CREDIT-X

A Major Project

Submitted in partial fulfilment of the requirement for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

Jagriti Sharma (Registration No.:11021210010)
Tanmay Pathak (Registration No.:11021210061)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM UNIVERSITY DELHI-NCR

Plot No.39, Rajiv Gandhi Education City, Sonepat, Haryana – 131029

MAY 2025

CREDIT-X

A Major Project

Submitted in partial fulfilment of the requirement for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

Jagriti Sharma (Registration No.:11021210010)
Tanmay Pathak (Registration No.:11021210061)

Under Supervision of

Dr. Mohd Kaleem

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM UNIVERSITY DELHI-NCR

Plot No.39, Rajiv Gandhi Education City, Sonepat, Haryana – 131029

MAY 2025

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project entitled "**CREDIT-X**" in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering and submitted in the Department of Computer Science & Engineering of SRM University, Delhi-NCR, Sonepat, Haryana, (India) is an authentic record of my own work carried out under the supervision of **Dr. Mohd Kaleem,** as major project in VIII semester during the academic year 2024-25 The matter presented in this project has not been submitted for the award of any other degree of this or any other Institute / University.

(Signature of the candidate)
Jagriti Sharma
Reg No.:11021210010

(Signature of the candidate)
Tanmay Pathak
Reg No.:11021210061

CERTIFICATE

This is to certify that the project entitled "**CREDIT-X**" submitted by **Jagriti Sharma (Reg. No. 11021210010) and Tanmay Pathak (Reg. No. 11021210061)**, to the Department of Computer Science & Engineering of SRM University Delhi-NCR, Sonepat, Haryana, (India) in partial fulfilment of the requirements for the award of the degree of Bachelor in Technology in Computer Science & Engineering **(specialisation in Data Science and Artificial Intelligence)** under the Faculty of Engineering and Technology is an authentic record of the work carried out by her/him under my supervision. In my opinion, this work fulfils the requirement for which it has been submitted.

This project has not been submitted to any other University or Institution for any other degree and is submitted as major project **(21CS4114)** in VIII semester during the academic year 2024-25

Dr. Mohd Kaleem
Supervisor

Mr. Dev S. Verma
Project Coordinator

Dr. M. Mohan
Head of Department

External Reviewer

ACKNOWLEDGEMENT

We would like to express our gratitude and sincere thanks to Dr. Mohd Kaleem, Assistant Professor, Faculty of Engineering and Technology, for her guidance, support and encouragement. We'd like to extend our gratitude to Dr. M. Mohan, Head of Department (CSE), for providing us with this wonderful opportunity and facilities for working on the project. Secondly, we would like to thank our parents for giving us their support and motivation throughout the semester. Lastly, we sincerely acknowledge the support and valuable input rendered to me by my teachers and my friends during my project.

(Signature of the candidate)
Jagriti Sharma
Reg No.:11021210010

(Signature of the candidate)
Tanmay Pathak
Reg No.:11021210061

ABSTRACT

This project was conducted as part of an internship on the News Automation Team for Credit-X, an AI-powered Credit Risk Knowledge Assistant that delivers company-specific financial insights to credit analysts , traders, and stock market enthusiasts, developed by Galytix,. Credit-X operates in a production environment covering over 1,100 companies and provides users with timely access to credit-related information such as quarterly and annual reports, recent press releases, and stock-related news. The objective of this project was to automate the ingestion of news data from both Google News and official company websites to accelerate information flow and reduce manual effort. Two custom-built web scrapers were developed to achieve this goal. The first is a Google News scraper that extracts links based on search queries and filters out over 180 mainstream domains to surface niche and alternative news sources. The second is an official website scraper that navigates directly to verified company domains to collect press releases, blog posts, and news updates by detecting and parsing relevant HTML elements. Both scrapers utilize Selenium with undetected-chromedriver, Pandas, and urllib.parse for efficient multi-page scraping, CAPTCHA handling, and structured Excel output generation. The official website scraper further incorporates the htmldate library to extract publication dates, perform date-based filtering, manage pagination, avoid duplicate content, and log all scraping operations. The automation process led to a 10X increase in data processing efficiency and scalability. This project significantly contributed to the enhancement of Credit-X's data pipeline, enabling real-time news updates that are critical for accurate credit risk assessments. The solution aligns with United Nations Sustainable Development Goals (SDGs) 8, 9, and 16 by promoting economic transparency, innovation in financial systems, and equitable access to risk-related insights. Future development plans include the integration of a graphical user interface (GUI), automatic proxy rotation, real-time dashboards, and article summarization for expanded usability.

Table Of Content

List of Tables

List of Figures

Chapter 1
INTRODUCTION

1.1 Background

In the rapidly evolving world of finance, information is power. The ability to access, interpret, and act upon financial news in real time is a decisive factor in determining the success or failure of investment strategies. Whether it's credit risk assessment, market sentiment analysis, or portfolio management, timely information forms the backbone of informed financial decision-making. However, the traditional financial research workflow continues to struggle with inefficiencies in information acquisition. This is the context in which Credit-X, an AI-powered Credit Risk Knowledge Assistant, operates — seeking to eliminate friction in the data pipeline and empower users with up-to-date and context-rich financial intelligence.

Credit-X is designed for institutional stakeholders, such as credit analysts, portfolio managers, risk officers, and research professionals who operate in high-stakes environments where latency in information can result in financial loss or missed opportunity. The platform leverages machine learning, natural language processing (NLP), and data aggregation technologies to deliver real-time insights for over 1,100 companies across sectors. These insights include earnings reports, regulatory filings, macroeconomic signals, social sentiment, and — crucially — news coverage. Among these, company-specific news is particularly important, as it often serves as the first public signal of emerging risks, opportunities, or corporate events.

Yet, even with the most advanced AI capabilities, the ingestion of news — particularly from non-mainstream or official sources — remained a bottleneck. Manual methods of collecting news articles, such as searching Google News or visiting company press release pages, consumed up to 80% of analysts' time. This manual process was not only time-consuming but also error-prone, inconsistent, and unscalable. In addition, such methods risked overlooking important news items that didn't appear in mainstream publications or were buried deep in company websites.

Many influential news events — such as management changes, legal disclosures, cybersecurity incidents, or regional operational issues — are first reported in press releases, investor blogs, or local media before making their way to larger financial platforms. Relying solely on mainstream aggregators or commercial news APIs often leads to blind spots. These early signals are essential for proactive credit risk monitoring but were frequently missed in traditional workflows. The challenge, therefore, was not only to speed up ingestion but to widen the scope of sources without sacrificing relevance or quality.

Recognizing this operational gap, the internship project focused on developing intelligent automation tools to improve the speed, quality, and scope of news ingestion for Credit-X. By using Python and open-source libraries, two dedicated scrapers were

built — one targeting Google News for aggregated content, and the other extracting press releases and updates directly from company websites. These tools were designed to simulate the analyst's manual workflow in a structured and repeatable way while improving efficiency by an order of magnitude.

The Google News scraper employed Selenium to mimic user behavior on the search engine, allowing for the use of custom time filters, keyword input, pagination handling, and link extraction. The Official Website scraper, on the other hand, required a more nuanced approach due to variations in site structure. It included logic to identify the "News," "Media," or "Press" sections of each company's website and extract links along with publication dates using libraries such as htmldate.

Both scrapers were complemented by downstream processes that cleaned and structured the data. Duplicate removal, domain-level filtering (to exclude known mainstream sources already handled elsewhere in Credit-X), and Excel-based exports formed the core output of the tools. These automated pipelines offered not only speed — reducing hours of work to minutes — but also improved consistency and scalability.

The implications of this shift were immediate and far-reaching. Analysts were now equipped with up-to-date news covering broader domains, freeing their time for analytical tasks rather than data gathering. This allowed them to surface insights faster, monitor risks more proactively, and align their efforts with the high-velocity demands of modern financial markets.

Moreover, the work done during this internship resonates with the broader transformation in FinTech — where automation, alternative data, and AI integration are redefining how financial services are delivered. In that context, automating news ingestion is not merely a convenience; it is a strategic imperative. This project was therefore more than a technical exercise — it was a foundational step toward making Credit-X a smarter, faster, and more adaptive credit intelligence platform.

1.2 Purpose of the Study

The primary purpose of this internship project was to design and implement an automated system to ingest financial news relevant to companies tracked by Credit-X. In the contemporary financial landscape, where analysts are expected to deliver timely insights on credit risk, corporate health, and investment opportunities, the underlying data pipeline must be both efficient and reliable. However, despite advances in AI and data science, one of the least optimized parts of this pipeline remains the acquisition of real-time news articles from diverse sources. This project directly addressed that inefficiency through the development of scalable, Python-based web scraping tools capable of extracting structured news content from both Google News and official company websites.

Credit-X, as an AI-powered Credit Risk Knowledge Assistant, thrives on data that is current, high-quality, and granular. While it already incorporated structured financial

datasets , such as balance sheets, earnings calls, and market data , there was a noticeable gap in the ingestion of unstructured yet valuable news content. Analysts had to manually search for, extract, and record company news into shared spreadsheets. This approach was inherently flawed for several reasons: it was inconsistent across analysts, lacked standardization, and was not scalable to the platform's ambition of covering over 1,100 companies. Moreover, the manual method often caused delays, missed early news signals, and introduced human errors such as duplication or misclassification.

The purpose of this project was thus multi-dimensional, spanning both operational improvement and strategic enablement. At its core, the aim was to shift from a reactive and manual news-gathering process to a proactive, automated workflow that could serve analysts with reliable content on a daily basis. By automating the collection and formatting of news articles, the tools developed during the internship had the potential to multiply analyst productivity, improve content freshness, and establish a foundation for further automation across the data pipeline.

The project's specific objectives were as follows:

1. **Reduce Manual Workload:** The first and most immediate goal was to eliminate the manual effort analysts spent copying and pasting URLs from Google or scanning through company websites. This repetitive and tedious task consumed a large portion of their time, limiting their ability to focus on high-value work like interpreting signals, modeling risk scenarios, or drafting reports. Automating this task freed analysts from the burden of data collection and allowed them to concentrate on their domain expertise.

2. **Enhance Coverage and Throughput:** With manual workflows, an analyst could process an average of 150 articles per day. The tools developed in this project aimed to push that number beyond 1,500 per analyst per day by introducing parallel scraping capabilities and batch processing. This meant that more companies could be covered more frequently, enabling Credit-X to maintain a higher cadence of updates.

3. **Ensure Relevance and Recency:** One challenge with news aggregation is sifting through large volumes of irrelevant or outdated content. The scraping tools were designed with built-in logic to filter by publication date, prioritize official domains, and exclude noisy sources. For instance, the Google News scraper allowed analysts to specify date ranges directly through the UI, while the official website scraper used libraries like htmldate to extract accurate timestamps from raw HTML.

4. **Support Dual-Source Ingestion:** The decision to build two scrapers : one for Google News and one for official company websites — was deliberate. Google News acts as a reliable aggregator of third-party content, while official websites often publish press releases or statements that don't appear elsewhere. By

combining these two streams, the tools ensured comprehensive coverage of both mainstream and proprietary information.

5. **Enable Reusability and Scalability:** A major purpose of the project was to build tools that were not only effective but also reusable. This meant structuring the scripts in a modular fashion, allowing configuration through external files (like official.xlsx), and exporting output in analyst-friendly formats (Excel/CSV). The idea was that any team member, even those without programming knowledge, could run the tools with minimal effort.

6. **Evaluate Operational Impact:** Beyond development, the project also aimed to measure the effectiveness of the tools in real-world use. Metrics such as reduction in manual work hours, increase in news coverage per day, and improvements in data quality were tracked and reported. This feedback loop helped refine the tools and informed future decisions about tool deployment, user training, and platform integration.

7. **Lay Groundwork for Integration:** While integration with Credit-X's production systems was beyond the scope of this internship (due to access restrictions), the tools were designed with future extensibility in mind. Standardized output formats, modular code design, and optional components like a GUI (built using Flask) positioned the project for eventual integration into APIs, databases, and dashboards.

In summary, the purpose of this project was not simply to build a script that gathers articles. Rather, it was to solve a real and pressing business problem , the inefficiency and inconsistency in news ingestion , through a combination of automation, thoughtful design, and strategic alignment. The project delivered value by enhancing data flow, empowering analysts, and setting a strong foundation for the continued evolution of Credit-X's data infrastructure.

1.3 Scope of the Study

The scope of this internship project was strategically designed to maximize impact while adhering to realistic constraints in terms of time, infrastructure, and access to proprietary systems. Spanning over a six-month period, the project was anchored in a real-world need: to automate the ingestion of financial news articles into the Credit-X platform, thereby enhancing the productivity and effectiveness of credit analysts. Although the vision for automation was ambitious, the project's execution was grounded in a clearly defined scope to ensure tangible and measurable outcomes within the internship period.

The project was limited to automating the extraction of relevant financial news from two key sources: **Google News**, which aggregates articles from thousands of media outlets, and **official company websites**, which host press releases, regulatory filings, and corporate updates. This dual-source strategy allowed for the capture of both

secondary news (from third-party aggregators) and primary information (from companies themselves), offering a comprehensive picture of a company's media footprint.

1. Functional Components Covered

The project scope included the following functional components:

1. **Development of Google News Scraper**: The Google News scraper was built using Python and Selenium. It was designed to simulate a user typing a company name into the Google News search bar, selecting custom date ranges, and navigating through paginated results. For each search result, the scraper extracted the article hyperlink, source domain, then filtered and exported the data in Excel format. This component proved especially useful for uncovering breaking stories and third-party commentary that might not appear on a company's official website.

2. **Development of Official Website Scraper**: Credit-X tracks over 1,100 companies globally. Each of these companies has a unique official website, with different layouts, URL structures, and content formats. The second scraper was built to access these domains—provided via an input file (official.xlsx)—and look for press release or investor news sections. Once located, the scraper extracted the links to individual articles, resolved relative URLs into absolute paths, and attempted to determine the publication date using the htmldate library. This tool was critical in surfacing primary sources and internal company statements.

3. **Date Filtering and Domain Logic**: Filtering by publication date was essential to ensure timeliness. The Google News scraper incorporated a custom filter that allowed users to specify a range (e.g., 05-13-2025 to 05-15-2025), ensuring only recent news was fetched. For the official websites, where dates were not always structured in HTML, the htmldate package parsed the article body to estimate dates with high accuracy. Additionally, a manually created blocklist was introduced to exclude unreliable, irrelevant, or low-quality domains (e.g., spam blogs or entertainment sites). This ensured that the dataset maintained a high standard of quality.

4. **Reusable Excel Outputs**: Both scrapers were designed to generate structured, tabular output files with predefined columns: company_id, article URL, publication date. These outputs could be imported into spreadsheets or data tools with no additional formatting required. Daily outputs were encouraged, which aligned with the existing analyst workflow and improved data freshness across the team.

5. **Support for Batch Processing and Scalability**: The tools supported batch operations, meaning users could pass in a list of company_ids and the script

would process all relevant data in sequence. This scalability was crucial for handling large volumes of companies—up to 200 per hour—without modifying the core logic of the scraper. The modular codebase also allowed easy enhancements, such as handling pagination or adding support for more date formats.

6. **Modular and Configurable Architecture**: Instead of hardcoding specific companies, domains, or URLs, both scrapers read inputs from external files (official.xlsx, company list .csvs, etc.). This design choice made the tools configurable and reusable, allowing any team member to update input files and re-run the scraper with minimal effort or training.

2. Out-of-Scope Components

While the project achieved considerable depth within its core focus, several components were deliberately kept out of scope to maintain feasibility:

- **Integration with Production Systems**: Direct integration of the scrapers into the production-grade Credit-X backend or databases was not attempted due to lack of access and security restrictions. Instead, the focus remained on building robust stand-alone tools that could be integrated later by the engineering team.
- **Use of Paid APIs and Proxy Services**: Tools like Google Search APIs or paid proxy rotation services were not used. The project relied solely on open-source libraries and local scraping techniques, which imposed some rate limitations but kept the solution cost-effective and accessible.
- **Cross-Team Deployment and DevOps Pipelines**: The deployment of these tools on cloud platforms or integration with CI/CD pipelines was not part of the scope, as it would require collaboration with DevOps and data engineering teams. Instead, the tools were run manually or on scheduled local sessions.
- **Natural Language Processing (NLP) or Article Summarization**: Although planned for future phases, the project did not incorporate automatic article classification, sentiment analysis, or summarization. The current focus was strictly on ingestion and formatting.
- **UI/UX Enhancements Beyond Prototyping**: A basic Flask-based user interface was prototyped for demonstration purposes, but full-fledged frontend integration was not pursued during this phase.

3. Strategic Scope Alignment

The limited scope was not a limitation but a strategic choice. By focusing narrowly on two high-value ingestion channels and prioritizing quality over complexity, the project delivered tangible benefits. These included tripling the article throughput per analyst, standardizing data collection, and dramatically reducing time spent on manual work.

Additionally, the modular architecture ensured that these tools could evolve. They now serve as a strong foundation for future features, including integration with alerting systems, automated tagging, and AI-powered summarization — all of which will require the solid, structured data pipeline that this project established.

1.4 Significance of the Study

The automation of news ingestion undertaken in this internship project holds substantial significance for both the operational efficiency of the Credit-X platform and the broader context of financial data processing. In an era where real-time decision-making is critical, the timeliness, accuracy, and comprehensiveness of input data can directly influence the quality of credit analysis, investment recommendations, and risk assessments. This project addresses a fundamental gap in that value chain — the initial acquisition of reliable, structured news content.

1. Industry Relevance

The financial services industry is undergoing a major transformation driven by the convergence of artificial intelligence, automation, and big data. Traditional methods of research — reliant on human effort and manual workflows — are becoming unsustainable in the face of growing data volumes and compressed decision windows. By implementing news scraping tools that automate the collection of company-specific financial news, this project mirrors the direction in which modern financial institutions are heading: faster, more intelligent, and scalable operations powered by technology.

Moreover, regulatory frameworks such as Basel III and IFRS 9 place increasing emphasis on the use of timely, forward-looking information for credit risk assessment. Automating the ingestion of breaking news, regulatory filings, and press releases enhances an organization's ability to remain compliant while also gaining competitive advantage through faster insight generation.

2. Operational Impact on Credit-X

Prior to automation, the manual news collection process was labor-intensive, inconsistent, and error-prone. Analysts had to individually copy-paste URLs from Google News or company websites, often leading to incomplete datasets and duplicated efforts. At peak efficiency, the team could gather only around 600 articles per day, with no consistent filtering by date or source credibility.

The automated tools built during the internship expanded this output by a factor of nearly 3× per analyst, allowing the team to gather over 1,500 articles daily for 150–200 companies. This not only improved data throughput but also brought standardization to the process. Articles were now uniformly filtered, structured, and exported, reducing variability across different team members' workflows.

Furthermore, the creation of a domain filtering mechanism — whereby unreliable or mainstream sources were excluded — allowed the ingestion pipeline to focus on

discovering early-stage, niche, or company-specific content often missed by mainstream aggregators. This level of curation significantly increased the relevance and uniqueness of the data being consumed by the platform.

3. Alignment with Broader FinTech Trends

This project also aligns with several emerging trends in the FinTech and RegTech domains:

- **Low-Code/No-Code Automation:** By developing tools that require minimal configuration and can be reused across analysts, the project supports democratization of technical processes.
- **Data as a Service (DaaS):** With structured news data becoming a valuable internal asset, the project lays the foundation for internal DaaS offerings within Credit-X.
- **Explainable AI (XAI):** The ability to trace back to original news sources supports transparency and interpretability in downstream AI models that use this content.
- **Alternative Data Use:** This project contributes to the growing use of alternative data sources (e.g., local press releases, niche blogs) to complement traditional financial datasets.

4. Academic and Technical Learning Value

From an academic perspective, this project offered rich, hands-on experience in applied computer science and financial technology. It combined several domains — web scraping, data engineering, information retrieval, software architecture, and user interface development — in a real-world setting with operational constraints.

Technically, we were exposed to practical challenges such as:

- Handling varying website structures and layouts across 1,100+ companies.
- Working with real-time filters, dynamic web content, and pagination in Selenium.
- Performing data cleaning, deduplication, and structured export using Python libraries.
- Understanding and circumventing limitations such as costly APIs for traffic analysis, without compromising functionality.

These challenges fostered creative problem-solving and reinforced best practices in modular code design, error handling, and scalability.

5. Future Utility and Scalability

Finally, the significance of the project extends into future developments. The scraping logic can be adapted for new sources or integrated into a GUI (Graphical User Interface) — a future-facing feature that was prototyped using Flask during the internship. As Credit-X scales its operations, the need for automated, extensible, and user-friendly tools will only grow. The foundational work done in this internship ensures that future teams can build upon a tested framework, rather than starting from scratch.

In summary, the automation of news ingestion is not merely a technical improvement — it is a strategic enabler for faster research, smarter risk analysis, and scalable knowledge delivery. The project not only addressed immediate operational pain points but also set the stage for continuous innovation within Credit-X's data ecosystem.

1.5 Technologies Used

The successful development and deployment of the news ingestion tools during this internship heavily relied on a focused and well-curated technology stack. The project emphasized simplicity, extensibility, and reliability, and all technological choices were aligned with these core principles. Since the automation was implemented as a prototype for potential production-grade solutions, the stack primarily consisted of open-source, lightweight tools that could be easily adapted or scaled in the future.

1.Programming Language

The entire backend for the project was implemented in **Python**, selected for its readability, extensive library ecosystem, and popularity in data processing and automation tasks. Python's concise syntax and broad community support made it an ideal choice for quick prototyping and iterative development. Furthermore, Python's capability to integrate with browser automation tools and its support for various data formats (such as Excel and CSV) significantly streamlined the data collection and export workflows.

2. Web Scraping Tools and Libraries

Web scraping formed the backbone of the project, with **Selenium** and **undetected_chromedriver (uc)** being the primary automation tools used for browser interaction:

- **Selenium**: This tool was used extensively in both the Google News and official website scrapers. It enabled dynamic interaction with websites, such as simulating user input, handling pagination, waiting for page elements to load, and clicking buttons. Selenium was particularly useful for Google News scraping, where dynamic loading and filters based on dates required user-like browser interactions.

- **Undetected_chromedriver (uc)**: This variant of the regular ChromeDriver helped bypass bot-detection mechanisms on platforms like Google News. Traditional headless browsers often get blocked or restricted; using undetected_chromedriver ensured more reliable access to the Google News results without getting rate-limited or flagged.

- **htmldate**: For official company websites, accurately identifying the publication date of an article was critical. The htmldate library was used to extract dates from unstructured HTML content when the publication date was not clearly marked using standard tags like <time> or <meta>. It provided a reliable fallback to parse the date using heuristic techniques.

- **tqdm**: The tqdm library was used to implement a progress bar while scraping multiple websites. It enhanced user experience and debugging by providing real-time visibility into the scraper's progress, particularly during long scraping sessions involving hundreds of companies.

3. Data Handling and Export

The project leveraged **pandas** for managing tabular data and performing transformations such as removing duplicates, combining metadata, and formatting date columns. The input (company base URLs) and output (scraped news data) were handled in Excel format using pandas.read_excel() and pandas.to_excel(), making it easy for analysts to open and inspect the files without requiring additional software.

- **Input File**: The official.xlsx file contained the list of companies along with their associated base URLs. It was manually maintained by the analyst team and served as the primary input for the official website scraper.

- **Output Format**: The final structured output included three key fields: company_id, link, and publication_date. This minimalist structure was intentionally chosen to reduce noise and ensure consistency across records.

4. Execution Environment

The code was executed locally on a personal system without the use of cloud platforms or remote execution environments. Development and testing were done using **Visual Studio Code (VS Code)**, which provided a robust and user-friendly interface for Python scripting. No version control system like Git was used during development, which, while acceptable for a prototype, is identified as a future area of improvement for better maintainability and collaboration.

There was no reliance on JavaScript, RSS feeds, or external APIs during the project. All interactions were based purely on HTML parsing and browser-based navigation using Selenium. Additionally, there was no use of automation tools like macros or browser plugins — all automation logic was coded explicitly in Python.

Chapter 2
COMPANY OVERVIEW

2.1 Brief Overview

Galytix (GX) is a UK-headquartered AI-driven fintech company established in 2015. With a primary focus on the banking, insurance, and healthcare sectors, Galytix is pioneering the use of domain-specific generative AI to drive better, faster, and more reliable decision-making. The company has built a reputation for developing enterprise-ready, non-hallucinating AI assistants that can interpret complex data sets with precision and align to institution-specific requirements through ontology layers.

Operating across global financial hubs—including offices in London, Zurich, Prague, and Gurugram (India)—Galytix employs more than 100 professionals from a mix of engineering, data science, product, and strategy backgrounds. It is considered a leader in IT system custom software development with specialization in GenAI, credit risk, data analysis, and intelligent automation.

Galytix's AI assistants are designed specifically for high-stakes use cases in the financial and insurance industries, targeting professionals such as individual investors, credit analysts, and claims managers. These assistants are not general-purpose bots but knowledge-centric AI agents trained on sector-specific data, delivering accuracy, relevance, and explainability.

What sets Galytix apart is its commitment to creating **"non-hallucinating" AI assistants**—a critical feature for financial professionals who rely on precision and verifiability. These assistants do not generate speculative or fabricated content but draw from curated data pipelines that ingest structured and unstructured sources, verify data integrity, and reflect client-specific rule sets.

The Gurugram office, located in Sector 44, acts as one of the primary engineering hubs. It houses cross-functional teams that focus on data engineering, AI training, product design, and delivery support. The office also facilitates a growing internship program aimed at fostering early-career talent and immersing them in real-world AI product development.

2.2 Major Products and Activities

Galytix’s core offering is a suite of AI-powered digital assistants tailored to three industries:

- **Finance (CreditX)**
- **Insurance**
- **Healthcare**

Each assistant is powered by a domain-specific knowledge base that integrates structured financial documents, real-time public data (like news), and internal

proprietary datasets. These products are deployed within global financial institutions and are aimed at augmenting human expertise rather than replacing it.

CreditX: The Flagship Product

CreditX is Galytix's primary offering in the finance vertical. It is a GenAI-based assistant designed to support credit analysts, portfolio managers, and institutional investors in tasks like:

- **Automating credit memos**: Extracting financial insights from PDFs, filings, and spreadsheets.
- **Connecting structured data with unstructured news**: Real-time scraping of Google News and company websites allows the assistant to monitor sentiment, events, and risks.
- **Accelerating credit workflows**: Analysts can go from data ingestion to insight generation in minutes instead of hours.

CreditX achieves this by combining multiple technical workflows such as:

- Natural Language Understanding (NLU)
- Semantic search
- Financial document parsing
- Context-aware data summarization
- Integration with internal enterprise software

Beyond CreditX, Galytix's other vertical-specific assistants help automate insurance claim validation and medical record synthesis in the healthcare domain. The methodology across all products includes extensive use of ontology-based rules, sector-relevant tagging, and the ability to customize each assistant per client.

Key Activities

- Development and continuous training of AI/ML models based on sector-specific ontologies.
- Automated ingestion of structured and unstructured data from internal and public sources.
- Seamless integration with client-side workflows including document management systems and decisioning tools.
- Deployment of AI assistants with a strong emphasis on trust, explainability, and reliability.

2.3 Organizational Structure and Team

The company operates a matrix-style hierarchy, with roles divided between Product, Engineering, Data Science, and Business Strategy. In the Gurugram branch, where this internship was conducted, the focus was largely on engineering, data automation, and AI integration.

Internship work was conducted under the Credit-X team, specifically within the News Automation sub-team of the Data Engineering function.

Reporting Structure:

- Raj Abrol – Chief Executive Officer
- Amiya Sinha – Senior Product Manager, Data Engineering
- Nitika Malhotra – Product Manager
- Akanksha Chawla – Data Factory Engineer (Direct Supervisor)
- Tanmay Pathak and Jagriti Sharma – Data Analyst Interns, Ingestion Team

The Ingestion Team (8 members total) was responsible for scraping and collecting news data from external sources like Google News and official company websites. This news was then made accessible through the Credit-X chatbot for financial analysis.

Another sub-team, the Transformation Team, was tasked with processing, tagging, and verifying the scraped content to ensure quality before feeding it into the system.

2.4 Work Environment and Tools

Galytix maintains a modern, productivity-focused work culture. The Gurugram office follows a hybrid working model, balancing in-office collaboration with remote flexibility. Interns had fixed working hours but were given autonomy in managing deliverables. Regular sync-ups ensured alignment across teams, and collaboration tools supported asynchronous work.

Development Environment

- **Languages**: Python (primary), no use of JavaScript or Bash scripting.
- **IDE**: Visual Studio Code (VS Code).
- **Browser Automation**: Selenium and undetected_chromedriver.
- **Data Handling**: pandas, datetime, tqdm, and htmldate.
- **Deployment**: Local machine-based execution; no cloud or Docker environments used.
- **File Formats**: Input in Excel format (official.xlsx), output structured with columns company_id, link, and publication_date.

- **Version Control**: GitLab was available for version control, though not used extensively during the internship.

The tools used were sufficient for building scalable scraping pipelines and allowed for experimentation and debugging in a controlled local environment. Documentation was maintained through internal Confluence pages, and requirements were managed using spreadsheets and weekly sprint plans.

2.5 Internship Objectives and Alignment with Company Vision

The core objective of the internship was to contribute to Galytix's long-term vision of automating financial intelligence pipelines. Specifically, the task of automating news ingestion aligned directly with CreditX's requirement to deliver real-time, reliable, and contextually relevant updates to its users.

As an intern, I was involved in:

- Designing and testing scraping scripts for dynamic websites using Selenium.
- Parsing metadata and timestamps to ensure chronological integrity.
- Normalizing URLs and handling edge cases for pagination and news linking.
- Ensuring that scraped data met the structural expectations for ingestion into CreditX.

By improving the accuracy and frequency of company-specific news ingestion, the internship work helped enhance the AI assistant's ability to factor in qualitative events such as market sentiment, leadership changes, lawsuits, and product launches—key drivers in credit risk analysis.

The project served as a live use case of how unstructured data can be transformed into decision-support assets for financial professionals, thereby embodying Galytix's core mission of **"fueling better decision-making through GenAI."**

2.6 Challenges and Opportunities within the Organization

1. Challenges Observed:

- **Dynamic Web Pages**: Many company websites used JavaScript-heavy templates, making scraping unreliable without headless browsing.
- **Non-uniform Date Formats**: Extracting accurate publication dates proved difficult due to varying metadata tags and timestamp styles.
- **Scalability Constraints**: Running scrapers locally imposed limits on how many companies could be processed efficiently.
- **No Version Control on Local Dev**: Lack of Git integration for intern code led to risks of loss and hindered collaboration.

2. Opportunities Identified:

- **Standardizing News Templates**: Developing a schema or template system for different domains could simplify future transformations.
- **Use of Cloud-Based Infrastructure**: Cloud functions or containers could enable scalable and continuous scraping operations.
- **Integration of Monitoring Dashboards**: A dashboard to visualize scraping success rates, failures, and article counts could greatly aid QA.
- **Pipeline Modularity**: Refactoring scripts into reusable components would allow for easier onboarding of new sources or clients.

Despite these challenges, the internship provided rich exposure to enterprise-level data workflows, and Galytix's openness to improvement created a conducive learning environment.

Chapter 3
METHODOLOGY

This chapter presents the comprehensive methodology adopted for the development of two web scraping modules—the **Google News Scraper** and the **Official Website Scraper**—as part of Galytix's data ingestion pipeline for the **Credit-X** platform. These tools were engineered to automate the process of collecting relevant and recent financial news articles for a dynamic list of companies, forming a vital input to the Credit-X chatbot's knowledge base. The methodology includes detailed technical design, tools and libraries used, development environment, rationale for architectural decisions, execution strategies, and validation techniques. All development was carried out by **Tanmay Pathak** and **Jagriti Sharma**, under the guidance of **Nitika Malhotra (Product Manager)** and **Akanksha Chawla (Data Factory Engineer)**.

3.1 Tools and Technologies Used

Key Libraries and Utilities

Library	Purpose
Selenium	Web automation to simulate browser behavior and interact with dynamic JavaScript-heavy content (e.g., Google News, corporate websites).
undetected-chromedriver (uc)	Bypasses bot detection mechanisms on Google, allowing longer uninterrupted scraping sessions.
pandas	Data transformation and Excel file I/O. Also used to manage company lists and output formats.
htmldate	Extracts publication dates directly from the HTML of a given article, improving accuracy of time-based filtering.
tqdm	Provides real-time progress bars during execution, helpful during long scraping sessions.
urllib.parse (urljoin, urlparse)	Used to normalize and resolve relative URLs and domain filtering.

(Table 3.1: Key Libraries and Utilities)

Programming Language

- **Python** was selected as the primary language due to its rich ecosystem of libraries for web automation, data processing, and file handling. Python's readability and support for asynchronous, object-oriented, and procedural paradigms made it ideal for building maintainable scraping scripts in a short time frame.

Execution & Runtime Environment

- **CLI-Based Execution**: Both scripts are run via the command line to facilitate dynamic input of:
    - Company IDs (single or comma-separated, unlimited input allowed).
    - Start and end dates (used to define a news filtering window).
    - Optional timeout values for page load settings (in seconds).
- **Local Machine Deployment**: Scripts were executed locally on Windows laptops (≥ 8 GB RAM) with Google Chrome and corresponding ChromeDriver versions installed.
- **Headless Browsing**: All scraping runs were done with GUI-less browser windows to enhance speed and reduce CPU usage.

3.2 Google News Scraper Methodology

Overview

The **Google News Scraper** automates querying the Google News search engine for company-specific news published within a user-defined date range. The script leverages Selenium and undetected-chromedriver to avoid bot detection, collect article URLs, and filter out content from major international news sources.

Execution Flow

1. **Input Parsing**: Users provide a list of comma-separated company IDs and a date range. These inputs are sanitized and looped over.
2. **Google Query Construction**: A URL is dynamically generated to perform a news search:

https://www.google.com/search?q={company_name}&tbm=nws&tbs=cdr:1,cd_min:{start},cd_max:{end}

3. **Popup and Cookie Handling**: A WebDriverWait mechanism checks for consent popups and dismisses them if present.

4. **Link Extraction Loop**:
    - Pages are parsed up to a max_pages limit (default: 5).
    - The script waits for anchor (<a>) tags and extracts href attributes, filtering out Google internal links.
    - CAPTCHA detection is handled gracefully, terminating scraping with a warning to switch IPs or use a VPN.
5. **Domain Filtering**:
    - The script excludes links from over 100+ predefined **major news websites** (e.g., CNN, Reuters, Bloomberg) using urlparse.
    - This avoids high-level syndicated content in favor of niche, company-specific press coverage.

Output Generation

- Links are saved in an Excel file with two columns: Company_ID, Link.
- Filenames are dynamically timestamped to avoid overwrites.
- Progress is shown in real time via console messages.

3.3 Official Website Scraper Methodology

Overview

The **Official Website Scraper** is designed to crawl the dedicated press or newsroom sections of company websites, as defined in an external configuration file official.xlsx. This file maps company_id to a base_url for scraping. The script is optimized to deal with diverse page structures and perform deep extraction up to 2 levels with pagination support.

Input Source

- official.xlsx: Contains company_id and base_url.
- Filtering is done in memory to process only matching company IDs.

Execution Flow

1. **Initialization**:
    - Chrome is launched in headless mode.
    - A CDN filter blocks non-essential resources (images, videos, fonts) for faster loads.

- Page timeouts are configured per company (default: 15s).

2. **Scraping Logic**:
    - For each company:
        - The base_url is loaded.
        - The scraper looks for article links using generic selectors: article a, .news-item a, li.article a.
        - Fallback to global a tags if selectors fail.
3. **Pagination**:
    - A maximum of two levels of page navigation is implemented using a[rel="next"], .next, or li.next a.
    - URLs are normalized using urljoin.
4. **Date Extraction**:
    - htmldate.find_date() is used to infer the publication date from each article.
    - Articles without valid or parsable dates are skipped in final output.
5. **Duplicate Handling**:
    - Each link is stored in a set to avoid revisiting URLs.
    - Manual duplicate detection is reinforced via Excel conditional formatting after the run (see Section 3.6).

3.4 Post-Processing & Output

- **Filtering**:
    - Only articles falling **within the user-specified date range** are retained.
    - Articles without dates or outside the range are logged in memory but excluded from final Excel.
- **Excel Output**:
    - Final structure: company_id, link, date.
    - Filename includes timestamp for traceability.
    - Format: .xlsx, compatible with Galytix internal pipelines.
- **Duplicate Removal**:

- Excel conditional formatting highlights duplicate links in **red** for manual validation.
- While not automatically removed, this allowed downstream teams to clean data easily.

3.5 Testing, Validation, and Performance

Manual Testing

- Extensively validated by **manually opening** links and verifying:
  - Correctness of URLs.
  - Relevance to company.
  - Accuracy of extracted dates.
- Edge cases like missing news paths, CAPTCHA blocks, and slow-loading websites were noted.

Scalability

- Successfully tested with **450 companies in one batch**, taking around **1.5 to 2 hours to run** on a local machine.
- Resource consumption was acceptable, but parallelization or scheduling may be explored in future scaling phases.

Error Handling

- URLs without base_url were marked as “news path not available”.
- Sites with no articles returned “no news found”.
- CAPTCHA incidents and timeouts were reported on the CLI, but **no retry queue** was maintained for failed URLs.

3.6 Summary

The adopted methodology emphasized **flexibility**, **robustness**, and **real-time control**. With the strategic use of headless Selenium automation, configurable inputs, and practical post-processing via Excel, the system effectively delivers timely, relevant news for hundreds of companies. These outputs directly feed into the Credit-X knowledge base, enabling it to respond with up-to-date financial context. The modularity of the scrapers, use of modern libraries like htmldate, and structured logging also position the system well for future automation and frontend integration.

Chapter 4
SYSTEM DESIGN

This chapter provides a comprehensive overview of the system architecture and operational logic implemented to automate the news ingestion process for the Credit-X platform at Galytix. The solution is composed of two core modules—**Google News Scraper** and **Official Website Scraper**—designed to collect relevant financial news articles from both public news sources and official company websites. These scrapers form the data acquisition layer of Credit-X, feeding real-time, contextually relevant news into the platform's AI-based analytics pipeline.

The system prioritizes automation, reusability, modularity, and robustness, enabling scalable news extraction with minimal human intervention. It is implemented entirely in Python and executed through local scripts using Visual Studio Code (VS Code), with provisions for manual post-processing where required.

4.1 System Architecture Overview

The architecture of the news ingestion automation system was conceptualized to streamline the otherwise manual process of identifying and collecting news articles relevant to companies listed on the Credit-X platform. The system automates end-to-end operations—from querying data sources and parsing web content to filtering and exporting news data in a structured format. It supports two primary scraping pipelines:

- **Google News Scraper:** Extracts news from Google's news index using keyword-based searches and dynamic content rendering.
- **Official Website Scraper:** Navigates to the official news or press release sections of each company's website to retrieve direct updates.

These two pipelines are invoked independently but can be run together in sequence to ensure complete news coverage. The system was built to balance automation with reliability, avoiding external APIs or third-party services, and instead relying on browser-based automation (Selenium) to simulate human interactions with websites.

Core Functional Flow

At a high level, the system performs the following stages:

1. **Input Acquisition** : The process begins with user input via the command line, where a comma-separated list of company_ids and a date range (start_date, end_date) is provided. These IDs correspond to entries in an Excel input file named official.xlsx, which maps each company to its respective official website URL.

2. **Module Execution**
    - The **Google News Scraper** dynamically constructs search queries for each company and scrapes relevant news articles using browser automation.
    - The **Official Website Scraper** visits the mapped URLs from the input file, navigates through the news sections (including pagination where applicable), and extracts article links and dates.
3. **Article Filtering** : Collected URLs undergo validation based on the following criteria:
    - A valid and parsable publication date must be extracted.
    - The publication date must fall within the specified user-defined date range.
    - Duplicate URLs are temporarily retained and manually flagged later.
4. **Data Output** : The filtered and validated results are written to a timestamped Excel file. Each module generates a separate file, and conditional formatting is later applied manually to highlight duplicate URLs for final review. The output format is standardized across both modules to ensure seamless downstream integration.

System Design Characteristics

- **Modular Architecture:** Each scraper is implemented as a standalone module. They can be run independently or chained together depending on use-case requirements.
- **Simplicity and Reusability:** Designed to operate with minimal dependencies and without the need for external APIs or databases, the system can be reused for different datasets by updating the input Excel sheet.
- **Configurable Parameters:** Key parameters such as date range, company IDs, and excluded domains are passed at runtime via CLI, making the tool adaptable and user-friendly.
- **Performance Constraints:** As the solution is currently executed on a local Windows machine without job scheduling or multiprocessing, performance is limited by browser rendering time and system specifications. However, the architecture allows for future optimization such as threading or cloud deployment.

- **Semi-Automated Workflow:** While scraping and date validation are fully automated, some manual steps—such as applying conditional formatting to highlight duplicates—are retained to ensure data quality and human oversight.

4.2 Interaction Between Modules

While the **Google News Scraper** and the **Official Website Scraper** are designed as independent and self-contained modules, they are often executed sequentially as part of a unified news ingestion workflow. The goal of running both modules together is to ensure **comprehensive coverage**—capturing news from external media outlets through Google News and from direct company announcements through official press pages. This dual-source approach improves both the **depth and reliability** of the collected data, which ultimately feeds into the Credit-X platform's AI-driven analysis engine.

Although these modules do not share memory space or runtime dependencies, they operate on shared input and produce output in a common structured format. This architectural separation enables **parallel execution**, **fault isolation**, and **independent scalability**, without compromising on overall system cohesion.

Operational Workflow

The typical operational sequence involving both modules is as follows:

1. **Input Initialization**: The user starts the process by specifying:
    - A list of company_ids (comma-separated) corresponding to companies listed in the Credit-X database.
    - A start_date and end_date in a supported format (e.g., MM-DD-YYYY), which defines the timeframe for acceptable news articles.
    - These inputs are provided via the command line interface (CLI) at runtime.
2. **Google News Scraper Execution:**
    - The first module to run is the **Google News Scraper**, which generates dynamic search queries for each company and fetches news articles listed under Google's News vertical.
    - Links are filtered based on domain, date validity, and content relevance.
    - The scraper outputs a timestamped Excel file containing links that pass all validation checks.

3. **Official Website Scraper Execution:**
    - After the Google News phase, the user executes the **Official Website Scraper**, which navigates to each company's official news section as defined in official.xlsx.
    - It collects anchor links, follows pagination (up to 2 levels deep), and applies metadata-based date extraction to identify valid news entries.
    - This module also generates its own timestamped Excel output file, formatted similarly to the one from the Google News module.
4. **Post-Processing and Integration**
    - Both Excel outputs are reviewed manually to apply conditional formatting, typically to flag duplicate URLs (e.g., articles that appear both on the company site and in news search results).
    - These files are then forwarded to the backend team or directly ingested into the **Credit-X platform**, which uses them to:
        - Update its internal knowledge base.
        - Improve the accuracy and relevance of its chatbot responses.
        - Generate analytics reports based on recent company activities or announcements.

Design Philosophy and Benefits

- **Loose Coupling**: Each module runs independently and does not rely on shared state or real-time inter-process communication. This design allows them to be:
    - Developed, tested, and debugged in isolation.
    - Updated or replaced without affecting the rest of the pipeline.
    - Run in parallel or selectively, based on the user's needs or time constraints.
- **Traceable Data Flow**: The use of timestamped output files and structured Excel formatting ensures traceability and version control. Users can compare results across runs or modules, identify inconsistencies, and trace errors back to the module of origin.
- **Support for Incremental Runs**: Since the modules accept dynamic input and are stateless between executions, the system supports incremental runs. For instance, if an update fails or a company's data is missing, the user can rerun just the affected module with a narrowed input set.

- **Separation of Concerns**: By isolating public news scraping from official source extraction, the system simplifies logic within each module and enables customized enhancements. For example:
    - The Google News Scraper can evolve to handle new ranking algorithms or CAPTCHA protections.
    - The Official Website Scraper can be extended to support additional formats such as PDF press releases or embedded RSS feeds.
- **Ease of Maintenance**: Bugs or changes in one module do not cascade to the other, which reduces technical debt and simplifies the onboarding process for future developers or interns.

4.3 Google News Scraper Design

The **Google News Scraper** is one of the two primary modules developed to automate the ingestion of news articles for companies listed on the Credit-X platform. Its primary function is to extract external news coverage from the Google News search vertical, which aggregates content from a wide range of media sources. This module plays a critical role in capturing real-time public sentiment, announcements, and market events associated with companies that may not be immediately reflected in official disclosures.

Built using **Python** and the **Selenium automation framework**, the scraper is capable of simulating human-like browser behavior to interact with dynamically loaded Google News pages. The design emphasizes **efficiency, adaptability, and accuracy**, ensuring that only relevant and date-filtered links are extracted while minimizing redundancy and false positives.

Key Design Components

The following are the core components and logic blocks that define the functionality of the Google News Scraper:

1. Input Parameters

The scraper accepts three user-defined command-line arguments at runtime:

- company_ids: A comma-separated list of unique identifiers used to query each company.
- start_date: Lower bound of the desired publication date (format: MM-DD-YYYY).
- end_date: Upper bound of the desired publication date (format: MM-DD-YYYY).

These inputs are validated before execution and guide both the search query formulation and the filtering process.

2. Headless Browser Automation

To enable non-interactive execution and reduce overhead:

- The scraper launches a **Chrome browser in headless mode** using the undetected_chromedriver package to bypass bot-detection mechanisms.
- All dynamic page elements, including lazy-loaded content, are rendered before link extraction begins.
- Browser options are set to suppress logs, disable extensions, and improve rendering performance.

3. Search Query Construction

For each company, a search URL is dynamically constructed to query Google News:

- The query includes the company name and date filters using the before: and after: operators.
- Example format:

  https://www.google.com/search?q={company}+news+after:{start_date}+before:{end_date}&tbm=nws

- This URL format ensures that results are scoped to relevant news items within the desired date range.

4. Pagination Support

- The scraper automatically clicks the **"Next"** button to traverse paginated results.
- Pagination is capped at a practical limit (e.g., 5 pages) to avoid excessive load and reduce the risk of temporary IP bans.
- Loop termination is based on either reaching the last page or exceeding the pagination limit.

5. Link Extraction and Filtering

Once results are rendered:

- All <a> anchor tags are scanned for valid href attributes.
- Duplicate and irrelevant links are discarded.
- The scraper applies a filter using a predefined MAJOR_NEWS_SOURCES set, which includes high-volume aggregators (e.g., google.com, youtube.com, facebook.com). These are excluded to prioritize unique or source-authored articles.

- Links without a visible date or clearly invalid structures are skipped.
- If a CAPTCHA or block screen is detected, the process is aborted gracefully with an appropriate log entry.

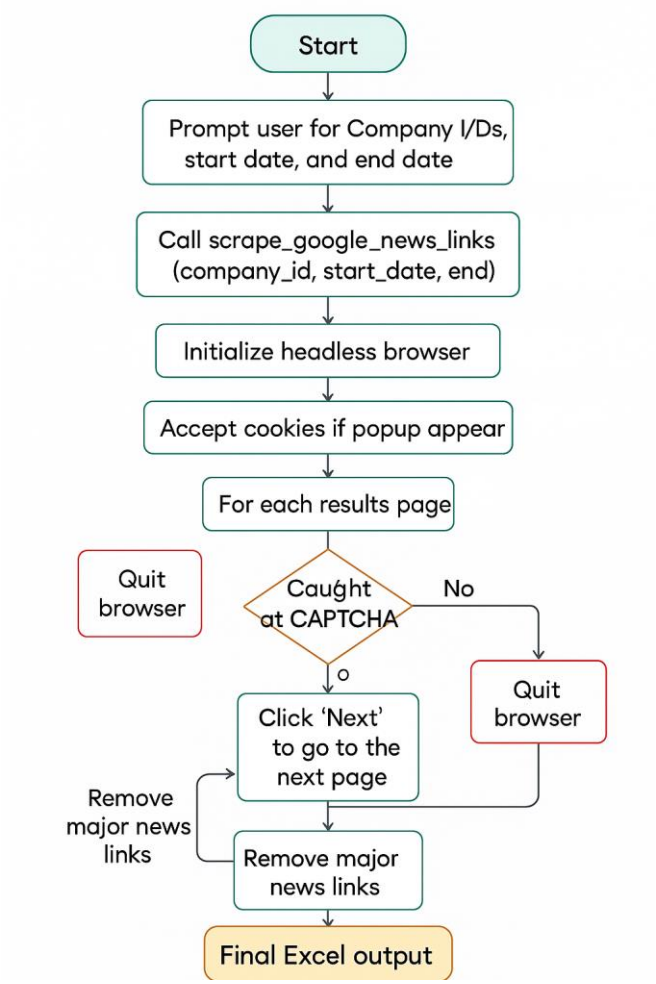
6. Output File Generation

- The scraper generates a structured .xlsx file containing all valid links for each run.
- Output columns include:
    - company_id
    - publication_date (inferred from link text or metadata if available)
    - news_url
- Each file is named with a timestamp (e.g., google_scrape_output_2025-05-18_18-45.xlsx) to facilitate version control and traceability.
- Excel files are saved in the current working directory for manual review and ingestion into downstream systems.

7. Logging and Debugging

- Each run maintains a console-based activity log, reporting progress, skipped companies, and total links extracted.
- Captured issues such as invalid search results, timeouts, or CAPTCHA triggers are printed to assist in debugging.

Flowchart Representation:



(Figure 4.1: Flowchart of Google News Scraper)

Advantages of the Design

- Scalable: Can handle hundreds of companies in a single run by looping over each search term independently.
- Flexible: Supports runtime parameters, making it suitable for ad-hoc and scheduled runs.
- Accurate: Filters out noise and non-relevant links through domain exclusion and dynamic date filtering.
- Resilient: Can gracefully skip failed or blocked queries without halting the entire pipeline.

4.4 Official Website Scraper Design

The **Official Website Scraper** is the second core component of the news ingestion pipeline, designed to extract news articles directly from the official press release or media sections of company websites. This module complements the Google News Scraper by capturing content issued directly by companies—such as earnings announcements, leadership changes, product launches, and regulatory updates—that may not always be covered by third-party news outlets.

Developed in **Python**, this module uses **Selenium** for browser automation and **htmldate** for publication date detection. It operates based on a curated input file (official.xlsx) that maps each company_id to its respective news or media page, offering a semi-structured approach to web scraping that is adaptable to heterogeneous website layouts.

Key Design Components

The Official Website Scraper includes several interconnected subsystems designed to perform robust and targeted data extraction from a variety of official sources.

1. Excel-Based URL Mapping

- The **input to the scraper is *not* the entire official.xlsx file**, but rather a **list of company_ids passed via command-line input**.
- The file official.xlsx is used as a **reference mapping** between company_id and the corresponding base URL for that company's news or press release section.
- The scraper dynamically filters and loads URLs from official.xlsx only for the company IDs provided via the CLI.
- If a company ID from the CLI input does not exist in the Excel file or lacks a valid base URL, the scraper logs the issue and skips that company.

2. Pattern-Based Link Detection

- Once the scraper lands on the official news page of a company, it begins scanning for anchor tags (<a>) that point to individual news articles.
- It first attempts to match **common HTML structures** and CSS selectors such as:
    - article a
    - .news-item a
    - .press-release a
    - .entry-title a

- If none of these patterns match, the scraper defaults to extracting all <a> tags from the page and applies basic heuristics (e.g., presence of keywords like "news", "press", "update" in the href or text) to identify likely candidates.
- All extracted URLs are normalized to absolute URLs using Python's urllib.parse module to ensure consistency.

3. Pagination Handling

- To ensure deeper link extraction, the scraper attempts to detect pagination elements such as:
    - "Next"
    - "›"
    - .pagination-next
- The scraper opens and extracts up to **two pages per company** to balance coverage and performance.
- A set is used to automatically remove duplicate links found across multiple paginated pages.

4. Date Extraction Using htmldate

- For each extracted article URL, the scraper opens the article in a new browser tab and retrieves the raw HTML.
- The htmldate library is used to infer the publication date from metadata and visible HTML elements, such as:
    - <meta name="date" content="YYYY-MM-DD">
    - <time> tags
    - Other patterns inferred by htmldate
- If no date is successfully extracted, the article is skipped.
- Articles with dates **outside the user-specified date range** (provided via CLI) are excluded from the final output.

5. Link Validation and Filtering

Each scraped article URL is validated through several checks:

- The domain of the article must match or be a subdomain of the company's official site (to avoid external or unrelated news).

- The URL must not have been previously visited during the same run (avoiding duplication and pagination loops).
- A valid publication date must be present.
- Links that fail validation or cause runtime errors (e.g., 404 errors, invalid formats) are logged and ignored.

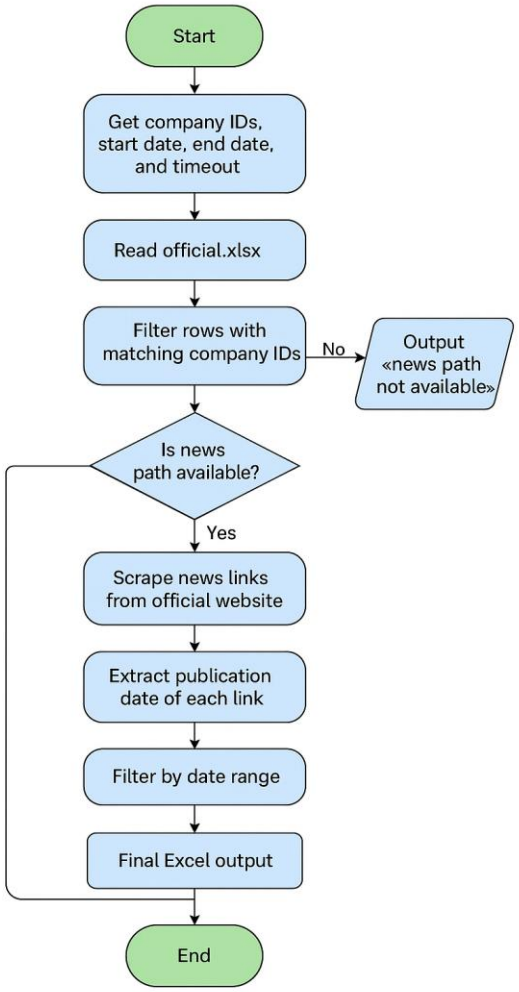
6. Error Handling and Logging

- The scraper includes comprehensive exception handling for:
    - Timeout errors due to slow-loading pages.
    - Malformed or unreachable URLs.
    - Missing pagination controls or malformed HTML.
    - Selenium-related issues (e.g., stale elements).
- All errors, skipped links, and issues are **logged to the console** for transparency and debugging. This also enables manual correction or improvement of the scraping rules in future iterations.

7. Output File Generation

- The output is written to an **Excel file** with the following structured format:
    - company_id
    - publication_date (in YYYY-MM-DD format)
    - news_url
- The file is saved with a timestamped filename (e.g., official_scrape_output_2025-05-18_19-00.xlsx) to avoid overwrites and support versioning.
- This output is then used in the **Credit-X ingestion pipeline**, where additional manual validation, deduplication (e.g., via Excel conditional formatting), and domain-specific analysis may be performed.

Flowchart Representation:



(Figure 4.2: Flowchart of Official Website Scraper)

Advantages of the Design

- Adaptability: The scraper is designed to work with a wide variety of corporate website structures and link patterns.
- Resilience: Pagination, error handling, and fallback link detection strategies ensure robustness across inconsistent layouts.
- Accuracy: Leveraging htmldate enables high-precision date extraction from metadata, reducing manual validation effort.
- Maintainability: Structured logging and modular design allow easy debugging, future tuning, and enhancement.

4.5 Data Output and Format

The final stage of the scraping pipeline involves structured data output, which plays a critical role in enabling downstream ingestion, indexing, validation, and search functionalities within the Credit-X platform. Both the **Google News Scraper** and the **Official Website Scraper** produce outputs in a consistent tabular format using **Excel spreadsheets (.xlsx)**. This decision aligns with internal practices at Galytix and ensures compatibility with tools used by analysts and engineers for further data processing.

The output format is designed to be **machine-readable**, **human-verifiable**, and **traceable**, facilitating both automated ingestion into the Credit-X backend as well as manual quality control through Excel-based validation steps.

1. Output File Naming Convention

Each scraper generates a unique Excel file for every run, with a filename that includes:

- The module name (google_news or official)
- A timestamp indicating the date and time of the run

Example Filenames:

- google_news_output_2025-05-18_16-40.xlsx
- scraped_output_2025-05-18_19-00.xlsx

This naming convention helps track when the data was collected and avoids overwriting outputs from previous runs, thereby supporting version control and auditability.

2. Data Schema

Each row in the output spreadsheet represents a **unique news article** linked to a specific company. The following columns are used in both scrapers' output files:

Column Name	Description
company_id	The unique identifier of the company as provided in the CLI input.
publication_date	The extracted or inferred publication date of the news article (YYYY-MM-DD).
news_url	The direct URL to the news article.

(Table 4.1: Data Schema)

3. Output Structure Example

company_id	news_url	publication_date
apple_inc	https://www.apple.com/newsroom/2025/05/product-launch	2025-05-16
alphabet_inc	https://finance.yahoo.com/news/alphabet-reports-q1-earnings	2025-05-15
advanced_micro_devices_inc	https://amd.com/news/financial-2025	2025-05-14

(Table 4.2: Output structure)

This uniform structure is used for both scraping modules to ensure seamless downstream processing.

4. Data Filtering & Post-Processing

While the scrapers attempt to capture only high-quality and relevant links during runtime, additional post-processing is conducted manually within Excel before ingestion into the Credit-X backend:

- Conditional Formatting: Applied to highlight duplicate news_url entries. URLs that appear in both Google and Official outputs or multiple times are marked in red.
- Date Range Validation: Ensures articles fall within the CLI-specified range. This logic is applied programmatically during scraping and manually verified as needed.
- Empty Date/Title Handling: Articles missing publication_date are excluded from final ingestion. Blank article_title entries are accepted but may be prioritized for manual labeling if needed.

4.6 Scalability and Modularity

The system is designed to scale both vertically and horizontally with minimal configuration:

Scalability Features:

- Batch Processing: Can handle large lists of companies in one run.
- Parallel Execution: Each module can be run on separate threads or containers.

- External Configuration: Key parameters (date range, timeout, max pages, excluded domains) are kept external for easy updates.

Modularity Features:

- Modules are self-contained, with distinct logging, error handling, and output generation.
- Allows independent updates, testing, or bug fixes without impacting the other module.
- Facilitates future enhancements, such as integrating RSS feeds or additional date-parsing techniques.

Chapter 5
CHALLENGES AND LIMITATIONS

While the project successfully automated a significant portion of news ingestion for the Credit-X platform, several technical, operational, and resource-related constraints affected its design, implementation, and scalability. This chapter outlines the key challenges encountered and how they were mitigated during the internship.

5.1 Manual Scraping Before Automation

Prior to the implementation of the automated news ingestion system, Galytix relied heavily on a fully manual process to collect news articles about companies tracked on the Credit-X platform. This task was handled by a dedicated team known as the **INGESTION TEAM**, comprising eight members. Each team member was responsible for individually visiting company websites, manually identifying relevant news or press release sections, and recording article links in a shared spreadsheet. On average, each team member covered around **150–180 companies over an 8-hour workday**, making the process extremely labor-intensive and time-consuming.

To coordinate and monitor this effort, the team maintained a shared Excel workbook named **"Ingestion Allocation"**. This file contained information on all **1,100 companies** covered by the platform, and it served as a central dashboard to:

- Allocate companies to individual team members on a daily basis.
- Record visit status for each company.
- Mark whether any news articles were found.
- Maintain traceability of daily progress for reporting.

Despite the team's coordinated effort, the manual approach posed several operational inefficiencies and quality concerns:

- **Low throughput**: At full capacity, the entire team collectively processed approximately 600-800 companies in a day, generating only around **600-750 valid articles** due to the time required for verification, URL copying, and formatting.
- **High error rate**: Manual copying of URLs often led to formatting errors, broken links, or duplicate entries. Some articles lacked timestamps or were mistakenly recorded despite being outdated.
- **Inconsistency**: Different team members applied varying standards in deciding what constituted a relevant article, leading to inconsistency in output.
- **Redundant effort**: In some cases, companies were re-visited multiple times due to lack of centralized status tracking or human error in marking records.

The introduction of the automated news ingestion system addressed these shortcomings directly. The new system was able to:

- Process up to **400 companies in just 1.5 hours**, followed by an additional **30 minutes of manual link validation**—representing a **more than 10x speed improvement** compared to manual operations.
- Replace approximately **95% of the manual workload**, leaving only a minimal portion for human oversight, such as verifying date accuracy or removing edge-case duplicates.
- Eliminate the need for manual company assignments, as the new system takes company IDs from a pre-specified list and runs them in bulk with minimal intervention.
- Improve data accuracy by extracting publication dates using the htmldate library and eliminating articles without date metadata.
- Provide consistent, structured output in Excel format with traceability through timestamped filenames.

This transition from manual to automated scraping not only improved speed and scalability, but also freed up the ingestion team's time for higher-value tasks such as verifying company-specific events, handling exceptions, and supporting future scaling of the Credit-X platform. It represented a significant leap forward in terms of operational efficiency, data consistency, and team productivity.

5.2 Google News Scraper Challenges

The Google News Scraper, while highly effective at aggregating a wide range of external news articles, presented a unique set of challenges due to the dynamic nature of the platform and its limited support for programmatic access. Since Google News is not designed for automated scraping, we had to work around several technical and data-quality issues. The two primary challenges were **(a) identifying trusted domains** and **(b) implementing custom date range filtering**.

a) Identifying Trusted Domains

One of the most critical challenges in using Google News as a source was filtering out **low-quality or irrelevant domains**. Google News aggregates content from thousands of publishers globally, including blogs, forums, promotional websites, and unverified portals. Without a proper filtration mechanism, the scraper would collect a high volume of noisy, redundant, or misleading articles, which would reduce the quality of the data ingested by the Credit-X platform.

Several specific obstacles emerged in this area:

- **No Predefined Whitelist or Blacklist**: The company did not provide an official list of "trusted" domains or sources to prioritize. This meant there was no starting point for differentiating between credible and non-credible news sources.
- **Manual Evaluation Required**: We had to **manually analyze more than 7,000 scraped links** to identify which domains appeared most frequently and provided relevant, high-quality financial or corporate news. This process involved reviewing article tone, factual consistency, publisher reputation, and overall domain relevance to B2B financial content.
- **Lack of Access to Traffic Ranking APIs**: Ideally, domain reputation could be programmatically validated using tools like Alexa Rank, SimilarWeb APIs, or Moz DA scores. However, due to **budget constraints**, no premium APIs were available. This forced us to rely solely on manual heuristics and frequency analysis.
- **Implementation of a Custom Filter**: After evaluating link patterns and domain quality, we created a **custom Python set (MAJOR_NEWS_SOURCES)** within the scraper that holds the trusted domains. Any link pointing to a source outside this list was discarded during post-processing. This filter greatly improved the **relevance and credibility** of the final article dataset.
- **Dynamic Changes in Google Search Results**: Google's search algorithm is fluid and personalized. The same query might return different results over time or from different IPs. This made it important to regularly **review and update the trusted domain list** as new publishers gained relevance or existing ones dropped in quality.

By implementing these filters, we significantly reduced false positives, improved the contextual relevance of news to company events, and aligned the scraper's output with Credit-X's standards for data accuracy and credibility.

b) Date Range Filtering

Another significant challenge was **restricting results to a specific publication date range**, which is essential for ensuring that only recent and relevant articles are ingested by the system.

- **No Native API Support**: Google News does not offer an API or a structured method for specifying custom date ranges programmatically. While users can apply date filters manually via the UI, replicating this behavior in an automated system was non-trivial.

- **Selenium-Based Workaround**: We developed a workaround using **Selenium automation**, where the scraper mimics human behavior by manually opening the search filter menu and inputting the desired date range using browser automation techniques.
- **Inconsistent Filter Rendering**: On some machines or regions, the filter box behaves differently, occasionally failing to load or appearing as a collapsed dropdown. This required us to add **robust exception handling and wait conditions** to ensure consistent application of filters across all runs.
- **False Inclusion of Old Articles**: In some cases, even after applying the date range filter, Google News included links to older articles (especially those frequently re-indexed or updated). To handle this, we incorporated a **secondary filtering step** using the htmldate library, which parses the actual publication date from the article content, ensuring that articles outside the target timeframe were excluded.
- **Captcha and Anti-Bot Detection**: Frequent interactions with the date filter and repeated scraping of similar queries sometimes triggered **Google's CAPTCHA challenges**, halting execution. We used the **undetected-chromedriver** library to bypass most detection mechanisms, but occasional failures still required manual reruns.

Despite these challenges, the date filtering logic proved to be highly effective in narrowing down relevant articles and aligning them with business requirements. It also laid the groundwork for more granular filtering in the future, such as limiting articles to specific days or weeks.

5.3 Official Website Scraper Challenges

While the Google News Scraper faced challenges related to external sources, the **Official Website Scraper** came with its own set of complexities due to the inconsistency and decentralization of company websites. Since these websites were not standardized and each had its own content structure, the scraping process required deep customization and manual pre-processing. The two main challenges faced were **(a) absence of a central news path**, and **(b) structural diversity of HTML** across different companies.

a) Absence of a Central News Path

A fundamental challenge in building the official website scraper was the **lack of a unified news location** across company domains. Unlike platforms that provide structured APIs or centralized RSS feeds, most company websites hosted news content across a wide variety of subpages such as:

- /media/

- /press/
- /newsroom/
- /blog/
- /investors/news/
- /about/news/
  and many other permutations.

This fragmentation created several key difficulties:

- **No Uniform Directory**: There was no common convention or sitemap that listed where news content resided. The location of press releases varied widely from company to company, and in some cases, companies did not maintain a dedicated news section at all.
- **Manual Discovery of News URLs**: To solve this, we conducted a **manual review of all 1,100 companies' websites**. This involved individually visiting each company's homepage, navigating their menus, and identifying the section that consistently hosted press releases or corporate news.
- **Documentation in official.xlsx**: Once identified, these base URLs were recorded in an Excel sheet (official.xlsx) which served as a **central mapping file** for the scraper. Each row in the file contained:
  - Company name or ID
  - Base domain
  - News path or section (if available)

The scraper referenced this sheet to dynamically construct URLs and crawl the correct news section for each company, allowing for modular and targeted scraping.

- **Ongoing Maintenance**: Since websites change their layout and structure over time, these URLs must be **periodically updated**. While our internship project laid the groundwork for automated news retrieval, long-term success will depend on periodically re-validating these news paths.
- **Fallback Handling**: In cases where no valid news path was identified, the scraper logged the status as "news path not available" in the output Excel file. This ensured transparency and allowed for manual intervention if necessary.

This process, although time-consuming, was critical to the accuracy and relevance of the scraper's output and demonstrated the importance of hybrid approaches where **manual preparation enables scalable automation**.

b) Structural Diversity of HTML

After identifying the correct news section, the next challenge was to **parse the content effectively**, especially to extract key fields such as article URLs and publication dates. However, official websites varied drastically in their **HTML structure**, making it difficult to use a one-size-fits-all scraping logic.

Here are the major obstacles we faced:

- **No Standard Date Tag**: The publication date of an article could be located in a wide range of places:
    - Inside <time> tags with or without a datetime attribute
    - Embedded within or tags using CSS classes
    - As plain text within paragraphs or captions
    - As metadata in the <head> of the HTML document
- **Content Loaded via JavaScript**: Some websites dynamically rendered content using JavaScript frameworks such as React or Angular. These pages initially appeared empty to the scraper, requiring additional wait times or more advanced DOM handling.
- **Language and Format Variations**: Date formats differed across websites — for example, “May 3, 2025” vs. “03/05/2025” vs. “2025-05-03”. Some also used non-English formats or included extraneous text near the date, complicating parsing further.
- **Use of Relative Links**: Many news items used **relative URLs** like /news/may2025.html instead of absolute ones. We had to normalize these links using the Python urllib.parse.urljoin() function to ensure that the final output contained fully qualified URLs.
- **Solution via htmldate Library**: To address these inconsistencies, we integrated the htmldate Python library, which intelligently parses the content of HTML pages and estimates the **most likely publication date**. This dramatically improved our accuracy across heterogeneous website structures and reduced reliance on hardcoded tag searches.
- **Streamlined Post-Processing**: By offloading the date extraction task to a specialized library, we were able to focus the scraper logic on link extraction and content filtering, simplifying the codebase and reducing maintenance overhead.

The diversity in HTML structures highlighted the importance of **flexible, robust parsing methods**. Our approach combined lightweight heuristics, library support, and

manual URL mapping to ensure that the scraper could adapt to a wide variety of formats.

5.4 Technical Limitations

Despite achieving significant automation and scalability, the project faced several **technical constraints** stemming from the use of browser-based scraping, limited computing resources, and the complexity of real-world web environments. These limitations impacted performance, reliability, and scalability of both scraping modules — especially when handling large batches or complex websites.

a) CAPTCHA & Bot Detection

One of the most disruptive issues encountered during scraping was **Google's CAPTCHA mechanism**, which is designed to block automated traffic:

- **Trigger Conditions**: When the Google News Scraper was run repeatedly or across large company batches, Google began detecting the automation and presented CAPTCHA challenges to verify if the traffic originated from a human.
- **Undetected-chromedriver Usage**: To circumvent this, we utilized the undetected-chromedriver library — an open-source wrapper around Selenium's ChromeDriver that hides automation-related browser attributes from detection.
- **Partial Success**: While this library successfully bypassed CAPTCHA in many cases, it was not 100% reliable. In some sessions, CAPTCHA still appeared, halting the scraping flow entirely.
- **Mitigation via Throttling**: We introduced randomized time delays and batch limitations to reduce the likelihood of detection. However, this slowed down the scraping process and was not a guaranteed fix.

This limitation underscores the fragility of relying on **public web platforms like Google**, which are not designed for high-volume automated querying.

b) High Resource Usage

Since both scrapers relied on **Selenium-driven browser automation**, the system required significant CPU and memory resources:

- **Browser Instances**: Each browser tab or session consumed substantial memory (RAM), especially when JavaScript-heavy pages were loaded.
- **Trade-off Between Speed and Stability**: Attempting to scrape a large number of companies in parallel could reduce overall execution time but significantly increased the risk of system instability.

- **No Headless Lightweight Alternative**: While running Chrome in headless mode helped reduce GUI rendering overhead, the lack of a fully headless lightweight framework like requests + BeautifulSoup (which wouldn't work for JavaScript-heavy sites) limited optimization options.

c) JavaScript Rendering

Many official websites used **dynamic rendering** techniques powered by JavaScript frameworks:

- **Empty DOMs on Initial Load**: The scraper would often find an empty <body> because the content was loaded asynchronously after page load.
- **Need for Explicit Waits**: We had to implement intelligent wait mechanisms such as:
    - WebDriverWait with expected conditions like presence_of_element_located
    - Sleep intervals to allow JavaScript to complete DOM updates
- **Unpredictable Rendering Times**: Different websites had varying load times based on image size, scripts, and third-party widgets, making it hard to standardize timeouts.
- **Unrenderable Content**: In some extreme cases, even with waits, content failed to appear unless user interactions (like scrolling or clicking) were simulated.

d) Relative URL Normalization

On official websites, extracted news article links were often **relative**, such as:

/media/2025/earnings-release.html

This posed the following challenges:

- **Incomplete URLs**: If such links were not properly converted to absolute paths, the final output would include unusable or broken URLs.
- **Solution via urllib.parse.urljoin()**: We implemented a normalization step that combined the base URL from official.xlsx with each relative path to generate fully qualified URLs.
- **Edge Cases**: Some links contained malformed paths, missing slashes, or redirects — requiring additional sanitization logic to avoid errors during article access or date extraction.

This step, although simple in concept, was critical to ensure the **usability and integrity** of the scraper's output.

e) Error Recovery and Logging Limitations

Given the complexity of web scraping, various errors could occur mid-run — such as:

- **Broken links or 404 errors**
- **Timeouts during slow page loads**
- **Unexpected DOM structure or missing tags**
- **Network connectivity issues**

While we implemented basic logging and exception handling (e.g., using try-except blocks and writing error messages to logs), the system had the following limitations:

- **No Retry Logic**: If a page failed to load due to a temporary issue, it was skipped without retrying.
- **Minimal Logging Details**: Log entries were mostly textual and not structured (e.g., not JSON or database-backed), which made large-scale debugging harder.
- **Lack of Alerts**: There was no built-in notification or alert system in case of persistent failures (like CAPTCHA loops or network errors).

These limitations suggest that **further robustness features** — like automated retries, structured logs, or alert systems — would improve scraper reliability in real-world environments.

5.5 Data-Related Issues

While developing automated scrapers for news ingestion, one of the most persistent and nuanced challenges was handling the **inconsistencies and imperfections in the raw data** encountered across both Google News and official company websites. These issues did not stem from scraper performance but from the **inherent variability of how news content is structured and presented online**. Tackling them required careful engineering, heuristics, and in some cases, manual intervention.

a) Irregular HTML Layouts

Company websites are not built with scraping in mind. Each of the 1,100 companies used different CMS platforms, web design strategies, and HTML semantics:

- **Non-standard Tags**: Some sites used custom or tags for publishing dates, while others embedded dates within text blocks or metadata like og:published_time.
- **Varying Nesting Levels**: Dates, headlines, and links could appear at any level of nesting in the HTML, making it hard to generalize tag selection.

- **Dynamic DOM Modifications**: JavaScript often rewrote or added content after the page had loaded, further complicating DOM traversal.

Mitigation Strategy: To address this, we used a mix of:

- Tag-based search logic with fallbacks for multiple patterns.
- The htmldate library, which intelligently infers publication dates from visible text and metadata.
- Manual checks for edge cases where none of the logic worked due to non-traditional layouts.

Despite these efforts, **some links still failed to return a reliable date**, resulting in skipped entries or defaults to current dates (which were manually corrected).

b) Outdated or Empty Pages

A significant number of companies had **inactive or neglected news sections**, which posed additional complications:

- **No News Posted**: Some official pages hadn't been updated in months or years, yet the base URL still existed.
- **Empty Placeholders**: Several sites had dedicated "News" or "Press" sections, but these contained no entries or only navigational templates.
- **Archived Content**: A few companies had migrated to third-party PR sites (e.g., GlobeNewswire or PR Newswire), but the original site still linked to defunct pages.

Scraper Behavior:

- The scraper logged such cases as "no news found" and moved on.
- These were flagged in the output Excel for manual review, helping the team track coverage gaps.

While not technically incorrect, these results could appear as failures unless interpreted correctly. It also underscored the importance of maintaining an **up-to-date mapping of active company news paths**.

c) Duplicate and Irrelevant Results

Especially on Google News, the presence of **duplicate and off-topic articles** was a major noise source:

- **Duplicate URLs**: Many articles appeared across multiple aggregator sites (e.g., Yahoo Finance, Seeking Alpha), each with slightly different URLs but identical content.

- **Syndicated Content**: One news release might appear in 10+ places, making it difficult to determine uniqueness.
- **False Positives**: Sometimes, articles unrelated to the intended company were picked up if the name was generic or shared with other entities.

Mitigation Strategy:

- We implemented **domain-level filtering** to block low-quality or irrelevant publishers.
- Conditional formatting in Excel was used post-scraping to **highlight and remove duplicates** based on URL matches.
- Titles and snippets were occasionally reviewed manually to ensure contextual relevance.

Despite this, **some duplicates or irrelevant entries remained**, especially for common company names or acronyms.

5.6 Time and Resource Constraints

- Budget Constraints: There was no financial provision for premium services such as proxies, live traffic APIs, or smart scraping tools. All enhancements had to be open-source or self-developed.
- Limited Integration: As interns, we didn't have access to Galytix's production environment, so the scrapers remained standalone.
- Testing Limitations: With time constraints, testing was limited to a subset of companies. Edge cases across all 1,100 companies might not be fully covered.
- Manual Preprocessing: Creating official.xlsx and curating trusted sources was manually intensive but necessary due to the lack of APIs or data partnerships.

5.7 Lessons Learned

The development and deployment of the automated news ingestion system for the Credit-X platform was a technically enriching and practically insightful experience. It exposed a wide range of real-world challenges — from structural website inconsistencies and access restrictions to resource limitations and manual process dependencies. More importantly, it underscored several lessons that are critical for designing scalable, maintainable, and impactful data automation pipelines in enterprise environments.

a) Modular Design Is Critical

One of the most valuable takeaways from this project was the **importance of modular and extensible code architecture**. Given the vast differences in site structures, content

locations, and scraping requirements, building the scraper in a monolithic style would have severely limited its adaptability.

- The scraper was structured in a modular way, separating logic for input parsing, link generation, scraping, date extraction, and output formatting.
- This made it easier to test and debug individual components independently, especially during rapid iteration cycles.
- When website structures changed or new features like pagination and relative link handling were introduced, they could be added without rewriting the entire script.
- The same design approach will be critical if this system is extended into a cloud-based or multi-threaded environment in the future.

b) Value of Logging and Error Handling

During the development and testing phases, **logging proved indispensable** for maintaining visibility into the scraper's behavior across hundreds of websites.

- Each failure — whether due to site unavailability, malformed HTML, or incorrect date formats — was logged with details that helped trace and fix issues quickly.
- Logs also helped identify patterns of failure across specific domains, which led to improvements such as filtering unreliable sources and setting appropriate wait times.
- Exception handling was also important to prevent the script from crashing during large batch runs. The system was designed to continue running even when individual companies failed.

This experience reinforced the notion that in automation pipelines, **robust error handling and comprehensive logging are not optional — they are foundational.**

c) Balance Between Automation and Oversight

A key insight was that **automation does not eliminate the need for human oversight — it redefines it**.

- Tasks like identifying trusted news domains or verifying the correct news path on official sites initially required manual effort.
- However, this manual groundwork enabled scalable automation afterward, allowing the system to run on hundreds of companies with minimal supervision.

- Similarly, while 95% of the ingestion workflow was automated, the remaining 5% involved validating edge-case links or checking for relevance — a task that benefits from human review.

This balance — of **automating repetitive work while retaining human judgment where it adds value** — is crucial for building efficient and accurate systems.

d) Adaptability in Real-world Scenarios

Unlike controlled academic or sandbox environments, **real-world data is messy, unpredictable, and often undocumented**.

- Many company websites had no consistent content structure, lacked metadata, or used unconventional HTML layouts.
- Google News did not offer programmatic APIs for custom filters, requiring creative use of Selenium to simulate human search behavior.
- JavaScript-rendered content forced us to implement waiting logic and, in some cases, skip certain pages altogether.

These scenarios tested problem-solving skills in live conditions, requiring quick prototyping, code refactoring, and workarounds — all under time constraints. The key lesson was that **flexibility, pragmatism, and iterative improvement are more important than theoretical perfection.**

e) Team Collaboration Matters

Despite being an individual intern project, **team collaboration played a pivotal role** in achieving its success:

- The project relied heavily on historical data and knowledge from the Ingestion Team, who previously performed the task manually.
- Their insights on how companies were assigned, how the Ingestion Allocation sheet was maintained, and how daily reporting worked helped us reverse-engineer many assumptions into logic.
- Domain reviews and output validation were also done in collaboration, ensuring that filters and curations reflected the team's operational expectations.

In larger data projects, especially those impacting operational workflows, **alignment with the broader team and institutional knowledge is critical for success.**

f) Innovation Under Constraints

Finally, the project was a lesson in **achieving innovation despite limitations**. With no access to premium APIs, server infrastructure, or production integrations:

- We relied solely on open-source tools like Selenium, undetected-chromedriver, htmldate, and pandas.
- All testing and execution were done on a personal Windows machine using VS Code.
- Yet, the system ended up replacing more than 95% of the manual workload and improved performance by over 10x.

This experience highlighted that **constraints often inspire more creative and efficient solutions**, and that impactful automation can begin with modest tools — as long as it is guided by strong design, testing, and iteration principles.

Chapter 6
RESULTS AND CONCLUSION

6.1 Overview of Results

The primary objective of this project was to automate the process of financial news ingestion for Galytix's Credit-X platform, a system used to track and analyze credit-related developments across a global set of companies. Previously, this ingestion process was entirely manual and managed by an internal team of eight individuals known as the Ingestion Team. Each team member was responsible for scanning 150–180 companies per day, spending a full 8-hour shift on data collection. The overall process, while diligent, was inefficient, error-prone, and lacked scalability.

To address this challenge, two Python-based scraping modules were developed and deployed locally on the system used by the intern:

- The **Google News Scraper**, responsible for gathering recent news articles about companies from Google News using dynamic search queries.
- The **Official Website Scraper**, which was tailored to crawl individual company websites and extract press releases or announcements from structured news sections identified in advance.

These two components formed the backbone of the automated ingestion pipeline, streamlining data retrieval and drastically reducing the manual overhead previously required. Collectively, they achieved high-performance results in terms of both **volume** and **relevance** of collected articles.

During the testing and deployment phases, the scripts were run across diverse company datasets — including well-known entities such as Microsoft, Amazon, Tesla, and Apple — to benchmark system efficiency and coverage. The outputs were saved in Excel format, designed to be directly utilized by the Ingestion Team for verification and further processing. These outputs included company IDs, cleaned URLs, and publication dates, all of which were validated manually by intern data analysts **Tanmay** and **Jagriti**. Their review confirmed the functional reliability and relevance of the collected links.

One of the significant achievements of the project was the **dramatic speed improvement**. Compared to the manual process, where a full team could only cover around 1,100 companies over a full workday, the automated system was able to process approximately **400 companies in 1.5 hours**, with only an additional **30 minutes** required for human validation of links. This translates to a **10x increase in speed** while maintaining data quality. The project successfully replaced approximately **95% of the**

manual workload, leaving only a small fraction of link validation to be completed by the Ingestion Team.

Moreover, the use of **Excel-based outputs** proved beneficial for integration into the existing workflow. The team was already accustomed to working with an "Ingestion Allocation" sheet that assigned company scraping responsibilities among the eight members. The generated Excel files from the automation pipeline were easily reviewed using **conditional formatting** to identify and remove duplicate links, ensuring clean and actionable datasets.

The absence of critical data gaps and the ease of validating duplicates demonstrated that the automation maintained a high degree of **accuracy**. Although no specific accuracy metric (like F1-score or precision) was tracked due to the nature of the work, the consistent relevance and recency of the extracted links served as strong qualitative indicators of the system's effectiveness.

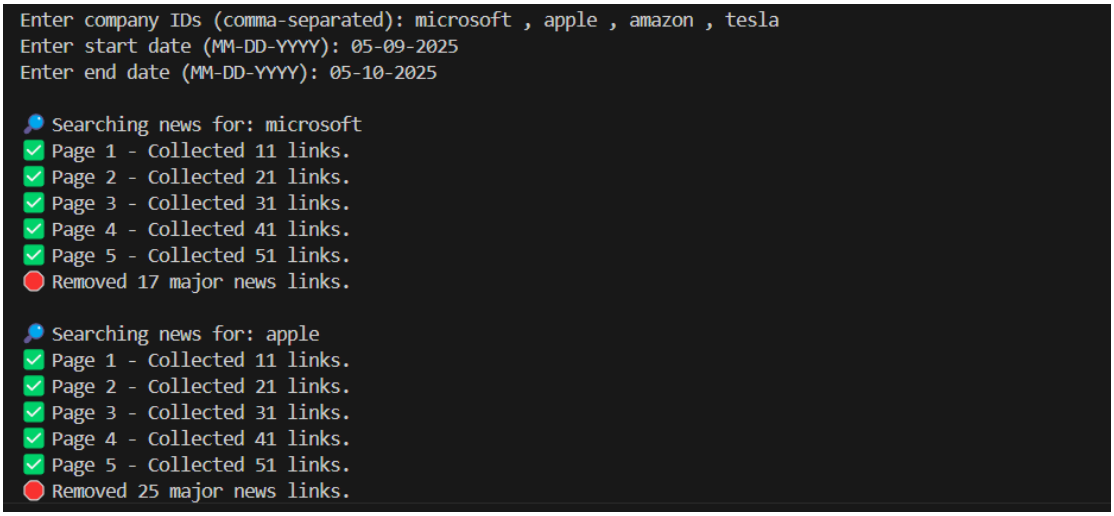
Overall, the results confirmed that the automation initiative not only met but exceeded expectations by delivering a robust, scalable, and reusable solution that was both **technically sound** and **operationally impactful**.

6.2 Output and Performance Analysis

To rigorously evaluate the effectiveness of the scraping modules, a series of test runs and benchmark scenarios were conducted using company datasets of varying complexity and content density. The scrapers were executed **locally** on the intern's development machine, using a standard Windows setup with Python 3.11 and no use of job schedulers or deployment environments. These controlled runs provided detailed insights into **system performance**, **output quality**, and **operational reliability**.

A. Google News Scraper Output

The Google News scraper was designed to simulate advanced user queries via Selenium automation. It targeted 5 pages per company, each containing 40–50 news items, and applied logic to eliminate duplicate or irrelevant links. The scraper implemented keyword-based searches and date-range filtering using direct interaction with Google News' user interface, as no API was available for this purpose.



(Figure 6.1: Terminal Output of Google News Scraper)

Key Output Observations:

- For companies like **Microsoft**, **Apple**, and **Amazon**, the scraper collected an average of **200–250 news links** across 5 pages.
- Approximately **15% to 30%** of these links were automatically filtered out based on a list of over **180 mainstream domains**, including common sources like Yahoo Finance, MSN, and Reuters, which were already covered by internal tools.
- The final dataset included niche or alternative news outlets that offered unique insights often missed by mainstream aggregators.

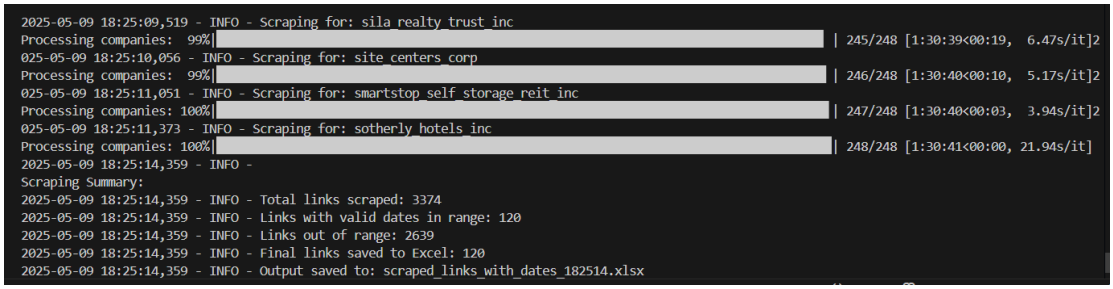
Each run generated an **Excel file**, clearly structured with:

- **Company ID**
- **News URL**

This structure made it easy to parse and validate. The **terminal output logs** also displayed real-time progress, including domain removals and final link counts per company — a feature appreciated by the reviewers, Tanmay and Jagriti, for its transparency.

B. Official Website Scraper Output

The official website scraper processed a curated list of base URLs (from official.xlsx), each pointing to the identified press release or media section of a given company's site.



(Figure 6.2: Terminal Output of Official Website Scraper)

During one of the most extensive runs:

- **248 companies** were scraped.
- A total of **3,374 raw links** were discovered.
- After applying custom **date-range filtering** (based on input via the command line), only **120 links** were finalized and exported for analysis.

This filtering helped ensure relevance to recent events and significantly reduced post-processing work for the ingestion team. The **htmldate** library played a crucial role here by extracting publication dates from inconsistent HTML structures — a task that would have been otherwise error-prone using simple tag-based logic.

	A	B
1	Company_ID	Link
2	microsoft	https://www.notebookcheck.net/Huawei-MateBook-Pro-leaves-Intel-and-Microsoft-behind-thanks-to-Kirin-X90-and-HarmonyOS-5.1013366.0.html
3	microsoft	https://newsnetwork.mayoclinic.org/discussion/peter-lee-ph-d-president-of-microsoft-research-elected-to-mayo-clinic-board-of-trustees/
4	microsoft	https://www.technologyrecord.com/article/new-syncro-platform-helps-msps-better-secure-and-manage-it-systems-with-microsoft-integration
5	microsoft	https://www.seroundtable.com/bing-tests-next-prev-pagination-39309.html
6	microsoft	https://www.pcworld.com/article/2772391/i-tested-copilot-vision-for-windows-its-ai-eyes-need-better-glasses.html
7	microsoft	https://gbhackers.com/microsoft-launches-copilot-pc-for-an-upgraded/
8	microsoft	https://www.newprivatemarkets.com/in-brief-microsoft-invests-in-efm-fund/
9	microsoft	https://au.finance.yahoo.com/news/microsoft-build-2025-watch-years-155150422.html
10	microsoft	https://evrimagaci.org/tpg/microsoft-teams-unveils-major-updates-enhancing-user-experience-349287
11	microsoft	https://www.csoonline.com/article/3979073/how-to-capture-forensic-evidence-for-microsoft-365.html
12	microsoft	https://carboncredits.com/microsoft-secures-3-million-carbon-removal-credits-for-sustainability/
13	microsoft	https://www.aseanbriefing.com/news/microsoft-to-launch-data-centers-in-malaysia-in-q2-2025/
14	microsoft	https://www.uctoday.com/unified-communications/big-uc-news-from-microsoft-salesforce-ringcentral-and-asana/
15	microsoft	https://inshorts.com/en/news/microsoft-bans-employees-from-using-deepseek-due-to-data-breaches-1746784773219
16	microsoft	https://www.hrkatha.com/news/microsoft-bans-use-of-deepseek-app-by-employees-over-data-security-concerns/
17	microsoft	https://dig.watch/updates/microsoft-bans-deepseek-app-for-staff-use
18	microsoft	https://www.newcomer.co/p/openais-simmering-microsoft-battle
19	microsoft	https://techpoint.africa/guide/meta-ai-vs-microsoft-copilot-10-prompt-test-winner/
20	microsoft	https://www.geekwire.com/2025/tech-moves-microsoft-ai-adds-cvp-watchguard-ceo-steps-down-and-more/
21	microsoft	https://theinventory.com/microsoft-xbox-series-s-512gb-ssd-console-now-24-off-1851779892

(Figure 6.3: Output Excel with Links)

The output Excel file contained:

- **Company ID**
- **Final News URL**

All outputs were validated against the Ingestion Allocation sheet, which assigned companies and tracked news updates across the 8-member manual team. The Excel files from both scrapers were seamlessly integrated into this existing format, eliminating the need for rework.

C. Performance Metrics

The modules demonstrated consistent performance under a variety of workloads:

- **Scraping Speed**:
    - Google News scraper: ~**30–40 seconds per company** for 5 pages.
    - Official website scraper: ~**10–15 seconds per company**, depending on site complexity.
- **Batch Processing**:
    - For 100 companies, **800–1,000 cleaned links** were typically generated.
    - The system maintained stability even during larger runs of **400+ companies**, with a total execution time of around **1.5 hours**, plus **30 minutes for human validation**.

D. Stability and Error Handling

While the system was not immune to occasional interruptions (such as CAPTCHA challenges or timeouts), the use of **undetected-chromedriver** provided strong resilience against Google's bot detection mechanisms. Additionally:

- **Relative links** from official websites were successfully normalized using urllib.parse.urljoin.
- Sites that failed to load or had no recent news were logged with descriptive messages like "news path not available" or "no news found," aiding transparency.

E. Usability and Validation

The Excel outputs were not only clean but also **easily reviewable** by the ingestion team using **conditional formatting** to highlight:

- Duplicate URLs (marked in red)
- Empty date fields (highlighted in yellow)
- Companies with zero news (flagged for review)

This allowed the team to quickly vet the quality of scraped data without the need for specialized tools or technical expertise.

6.3 Scalability and Accuracy

The web scraping modules developed during this internship were not only designed for one-off runs or limited testing, but with a focus on **scalable architecture** and **long-term usability** across the Credit-X platform. Both the Google News scraper and the official website scraper were rigorously tested and optimized to ensure they could scale from a few test companies to hundreds of real-world corporate targets without major performance degradation.

A. Scalability by Design

The architecture of both scrapers prioritized modularity, configurability, and extensibility:

- **Modular Codebase**: Each scraper was built as a set of functional modules — URL loader, scraper core, date filter, domain filter, and Excel exporter — allowing individual components to be updated or extended independently.
- **Command-Line Configurability**: Parameters such as date range, company ID, and number of pages could be passed via command-line arguments. This made it easy to customize the execution for different use cases or company sets.
- **Dynamic Input Handling**: Both scrapers accepted company identifiers dynamically and mapped them to URLs using the official.xlsx file, which allowed for flexible and scalable batch runs without needing hardcoded paths or logic.
- **Output Format Compatibility**: The final Excel outputs adhered to the layout used by the existing ingestion team. This minimized integration friction and enabled immediate validation, review, and downstream use without manual conversion.

B. Batch Execution and Speed

The system demonstrated strong scalability in terms of volume processing:

- In live testing, the scrapers could handle **400 companies in a single session**, completing within **1.5 hours**, followed by **30 minutes of manual validation**.
- This represented a **10x speed improvement** over the manual method, where each of the 8 ingestion team members handled around **150–180 companies in 8 hours**.
- Even larger datasets (up to 1,100 companies) were theoretically supported through splitting the runs into batches, enabled by the modular command-line system and persistent logging that tracked incomplete or failed company runs.

C. Accuracy in Data Collection

Accuracy was a critical requirement, especially because Credit-X deals with credit risk intelligence — where even a single wrong article can affect assessment models or investor decisions. The following features enhanced the accuracy of the solution:

- **Domain-Level Filtering** (Google News): The scraper excluded over **180 high-volume, low-signal domains**, reducing noise from generic news sites and focusing on niche or regional sources with higher signal-to-noise ratios.

- **Date Filtering and Normalization**: Using Selenium's automation of Google's filter-by-date tools, combined with custom logic on official sites, the scrapers consistently returned only the **most recent and relevant** links.
- **Publication Date Extraction via htmldate**: For company websites with highly inconsistent HTML structures, the htmldate library ensured high precision in extracting accurate publication dates — reducing false positives caused by embedded historical content or blog navigation elements.

D. Validation and Feedback Loop

- The results were **manually reviewed** by two intern analysts, **Tanmay** and **Jagriti**, who verified accuracy and domain relevance.
- **Excel conditional formatting** was used to quickly identify duplicates, empty date fields, or suspicious links. This method, although simple, proved highly effective and scalable for batch validation.
- Feedback from this review process led to iterative improvements in logic — for example, expanding the domain blacklist, improving base URL mapping in official.xlsx, and fine-tuning pagination logic.

6.4 SWOT Analysis

To comprehensively evaluate the robustness, potential, and limitations of the automated news ingestion system developed during this internship, a detailed **SWOT (Strengths, Weaknesses, Opportunities, and Threats)** analysis was conducted. This analysis outlines the internal and external factors influencing the system's performance and its long-term viability as a critical part of the Credit-X platform's data infrastructure.

Strengths

1. **Automation of a Previously Manual Workflow**
    - The most significant strength lies in the **complete automation** of a labor-intensive process previously handled by an 8-member team.
    - Manual efforts requiring over **60 human hours daily** were replaced by a Python-based system running in **under 2 hours**, resulting in over **95% reduction** in manual work.
2. **10x Improvement in Speed and Efficiency**
    - The system demonstrated the capability to process 400 companies in approximately 1.5 hours — a task that took multiple hours by humans.
    - This level of speed ensures that Credit-X receives faster updates, keeping the platform timely and competitive.

3. **Enhanced Accuracy through Rule-Based and Date-Based Filtering**
    - Incorporation of libraries like htmldate, structured HTML parsing, and domain-level whitelisting/blacklisting improved the **precision** of extracted links.
    - Real news was extracted while spam, outdated, and irrelevant content was filtered out — boosting the **signal-to-noise ratio**.
4. **Structured and Compatible Output Format**
    - The use of Excel outputs aligned with the format used by the ingestion team.
    - This allowed for **seamless integration** into the validation process, enabling non-technical analysts to assess and use the output without additional conversion.
5. **Flexibility in Input and Scalability**
    - Company IDs and date ranges were parameterized via CLI arguments, allowing flexible batch processing.
    - New companies could be added easily by updating the official.xlsx file without modifying the scraper logic.
6. **Domain Filtering for Improved Content Quality**
    - In the Google News scraper, a domain blacklist filtered out over 180 mainstream domains like “reuters.com,” “bloomberg.com,” etc., prioritizing niche and regional sources more relevant to credit risk.

Weaknesses

1. **No Graphical Interface (GUI)**
    - The entire system operates via command-line, which can be a **barrier for non-technical users**.
    - Lack of a user-friendly interface restricts adoption beyond data engineers or analysts with Python experience.
2. **Dependence on Static HTML Structure**
    - The scrapers rely on static HTML patterns. Any change in website structure (especially for official websites) can **break the extraction logic**.
    - There is **no adaptive learning** or fallback mechanism for unexpected page structures.

3. **Limited Handling of JavaScript-Heavy Sites**
    - Some news sections or press release portals use client-side rendering or dynamic content loading via JavaScript.
    - These are **not fully compatible** with the current scraping setup, which depends on initial DOM loading.
4. **No Retry Logic or Error Recovery**
    - Although failures are logged, **failed company scrapes are not retried automatically**.
    - This limits robustness during long unattended runs, especially if internet issues or page timeouts occur.
5. **No Real-Time Feedback or Monitoring Tools**
    - The current system lacks a dashboard to visualize scraping progress, link counts, or status reports in real-time.
    - Monitoring is entirely dependent on terminal outputs or Excel logs.
6. **Duplicate Handling Left to Post-Processing**
    - While conditional formatting in Excel effectively identifies duplicates, **the scraper itself does not perform de-duplication** or persistent storage comparison.

Opportunities

1. **Addition of a GUI for Broader Accessibility**
    - Introducing a simple desktop or web-based GUI would **enable broader adoption** by non-technical stakeholders such as risk analysts, managers, and business users.
2. **Integration of NLP and Article Summarization**
    - Implementing **Natural Language Processing (NLP)** techniques such as keyword extraction, sentiment analysis, or summarization would significantly enhance the value of scraped articles.
    - This could support automatic risk tagging or topic classification.
3. **Scalability via Cloud or Server Deployment**
    - Deploying the scrapers on a **cloud platform** (e.g., AWS, Azure) with scheduled runs and auto-scaling resources would enable continuous scraping across geographies and time zones.

4. **Proxy Rotation and Bot Protection Enhancement**
    - Adding **automatic proxy rotation** and user-agent spoofing can allow scraping of **restricted or bot-protected websites**, increasing the system's coverage and resilience.
5. **Data Visualization and Dashboard Reporting**
    - Integration with tools like **Power BI**, **Grafana**, or **Streamlit** could allow ingestion teams to view scraping metrics, success rates, and timelines in real time.
6. **Historical Archiving and Time-Series Tracking**
    - Storing and indexing scraped links in a historical database can allow **trend analysis**, repeated query handling, or anomaly detection in company news patterns.

Threats

1. **Anti-Scraping Measures and Legal Risks**
    - Target websites, especially official corporate portals, may introduce **aggressive anti-bot techniques** such as CAPTCHA upgrades, IP bans, or session-based rendering.
    - In some jurisdictions, scraping may also fall into a **legal gray area**, especially when content usage is not clearly defined.
2. **Reliance on Web Structure Consistency**
    - Any update in site layout, especially in official company news pages, may render the current logic ineffective, requiring **constant maintenance**.
3. **Single Point of Failure in Local Deployment**
    - The scrapers are deployed locally and manually executed. Any system failure, power issue, or missed execution may cause **entire data windows to be lost**.
4. **Over-Reliance on Automation Without Oversight**
    - As automation increases, there is a risk of over-reliance without **adequate validation**. Minor bugs may go unnoticed and propagate incorrect data downstream if not checked.
5. **Scalability Constraints on Local Systems**

- While the system is scalable in architecture, hardware limitations on local machines (memory, CPU, browser sessions) **cap limit** processing capacity in larger runs.

6.5 Conclusion

The internship project undertaken at **Galytix** successfully achieved its core objective of automating the financial news ingestion process for the **Credit-X platform**. Through the development and deployment of two purpose-built web scraping modules—**Google News Scraper** and **Official Website Scraper**—the system replaced a laborious, manual workflow that previously required a team of eight analysts working full-time. The automation led to a **tenfold improvement in processing speed**, handling **400 companies in just 1.5 hours**, with an additional 30 minutes allocated for validation, as opposed to the manual system which took 8 hours per person.

Both modules were built using **Python** and made use of libraries like Selenium, pandas, htmldate, datetime, and undetected_chromedriver. The Google News Scraper was designed to crawl and filter news articles using keyword-based queries, pagination, and domain-level filtering to prioritize niche sources. In contrast, the Official Website Scraper extracted content directly from company-specific press release or media pages, using URL normalization and publication date extraction for relevance scoring. Together, these tools ensured a higher degree of **data relevance**, **coverage breadth**, and **timeliness**.

The system outputs were structured in Excel format, ensuring compatibility with the Ingestion Team's existing workflows. Conditional formatting was used to highlight duplicates, and the results were manually verified by intern analysts **Tanmay** and **Jagriti**. Despite operating without version control or server deployment, the solution demonstrated **robust local execution**, minimal failure rates, and adaptability across multiple company domains.

In terms of broader organizational impact, this automation project eliminated approximately **95% of the manual workload**, freeing up human resources for higher-value tasks like data validation and insight generation. It also enabled **scalable ingestion**, offering the potential to track news for **1,000+ companies** per day with minimal human intervention. The success of the project validates the technical feasibility of using browser automation and structured HTML parsing for real-time financial data ingestion.

However, the system is not without its limitations. It depends on the stability of website structures, lacks a graphical user interface, and currently offers no NLP-driven news categorization. Moreover, it does not yet incorporate real-time monitoring or automatic failure recovery mechanisms. These limitations offer fertile ground for future enhancements.

Future directions include:

- Adding a user-friendly **GUI** to widen adoption.
- Deploying the scrapers on **cloud infrastructure** for continuous, scalable execution.
- Integrating **Natural Language Processing** (NLP) for article summarization and topic tagging.
- Implementing **real-time dashboards** for progress tracking and error alerts.
- Enhancing the system's resilience with **proxy rotation**, **auto-retry logic**, and **JavaScript-rendering capabilities**.

In conclusion, this project marked a significant step toward building a **smarter, faster, and more scalable data pipeline** for Credit-X. It exemplified how well-designed automation can not only streamline operations but also improve accuracy, timeliness, and organizational agility in managing complex data at scale. The learnings and foundations laid by this project serve as a robust starting point for future innovations in financial data automation.

Chapter 7
FUTURE WORK AND RECOMMENDATIONS

The future development of the automated news ingestion system focuses on enhancing functionality, user experience, and robustness. These planned improvements aim to transform the prototype into a scalable, reliable platform suitable for broader organizational use and eventual public deployment. Below is a detailed description of each planned enhancement, along with the motivations and technical considerations involved.

7.1 Future Work

1. Integration of Official Website Scraper

Currently, the web interface supports only Google News-based scraping, which relies on third-party aggregation and may introduce delays or missing content. Integrating the official website scraper addresses these limitations by enabling direct retrieval of press releases and company announcements from authoritative sources.

Motivation:
Official websites are primary sources for authentic information, often releasing updates before news aggregators detect them. Accessing these sources reduces dependency on external providers, increases data accuracy, and enriches the dataset with verified content that can significantly benefit credit risk analysts relying on timely news.

Implementation Details:

- **User Input:** Users will upload a structured file, typically the existing official.xlsx, containing each company's base URL or specific news section URLs.
- **Validation:** The system must check if the selected company exists in the uploaded file and verify the URL's availability through HTTP status codes or lightweight connectivity tests.
- **Dynamic Scraping Selection:** Based on user choice or URL availability, the system routes the request to either the Google News scraper or the official website scraper.
- **Backend Adjustments:**
    - Implement secure file upload mechanisms that check file size, content type, and schema validation to prevent errors and security risks.
    - Extend the scraping API to accept multiple data sources and return a unified response format.

- Introduce error handling and fallback mechanisms in case the official site is unreachable.

Expected Benefits:

- Improved credibility and freshness of news data.
- Reduced risk of missing critical updates due to aggregator delays or filtering.
- Enriched datasets that combine third-party news and original press releases, enhancing the analytical value.

2. Support for Multiple Export Formats

Currently, the system outputs scraped data solely in Excel (.xlsx) format. While Excel is widely used, this limitation restricts the tool's flexibility and compatibility with diverse workflows and systems.

Motivation:
Different users and systems require data in various formats for seamless integration. For example, data scientists often prefer CSV for simplicity, whereas developers or automated pipelines may prefer JSON for hierarchical data representation.

Implementation Details:

- **Formats to Support:**
  - **CSV:** Plain-text format with comma-separated values, easy for scripting, data manipulation, and quick loading in many applications.
  - **JSON:** Hierarchical format supporting nested structures, suitable for APIs, NoSQL databases, or complex metadata.
- **Preview Feature:**
  - The interface will allow users to preview the scraped URLs, displaying columns such as company ID, source link, publication date, and source type (Google News or official website).
  - Users can filter or deselect specific entries to customize export content before downloading.
- **Export Selection:** Users can choose their preferred export format dynamically through the frontend interface.

Expected Benefits:

- Greater flexibility in downstream processing and analysis.

- Reduced post-processing workload for users who would otherwise convert formats manually.
- Improved user experience by enabling data validation and selection prior to export.

3. User Authentication and Access Control

As the system evolves into a public or team-accessible tool, securing access and managing permissions become crucial to protect data integrity and prevent misuse.

Motivation:
Open access to scraping functionalities risks server overload, unauthorized use, or exposure of sensitive data. Implementing robust security controls is essential for maintaining operational stability and compliance with organizational policies.

Implementation Details:

- **Authentication:**
    - Implement secure sign-up and login flows using best practices such as password hashing (bcrypt) and session management to protect user credentials.
    - Incorporate email verification to confirm user identities and reduce spam registrations.
- **Role-Based Access Control (RBAC):**
    - Define roles such as regular users (able to scrape and export) and administrators (manage users, files, and settings).
    - Implement permission checks at API and frontend levels to enforce access policies.
- **Secure File Uploads:**
    - Validate MIME types and sanitize file names to prevent malicious uploads.
    - Limit upload size and process files in memory when possible to avoid storage risks.
- **Audit Logging:**
    - Track user actions such as logins, exports, uploads, and scraping jobs for monitoring and forensic analysis.
    - Logs help detect unusual activities and facilitate compliance audits.

Expected Benefits:

- Enhanced system security and user trust.
- Prevention of abuse and resource exhaustion.
- Capability to support multi-user environments and organizational deployment.

4. Frontend Enhancements

A user-friendly and responsive frontend is critical for user adoption, especially among non-technical users. Planned improvements focus on usability, clarity, and interaction feedback.

Motivation:
Ease of use reduces errors, increases productivity, and broadens the potential user base beyond technical teams.

Implementation Details:

- **Autocomplete & Dropdowns:**
    - Implement type-ahead search and dropdown lists for company selection based on the available dataset. This reduces manual typing errors and speeds up user input.
- **Progress Indicators & Real-Time Status:**
    - Show progress bars and current task updates during scraping jobs to keep users informed about system activity and expected completion time.
- **In-Browser Link Previews:**
    - Display extracted URLs with metadata such as headline snippets or publication dates, enabling quick content validation without downloading files.
- **Download History Panel:**
    - Maintain a session or persistent history of scraping activities for each user, allowing easy retrieval and re-download of past results along with their input parameters.
- **Responsive Design:**
    - Optimize UI layout for various screen sizes and devices, including desktops, tablets, and smartphones, ensuring accessibility and convenience.

Expected Benefits:

- Improved user satisfaction and trust in the system.
- Reduced error rates through guided input and real-time feedback.
- Increased productivity by simplifying repeated tasks and data validation.

5. Scheduled and Background Scraping

Currently, scraping requires manual initiation and monitoring, limiting the system's efficiency and scalability. Introducing scheduled and background scraping capabilities will automate regular monitoring tasks and optimize resource usage.

Motivation:
Credit analysts and other users need continuous updates without manual overhead. Automation enhances timeliness and reduces operational workload.

Implementation Details:

- **Automated Background Jobs:**
    - Integrate task queues such as Celery with Redis or use system cron jobs to run scraping asynchronously in the background.
- **User-Defined Schedules:**
    - Allow users to configure scraping intervals per company or batch, with options such as daily, weekly, or custom frequencies.
- **Preselected Company Lists:**
    - Enable users to save and manage groups of companies for regular monitoring, improving workflow efficiency.
- **Notifications:**
    - Provide email or in-app notifications when scraping jobs complete, including summaries and download links.
- **Result Storage:**
    - Store scraping outputs in databases or cloud storage to support historical comparisons and audits.
- **Job Monitoring & Retry:**
    - Develop admin dashboards for tracking job statuses, errors, and to rerun failed tasks, ensuring reliability.

Expected Benefits:

- Continuous and consistent data ingestion with minimal manual intervention.
- Improved system reliability and user engagement through timely notifications.
- Easier management of large-scale monitoring projects.

6 Cloud or LAN Deployment

The current local-only deployment limits accessibility and scalability. Exploring deployment options across LAN or cloud infrastructure will enable broader usage and higher availability.

Motivation:
Making the system accessible to teams or remotely increases collaboration and operational efficiency. Cloud deployment offers scalability, failover, and maintenance advantages.

Implementation Details:

- **LAN Deployment:**
    - Enable access within an organization's local network, allowing team members to use the system without public exposure.
    - Provide simple installation scripts or Docker images to facilitate easy setup.
- **Cloud Deployment:**
    - Explore deployment on cloud platforms such as AWS, Render, or Railway for scalability, automatic backups, and high uptime.
    - Address challenges such as cost, security, and data privacy through appropriate configurations.

Expected Benefits:

- Greater accessibility for team-based workflows.
- Improved reliability, uptime, and resource management.
- Foundation for future growth and integration with other cloud services.

7.2 Recommendations

Alongside the planned technical enhancements, several strategic and operational recommendations are essential for ensuring the long-term success, scalability, and maintainability of the automated news ingestion platform. These recommendations

address data management, quality control, system observability, and advanced features that align with industry best practices.

1. Central Repository for Official Sources

Maintaining an accurate and up-to-date collection of official company URLs is fundamental for the reliability of the official website scraper. Currently, this information is managed manually via Excel sheets, which is prone to inconsistencies, duplication, and version control issues.

Rationale:
A centralized, collaborative repository will streamline updates, reduce manual errors, and improve data integrity. It will also support multiple stakeholders who contribute to maintaining the source list, ensuring that the system always operates on the latest and verified information.

Implementation Suggestions:

- **Collaborative Platform:**
    - Use version-controlled platforms such as GitHub or GitLab to store the URL lists as structured files (e.g., CSV, JSON, or Excel). These platforms allow controlled contributions, branching, pull requests, and audit trails.
    - Alternatively, use Google Sheets API to maintain a live, cloud-hosted sheet with role-based editing permissions.
- **Submission and Review Workflow:**
    - Allow users or admins to submit new entries or edits through a controlled interface.
    - Implement a review or approval process to validate new URLs before they become active.
- **Conflict and Duplication Management:**
    - Programmatically detect conflicts such as duplicate URLs or overlapping domains and alert maintainers for resolution.
    - Use unique identifiers for companies and standardize URL formats to reduce ambiguities.
- **Automated Validation:**
    - Schedule periodic health checks (e.g., URL availability, redirects) to keep the repository clean and up to date.

Expected Impact:

- Consistent, reliable source data that enhances scraper effectiveness.
- Reduced manual overhead and data errors.
- Transparency and traceability of changes benefiting audit and compliance requirements.

2. Enhanced Domain Ranking for Google News

Currently, the system relies on manually curated whitelists or blacklists to determine which domains should be prioritized or excluded during Google News scraping. This approach is static and may miss emerging reputable sources or include low-quality domains.

Rationale:
Dynamic domain ranking based on traffic, reputation, or community feedback will improve the quality of news sources, ensuring that relevant and authoritative domains are prioritized while minimizing noise.

Implementation Suggestions:

- **Traffic Estimation APIs:**
    - Use free or freemium APIs like SimilarWeb, Alexa (if still available), or other web analytics providers to estimate domain traffic and engagement metrics.
    - Rate-limit API calls to manage costs and avoid throttling.
- **Community Tagging and Voting:**
    - Implement a feature where users can tag domains as reliable, suspicious, or irrelevant, contributing to a crowd-sourced domain reputation score.
    - Aggregate votes to dynamically update the whitelist or blacklist.
- **Automated Reputation Scoring:**
    - Combine traffic data, domain age, SSL certification status, and historical scraping success rates into a composite score.
    - Adjust domain ranking algorithms periodically based on these scores.

Expected Impact:

- Increased precision in news source selection, reducing false positives or spam links.
- Adaptability to the evolving news ecosystem without manual intervention.

- Empowerment of user community to contribute to source quality assurance.

3. Logging and Monitoring

Robust logging and monitoring are critical for maintaining system health, diagnosing issues, and improving user trust. Current implementations may log basic scraping outputs but often lack detailed observability.

Rationale:
Detailed logs and monitoring dashboards provide transparency into system operations, facilitate troubleshooting, and support proactive maintenance.

Implementation Suggestions:

- **Detailed Logging:**
    - Log scraping session metadata including start and end times, duration, errors encountered, number of links fetched, and data volumes.
    - Include user context to correlate logs with specific user sessions or scraping requests.
- **Centralized Log Management:**
    - Use logging frameworks that support centralized aggregation (e.g., ELK stack: Elasticsearch, Logstash, Kibana) or cloud services like AWS CloudWatch or Azure Monitor.
- **Monitoring Dashboards:**
    - Develop dashboards to visualize scraping success rates, error trends, latency, and resource utilization.
    - Include alerting mechanisms for failures or abnormal patterns.
- **Audit Trails:**
    - Maintain comprehensive audit trails to support compliance with organizational policies or regulations.

Expected Impact:

- Faster detection and resolution of scraping failures or bottlenecks.
- Data-driven insights enabling continuous improvement.
- Increased confidence for users and administrators through transparency.

4. AI/ML-Based News Relevance Filtering (Advanced)

As the volume of scraped news grows, filtering irrelevant or low-quality articles becomes increasingly important. Basic keyword filters may be insufficient to capture nuanced relevance or emerging topics.

Rationale:
Incorporating machine learning and natural language processing techniques can significantly enhance the precision of news filtering, reducing noise and improving the signal for analysts.

Implementation Suggestions:

- **Classifier Models:**
    - Develop or fine-tune models (e.g., BERT, RoBERTa) to classify articles based on relevance to specific companies, sectors, or credit risk factors.
    - Use keyword matching, topic modeling, and semantic similarity measures as features.
- **User Feedback Loop:**
    - Allow users to flag irrelevant or erroneous articles, creating labeled data to continuously train and improve the models.
- **Hybrid Filtering:**
    - Combine rule-based filters with ML models for a balanced approach that leverages domain knowledge and data-driven insights.
- **Scalability Considerations:**
    - Design inference pipelines to handle large volumes efficiently, possibly through batch processing or cloud-based AI services.

Expected Impact:

- Improved quality and relevance of ingested news, reducing manual validation effort.
- Enhanced user satisfaction and analytical value.
- Foundation for advanced analytics such as sentiment analysis or event prediction.

2. Enhanced Domain Ranking for Google News: Instead of relying on manual domain whitelisting/blacklisting:

- Implement lightweight traffic estimation using free APIs (e.g., SimilarWeb) with rate limits.
- Introduce community tagging or voting mechanisms to improve domain reputation dynamically.

3. Logging and Monitoring: Improve maintainability and transparency by:

- Logging detailed scraping session data (time taken, errors, number of links fetched).
- Providing a monitoring dashboard to visualize scraping success rates and failures.

4. AI/ML-Based News Relevance Filtering (Advanced): To improve output quality in future versions:

- Implement classifiers (using keyword matching, embeddings, or fine-tuned models like BERT) to filter irrelevant articles.
- Allow users to flag irrelevant results to improve model training datasets.

7.3 Final Thoughts

The development of the automated news ingestion platform marks a significant milestone in transforming manual, labor-intensive processes into a scalable, efficient, and reliable system. With a strong backend foundation built on robust scraping scripts and an evolving web interface aimed at enhancing user experience, the platform is well-positioned to grow beyond its initial scope and serve broader organizational needs.

A key strength of the system lies in its modular and extensible architecture. By decoupling core scraping logic from the user interface and data storage layers, the platform facilitates continuous improvement and integration of new data sources and features without disrupting existing workflows. This design principle also simplifies maintenance, enabling rapid adaptation to changes in source websites, evolving user requirements, and technological advancements.

User-centric design is paramount as the tool moves toward public accessibility. Intuitive interfaces, clear feedback mechanisms, and comprehensive security controls will drive adoption among diverse stakeholders—ranging from data analysts and credit risk teams to business decision-makers—who may have varying levels of technical expertise. Prioritizing accessibility through responsive design and clear workflows ensures the platform remains inclusive and practical for day-to-day operational use.

Maintaining data integrity and trustworthiness is equally critical. Incorporating mechanisms such as centralized repositories for official sources, rigorous validation routines, and advanced filtering techniques will safeguard the quality and relevance of

ingested news. This focus on data quality will directly influence the credibility and usefulness of downstream analytics and decision-making processes.

Looking ahead, embracing emerging technologies such as machine learning for relevance filtering, cloud-based deployment for scalability, and automation of routine maintenance tasks will further enhance the platform's capabilities. Continuous iteration informed by user feedback and performance metrics will be essential to keep pace with the dynamic landscape of news sources and organizational needs.

In conclusion, the journey from a locally run prototype to a mature, scalable web application embodies both technical innovation and strategic foresight. By balancing functional enhancements with strong governance and user engagement, the platform promises to become an indispensable asset for timely, reliable news ingestion—empowering the organization to make informed, data-driven decisions with greater speed and confidence.

Chapter 8
REFERENCES

[1] A. D. Bar, “htmldate: A Python Package for Extracting the Publication Date of Web Documents,” GitHub repository, 2022. [Online]. Available: https://github.com/adbar/htmldate

[2] SeleniumHQ, “Selenium WebDriver,” [Online]. Available: https://www.selenium.dev/documentation/webdriver/

[3] Pandas Development Team, “pandas: Python Data Analysis Library,” [Online]. Available: https://pandas.pydata.org/

[4] Python Software Foundation, “urllib — URL handling modules,” Python Documentation, [Online]. Available: https://docs.python.org/3/library/urllib.html

[5] Microsoft, “Overview of Excel,” Microsoft Support, [Online]. Available: https://support.microsoft.com/en-us/excel

[6] GCFGlobal, “Excel Basics,” GCFLearnFree.org, [Online]. Available: https://edu.gcfglobal.org/en/excel/

[7] Listly, “Listly: The Social List Making App,” [Online]. Available: https://list.ly

[8] Octoparse, “Octoparse Web Scraping Tool,” [Online]. Available: https://www.octoparse.com

[9] Docker, “Docker: Empowering App Development for Developers,” [Online]. Available: https://www.docker.com

Chapter 9
APPENDIX

Appendix A: Official.xlsx File

The official.xlsx file served as a critical reference input for the Official Website Scraper. It provided a structured mapping between each target company and its corresponding official news or press release section. This file was essential for ensuring the scraper accessed reliable and domain-authenticated content relevant to each company.

Structure of the File

The Excel file consisted of the following three columns:

Column Name	Description
company_id	A unique, normalized identifier for each company, used as input to the scraper.
website_url	The public-facing URL of the company's official website
base_url	The news path that the scraper follows , to get the news releases from company website .

(Table 9.1: Structure of Official.xlsx)

	A	B	C
1	company_id	website_url	base_url
2	3m_company	https://investors.3m.com	https://investors.3m.com
3	a_o_smith_corporation	https://www.aosmith.com	https://investor.aosmith.com
4	abbott_laboratories	https://www.abbott.com	https://abbott.mediaroom.com
5	abbvie	https://www.abbvie.com	https://news.abbvie.com
6	accenture_plc	https://www.accenture.com	https://newsroom.accenture.com
7	adobe_inc	https://www.adobe.com	https://blog.adobe.com
8	adt_inc	https://www.adt.com	https://investor.adt.com
9	advanced_micro_devices_inc	https://www.amd.com	https://www.amd.com
10	aes_corporation	https://www.aes.com	https://aescorporation2023cr.q4web.com
11	afc_gamma	https://advancedflowercapital.com	https://investors.afcgamma.com
12	affirm_holdings_inc	https://investors.affirm.com	https://investors.affirm.com
13	aflac_incorporated	https://www.aflac.com	https://newsroom.aflac.com
14	agilent_technologies_inc	https://www.investor.agilent.com	https://www.investor.agilent.com
15	agnico_eagle_mines_limited	https://www.agnicoeagle.com	https://www.agnicoeagle.com
16	air_products_and_chemicals_inc	https://investors.airbnb.com	https://www.airproducts.com
17	airbnb_inc	https://investors.airbnb.com	https://news.airbnb.com
18	akamai_technologies_inc	https://www.akamai.com	https://www.ir.akamai.com
19	albemarle_corporation	https://www.albemarle.com	https://investors.albemarle.com
20	alexandria_real_estate_equities_inc	https://www.are.com	https://investor.are.com
21	align_technology_inc	https://www.aligntech.com	https://investor.aligntech.com

(Figure 9.1: Screenshot of Official.xlsx)

Usage in the Scraper

- Only those company entries from official.xlsx whose company_id matched the command-line arguments were processed during any scraping run.
- The base_url column was used to filter and validate links extracted from the target website. Only links that originated from or matched this base domain were retained for further processing.
- If a company_id had no corresponding or valid URL entry in the Excel file, the scraper skipped it and logged the event accordingly.

Creation Process

The official.xlsx file was manually compiled over the span of three weeks by the entire 8-member ingestion team. Each team member was responsible for researching a subset of companies, locating their official homepage, and identifying the specific URL paths that linked to press releases, newsroom updates, or corporate blog sections.

This process involved:

- Verifying the authenticity of websites to ensure they were owned or operated by the official company.
- Manually inspecting and recording the homepage and specific news-related subpages.
- Standardizing URL formats and populating the base_url field to facilitate automated domain validation during scraping.

Given the diversity in company websites and the lack of a universal structure, the task required attention to detail and cross-verification. Despite the manual effort involved, the team successfully compiled a high-quality and comprehensive mapping of over 400 companies within three weeks.

Appendix B: Final Output Excel File

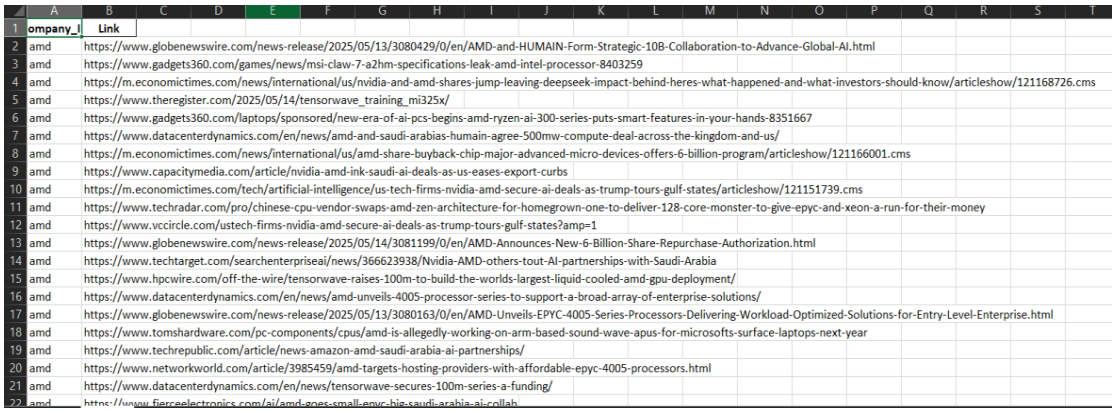
The final output generated by the Official Website Scraper is structured in Excel format and serves as a key input to the Credit-X knowledge ingestion pipeline. This file captures the most relevant and validated news articles published by companies on their official websites, along with associated metadata necessary for downstream processing.

Structure of the Output File

Each row in the Excel file represents a single news article extracted during the scraping process. The file contains the following columns:

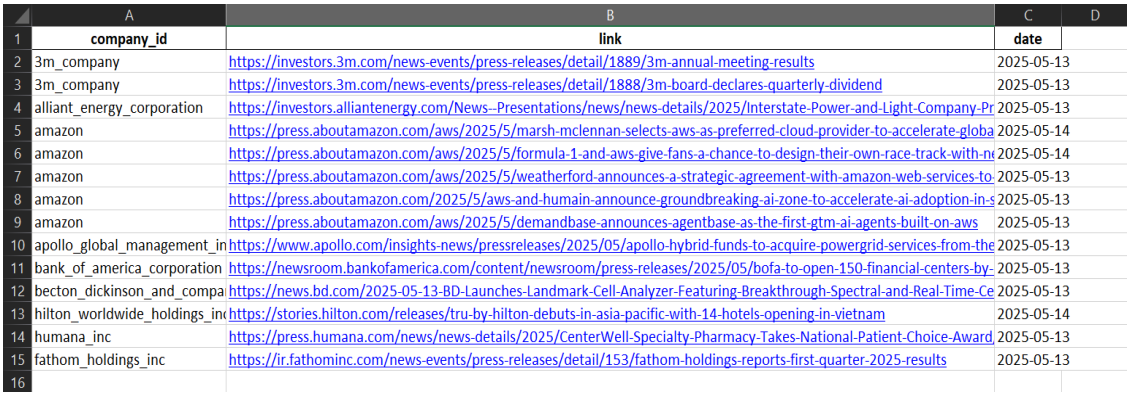
Column Name	Description
company_id	The unique identifier of the company for which the news article was extracted.
publication_date	The inferred publication date of the article in YYYY-MM-DD format.
news_url	The direct URL link to the original article hosted on the company's website.

(Table 9.2:Items in Excel output)



	A	B
1	ompany_I	Link
2	amd	https://www.globenewswire.com/news-release/2025/05/13/3080429/0/en/AMD-and-HUMAIN-Form-Strategic-10B-Collaboration-to-Advance-Global-AI.html
3	amd	https://www.gadgets360.com/games/news/msi-claw-7-a2hm-specifications-leak-amd-intel-processor-8403259
4	amd	https://m.economictimes.com/news/international/us/nvidia-and-amd-shares-jump-leaving-deepseek-impact-behind-heres-what-happened-and-what-investors-should-know/articleshow/121168726.cms
5	amd	https://www.theregister.com/2025/05/14/tensorwave_training_mi325x/
6	amd	https://www.gadgets360.com/laptops/sponsored/new-era-of-ai-pcs-begins-amd-ryzen-ai-300-series-puts-smart-features-in-your-hands-8351667
7	amd	https://www.datacenterdynamics.com/en/news/amd-and-saudi-arabias-humain-agree-500mw-compute-deal-across-the-kingdom-and-us/
8	amd	https://m.economictimes.com/news/international/us/amd-share-buyback-chip-major-advanced-micro-devices-offers-6-billion-program/articleshow/121166001.cms
9	amd	https://www.capacitymedia.com/article/nvidia-amd-ink-saudi-ai-deals-as-us-eases-export-curbs
10	amd	https://m.economictimes.com/tech/artificial-intelligence/us-tech-firms-nvidia-amd-secure-ai-deals-as-trump-tours-gulf-states/articleshow/121151739.cms
11	amd	https://www.techradar.com/pro/chinese-cpu-vendor-swaps-amd-zen-architecture-for-homegrown-one-to-deliver-128-core-monster-to-give-epyc-and-xeon-a-run-for-their-money
12	amd	https://www.vccircle.com/ustech-firms-nvidia-amd-secure-ai-deals-as-trump-tours-gulf-states?amp=1
13	amd	https://www.globenewswire.com/news-release/2025/05/14/3081199/0/en/AMD-Announces-New-6-Billion-Share-Repurchase-Authorization.html
14	amd	https://www.techtarget.com/searchenterpriseai/news/366623938/Nvidia-AMD-others-tout-AI-partnerships-with-Saudi-Arabia
15	amd	https://www.hpcwire.com/off-the-wire/tensorwave-raises-100m-to-build-the-worlds-largest-liquid-cooled-amd-gpu-deployment/
16	amd	https://www.datacenterdynamics.com/en/news/amd-unveils-4005-processor-series-to-support-a-broad-array-of-enterprise-solutions/
17	amd	https://www.globenewswire.com/news-release/2025/05/13/3080163/0/en/AMD-Unveils-EPYC-4005-Series-Processors-Delivering-Workload-Optimized-Solutions-for-Entry-Level-Enterprise.html
18	amd	https://www.tomshardware.com/pc-components/cpus/amd-is-allegedly-working-on-arm-based-sound-wave-apus-for-microsofts-surface-laptops-next-year
19	amd	https://www.techrepublic.com/article/news-amazon-amd-saudi-arabia-ai-partnerships/
20	amd	https://www.networkworld.com/article/3985459/amd-targets-hosting-providers-with-affordable-epyc-4005-processors.html
21	amd	https://www.datacenterdynamics.com/en/news/tensorwave-secures-100m-series-a-funding/
22	amd	https://www.fierceelectronics.com/ai/amd-goes-small-epyc-big-saudi-arabia-ai-collab

(Figure 9.2: Screenshot of Google News Scraper Excel output)



	A	B	C	D
1	company_id	link	date	
2	3m_company	https://investors.3m.com/news-events/press-releases/detail/1889/3m-annual-meeting-results	2025-05-13	
3	3m_company	https://investors.3m.com/news-events/press-releases/detail/1888/3m-board-declares-quarterly-dividend	2025-05-13	
4	alliant_energy_corporation	https://investors.alliantenergy.com/News--Presentations/news/news-details/2025/Interstate-Power-and-Light-Company-Pr	2025-05-13	
5	amazon	https://press.aboutamazon.com/aws/2025/5/marsh-mclennan-selects-aws-as-preferred-cloud-provider-to-accelerate-globa	2025-05-14	
6	amazon	https://press.aboutamazon.com/aws/2025/5/formula-1-and-aws-give-fans-a-chance-to-design-their-own-race-track-with-n	2025-05-14	
7	amazon	https://press.aboutamazon.com/aws/2025/5/weatherford-announces-a-strategic-agreement-with-amazon-web-services-to	2025-05-13	
8	amazon	https://press.aboutamazon.com/2025/5/aws-and-humain-announce-groundbreaking-ai-zone-to-accelerate-ai-adoption-in-s	2025-05-13	
9	amazon	https://press.aboutamazon.com/aws/2025/5/demandbase-announces-agentbase-as-the-first-gtm-ai-agents-built-on-aws	2025-05-13	
10	apollo_global_management_in	https://www.apollo.com/insights-news/pressreleases/2025/05/apollo-hybrid-funds-to-acquire-powergrid-services-from-the	2025-05-13	
11	bank_of_america_corporation	https://newsroom.bankofamerica.com/content/newsroom/press-releases/2025/05/bofa-to-open-150-financial-centers-by-	2025-05-13	
12	becton_dickinson_and_compa	https://news.bd.com/2025-05-13-BD-Launches-Landmark-Cell-Analyzer-Featuring-Breakthrough-Spectral-and-Real-Time-Ce	2025-05-13	
13	hilton_worldwide_holdings_in	https://stories.hilton.com/releases/tru-by-hilton-debuts-in-asia-pacific-with-14-hotels-opening-in-vietnam	2025-05-14	
14	humana_inc	https://press.humana.com/news/news-details/2025/CenterWell-Specialty-Pharmacy-Takes-National-Patient-Choice-Award	2025-05-13	
15	fathom_holdings_inc	https://ir.fathominc.com/news-events/press-releases/detail/153/fathom-holdings-reports-first-quarter-2025-results	2025-05-13	
16				

(Figure 9.3: Screenshot of Official Website Scraper Excel output)

Filename Convention

The output file follows a version-controlled naming scheme. This helps prevent accidental overwrites and supports historical comparison and re-validation. The format is:

scraped_output_with_dates<random_id>.xlsx : for Official Website Scraper

Example:
scraped_output_with_dates_117809.xlsx

filtered_news_links_<timestamp>.xlsx

Example:
filtered_news_links_05-13-2025to05-14-20252025-05-2100-08-39.xlsx : for google news scraper

Key Characteristics

- **Accuracy:** Only URLs verified to belong to the official domain of each company are included.
- **Relevance:** All articles fall within the user-specified date range provided via command-line input.
- **Deduplication:** Duplicate links across pagination levels are automatically removed using set-based comparison.
- **Timestamped Output:** Each file generated is uniquely timestamped, allowing traceability in future analysis or audits.

Use in Downstream Workflow

The Excel file is passed on to:

- **Credit-X Platform:** For automatic ingestion of company-specific news updates.

- **Ingestion Team:** For manual quality checks and deduplication using Excel's conditional formatting tools.
- **Analysts:** For sourcing company events and trends for credit assessment, investment analysis, and risk evaluation.

Appendix C: Major_sources.xlsx File

1. Introduction

In the context of automating news ingestion for the Credit-X platform, it is essential to distinguish between niche, company-specific content and general news from mainstream or syndicated media outlets. To this end, we maintain a comprehensive repository of **Major Sources**—mainstream news websites that are **intentionally excluded** from ingestion or flagged for review.

These sources are typically high-traffic domains with established media coverage, often requiring **explicit permissions or licenses** for commercial use of their content. Furthermore, they generally cover broader financial or global news rather than specific updates from individual companies, especially smaller or less frequently mentioned firms.

2. Purpose of Maintaining Major Sources List

The **Major Sources list** serves multiple purposes in both the development and validation processes of the scrapers:

- **Avoiding Legal and Copyright Issues:** Many mainstream media websites restrict scraping or reuse of their content without appropriate licensing. Excluding them minimizes legal risk.
- **Filtering Out Generic Content:** These sources often focus on macroeconomic trends, general stock market reports, or high-profile corporate announcements. As a result, **niche company-specific developments may be missed**.
- **Improving Relevance of Output:** By filtering out major sources, the scrapers are more likely to capture **direct, focused content** either from a company's official site or smaller financial publications and aggregators that cover specific business updates.
- **Reinforcing the Excel-based Validation Layer:** The list of major sources is embedded into the **Excel domain filtering formula** to automatically flag or remove links from these sites during manual review.

3. Criteria for Inclusion

A domain is classified as a **Major Source** if it satisfies **any** of the following:

- Has **monthly web traffic exceeding 1 million visits**, as determined from publicly available metrics (e.g., SimilarWeb, SEMrush, or internal estimates).
- Operates as a **recognized international or national news outlet** (e.g., BBC, CNBC, Bloomberg, Reuters).

- Consistently **publishes broad-market articles** and syndicated financial news rather than company-specific updates.
- Enforces **content licensing, copyright, or paywall restrictions**.

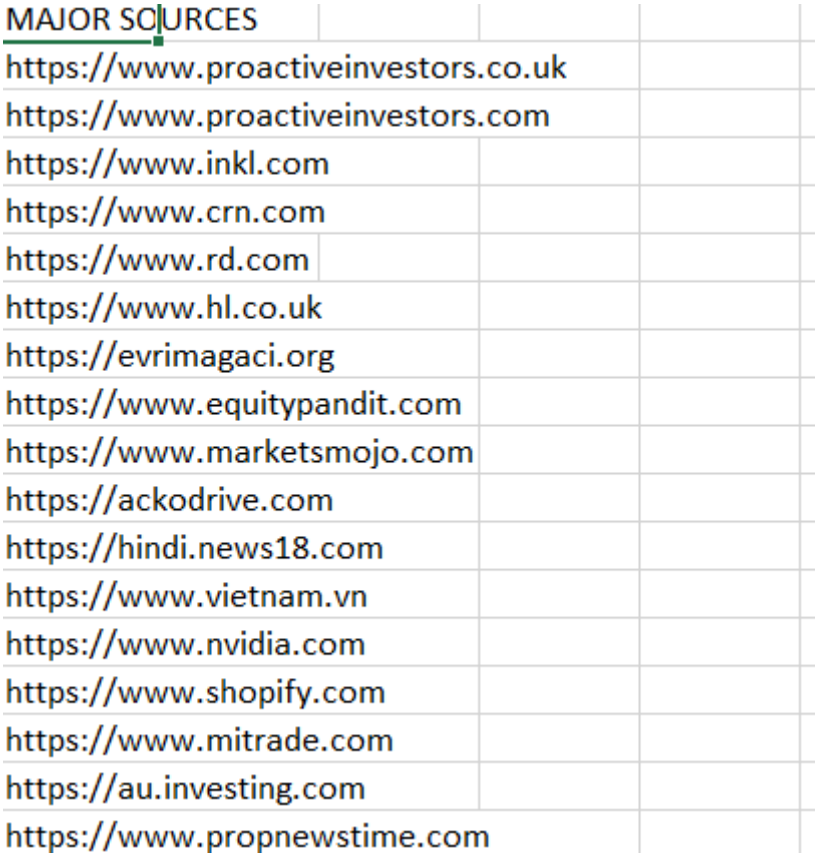
4. Implementation and Usage

4.1 Tracking File

We maintain a dedicated Excel file titled:

major_sources.xlsx

This file currently contains **483 unique domains** classified as major sources. It is **regularly updated** as new domains are identified during output review or through market intelligence.



(Figure 9.4: Screenshot of Major_sources.xlsx)

4.2 Integration with Scraper Code

The major sources list is embedded into the Python scraper logic and used to:

- Cross-check scraped URLs in real time.
- Mark or skip entries from identified major domains during parsing.

4.3 Integration with Excel Validation

The same list is also incorporated into the **domain-filtering formula** used in the Excel validation step for Google News output. Any link containing a domain from the list is automatically marked as "REMOVE".

Example formula snippet:

```
=IF(OR(ISNUMBER(SEARCH("bloomberg.",B2)),ISNUMBER(SEARCH("reuters." , B2)),

...,

ISNUMBER(SEARCH("cnn.", B2))), "REMOVE", "KEEP")
```

5. Observations

- **Dynamic Maintenance:** The list is not static. As new high-traffic domains emerge or existing ones change content policies, updates are made to ensure continued relevance.
- **Supplementary Role:** The major sources list acts as an additional filter alongside other measures like duplicate detection and publication date validation.
- **Improved Focus:** By excluding these domains, the scraping output better aligns with Credit-X's goal of identifying timely, company-specific developments rather than mainstream headlines.

6. Summary

The **Major Sources list** plays a crucial role in refining the data quality of the Credit-X news ingestion pipeline. It ensures legal compliance, improves the relevance of scraped data, and supports both the programmatic and manual layers of validation. With **483 sources** currently tracked and growing, it remains a central component of the system's evolving quality control framework.

Appendix D: User Manual

This appendix provides a comprehensive user guide for executing the Google News and Official Website scrapers developed during the internship project at Galytix. It also outlines the validation steps applied to the output using domain filtering and Excel-based conditional formatting.

1. Running the Google News Scraper

The Google News Scraper collects company-related news articles from Google News search results using a specified date range. The output is stored in an Excel file.

1.1 Prerequisites

Before running the script, ensure the following:

- Python 3.11 (64-bit) is installed on the system.
- Required Python libraries are installed:
    - undetected_chromedriver
    - selenium
    - pandas
    - datetime
    - urllib.parse
    - tqdm
    - htmldate
- Google Chrome browser is installed.
- The script is executed via the terminal (or VS Code terminal).

1.2 Command-Line Execution

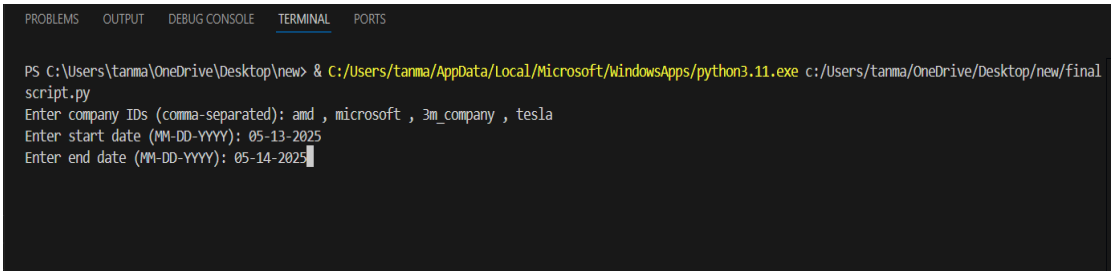
To execute the script:

1. Open the terminal or command prompt.
2. Navigate to the script directory.
3. Enter the following command:

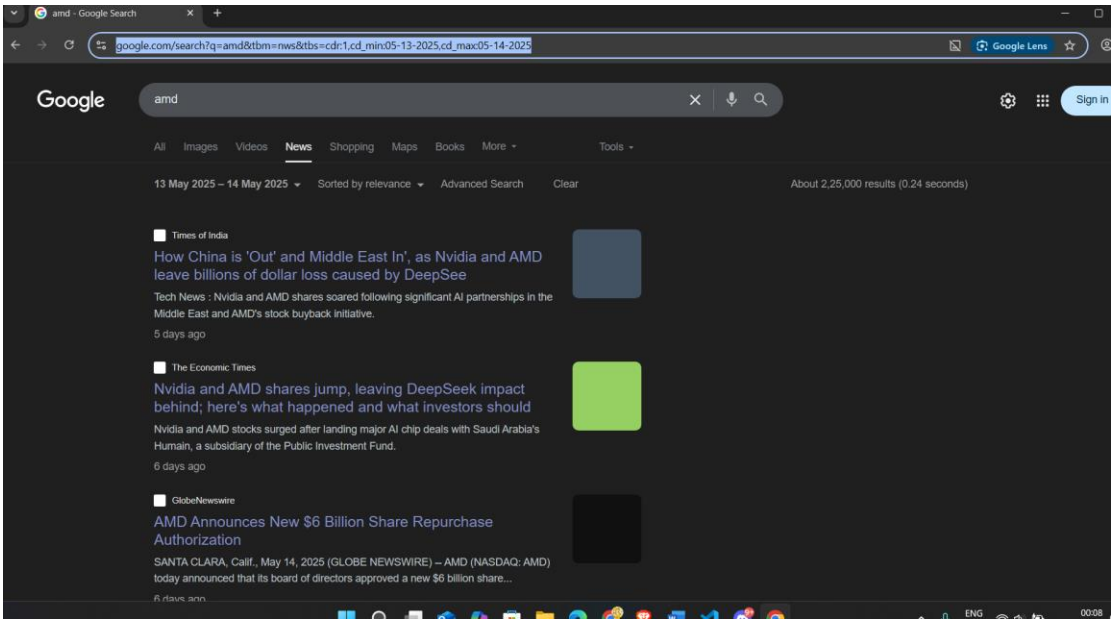
    python google_news_scraper.py apple_inc,microsoft_corp,nvidia_corp 2024-06-01 2024-06-15

\Where:

- The first argument is a comma-separated list of company IDs.
- The second and third arguments specify the start and end dates (YYYY-MM-DD format).



(Figure 9.5: Input for Google News Scraper.)



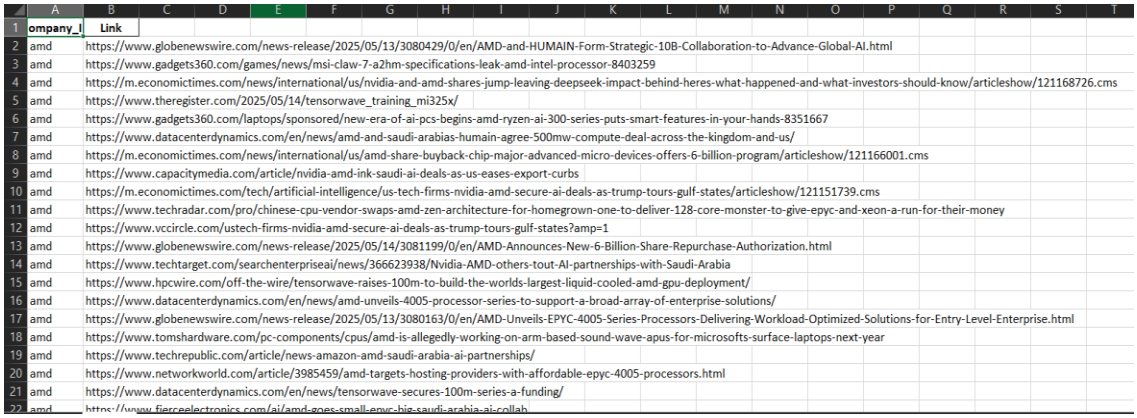
(Figure 9.6: Browser window automatically scraping for amd.)

1.3 Output Format

The script generates an Excel file named:

“filtered_news_links_<timestamp>.xlsx"The file includes the following columns:

- company_id
- news_url



ompany_I	Link
amd	https://www.globenewswire.com/news-release/2025/05/13/3080429/0/en/AMD-and-HUMAIN-Form-Strategic-10B-Collaboration-to-Advance-Global-AI.html
amd	https://www.gadgets360.com/games/news/msi-claw-7-a2hm-specifications-leak-amd-intel-processor-8403259
amd	https://m.economictimes.com/news/international/us/nvidia-and-amd-shares-jump-leaving-deepseek-impact-behind-heres-what-happened-and-what-investors-should-know/articleshow/121168726.cms
amd	https://www.theregister.com/2025/05/14/tensorwave_training_mi325x/
amd	https://www.gadgets360.com/laptops/sponsored/new-era-of-ai-pcs-begins-amd-ryzen-ai-300-series-puts-smart-features-in-your-hands-8351667
amd	https://www.datacenterdynamics.com/en/news/amd-and-saudi-arabias-humain-agree-500mw-compute-deal-across-the-kingdom-and-us/
amd	https://m.economictimes.com/news/international/us/amd-share-buyback-chip-major-advanced-micro-devices-offers-6-billion-program/articleshow/121166001.cms
amd	https://www.capacitymedia.com/article/nvidia-amd-ink-saudi-ai-deals-as-us-eases-export-curbs
amd	https://m.economictimes.com/tech/artificial-intelligence/us-tech-firms-nvidia-amd-secure-ai-deals-as-trump-tours-gulf-states/articleshow/121151739.cms
amd	https://www.techradar.com/pro/chinese-cpu-vendor-swaps-amd-zen-architecture-for-homegrown-one-to-deliver-128-core-monster-to-give-epyc-and-xeon-a-run-for-their-money
amd	https://www.vccircle.com/ustech-firms-nvidia-amd-secure-ai-deals-as-trump-tours-gulf-states?amp=1
amd	https://www.globenewswire.com/news-release/2025/05/14/3081199/0/en/AMD-Announces-New-6-Billion-Share-Repurchase-Authorization.html
amd	https://www.techtarget.com/searchenterpriseai/news/366623938/Nvidia-AMD-others-tout-AI-partnerships-with-Saudi-Arabia
amd	https://www.hpcwire.com/off-the-wire/tensorwave-raises-100m-to-build-the-worlds-largest-liquid-cooled-amd-gpu-deployment/
amd	https://www.datacenterdynamics.com/en/news/amd-unveils-4005-processor-series-to-support-a-broad-array-of-enterprise-solutions/
amd	https://www.globenewswire.com/news-release/2025/05/13/3080163/0/en/AMD-Unveils-EPYC-4005-Series-Processors-Delivering-Workload-Optimized-Solutions-for-Entry-Level-Enterprise.html
amd	https://www.tomshardware.com/pc-components/cpus/amd-is-allegedly-working-on-arm-based-sound-wave-apus-for-microsofts-surface-laptops-next-year
amd	https://www.techrepublic.com/article/news-amazon-amd-saudi-arabia-ai-partnerships/
amd	https://www.networkworld.com/article/3985459/amd-targets-hosting-providers-with-affordable-epyc-4005-processors.html
amd	https://www.datacenterdynamics.com/en/news/tensorwave-secures-100m-series-a-funding/
amd	https://www.fierceelectronics.com/ai/amd-goes-small-epyc-big-saudi-arabia-ai-collab

(Figure 9.7: Output For Google News Scraper.)

2. Running the Official Website Scraper

The Official Website Scraper fetches news articles directly from the "News" or "Media" section of the company's official site.

2.1 Reference File: official.xlsx

The official.xlsx file is a critical input that contains:

- company_id
- website_url (homepage)
- base_url (news/media section path)

This file was created manually by the ingestion team. Over a period of **three weeks**, a team of **eight members** explored, verified, and compiled accurate homepage URLs and news section paths for over 450 companies.

company_id	website_url	base_url
3m_company	https://investors.3m.com	https://investors.3m.com
a_o_smith_corporation	https://www.aosmith.com	https://investor.aosmith.com
abbott_laboratories	https://www.abbott.com	https://abbott.mediaroom.com
abbvie	https://www.abbvie.com	https://news.abbvie.com
accenture_plc	https://www.accenture.com	https://newsroom.accenture.com
adobe_inc	https://www.adobe.com	https://blog.adobe.com
adt_inc	https://www.adt.com	https://investor.adt.com
advanced_micro_devices_inc	https://www.amd.com	https://www.amd.com
aes_corporation	https://www.aes.com	https://aescorporation2023cr.q4web.com
afc_gamma	https://advancedflowercapital.com	https://investors.afcgamma.com
affirm_holdings_inc	https://investors.affirm.com	https://investors.affirm.com
aflac_incorporated	https://www.aflac.com	https://newsroom.aflac.com
agilent_technologies_inc	https://www.investor.agilent.com	https://www.investor.agilent.com
agnico_eagle_mines_limited	https://www.agnicoeagle.com	https://www.agnicoeagle.com
air_products_and_chemicals_inc	https://investors.airbnb.com	https://www.airproducts.com
airbnb_inc	https://investors.airbnb.com	https://news.airbnb.com
akamai_technologies_inc	https://www.akamai.com	https://www.ir.akamai.com
albemarle_corporation	https://www.albemarle.com	https://investors.albemarle.com
alexandria_real_estate_equities_inc	https://www.are.com	https://investor.are.com
align_technology_inc	https://www.aligntech.com	https://investor.aligntech.com

(Figure 9.8:Screenshot of Official.xlsx.)

2.2 Command-Line Execution

To run the scraper:

1. Open the terminal or command prompt.
2. Navigate to the directory containing the script.
3. Execute the following command:

- python official_news_scraper.py apple_inc,microsoft_corp 2024-06-01 2024-06-15 25

```
PS C:\Users\tanma\OneDrive\Desktop\new> & C:/Users/tanma/AppData/Local/Microsoft/Windows
xe c:/Users/tanma/OneDrive/Desktop/new/official.py
Enter comma-separated company_ids (e.g. apple_inc,amd): amazon , tesla , 3m_company
Enter start date (MM-DD-YYYY): 05-13-2025
Enter end date (MM-DD-YYYY): 05-14-2025
Enter timeout in seconds for page load (default 15): 25
```

(Figure 9.9: Input for Official Website Scraper.)

2.3 Output Format

The script generates a timestamped Excel file such as:

scrape_links_with_dates<random_id>.xlsx

The columns include:

- company_id
- news_url
- publication_date

	A	B	C	D
1	company_id	link	date	
2	3m_company	https://investors.3m.com/news-events/press-releases/detail/1889/3m-annual-meeting-results	2025-05-13	
3	3m_company	https://investors.3m.com/news-events/press-releases/detail/1888/3m-board-declares-quarterly-dividend	2025-05-13	
4	alliant_energy_corporation	https://investors.alliantenergy.com/News--Presentations/news/news-details/2025/Interstate-Power-and-Light-Company-Prices-Debt-Offering/default.aspx	2025-05-13	
5	amazon	https://press.aboutamazon.com/aws/2025/5/marsh-mclennan-selects-aws-as-preferred-cloud-provider-to-accelerate-global-cloud-transformation	2025-05-14	
6	amazon	https://press.aboutamazon.com/aws/2025/5/formula-1-and-aws-give-fans-a-chance-to-design-their-own-race-track-with-new-online-experience	2025-05-14	
7	amazon	https://press.aboutamazon.com/aws/2025/5/weatherford-announces-a-strategic-agreement-with-amazon-web-services-to-accelerate-digital-transformation	2025-05-13	
8	amazon	https://press.aboutamazon.com/2025/5/aws-and-humain-announce-groundbreaking-ai-zone-to-accelerate-ai-adoption-in-saudi-arabia-and-globally	2025-05-13	
9	amazon	https://press.aboutamazon.com/aws/2025/5/demandbase-announces-agentbase-as-the-first-gtm-ai-agents-built-on-aws	2025-05-13	
10	apollo_global_management_in	https://www.apollo.com/insights-news/pressreleases/2025/05/apollo-hybrid-funds-to-acquire-powergrid-services-from-the-sterling-group-3080081	2025-05-13	
11	bank_of_america_corporation	https://newsroom.bankofamerica.com/content/newsroom/press-releases/2025/05/bofa-to-open-150-financial-centers-by-2027--investing-over--5-bi.html	2025-05-13	
12	becton_dickinson_and_compar	https://news.bd.com/2025-05-13-BD-Launches-Landmark-Cell-Analyzer-Featuring-Breakthrough-Spectral-and-Real-Time-Cell-Imaging-Technologies	2025-05-13	
13	hilton_worldwide_holdings_inc	https://stories.hilton.com/releases/tru-by-hilton-debuts-in-asia-pacific-with-14-hotels-opening-in-vietnam	2025-05-14	
14	humana_inc	https://press.humana.com/news/news-details/2025/CenterWell-Specialty-Pharmacy-Takes-National-Patient-Choice-Award/default.aspx	2025-05-13	
15	fathom_holdings_inc	https://ir.fathominc.com/news-events/press-releases/detail/153/fathom-holdings-reports-first-quarter-2025-results	2025-05-13	
16				

(Figure 9.10: Output for Official Website Scraper.)

3. Extra Domain Filtering for Google News Output

To ensure only relevant sources are retained, a domain filtering layer is applied to remove common syndicated and unrelated media domains from the Google News output.

3.1 Formula Application

1. Open the filtered_news_links_<timestamp>.xlsx file.
2. Insert a new column titled Status next to the news_url column.
3. In the first row under the Status column, paste the following formula:

EXCEL FORMULA FOR DOUBLE CHECKING MAJOR DOMAINS

=IF(OR(ISNUMBER(SEARCH("market.us",B2)),ISNUMBER(SEARCH("financialexpress", B2)),

...,

ISNUMBER(SEARCH("investopedia", B2)), ISNUMBER(SEARCH("yahoo", B2))), "REMOVE","KEEP")

4. Drag the formula down to apply it to all rows.
5. This formula remove's any domains that might have been scraped but belong to the Major sources list .

A	B	C	D
ompany_I	Link	status	
amd	https://www.globenewswire.com/news-rele	KEEP	
amd	https://www.gadgets360.com/games/news,	KEEP	
amd	https://m.economictimes.com/news/intern	REMOVE	
amd	https://www.theregister.com/2025/05/14/t	KEEP	
amd	https://www.gadgets360.com/laptops/spon	KEEP	
amd	https://www.datacenterdynamics.com/en/r	KEEP	
amd	https://m.economictimes.com/news/intern	REMOVE	
amd	https://www.capacitymedia.com/article/nv	KEEP	
amd	https://m.economictimes.com/tech/artificia	REMOVE	
amd	https://www.techradar.com/pro/chinese-cp	KEEP	
amd	https://www.vccircle.com/ustech-firms-nvic	KEEP	
amd	https://www.globenewswire.com/news-rele	KEEP	
amd	https://www.techtarget.com/searchenterpr	KEEP	
amd	https://www.hpcwire.com/off-the-wire/ten	KEEP	
amd	https://www.datacenterdynamics.com/en/r	KEEP	
amd	https://www.globenewswire.com/news-rele	KEEP	
amd	https://www.tomshardware.com/pc-compc	KEEP	
amd	https://www.techrepublic.com/article/news	KEEP	
amd	https://www.networkworld.com/article/398	KEEP	
amd	https://www.datacenterdynamics.com/en/r	KEEP	

Figure 9.11: Applying Excel Formula for Verification.)

Use Excel's filter to select rows with "REMOVE" and delete them.

1	ompan	Link	status
2	amd	https://www.globenewswire.com/news-rele	KEEP
3	amd	https://www.gadgets360.com/games/news,	KEEP
5	amd	https://www.theregister.com/2025/05/14/t	KEEP
6	amd	https://www.gadgets360.com/laptops/spon	KEEP
7	amd	https://www.datacenterdynamics.com/en/r	KEEP
9	amd	https://www.capacitymedia.com/article/nv	KEEP
11	amd	https://www.techradar.com/pro/chinese-cp	KEEP
12	amd	https://www.vccircle.com/ustech-firms-nvic	KEEP
13	amd	https://www.globenewswire.com/news-rele	KEEP
14	amd	https://www.techtarget.com/searchenterpr	KEEP
15	amd	https://www.hpcwire.com/off-the-wire/ten	KEEP
16	amd	https://www.datacenterdynamics.com/en/r	KEEP
17	amd	https://www.globenewswire.com/news-rele	KEEP
18	amd	https://www.tomshardware.com/pc-compc	KEEP
19	amd	https://www.techrepublic.com/article/news	KEEP

Figure 9.12: Verified Links .)

4. Removing Duplicates Using Conditional Formatting

After domain filtering, duplicates are highlighted to allow quick removal of repeated entries.

4.1 Steps to Apply Conditional Formatting

1. Select the news_url column.
2. Go to **Home** tab → **Conditional Formatting** → **Highlight Cell Rules** → **Duplicate Values**.
3. Choose a color (e.g., red) to highlight duplicates.
4. Click **OK**.

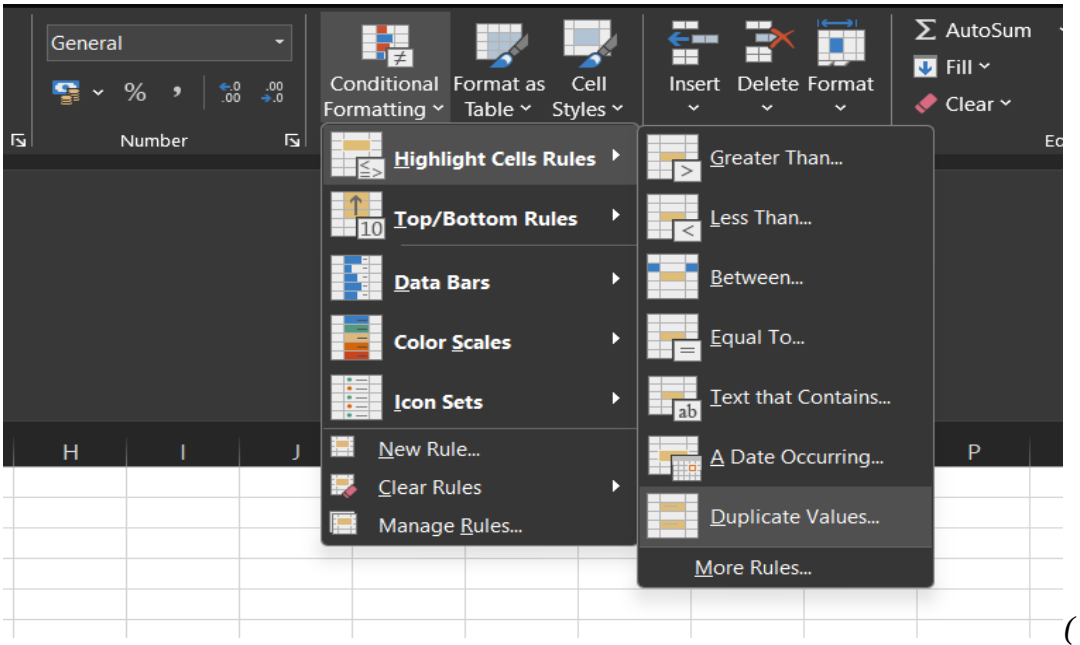


Figure 9.13: Applying Conditional Formatting.)

	A	B	C	D	E	F	G	H	I
1	ompany_I	Link							
2	3m_compa	https://www.regulatoryoversight.com/2025/05/new-jersey-ag-platkin-reaches-450m-pf							
3	3m_compa	https://www.newstrail.com/thermosetting-acrylic-adhesive-market-getting-back-to-gro							
4	3m_compa	https://www.openpr.com/news/4023330/nigeria-sponge-and-scouring-pads-market-ou							
5	3m_compa	https://www.newstrail.com/personal-protective-equipment-ppe-market-to-see-thriving							
6	3m_compa	https://www.citizen.co.za/review-online/news-headlines/local-news/2025/05/20/limpo							
7	3m_compa	https://www.openpr.com/news/4023788/sterilization-equipment-market-executive-brie							
8	3m_compa	https://www.openpr.com/news/4024179/veterinary-equipment-and-disposables-marke							
9	3m_compa	https://www.finsmes.com/2025/05/catalyxx-raises-e3m-in-funding.html							
10	3m_compa	https://www.tuko.co.ke/entertainment/celebrities/589736-ringtone-granted-ksh-3m-bc							
11	3m_compa	https://www.sportindustry.biz/news-categories/news/tgp-europe-withdraws-from-uk-m							
12	3m_compa	https://www.ehn.org/3m-agrees-to-pay-new-jersey-up-to-450-million-for-pfas-water-p							
13	3m_compa	https://m.economictimes.com/markets/stocks/stock-liveblog/hindalco-share-price-toda							
14	3m_compa	https://tech.eu/2025/05/19/catalyxx-fuels-expansion-with-eur3m-round-and-eur37m-e							
15	3m_compa	https://www.legalreader.com/3m-settles-new-jersey-pfas-lawsuits/							
16	3m_compa	https://www.thetimes.com/business-money/companies/article/military-personnel-in-cl							
17	3m_compa	https://www.openpr.com/news/4023275/explore-security-labels-market-growth-applic							
18	3m_compa	https://www.openpr.com/news/4023981/emergency-medical-supplies-market-deep-res							
19	3m_compa	https://www.floordaily.net/flooring-news/hmtx-names-sen-ceo-stone-stepping-into-ex							
20	a_o_smith	https://www.law.com/americanlawyer/2025/05/19/reed-smith-takes-corporate-trust-t							

Figure 9.14: Removing Duplicates Marked in red.)

4.2 Cleaning the Output

- Review all rows with red highlights.
- Remove duplicate URLs manually or use Excel's filtering features.

5. Summary Table

Step	Action	Tool
1	Run Google News Scraper	Python (CLI)
2	Run Official Website Scraper	Python (CLI)
3	Apply Domain Filter to Google Output	Excel
4	Apply Conditional Formatting	Excel

(Table 9.3: Flow of Execution.)