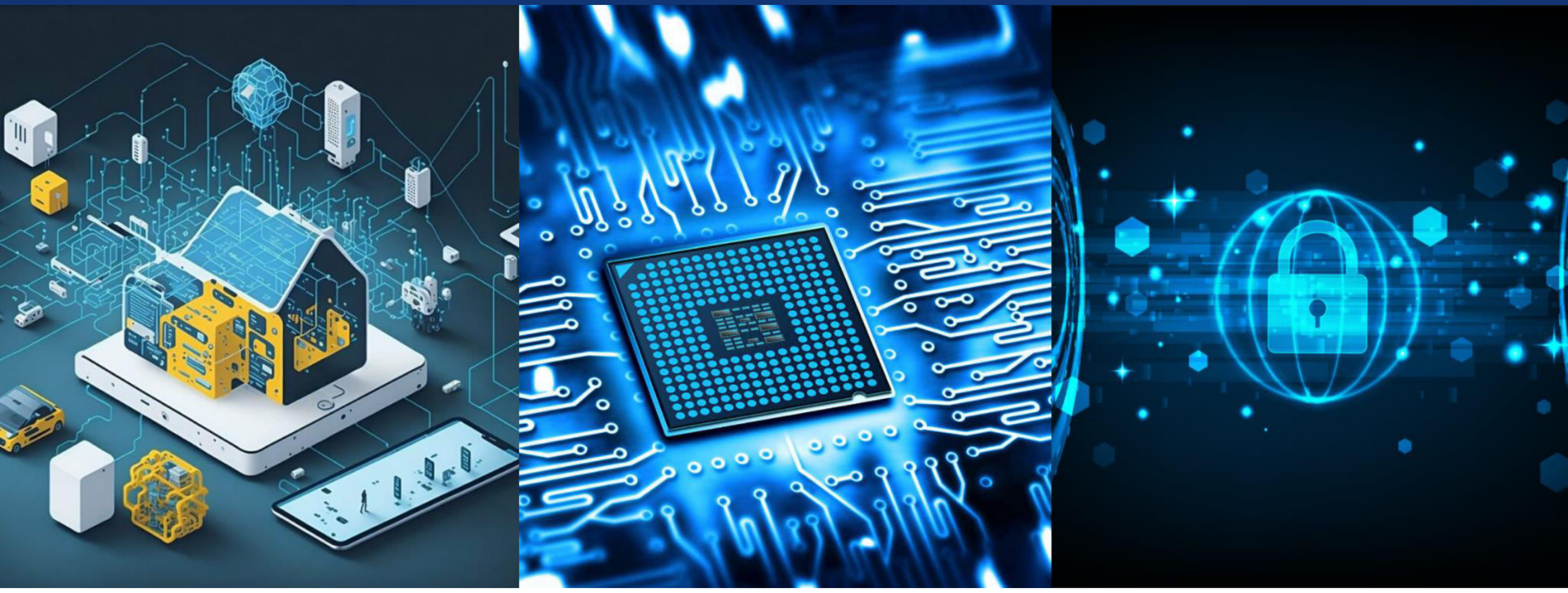
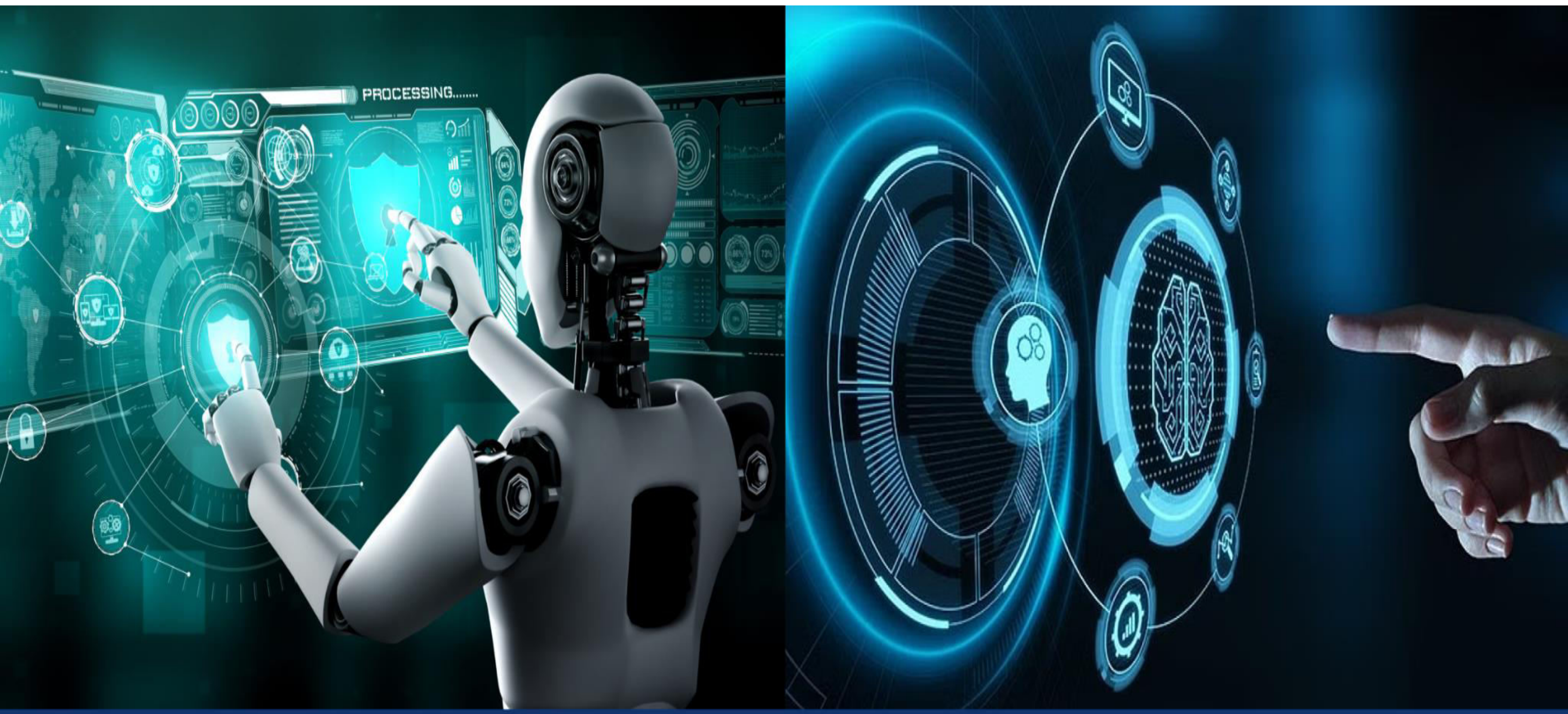
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 4, April 2025

Parser for Social Media Feeds

Prof. Sathish G C, Vivek, Punith V, Abhiram C, Hrishikesh Naik

Department of CSE, Reva University, Bangalore, Karnataka, India

ABSTRACT: Social media platforms have become central hubs of digital interaction, consequently turning into vital sources of evidence for forensic investigations. These platforms contain vast amounts of user-generated information critical for addressing cybercrime, fraud, and other offenses. However, gathering this evidence manually—collecting posts, messages, connection details, and profile data—is a challenging task. It's not only time-intensive but also susceptible to human error and often fails to capture the fleeting nature of online content effectively. Addressing these shortcomings, this paper details the development and architecture of an integrated system designed to automate the process of parsing and documenting social media activity for forensic use. Our solution brings together a secure web application for investigators to manage their work, a robust backend system that uses asynchronous processing for efficiency and scale, and a browser extension that works within the investigator's own browser to carefully extract data and capture screenshots. The system compiles the gathered information into structured PDF reports, offering a clear, timestamped record of relevant social media profiles (initially targeting Facebook, Twitter/X, and Instagram). The ultimate goal is to streamline the evidence collection workflow, minimize manual effort and potential inaccuracies, improve the thoroughness of data capture, and provide forensic professionals with well-documented evidence ready for analysis.

KEYWORDS: Digital forensics, Social networking (online), Evidence collection, Web scraping, Browser extension, Automated data extraction, Asynchronous processing, PDF generation, Evidence preservation, Web application, Node.js, React, Job Queue.

I. INTRODUCTION

The explosion of social media over the past decade has fundamentally reshaped how individuals communicate and share information, inadvertently creating rich, complex digital environments that often hold crucial evidence for legal and investigative matters. Platforms like Facebook, Instagram, and Twitter (now X) serve as digital chronicles, capturing interactions, relationships, opinions, and activities that can be pivotal in criminal investigations, corporate inquiries, incident response analyses, and civil disputes [III]. The content housed within these platforms – ranging from public posts and private communications to intricate networks of followers and detailed user profiles – frequently provides invaluable context, helping investigators piece together timelines, understand motives, map associations, verify locations, and decipher communication patterns.

However, the practical reality of harnessing this potential evidence is often hindered by the cumbersome nature of manual collection methods. Traditionally, investigators have had to undertake the laborious process of navigating through profiles, manually taking numerous screenshots, carefully transcribing relevant text, and attempting to assemble these disparate pieces into a coherent and verifiable report. This approach is inherently inefficient, demanding significant investigator time and resources. More importantly, it is highly susceptible to human error – inconsistent capture procedures, accidental omissions, transcription mistakes, and subjective biases can all compromise the quality and completeness of the collected evidence. Furthermore, establishing the integrity and provenance of manually gathered materials for admissibility in court proceedings can be challenging.

The inherent volatility of social media content, where information can be altered or deleted with alarming speed, adds a critical time-sensitive dimension to the collection process. Operational constraints, such as platforms restricting certain views or functionalities on desktop browsers, sometimes force investigators to use specific devices, further complicating standardization and efficiency [Problem Statement].

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025 |DOI:10.15680/IJIRCCE.2025.1304264

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. BACKGROUND

Navigating the acquisition of evidence from social media platforms presents unique challenges within the field of digital forensics. The sheer volume of data, its dynamic nature, platform-specific structures, and privacy considerations demand specialized approaches. Initial attempts relying solely on manual documentation quickly proved insufficient due to inherent limitations in scalability, the difficulty in ensuring reproducible results, the risk of inadvertently altering the evidence during interaction (spoliation), and the struggle to produce verifiable, timestamped records suitable for legal scrutiny. Consequently, the focus shifted towards developing automated techniques to streamline and improve the reliability of this process. Currently, two primary automated strategies dominate the landscape of social media data extraction: leveraging official Application Programming Interfaces (APIs) where available, and employing direct web scraping techniques [Abstract, PDF 1].

API-Based Extraction: When platforms like Twitter or Facebook offer APIs, they provide a formal, structured channel for accessing certain types of data [IV, PDF 1]. Interacting with an API typically returns data in predictable formats (like JSON), which simplifies subsequent parsing and integration into forensic tools. Operating within the defined boundaries of an API generally aligns with platform policies, mitigating the risk of violating terms of service. However, the utility of APIs for comprehensive forensic collection is often constrained. Platforms frequently impose strict rate limits, capping the amount of data that can be retrieved within specific time windows, which can hinder large-scale investigations. Perhaps more critically, APIs often grant access to only a subset of the information visible on the platform itself. Historical data, certain types of user interactions (e.g., reactions to older content), ephemeral content (like Instagram Stories), or information deemed private might be inaccessible via the API, even if readily viewable by an authenticated user logged into the platform's web interface. Furthermore, APIs are inherently dynamic; platforms continuously evolve their APIs, modify access policies, introduce new versions, or deprecate older endpoints, sometimes with limited notice. This necessitates ongoing maintenance and adaptation for any tool relying on these interfaces [IV, PDF 1]. Instagram, in particular, serves as a prominent example of a platform that has significantly curtailed its API offerings for general third-party access over time.

Web Scraping: In situations where APIs are non-existent, provide insufficient access for forensic needs, or are overly restrictive, web scraping emerges as a common, albeit technically demanding and often fragile, alternative [IV, PDF 1]. This approach involves programmatically interacting with the target platform's user-facing website, simulating user actions (using browser automation tools like Selenium or Puppeteer) or directly fetching and parsing the underlying HTML code of web pages (using libraries such as BeautifulSoup or Cheerio). The primary advantage of web scraping is its potential ability to access *any* data rendered and visible within a standard web browser, potentially capturing information unavailable through APIs. However, this method is fraught with considerable technical and practical obstacles:

High Sensitivity to Change: Scraping scripts are typically built upon specific assumptions about the website's structure (DOM layout, CSS class names, element IDs). Even minor UI redesigns or code refactoring by the platform can easily break the scraping logic, requiring frequent, time-consuming, and technically skilled maintenance efforts.

Platform Defenses: Social media companies actively invest in sophisticated anti-scraping technologies to protect their platforms and user data. These countermeasures can include CAPTCHA verification, IP address blocking or rate-limiting, browser fingerprinting analysis, monitoring user interaction patterns for non-human behaviour, and detecting headless browser automation. Successfully bypassing these defenses requires advanced techniques and carries inherent risks.

Handling Client-Side Dynamics: Modern websites are heavily reliant on JavaScript frameworks to load and update content dynamically after the initial page load (e.g., infinite scrolling feeds, comments loaded on demand via AJAX). Effectively scraping this dynamic content often requires executing JavaScript within the scraper's environment or intercepting network traffic, adding significant complexity beyond parsing static HTML.

Authentication Difficulties: Managing user logins and maintaining active sessions programmatically for server-side scraping can be complex, potentially involving insecure credential storage or triggering multi-factor authentication prompts and security alerts on the target accounts.

Legal and Ethical Considerations: Undertaking web scraping, particularly at scale or aggressively, may constitute a violation of the platform's Terms of Service, potentially leading to account suspension, legal action, or reputational harm. Furthermore, investigators must navigate complex ethical considerations regarding user privacy and consent, especially when collecting potentially sensitive information [IV, PDF 1]. Compliance with data privacy regulations like GDPR and CCPA is not optional but a strict legal requirement [V, PDF 1].
Leading commercial digital forensic suites, including well-known tools like X1 Social Discovery and Magnet AXIOM, often adopt hybrid approaches, attempting to combine the benefits of available API access with sophisticated web scraping techniques to offer comprehensive data collection capabilities [IV, V, PDF 1]. While these tools are powerful, they often represent a significant financial investment and remain subject to the fundamental limitations imposed by platform API restrictions, the inherent technical challenges of reliable scraping, and the constantly shifting legal and technical environment.

III. PROBLEM STATEMENT

The central challenge this work aims to resolve lies in the significant inefficiencies, potential for inaccuracies, and lack of robust, verifiable documentation associated with the traditional manual methods used to collect digital evidence from dynamic social media platforms within forensic investigations. The existing manual approaches struggle to meet the rigorous demands of modern digital forensics concerning speed, comprehensiveness, reliability, and the ability to produce evidence likely to withstand legal scrutiny.
Therefore, the overarching goal is to develop and validate an automated system specifically designed to overcome these shortcomings. To achieve this, the system must fulfill the following essential requirements:

Automate Data Extraction and Visual Capture: Establish a reliable automated mechanism to parse relevant textual information (including, but not limited to, user posts, profile details, timeline entries, connection lists, and associated timestamps) and simultaneously capture contextual screenshots from designated social media profiles.

Enable Targeted Platform and Profile Selection: Provide investigators with the capability to clearly specify the target social media platform (initially supporting key platforms like Facebook, Twitter/X, and Instagram) and identify the specific profile under investigation using standard identifiers such as usernames or direct profile URLs.

Generate Structured, Comprehensive Reports: Automatically assemble the collected textual data and captured visual evidence (screenshots) into a coherent, well-structured, and easily searchable report format, ideally the Portable Document Format (PDF), designed for effective forensic documentation, subsequent analysis, and potential presentation in legal contexts.

Minimize Human Error and Enhance Reliability: Drastically reduce the likelihood of errors commonly associated with manual processes, such as transcription mistakes, inconsistent screenshotting, data omission, and subjective selection, thereby improving the overall reliability and trustworthiness of the collected digital evidence.

IV. RESEARCH GAP

While the field of social media forensics has seen progress, and various tools exist, significant gaps remain that hinder the creation of truly efficient, reliable, and universally applicable collection systems. Our research specifically targets several of these gaps:

Incompleteness of API-Based Data: A primary gap exists because official platform APIs, while structured, often deliberately restrict access to the full spectrum of data potentially relevant for forensic analysis (e.g., older posts, specific interaction types, deleted content remnants visible temporarily, UI elements). Investigators often need access to data precisely *as it appears* to an authenticated user, which APIs frequently do not provide [II].

Fragility and of Server-Side Scraping: Conventional server-side scraping tools, while attempting to access data beyond API limits, suffer from extreme brittleness due to UI changes and face complex challenges in bypassing anti-bot measures and handling dynamic web content [II]. Furthermore, operating server-side often requires managing authentication credentials insecurely or dealing with complex session emulation, making it difficult to truly replicate an authentic user view. There's a need for approaches that offer potentially wider access without the same level of fragility or authentication complexity.

Leveraging Authenticated Context Securely: The most comprehensive view of a social media profile is often available to a logged-in user. However, securely automating data collection *from within* that authenticated browser session without compromising the user's credentials or browser security, and without relying on fragile server-side emulation, represents a significant technical and architectural gap. Existing solutions rarely integrate this client-side context effectively with backend processing.

Lack of Integrated Forensic Workflow Systems: Many tools excel at either extraction *or* analysis *or* reporting. A gap exists for seamlessly integrated systems designed specifically for the forensic workflow. Such a system should encompass secure user management, intuitive case/request initiation, robust handling of potentially long-running collection tasks (including coordination between client and server), reliable and verifiable evidence storage, and integrated report management and retrieval capabilities, all within a cohesive interface.

Bridging Client-Side Capture and Backend Orchestration: Architecturally, effectively coordinating actions between a client-side component (like a browser extension capturing data opportunistically) and a sophisticated backend system (handling job queuing, complex PDF generation, secure storage, user verification) poses unique challenges. Documented examples of architectures successfully bridging this divide for forensic purposes, particularly ensuring data integrity and user attribution across the boundary, are scarce.

Our research directly confronts these gaps. By centering the data capture process around a browser extension (Gap 3), we aim to leverage the authenticated user context to potentially access more comprehensive data than APIs allow, while avoiding the pitfalls of complex server-side authentication management (addressing Gap 2 indirectly). This extension acts as the client-side capture mechanism, directly interacting with the rendered web page. Crucially, we integrate this client-side component with a comprehensive backend system featuring secure user authentication, asynchronous job queuing for scalability, dedicated worker processes for complex tasks like PDF generation, and secure cloud storage (addressing Gap 4 and 5). The explicit verification step, where the backend validates the examiner's username submitted by the extension against its own authenticated user database, provides a critical bridge ensuring data attribution and integrity across the client-server boundary, filling a key aspect of Gap 5. This integrated architecture offers a novel approach compared to purely API-based or traditional server-side scraping solutions.

V. RESEARCH OBJECTIVES

Stemming from the identified challenges and the desire to fill the existing research gaps, we established the following specific objectives for this project:

Architect and Implement a Cohesive Multi-Component System: To design, build, and validate a system architecture that effectively integrates three distinct but interacting components: (a) a user-friendly frontend web application for investigator control and report management, (b) a scalable backend API server responsible for business logic, task orchestration, and data persistence, and (c) a specialized browser extension designed for targeted data capture directly from social media websites.

Establish Secure User Authentication and Authorization: To implement robust mechanisms for user registration and login within the web application, employing industry-best practices such as strong password hashing (bcrypt) and token-based authentication (JWT) to ensure that only verified and authorized investigators can access system functionalities, initiate data collection tasks, and retrieve generated forensic reports.

Develop an Asynchronous Task Processing Backend: To engineer and integrate an efficient asynchronous processing workflow utilizing a message queue (e.g., BullMQ/Redis) and dedicated background worker processes. This architecture aims to handle computationally intensive or time-consuming operations (like coordinating scraping, generating complex PDFs, interacting with cloud storage) without blocking the primary API server, thereby ensuring a responsive user interface and enabling the system to scale effectively under concurrent load.

Create Client-Side Data Extraction Capabilities via Browser Extension: To develop a browser extension capable of dynamically injecting code (content scripts) into the web pages of supported social media platforms (initially Facebook, Twitter/X, Instagram). This extension must be able to accurately identify the platform context, locate and extract relevant information (profile details, visible posts) by parsing the page's Document Object Model (DOM), and capture contextual screenshots, all while operating within the technical constraints and security sandbox of the browser environment.

Implement Structured Report Generation and Secure Cloud Storage: To build a reliable module, executed by the backend worker, that can process the diverse data (text, metadata, images) collected by the extension, systematically compile it into a standardized, structured PDF report format suitable for forensic documentation and review, and implement secure, scalable storage mechanisms for these reports, prioritizing the use of cloud-based object storage (like Google Cloud Storage) for persistence and accessibility.

Ensure Verifiable Data Provenance and User Attribution: To design and implement mechanisms that establish a clear and verifiable link between the data captured by the browser extension and the authenticated investigator who initiated the specific collection request through the web application. This includes backend verification of the examiner's identity submitted alongside the captured data.

Conduct Functional Evaluation and Identify Operational Limitations: To perform thorough functional testing of the fully integrated system, validating the complete end-to-end workflow from request initiation to report download. This evaluation aims to assess the system's effectiveness in meeting the automation objectives and, equally importantly, to critically identify and document the practical challenges, inherent limitations (especially concerning the reliability and maintenance of web scraping), and potential areas requiring future improvement or alternative approaches.

VI. METHODOLOGY

To achieve the research objectives, we adopted a methodology centered on building a distributed system with clearly defined components and interactions. The architecture was designed for modularity, allowing different parts (like the frontend UI, backend API, data capture extension, and processing worker) to be developed and potentially scaled somewhat independently.

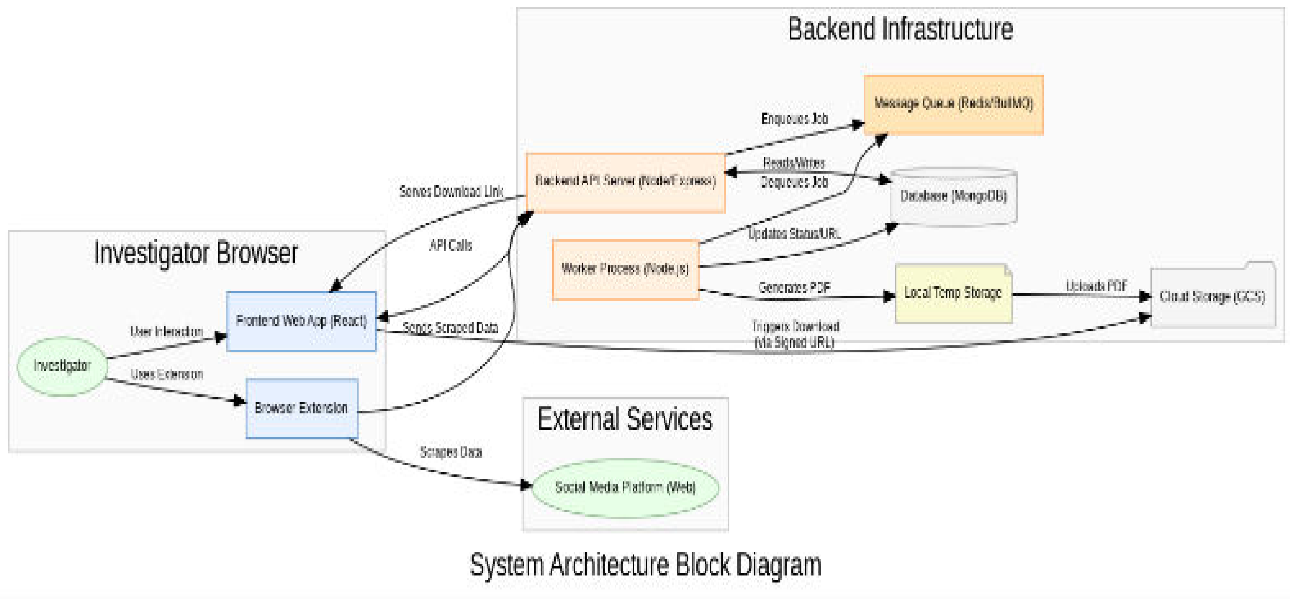


Fig. 1: High-Level System Architecture Block Diagram

The core components and their implementation approach are detailed below:

Frontend Web Application: We chose React, a popular JavaScript library, combined with the Vite build tool for its fast development experience and efficient production builds. This allowed us to create a responsive Single Page Application (SPA). Key interface elements included forms for authentication, a dashboard for initiating report requests (specifying platform and target identifier), and a dynamic list displaying report status with download capabilities. Communication with the backend relied on the Axios library, configured with interceptors to handle authentication tokens (JWT) automatically and manage session expirations gracefully. State management within components handled loading indicators, error messages, and the display of fetched report data. Polling was implemented within the report list component to provide near real-time status updates for ongoing jobs.

Backend API Server: Node.js served as the runtime, with the Express.js framework providing the structure for building RESTful API endpoints. Mongoose was used as the Object Data Mapper (ODM) to interact with a MongoDB database, defining schemas for User and Report collections. The User schema included username and a password field automatically hashed using a pre-save hook with the bcrypt library. The Report schema stored references to the user, platform details, target identifier, processing status (using an enum: 'Received', 'Generating', 'Completed', 'Failed'), error messages, the final PDF URL, and timestamps. Authentication endpoints (/api/auth/register, /api/auth/login) handled user credential management and JWT issuance/verification. Report endpoints (/api/report) managed incoming data submissions from the extension (including crucial username verification against the User collection), handled requests from the frontend to list reports (filtering by the authenticated user ID extracted from the JWT by protect middleware), and generated secure, time-limited download URLs for completed reports stored in Google Cloud Storage. A critical function of the API was to enqueue processing jobs into the BullMQ message queue upon successful data reception and verification, rather than processing synchronously.

Asynchronous Task Processing: We implemented a message queue using BullMQ, leveraging a Redis instance as the backend broker. This queue, named pdf-processing, received job messages from the API server. Each job contained the necessary information (like the MongoDB reportId, platform, the full content payload received from the extension, and target identifier) for the worker to process. A separate Node.js worker process was created to listen exclusively to this queue. This worker employed logic to handle concurrency and retries as configured in BullMQ. Its primary responsibility was to dequeue jobs, orchestrate PDF generation using the received data, handle the upload to cloud storage, and update the corresponding report's status and URL in the MongoDB database. This decoupling was essential for maintaining API responsiveness and enabling background processing.

Browser Extension: Developed following Chrome Extension Manifest V3 standards.
Content Script: Injected JavaScript code designed to run within the context of loaded social media pages. It used standard DOM manipulation methods (document.querySelector, querySelectorAll, innerText) to locate and extract textual data based on known (though potentially unstable) CSS selectors or element structures specific to each supported platform (Twitter/X, Instagram, Facebook). It communicated with the Service Worker via chrome.runtime.sendMessage.
Service Worker: The extension's background event handler. It managed the overall workflow initiated by the popup. It coordinated communication with the content script, requesting profile and post data sequentially. It utilized the chrome.tabs.captureVisibleTab API to capture a PNG screenshot of the currently visible tab content. After aggregating the data from the content script and the screenshot, it constructed the final JSON payload and used the fetch API to send it via POST request to the configured backend API endpoint, ensuring the examiner's username (received from the popup) was included in the payload for backend verification. It also relayed status messages back to the popup UI.
Popup: An HTML page with associated JavaScript providing the user interface upon clicking the extension icon. It included input fields for the mandatory examiner username and the configurable backend API URL (with persistence using chrome.storage.local). The main button triggered the generateFullReport message to the service worker. It also listened for status update messages from the service worker to provide feedback to the user.

PDF Generation and Storage: The backend worker process employed the pdfkit Node.js library for creating PDF documents dynamically. The code structured the PDF logically with a title page containing metadata, followed by sections detailing the extracted profile information. Screenshots received from the extension (as base64 data URLs) were temporarily saved to the worker's local filesystem as image files (e.g., PNG) before being embedded into the PDF

using pdfkit's image embedding capabilities. Extracted posts were formatted and added sequentially. Error handling was included to note scraping failures within the PDF. After successful generation, the worker used the official Google Cloud Storage Node.js client library to upload the locally generated PDF file to a designated GCS bucket. The worker required appropriate service account credentials (configured via the GOOGLE_APPLICATION_CREDENTIALS environment variable) to authenticate with GCS. Upon successful upload, the standard GCS URL (https://storage.googleapis.com/...) was stored in the corresponding Report document in the database. Download access was managed by the backend API generating short-lived signed URLs upon request from an authenticated user.
This methodology emphasized modular design, asynchronous processing for performance, leveraging the browser environment for data capture via the extension, and secure handling of user authentication and data storage.

VII. TOOLS AND TECHNIQUES USED

The realization of the Social Media Feed Parser system drew upon a carefully selected combination of contemporary web technologies, programming languages, libraries, and architectural principles. The choices were driven by the specific requirements of the project, including the need for a responsive user interface, scalable backend processing, secure authentication, interaction with the browser environment, document generation, and cloud integration.

Frontend Development: To build the user-facing application, we utilized React, a popular JavaScript framework chosen for its efficiency in creating dynamic, component-based interfaces and its extensive supporting ecosystem. The Vite build tool was employed to provide a fast development server environment and generate highly optimized code bundles for production deployment.
For handling communication with the backend API, the Axios library facilitated making asynchronous HTTP requests, offering helpful features like interceptors for seamless authentication header management. Client-side navigation within the single-page application, including the implementation of protected routes requiring authentication, was managed using React Router. UI state, such as tracking loading statuses, displaying error messages to the user, and managing the fetched report data, was handled effectively using React's built-in state management capabilities like useState and useContext.

Backend Development:
The core backend system was built upon the Node.js runtime environment, allowing us to leverage JavaScript for scalable server-side application development. We used the Express.js web framework for its minimalist and flexible approach to structuring the backend RESTful API, defining routes, and managing middleware functions. For data persistence, MongoDB was selected as the NoSQL database, primarily due to its flexibility in handling potentially varied report structures and its strong integration with the Node.js ecosystem. Interaction with MongoDB was streamlined using the Mongoose Object Data Mapper (ODM), which provides convenient methods for schema definition, data validation, and implementing middleware hooks, such as the one used for automatic password hashing. Robust user authentication was achieved using JSON Web Tokens (JWT) for stateless session management, while the bcrypt library ensured that user passwords were securely hashed before being stored in the database.
Asynchronous Task Processing:
To ensure the application remained responsive during potentially long-running operations, an asynchronous task processing architecture was implemented. We utilized the BullMQ message queue library, backed by a Redis instance, to effectively decouple tasks like PDF generation and cloud uploads from the main API request-response cycle. This approach enhances scalability and provides reliable background job execution.

Browser Extension:
The critical client-side data capture component was developed as a browser extension using standard WebExtension APIs (specifically following Chrome's Manifest V3 standards). These APIs provided the necessary capabilities for DOM manipulation via content scripts, background processing logic within a service worker, creating a user interface element (the popup), managing local extension storage, and facilitating secure messaging between extension components. The extension's internal logic and user interface elements were constructed using fundamental web technologies: JavaScript, HTML, and CSS.

Document Generation:
The programmatic creation of the final forensic reports was handled by the pdfkit library within the backend worker process. This powerful Node.js library allowed for detailed control over PDF document creation, including text formatting, precise layout management, and the embedding of images captured by the browser extension.

Cloud Infrastructure:
For scalable, durable, and secure storage of the generated PDF reports, essential for operational deployment, we integrated Google Cloud Storage (GCS). Secure authentication between the backend worker process and the GCS APIs was managed using a GCS Service Account key file. To enable secure downloads of the potentially sensitive reports, we leveraged GCS's URL Signing feature, which generates time-limited, unique URLs granting temporary access to private objects, thereby avoiding the need to make the reports publicly accessible.

General Tools:
Visual Studio Code served as the primary Integrated Development Environment (IDE), providing a comprehensive environment for coding, debugging, and project management. Sensitive configuration parameters, such as database connection strings, API keys, and cloud credentials, were managed securely outside the application's source code using environment variables, typically loaded via .env files during development.

VIII. RESULTS AND DISCUSSION

The implementation phase culminated in a functional system capable of executing the designed workflow for automated social media evidence collection and documentation. Testing confirmed that the core components interacted as intended, allowing authenticated investigators to initiate report requests through the web application, trigger data capture via the browser extension on supported platforms (Twitter/X, Facebook, Instagram), have the requests processed asynchronously by the backend, and ultimately download structured PDF reports containing the collected information.

Evaluation of Functional Achievements:

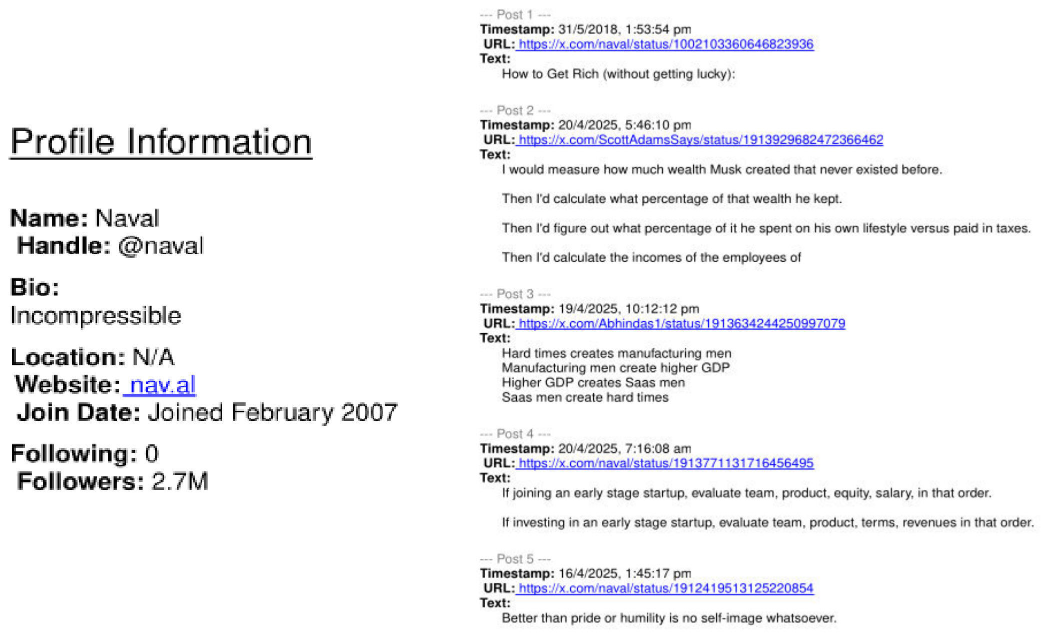
Successful Automation: The primary goal of automating the collection process was achieved. The system effectively replaces the manual tasks of navigating profiles, taking screenshots, and transcribing basic profile details and visible posts (with varying success based on platform stability). This demonstrably saves investigator time and effort.

Asynchronous Architecture Performance: The message queue and worker process architecture performed exceptionally well. Requests initiated via the web application received immediate confirmation, and the UI remained responsive regardless of the processing load on the backend worker. This validated the choice of an asynchronous design for handling potentially long-running tasks.

Structured Reporting: The use of pdfkit enabled the generation of consistent and organized PDF reports. These reports included essential metadata, logically separated sections for profile information and posts, and embedded screenshots, providing a standardized and more easily reviewable format compared to ad-hoc manual documentation.

User Authentication and Data Linkage: The JWT-based authentication secured the web application effectively. The implemented workflow, requiring the investigator to provide their username in the extension popup for backend verification before processing, successfully established a traceable link between the captured data and the authorized initiating user, addressing a crucial aspect of data provenance within the system's context.

Secure Storage and Access: Employing Google Cloud Storage for report persistence and utilizing signed URLs for download access provided a secure and scalable solution for managing the final evidence artifacts.



Profile Information

Name: Naval
Handle: @naval

Bio:
Incompressible

Location: N/A
Website: nav.al
Join Date: Joined February 2007

Following: 0
Followers: 2.7M

--- Post 1 ---
Timestamp: 31/5/2018, 1:53:54 pm
URL: https://x.com/naval/status/1002103360646823936
Text:
How to Get Rich (without getting lucky):

--- Post 2 ---
Timestamp: 20/4/2025, 5:46:10 pm
URL: https://x.com/ScottAdamsSays/status/1913929682472366462
Text:
I would measure how much wealth Musk created that never existed before.

Then I'd calculate what percentage of that wealth he kept.

Then I'd figure out what percentage of it he spent on his own lifestyle versus paid in taxes.

Then I'd calculate the incomes of the employees of

--- Post 3 ---
Timestamp: 19/4/2025, 10:12:12 pm
URL: https://x.com/Abhindas1/status/1913634244250997079
Text:
Hard times creates manufacturing men
Manufacturing men create higher GDP
Higher GDP creates Saas men
Saas men create hard times

--- Post 4 ---
Timestamp: 20/4/2025, 7:16:08 am
URL: https://x.com/naval/status/1913771131716456495
Text:
If joining an early stage startup, evaluate team, product, equity, salary, in that order.

If investing in an early stage startup, evaluate team, product, terms, revenues in that order.

--- Post 5 ---
Timestamp: 16/4/2025, 1:45:17 pm
URL: https://x.com/naval/status/1912419513125220854
Text:
Better than pride or humility is no self-image whatsoever.

IX. DISCUSSION OF CHALLENGES AND OPERATIONAL REALITIES

While functionally successful, the system's reliance on browser-extension-based web scraping surfaced significant operational challenges and limitations that warrant careful consideration:

The Achilles' Heel: Scraping Fragility: By far the most critical challenge is the inherent instability of the data extraction logic within the browser extension's content script. Social media platforms continuously iterate on their frontend code, layouts, and class names. Any such change, even minor ones, has a high probability of breaking the specific DOM selectors relied upon by the scraper, rendering data collection inaccurate or impossible until the extension is updated. This necessitates a reactive and potentially resource-intensive maintenance cycle, constantly adapting the extension to keep pace with platform evolution. During our testing, Instagram and Facebook selectors proved particularly volatile.

Handling Web Dynamism: Modern social media interfaces heavily utilize JavaScript for dynamic content loading (e.g., infinite scrolling feeds, lazy-loaded images, comments appearing on demand). The current implementation primarily captures the static or initially loaded content well. Reliably capturing data that requires user interaction (like scrolling) or complex JavaScript execution from within the content script requires more sophisticated techniques (e.g., programmatic scrolling, Mutation Observers, waiting for specific network events) which add complexity and can further reduce reliability.

Platform Defenses and Ethical Use: Although leveraging the user's session bypasses direct login issues, the automated nature of the extension's interactions could still be detected by platform anti-bot systems if performed too rapidly or predictably. This might lead to temporary restrictions, CAPTCHA challenges, or account scrutiny. Use of the tool must be judicious, respecting platform resources and, most importantly, adhering strictly to legal authorizations and ethical guidelines for investigations. The tool provides a *means* of collection, not the *authority* to collect.

Screenshot Limitations: Capturing truly comprehensive visual evidence via automated screenshots is challenging. The current captureVisibleTab provides only the viewport. Full-page, scrolling screenshots or capturing specific dynamic elements accurately across various screen sizes and states often require more advanced browser automation capabilities or client-side libraries (like html2canvas), which can impact performance and complexity.

Workflow Dependencies: The system requires conscious user action: the investigator must navigate to the correct page, ensure the extension is active, invoke its functionality, and provide their username. This introduces manual steps and potential inconsistencies compared to a hypothetical, fully autonomous server-side ideal (which faces its own significant hurdles).

Performance Notes: The system demonstrated good responsiveness in user-facing interactions due to the backend's asynchronous design. The bottleneck lies in the worker process, where PDF generation and cloud uploads for complex reports can take noticeable time (seconds to potentially minutes). The browser extension itself remained relatively lightweight during testing, though very complex pages could theoretically cause minor browser slowdown during intensive DOM parsing or screenshotting.
In essence, the project successfully demonstrated the *technical feasibility* of the proposed architecture but also underscored the significant *operational challenges* associated with maintaining web scraping tools in the dynamic social media environment. The automation offers tangible benefits, but practical deployment requires acknowledging and planning for the continuous maintenance effort involved.

X. CONCLUSION

In conclusion, this paper has thoroughly documented the design, implementation, and operational characteristics of an automated system developed for the parsing and documentation of social media feeds, tailored specifically to enhance digital forensic investigations. The system successfully integrates a secure web application for investigator control, a scalable backend employing asynchronous processing through message queues, and an innovative browser extension that performs data capture directly within the user's authenticated browser context. Through this architecture, the system automates the collection of profile information, visible posts, and contextual screenshots from major platforms like Facebook, Twitter/X, and Instagram, subsequently compiling this evidence into structured, standardized PDF reports stored securely in cloud infrastructure.

XI. FUTURE DIRECTIONS

Based on the insights gained during development and testing, several promising avenues exist for future enhancements to improve the system's robustness, expand its capabilities, and increase its overall value within the digital forensic toolkit:

Improving Scraping Robustness: The most critical area for future work is mitigating the fragility of the browser extension's scraping logic. Research and implementation could focus on:
Visual Selectors: Employing techniques that identify elements based on visual characteristics (e.g., relative position, appearance) rather than relying solely on potentially volatile CSS selectors or DOM paths.
AI/ML Element Identification: Training lightweight machine learning models to recognize common social media components (posts, profile headers, comment sections) even when underlying code structures change moderately.
Heuristic Approaches: Developing more flexible parsing logic that uses multiple potential selectors or structural patterns to find target data, increasing resilience to minor changes.

Expanding Platform Support and Methods: Systematically assess and add support for other social media or online platforms frequently encountered in investigations. For platforms notoriously difficult to scrape via browser extensions (e.g., end-to-end encrypted messaging applications like WhatsApp, Signal, Telegram), investigate alternative integration strategies, such as interoperability with specialized mobile forensic extraction tools or analysis of desktop application data artifacts, where legally permissible and technically feasible.

Advanced Reporting and Integrity: Enhance the capabilities of the PDF reporting module:
Customization: Allow investigators to configure report templates or select specific data fields for inclusion.
Data Hashing: Automatically compute and embed cryptographic hash values (e.g., SHA-256) for individual extracted text snippets and captured screenshots within the PDF metadata or content, providing stronger evidence integrity verification.
Interactivity/Analysis: Explore embedding basic interactive elements or preliminary data visualizations (e.g., timelines, simple charts) within the PDF reports.

Sophisticated Dynamic Content Handling: Implement more advanced techniques within the extension's content scripts to reliably capture dynamically loaded content. This could involve controlled programmatic scrolling combined with mutation observers, intelligent waiting strategies for specific network events or element appearances, or potentially intercepting relevant frontend API calls made by the platform itself (if technically possible and compliant with terms).

Integration of Natural Language Processing (NLP): As suggested by related research [II], integrate NLP functionalities into the backend worker process. This could enable automated analysis during report generation, such as performing sentiment analysis on posts, identifying named entities (people, locations, organizations), extracting relevant keywords based on case context, or applying topic modeling to large volumes of captured text, thereby providing initial analytical insights to the investigator.

Enhanced Provenance and Audit Trails: Bolster the system's ability to provide a detailed and verifiable record of the collection process. Beyond basic timestamps and user attribution, explore logging more granular actions performed by the extension and worker (e.g., specific selectors used, elements captured). Investigate the feasibility of utilizing blockchain technology [V] to create immutable, timestamped records of collection metadata and actions, potentially strengthening admissibility arguments.

System Monitoring and Management: Develop comprehensive monitoring dashboards for administrators or lead investigators. These dashboards could track job queue statistics, worker performance metrics, identify patterns of scraping failures across different platforms (indicating a need for extension updates), and provide overall system health monitoring to facilitate proactive maintenance.

Usability Enhancements: Conduct formal usability studies involving target end-users (digital forensic investigators) to gather feedback on both the web application interface and the browser extension workflow. Use this feedback to iteratively refine the user experience, focusing on clarity, intuitiveness, and efficiency within the context of typical investigative procedures.

Native Application Feasibility Study: Undertake a thorough investigation into the necessity and feasibility of developing dedicated native applications (for Windows, macOS, or potentially Android) as companions or alternatives to the browser-based approach. This would be justified if specific platforms or data types critical to investigations are demonstrably inaccessible via standard web browsers and extensions.

XII. ACKNOWLEDGMENTS

We wish to express our sincere and profound gratitude to Reva University. The institution's provision of essential resources, access to necessary infrastructure, and the fostering of a supportive academic climate were absolutely instrumental in the successful conception, development, and completion of this research project.
We are deeply indebted to our dedicated mentors, knowledgeable professors, and insightful research advisors; their expert guidance, constructive criticism, and unwavering encouragement provided invaluable direction throughout the project's duration. Furthermore, this work stands on the shoulders of the vibrant global open-source software community. We gratefully acknowledge the critical role played by the numerous freely available tools, libraries, and frameworks – including the Node.js runtime, the React library, the Express framework, MongoDB, Redis, BullMQ, pdfkit, the Google Cloud Client Libraries, and the foundational WebExtension APIs – whose collective power and accessibility made the implementation of this complex system feasible. The spirit of collaboration inherent in open source significantly accelerated our progress and enriched our development experience.

REFERENCES

[1] J. D. Wang and P. Luo, "A Multi-Layer Semantic Approach for Digital Forensics Automation for Online Social Networks," *Sensors*, vol. 22, no. 3, p. 964, Feb. 2022. doi: 10.3390/s22030964.
[2] S. Prakash, Z. Shahbazi, and Y.-C. Byun, "NLP-Based Digital Forensic Analysis for Online Social Network Based on System Security," *Int. J. Environ. Res. Public Health*, vol. 19, no. 12, p. 7421, June 2022. doi: 10.3390/ijerph19127421.
[3] N. Shashidhar, "A Comprehensive Survey on Artifact Recovery from Social Media Platforms: Approaches and Future Research Directions," *Information*, vol. 14, no. 12, p. 668, Dec. 2023. doi: 10.3390/info14120668
[4] K. Maitra, "Digital Forensic Analysis of Social Media Platforms for Enhanced Investigation and Evidence Collection," *Int. J. Innovation and Applied Studies*, vol. 43, no. 4, pp. 1194-1207, Aug. 2024. [Online]. Available: