form the baseline framework for bot operation.

a. Use of Lightweight HTTP Retrieval (GET Requests vs. Headless Browsers)

113. Crawlers typically rely on lightweight HTTP `GET` requests to retrieve publicly available HTML content when possible. A lightweight HTTP `GET` request simply requests a resource and receives only the HTTP response; it does not initiate additional requests for resources, such as JavaScript, CSS, images or fonts. This approach avoids the additional

[70] *See Overview of OpenAI Crawlers*, OpenAI Developers, https://developers.openai.com/api/docs/bots/ ("OpenAI uses web crawlers ('robots') and user agents to perform actions for its products, either automatically or triggered by user request.").

[71] *See* University of Pittsburgh Library, *Web Scraping Best Practices*, Research Guide (Oct. 6, 2023), https://pitt.libguides.com/webscraping/bestpractices; *see also Best Practices for Ethical Web Crawlers*, AWS, https://docs.aws.amazon.com/prescriptive-guidance/latest/web-crawling-system-esg-data/best-practices.html; Jason Struck*, Web Scraping Best Practices, Statistical Software & Services Collaboratory*, University of Wisconsin–Madison (Feb. 2025), https://sscc.wisc.edu/sscc/pubs/webscraping-r/index.html.

server-side processing required to render pages and run scripts, therefore helping to reduce computational load on the target server and minimize unnecessary resource consumption.[72]

114. In contrast, headless browsers replicate full browser environments, including JavaScript execution and dynamic rendering. At scale, such tools can impose materially greater burdens on server infrastructure. Additionally, from the server's perspective, a headless browser is often indistinguishable from a real human user, meaning that using headless browsers makes bot activity much more difficult for website administrators to detect. Responsible operators therefore limit headless browser usage to circumstances where dynamic rendering is necessary.

b. Accurate and Transparent User Agent Identification

115. As I explained in **Section V.A**, the HTTP `User-agent` header identifies the software making a request on behalf of the user or organization operating the bot. Responsible bot operators include a descriptive and accurate user agent string in each request. This is functionally equivalent to walking into a building with a name tag: by including a descriptive and accurate user agent string in an HTTP request, the bot is identifying itself by name. This makes it very easy for website owners to express their preferences with respect to bot access to their sites, either by updating their `robots.txt` file to disallow specific named bots or by implementing code that rejects any HTTP requests containing user agent strings associated with the bots they want to disallow.[73]

116. The declaration of the actual user agent string that is performing the HTTP request promotes transparency and is foundational to cooperative interaction between automated systems such as crawlers and scrapers and website administrators.[74]

[72] *See* Jerome Mottet, *Optimizing Web Scraping: When to Use (or Not Use) Headless Browsers,* Developer's Tavern (June 2, 2024), https://www.developerstavern.com/optimizing-web-scraping-when-to-use-headless-browsers/.
[73] *See Best Practices for Ethical Web Crawlers*, AWS, https://docs.aws.amazon.com/prescriptive-guidance/latest/web-crawling-system-esg-data/best-practices.html.
[74] *See* Roy T. Fielding et al., *RFC 9110 HTTP Semantics,* Internet Engineering Task Force (June 2022), https://www.rfc-editor.org/rfc/rfc9110.html (addressed in section 10.1.5. "User-Agent").

117. Departures from this practice include omitting the `User-agent` header, using vague or misleading identifiers, or intentionally impersonating common consumer browsers or unrelated services. Such practices obscure the identity of the requesting entity and interfere with a website administrator's ability to manage automated access..

c. Public Documentation and Opt-Out Instructions

118. In addition to declaring a user agent in their HTTP `GET` requests, operators that follow industry norms should publicly document their bot's identity and behavior. This documentation typically includes the user agent string, a description of the bot's purpose, and clear instructions explaining how a website administrator may opt out, often by indicating preferences through `robots.txt` configuration or direct contact.[75]

119. This publication ensures that website administrators have the information necessary to identify both the origin and purpose of bots and decide whether to exclude or block them.[76]

120. Departures from this practice, including failing to publish identifying information, providing no mechanism for opt-out, or structuring operations to avoid easy identification, undermine transparency and impair site-level governance mechanisms.

d. Publication of Egress IP Ranges and Bot Verification

121. As I explained in **Section V.A,** IP addresses identify the network origin of traffic. Responsible bot operators frequently publish their egress IP ranges.[77] This practice serves two important purposes. First, it enables website administrators to verify that traffic

[75] *See* Google for Developers, *List of Google's common crawlers*, Crawling infrastructure, https://developers.google.com/crawling/docs/crawlers-fetchers/google-common-crawlers (last updated Apr. 23, 2026); *see also CCBot*, Common Crawl, https://commoncrawl.org/ccbot; *Overview of OpenAI Crawlers*, OpenAI Developers, https://developers.openai.com/api/docs/bots/.

[76] *See* Martijn Koster et al., *RFC 9309 Robots Exclusion Protocol,* Internet Engineering Task Force (Sept. 2022), https://www.rfc-editor.org/rfc/rfc9309.html.

[77] *See* Google for Developers, *Verify requests from Google crawlers and fetchers*, Crawling infrastructure, https://developers.google.com/crawling/docs/crawlers-fetchers/verify-google-requests (last updated Mar. 20, 2026); *see also Egress policies*, Cloudflare, https://developers.cloudflare.com/cloudflare-one/traffic-policies/egress-policies/ (last updated Apr. 7, 2026) (explaining how "Dedicated egress IPs" can solve the problem of third party services only allowing connections from a known list of IP addresses access to their websites); *Overview of OpenAI Crawlers*, OpenAI Developers, https://developers.openai.com/api/docs/bots/.

claiming to originate from a particular bot is in fact coming from the legitimate operator, thereby mitigating the risk of bot spoofing (i.e., falsely claiming to be a different bot or service). Second, it allows administrators to implement IP-based ingress filtering if they choose to allow or block traffic from specific entities, at a network level.[78]

122. Departures from this practice include concealing IP ranges, issuing HTTP requests through undisclosed proxy networks, or changing the bots' IP addresses periodically to circumvent filtering, server-side rate limiting, or other forms of access restriction. Such practices interfere with verification and prevent website administrators from implementing effective network controls.

e. Rate Limiting and Load Management

123. Rate limiting is central to responsible crawling. Bot operators typically limit the number and frequency of requests sent to a target website and adjust crawl rates to avoid overwhelming the target website with requests.[79] Smaller or resource-constrained sites are typically accessed at lower frequencies than large-scale platforms, as this practice reduces the risk of server overload and preserves access for human users.[80]

124. Departures from this practice, including high-frequency request bursts, ignoring backoff signals from servers, or distributing requests across rotating IP addresses increase the likelihood of service degradation and resource overconsumption.

125. Collectively, these industry norms establish a coherent framework for automated access to publicly available web content. As I elaborate in **Section VII.C**, OpenAI's bots implement each of these practices, enabling their bots to be responsible participants in the broader web ecosystem.

---

[78] *See What is Egress Filtering?*, Twingate (Oct. 2, 2024), https://www.twingate.com/blog/glossary/egress-filtering (which explains that ingress filtering "focuses on incoming traffic, blocking unauthorized access and malicious data from entering.").

[79] *See Best Practices for Ethical Web Crawlers*, AWS, https://docs.aws.amazon.com/prescriptive-guidance/latest/web-crawling-system-esg-data/best-practices.html.

[80] *See id.* ("For small or medium-sized websites, 1 request every 10–15 seconds might be appropriate. For larger websites or those with explicit crawl permissions, 1–2 requests per second might be appropriate.").

ii. Main Features and Interpretation of a Robots.txt File

133. The REP is simple and universal. As a plain text file located at the root of a website, `robots.txt` is easy to identify, edit and update without complicated code compilation or further database work.

134. For example, **Figure 9** shows the `robots.txt` file for the Weather Channel website.[94] The file first specifies user agents that are explicitly allowed to crawl the entire site, as indicated by the directive "`Allow: /.`" In this case, those user agents include `ChatGPT-User` and `AmazonAdBot`.

135. Next, the file defines a "catch-all" section using the "`User-agent: *`" instruction, where the asterisk indicates that the rules apply to all bots not explicitly listed above. Under this section, crawling is generally permitted, but with restrictions on specific paths, specified by the "`Disallow:`" commands. These paths are `/includes/`, `/life/`, and others listed in the file.

136. Note that the `robots.txt` file operates on an opt-out principle: if there is no rule matching a specific user agent token, and no "catch-all" section denoted by an asterisk, the website is understood to allow all access by default.[95]

[94] *See The Weather Channel robots.txt file*, The Weather Channel (Mar. 16, 2026), weather.com/robots.txt.

[95] *See* Martijn Koster et al., *RFC 9309 Robots Exclusion Protocol,* Internet Engineering Task Force (Sept. 2022), https://www.rfc-editor.org/rfc/rfc9309.html ("If no group matches the product token and there is no group with a user-agent line with the "*" value, or no groups are present at all, no rules apply."); *see also* Martijn Koster et al., untitled mailing list archive (1994), https://www.robotstxt.org/robots-nexor-mbox.txt (which states that "empty file = all robots permitted? [...] Yes.").

Figure 9: Excerpt of the Weather Channel robots.txt file

```
#
# /robots.txt
#

#
# Last updated by nickhstr 03/16/2026
#

# Explicit Allows

User-agent: ChatGPT-User
Allow: /

User-agent: AmazonAdBot
Allow: /

# Disallowed for PhantomJS

User-agent: *
# Crawl-delay: 10

User-agent: CriteoBot/0.1
Disallow:
Crawl-delay: .02

# Directories
Disallow: /includes/
Disallow: /life/
Disallow: /misc/
Disallow: /modules/
Disallow: /profiles/
Disallow: /scripts/
Disallow: /themes/
Disallow: /api/
Disallow: /taxonomy/
```

137. The lines preceded by the "#" character denote comments that can provide contextual information for human readers.[96] For example, the comment at the top of the file in **Figure 9** indicates that the file was last updated on "03/16/2026". These comments do not constitute machine-readable instructions and are therefore ignored by bots during

[96] *See* Martijn Koster et al., *RFC 9309 Robots Exclusion Protocol,* Internet Engineering Task Force (Sept. 2022), https://www.rfc-editor.org/rfc/rfc9309.html ("# . . . [d]esignates a line comment.").

processing. This reflects standard practice for code comments more broadly,[97] and is explicitly noted in some `robots.txt` guidelines, such as those published by Google.[98]

138. If a website administrator provides access instructions through the comments, it is not within the REP specification for those comments to influence or change how bots access that website. For example, the New York Times `robots.txt` file includes a comment with a link to its Terms of Service, as shown in **Figure 10**. Nothing in the REP specification requires bots to navigate to that link. The New York Times' inclusion of a hash character (#) at the beginning of each line of this comment is an instruction that indicates that bots should ignore the comment.[99]

Figure 10: Comments included in the NY Times `robots.txt` file

```
# New York Times content is made available for your personal, non-commercial
# use subject to our Terms of Service here:
# https://help.nytimes.com/hc/en-us/articles/115014893428-Terms-of-Service.
# Use of any device, tool, or process designed to data mine or scrape the content
# using automated means is prohibited without prior written permission from
# The New York Times Company.  Prohibited uses include but are not limited to:
# (1) text and data mining activities under Art. 4 of the EU Directive on Copyright in
# the Digital Single Market;
# (2) the development of any software, machine learning, artificial intelligence (AI),
# and/or large language models (LLMs);
# (3) creating or providing archived or cached data sets containing our content to others; and/or
# (4) any commercial purposes.
# Contact https://nytlicensing.com/contact/ for assistance.
```

D. Website Content Controls

139. Website owners can specify how and under what conditions visitors (including human users and bots) should access content on their website, and can choose from several

[97] *See, e.g.*, Diana Chien and Paul Clemons, *Coding and Comment Style,* Broad Institute of MIT and Harvard, https://mitcommlab.mit.edu/broad/commkit/coding-and-comment-style/ ("Your audience is the human who will do something useful with your code."); *see also* Gustavo Gallas, *HTML Comments: How to use the comment tags,* CopaHost (June 21, 2023), https://www.copahost.com/blog/html-comments/ ("HTML comments add notes or annotations within the HTML code. The web browser ignores such code when it renders the webpage. They are primarily intended for developers and designers to provide explanations, reminders, or instructions within the code.").

[98] *See How to write and submit a robots.txt file*, Google for Developers, https://developers.google.com/crawling/docs/robots-txt/create-robots-txt (last updated Nov. 21, 2025) ("The # character marks the beginning of a comment. Comments are ignored during processing.").

[99] *See, e.g.*, *How Google Interprets the robots.txt Specification - Syntax*, Google Developers, https://developers.google.com/crawling/docs/robots-txt/robots-txt-spec#syntax (last updated Apr. 14, 2026) ("To include comments, precede your comment with the # character. Keep in mind that everything after the # character will be ignored.").

RAG from model training, where content in training would be used to permanently modify the model's learned weights through gradient descent and backpropagation.[129]

171. Since OpenAI's release of ChatGPT in 2022, it has released two services that allow the models underlying ChatGPT to search the web and retrieve information before generating an answer to a user query: the "Browse" feature initially released experimentally in March 2023 ("Browse"), and the "ChatGPT Search" feature which began to roll out in Fall 2024 ("ChatGPT Search"). I collectively refer to these features as OpenAI's "Browse/Search Functionality." These are examples of RAG configuration. For a technical description of how these services implemented this type of dynamic retrieval at inference time, see **Section VI.A** below.

VI. OPENAI'S SEARCH SERVICES

172. In this section, I examine the operation of ChatGPT, along with its Browse/Search Functionality, and demonstrate that OpenAI's practices are consistent with the industry standards governing web crawling and scraping.

A. Overview of ChatGPT and Its Search Capabilities

173. OpenAI operates an AI-powered chatbot known as ChatGPT. The models that drive ChatGPT are LLMs pre-trained on large amounts of training data.[130] Through this training process, the models learn patterns in text that enable them to generate responses to user prompts. As a result, once the models are "post-trained" (i.e., trained to follow a desired behavior, in accordance with policy), they are able to perform a variety of tasks, such as answering questions, summarizing information, generating images, and more.[131]

174. As explained in **Section V.E**, in addition to relying on their training data to generate a response, LLMs can consult external sources by conducting a web search when the user's

[129] Both gradient descent and backpropagation are core learning methods for neural networks which assist the model with optimization and adjusting its parameters to reduce errors. *See* Richmond Alake, *A Data Scientist's Guide to Gradient Descent and Backpropagation Algorithms,* Nvidia Developer Technical Blog (Feb. 9, 2022), https://developer.nvidia.com/blog/a-data-scientists-guide-to-gradient-descent-and-backpropagation-algorithms/.
[130] *See* OpenAI, *How ChatGPT and our foundation models are developed*, General FAQs (Feb. 2026), https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-foundation-models-are-developed.
[131] *See* OPCO_MDL_000000300.

query calls for it.[132] This capability is also part of ChatGPT and is an example of retrieval-augmented generation (RAG), which conducts a web search, retrieves relevant results, and incorporates those results into the model's context window before generating a response. Unlike conventional search engines, this process uses the retrieved content as an intermediate input to generate a new, synthesized response tailored to the user's query.

175. OpenAI leverages bots to crawl or scrape publicly available websites. OpenAI uses three user agents to identify those bots: `GPTBot`, `OAI-SearchBot` and `ChatGPT-User`.[133]

176. `GPTBot` is used to automatically crawl publicly accessible web content for use in training OpenAI's AI models. It is not used in connection with OpenAI's Browse/Search Functionality. This report does not address the behavior of `GPTBot` or OpenAI's efforts to collect publicly accessible web content for training AI models.

177. `OAI-SearchBot` is used to automatically discover and retrieve public content that may appear in ChatGPT search results in response to a user query.[134] OpenAI began using `OAI-SearchBot` in 2024 in connection with the rollout of the ChatGPT Search product.[135]

178. The third crawler, `ChatGPT-User`, is used to take direct actions on behalf of ChatGPT users and when conducting a web search.[136] It is not used for crawling the web in any automatic fashion.[137] This is the user agent string that OpenAI used to identify requests issued by the "web browsing plugin" that it announced in March 2023,[138] and continued

[132] *See*, e.g., AWS, *What is RAG (Retrieval-Augmented Generation)?*, https://aws.amazon.com/what-is/retrieval-augmented-generation/ ("The LLM uses the new knowledge and its training data to create better responses [...] The new data outside of the LLM's original training data set is called *external data.*").
[133] *See generally Overview of OpenAI Crawlers*, OpenAI, https://developers.openai.com/api/docs/bots/.
[134] *See Overview of OpenAI Crawlers*, OpenAI, https://developers.openai.com/api/docs/bots/ ("OAI-SearchBot is for search. OAI-SearchBot is used to surface websites in search results in ChatGPT's search features.").
[135] Conversations with Philip Pronin and Adam Perelman.
[136] Conversation with Philip Pronin.
[137] *See* Sandhini Agarwal et al., *ChatGPT plugins*, OpenAI (Mar. 23, 2023), https://openai.com/index/chatgpt-plugins/.
[138] *See* Sandhini Agarwal et al., *ChatGPT plugins*, OpenAI (Mar. 23, 2023), https://openai.com/index/chatgpt-plugins/.

to use as part of its "Browse" service throughout 2024 and as part of its "ChatGPT Search" service launched in late 2024.[139]

179. In the following sections, I explain how OpenAI's Browse/Search Functionality and its crawlers operate. I then demonstrate that OpenAI's bots adhere to a website owner's `robots.txt` preferences and implement blocklists as an additional method to restrict bot access.

180. To do so, I rely on a dataset of 20 million ChatGPT conversation records that were provided to me by OpenAI. These records contain conversation transcripts along with their associated *metadata*. Metadata refers to information *about* the conversations, such as timestamps and user identifiers. The metadata also includes fields reflecting operational events that are not visible to the user, such as whether certain system tools were involved, the type of message generated, and other structured indicators associated with each message, which provide meaningful insight into how the system behaves in practice.

181. I analyze these data samples in conjunction with deposition testimony and my conversations with OpenAI technical personnel to form an understanding of the design and implementation of ChatGPT's browsing functionality.

B. Overview of ChatGPT Conversation Records

182. I analyzed 20 million ChatGPT conversation records for this report corresponding to conversations initiated between December 2022 through November 2024 – i.e., Sampling Period. It is my understanding that these records were selected by Plaintiffs through a uniform random sampling process from a list OpenAI provided of ChatGPT conversations between December 2022 and November 2024.[140]

[139] Conversations with Philip Pronin and Adam Perelman.

[140] Declaration of John Vincent Monaco in Support of OpenAI Defendants' Opposition to News Plaintiffs' Motion regarding Output Logs, MDL ECF 436-1 (filed Aug. 4, 2025), ¶ 3.

technique,[297] and plays an important part in "public-spirited information-gathering online."[298] The arguments made in these briefs demonstrate the overwhelming understanding that both automated crawling and scraping are invaluable tools used by many different information technology professionals, researchers and journalists in a range of important contexts.

278. In particular, these various amicus briefs submitted in *Van Buren* support the conclusion that web scraping of publicly accessible content is a legitimate and widely-practiced activity that serves important research, innovation, and public interest purposes. From the examples outlined above, automated web crawling is similarly a common and widely-accepted practice, and represents a foundational component of the modern Internet's architecture and functionality. The discoverability, organization, and accessibility of online information depend fundamentally on automated systems that retrieve and index web content. Without these practices, the Internet would not function as a searchable, structured ecosystem of information.

iii. Many News Outlets, Including the Plaintiffs, Implement and Rely on Web Scrapers

279. News publishers routinely employ web scraping tools to collect and process information from websites. While news publishers do not always release their web scrapers to the public, both the Chicago Tribune and The New York Times have publicly released some of the web scrapers they have developed. I reviewed the source code of five such scrapers built for journalistic purposes, which the Plaintiffs themselves developed in-house or otherwise are obtained through third-party providers: (1) `nyt-nj-campfin`, a campaign finance scraper published by the New York Times;[299] (2) `nyt-prb-scraper`, a scraper published by the New York Times for retrieving documents from

[297] *See* Brief for the Reporters Committee for Freedom of the Press et al. as Amicus Curiae Supporting Petitioner at 2, 4, *Van Buren*, No. 19-783 (July 8, 2020), https://www.rcfp.org/wp-content/uploads/2020/07/2020-07-08-RCFP-Supreme-Court-amicus-brief-in-Van-Buren-v.-U.S.pdf.
[298] *See For data journalists, an important legal victory,* Reporters Committee for Freedom of the Press (May 2, 2022), https://www.rcfp.org/data-journalism-legal-victory/.
[299] *See* Rachel Shorey and Gautam Hathi, *New York Times Newsroom Developers - nyt-nj-campfin - Scrapers for NJ Campaign Finance Data*, GitHub (Aug. 4, 2017), https://github.com/newsdev/nyt-nj-campfin (describing code used to scrape and structure New Jersey campaign finance filings into JSON format).

the Periodic Review Secretariat concerning Guantanamo detainees;[300] (3) `pandascrapers`, a scraper published by the Chicago Tribune to capture criminal warrants from official Cook County website;[301] (4) `Upton`, a general-purpose web scraping framework published by investigative journalism outlet, ProPublica;[302] and (5) `klaxon`, published by the Marshall Project to allow journalists to monitor website changes such as freedom of information disclosure logs and court records.[303]

280. Scraping by news outlets has been a longstanding journalistic practice. Propublica's open-source project, `Upton`, was released nearly 13 years ago, with the aim of making the common practice of web scraping simpler.[304] Similarly, the open-source reporting tool, `Klaxon`, launched by the Marshall Project, has been used by over 50 news organizations, including the New York Times, the Washington Post, and the Associated Press.[305] While web scraping has a long history within the field of journalism, the scrapers implemented by these organizations do not always follow industry norms.

281. The New York Times uses these scrapers to collect public data from government and commercial websites for products such as election pages, COVID-19 trackers, information on Guantanamo Detainees, and New Jersey campaign finance filings.[306] In

---

[300] *See* Jeremy Bowers, *New York Times Newsroom Developers - nyt-prb-scraper -A Scraper and Parser for the Periodic Review Secretariat's Web Pages for Guantanamo Detainees*, GitHub (Sept. 15, 2016), https://github.com/newsdev/nyt-prb-scraper (describing code that "scrap[es] and pars[es]" PRS webpages and outputs structured data such as CSV, JSON, or TSV).

[301] *See* David Eads, *Chicago Tribune - pandascrapers*, GitHub (Mar. 1, 2013), https://github.com/newsapps/pandascrapers (describing scrapers that collect data, e.g., Cook County criminal warrants, from public websites and store it in a structured database to support newsroom data workflows).

[302] *See* Jeremy B. Merrill et al., *ProPublica - Upton: A Batteries-Included Framework for Easy Web Scraping,* GitHub (May 25, 2013), https://github.com/propublica/upton.

[303] *See* Ryan Murphy et al., *The Marshall Project - Klaxon*, GitHub (Apr. 20, 2026), https://github.com/themarshallproject/klaxon.

[304] *See* Jeremy B. Merrill, *Upton: A Web Scraping Framework,* ProPublica (July 22, 2013), https://www.propublica.org/nerds/upton-a-web-scraping-framework ("Web scraping is a common task in data journalism and news-application development.").

[305] *See* Tom Meagher, *The Marshall Project - Klaxon - News organizations using Klaxon,* GitHub (Mar. 6, 2021), https://github.com/themarshallproject/klaxon/blob/main/NEWSROOMS.md.

[306] *See Bot Information for New York Times Newsgathering*, Cloudflare Radar, https://radar.cloudflare.com/bots/directory/new-york-times-newsgathering; *New York Times Newsgathering*, DataDome, https://datadome.co/bots/new-york-times-newsgathering/; Rachel Shorey and Gautam Hathi, *New York Times Newsroom Developers - nyt-nj-campfin - Scrapers for NJ Campaign Finance Data*, GitHub (Aug. 4, 2017), https://github.com/newsdev/nyt-nj-campfin; Jeremy Bowers, *New York Times Newsroom Developers - nyt-prb-scraper -A Scraper and Parser for the Periodic Review Secretariat's Web Pages for Guantanamo Detainees*, GitHub (Sept. 15, 2016), https://github.com/newsdev/nyt-prb-scraper.

particular, these tools have been used for investigative reporting on cases such as the widespread abuse of tax breaks in Puerto Rico, where the New York Times used scrapers to "automatically Google the names, inspect the results, and identify people whose financial history would be worth digging into further."[307]

282. The New York Times also has an internal team called the "AI Initiatives Team" which develops AI tools and product features to assist with reporting.[308] Most recently, the New York Times has used a custom built LLM tool that scrapes, transcribes and summarizes new podcast episodes, in order to create a report that tracks the growth of the "Manosphere" and right-wing rhetoric in popular podcast forums.[309]

iv. Websites Benefit From and Actively Encourage Crawling and Scraping

283. Websites do not merely tolerate crawling and scraping; they actively depend on it for their business success. Search engine optimization (SEO) is a multi-billion dollar industry precisely because appearing in search results is critical for driving traffic to websites.[310]

284. This dynamic creates an important context for understanding web crawling and scraping norms: most websites want to be crawled or scraped. They submit sitemaps to search engines, optimize their HTML structure for bot readability, and monitor their search rankings obsessively. A website that is not indexed by major search engines is effectively invisible to most potential visitors.

285. Website administrators employ numerous techniques to invite and facilitate crawling. Websites publish XML sitemaps that explicitly list all pages they want bots to discover

[307] *See* Andrew Deck, *How The New York Times Uses a Custom AI Tool to Track the Manosphere*, NiemanLab (Feb. 11, 2026) https://www.niemanlab.org/2026/02/how-the-new-york-times-uses-a-custom-ai-tool-to-track-the-manosphere/.
[308] *See* Press Release, N.Y. Times, *Introducing the A.I. Initiatives Team*, (May 9, 2024), https://www.nytco.com/press/introducing-the-a-i-initiatives-team/.
[309] *See* Andrew Deck, *How The New York Times Uses a Custom AI Tool to Track the Manosphere*, NiemanLab (Feb. 11, 2026) https://www.niemanlab.org/2026/02/how-the-new-york-times-uses-a-custom-ai-tool-to-track-the-manosphere/ (describing The New York Times's use of automated tools, including scraping and analysis systems, to collect and monitor online network data at scale).
[310] *See SEO Software Market (2024 - 2030)*, Grand View Research (Jan. 24, 2024), https://www.grandviewresearch.com/industry-analysis/seo-software-market-report (analyzing the global SEO market).