BCSE497J Project-I

ADVANCED WEB SCRAPPING SOFTWARE WITH OPTICAL CHARACTER RECOGNITION (OCR) AND SPONSOR DETECTION

Reg. No.	Name
22BCE2593	ARYAN

Under the Supervision of

Dr. RADHAKRISHNAN DELHIBABU

Professor Grade 1

School of Computer Science and Engineering (SCOPE)

B.Tech.

in

Computer Science and Engineering

Core

School of Computer Science and Engineering

September 2025

TABLE OF CONTENTS

ABSTRACT

In today's digital age, corporate websites are powerful platforms for marketing and advertising. For many companies, sponsorships are not just an additional income source but a central part of their business model. To manage these sponsorships fairly, businesses must know exactly how long a sponsor is visible on their website and in what form. This information is also critical to measure the return on investment for sponsors. Traditional web scraping methods struggle with this task because modern websites often use JavaScript to load content dynamically, and sponsor names or logos are frequently embedded in images. These limitations mean that important sponsor data is often missed. To overcome these issues, we present a software framework designed to capture sponsor visibility in a complete and reliable way.

The framework is built on a dual scraping engine. The first part uses BeautifulSoup to extract content from static HTML pages, while the second part uses Selenium to handle websites that rely on JavaScript for loading data. This combination ensures that no content is overlooked, whether it appears at the beginning or is loaded later. A text analysis module then scans through the collected content to detect sponsor names, keywords, and other recognizable information. At the same time, an image collection module gathers all image links from the site, ensuring that the system also accounts for visual material.

To deal with sponsors presented inside images such as logos, banners, or promotional graphics, the framework includes an Optical Character Recognition module. This component processes the images, reads the text inside them, and converts it into machine-readable data. This step is crucial because it allows the system to identify sponsors even if their names never appear in the written text of the site. By combining the results from the OCR module with the text analysis, the system creates a full profile of each sponsor's presence.

The final and most important part of the framework is the Sponsor Presence Detection module. This module continuously monitors the website, recording the exact time when a sponsor first appears and when it disappears. It calculates the total time that each sponsor is visible and produces a detailed log with timestamps. The output includes the sponsor's name, the complete duration of their presence, and every interval of appearance and absence.

In conclusion, this framework provides a practical and automated solution for tracking sponsors on modern, dynamic websites. By combining web scraping, image processing, and time-based tracking, it offers businesses reliable and detailed data that supports accurate billing and better measurement of sponsor performance. This not only improves operational efficiency but also strengthens decision making with trustworthy insights.

1. INTRODUCTION

1.1. BACKGROUND

The proliferation of digital platforms has made web content a primary source of data for various applications, including market analysis, academic research, and business intelligence. As the volume and complexity of online data have grown, so has the demand for automated data extraction solutions. Traditional web scraping methods, while effective for static content, often fail when confronted with the dynamic nature of modern websites, which are increasingly built with technologies like JavaScript that render content in real time. This limitation has spurred a new wave of research focused on creating more robust and intelligent web scraping frameworks.

As highlighted by Singrodia et al. (2019), the vast majority of web data is unstructured and difficult to collect manually, necessitating automated tools to transform this data into an organized format for analysis. The work of Glez-Peña et al. (2013) further emphasizes the continued relevance of web scraping, even in a world of APIs, particularly for data sources in fields like bioinformatics that may lack programmatic interfaces. This suggests a persistent need for versatile scraping solutions that can adapt to diverse data landscapes.

Furthermore, the challenge of data extraction is not limited to text. Patnaik et al. (2021) demonstrate the need for systems that can adapt to dynamic website changes, particularly in e-commerce, where product details are often presented within images. Their research introduces a system that uses deep learning and OCR to extract data from images, showcasing a significant advancement beyond simple text scraping. Similarly, Oussaleh and Taoufik (2024) use a combination of web scraping and OCR to extract structured data from unstructured scanned documents, further proving the value of integrating image processing into data collection workflows.

1.2. MOTIVATIONS

The central motivation for this project is to address a significant and unresolved business problem within the digital advertising and sponsorship sector. In today's highly competitive market, where a company's web presence is a primary source of revenue, there is a critical need for a reliable, automated method to monitor the visibility of sponsors. The current practice of manual, labor-intensive tracking is not only inefficient and costly but also highly susceptible to human error. This lack of a precise, data driven system results in a number of critical issues for businesses, including inaccurate billing, a basis for disputes over service delivery, and an inability to provide sponsors with the clear, quantifiable return on investment (ROI) data they increasingly demand.

Existing web scraping tools and techniques, while useful for general data collection, are fundamentally ill-equipped to handle this specific challenge. They are often unable to navigate the dynamic content and complex structures of modern websites, a limitation highlighted by prior research. More importantly, conventional scrapers cannot process visual information, leaving them blind to sponsor logos, branded images, and banner ads, which constitute a significant portion of digital sponsorships.

The problem is not merely about extracting data but about capturing the right data in real time, with the crucial addition of temporal context. Therefore, there is a clear and compelling need for a solution that can not only handle dynamic text and content but also integrate image recognition (OCR) to provide a complete and accurate picture of sponsor presence. Our project is motivated by the opportunity to fill this void, providing a comprehensive, intelligent, and time-sensitive solution that will transform how businesses manage and monetize their digital sponsorship arrangements.

1.3. SCOPE OF THE PROJECT

1.3.1. Web Scraping Module

1.3.1.1. *Text Data Extraction:* This module will use Python libraries, specifically BeautifulSoup and Selenium, to crawl websites and extract all textual data. It will be capable of handling both static HTML content and dynamic content rendered by JavaScript. The scope is limited to retrieving text, not interpreting its meaning at this stage.

1.3.1.2. *Image URL Extraction:* This module, also utilizing BeautifulSoup and Selenium, will identify and collect all image URLs present on a webpage. The scope is confined to URL collection and will not involve downloading or processing the images themselves.

1.3.2. Image Processing and OCR Module

1.3.2.1. *Optical Character Recognition (OCR):* This module will take the image URLs from the scraping module and process them using an OCR engine. The scope is to convert any text embedded within the images (such as logos, banners, and brand names) into machine-readable text data. This module is focused solely on character recognition and does not include image analysis beyond text detection.

1.3.3. Sponsor Presence Detection Module

1.3.3.1. *Data Integration and Analysis:* This core module will integrate the text data from the web scraping module and the text from the OCR module. It will be responsible for identifying sponsor names based on a predefined list or specific keywords.

1.3.3.2. *Temporal Analysis:* This is a crucial part of the scope. The module will log the precise timestamp of a sponsor's presence (first detection) and absence (no longer detected). The scope is to calculate and record the cumulative duration of each sponsor's visibility on the website.

1.3.4. Output File

1.3.4.1. *Structured Data Generation:* The final scope is to generate a structured output file (e.g., CSV, JSON) that contains specific, verifiable data. This data will include the sponsor's name, the URLs where they were detected, a log of their presence and

absence timestamps, and the total calculated duration of their visibility. This module is limited to data formatting and presentation, not visualization or external reporting.

2. PROJECT DESCRIPTION AND GOALS

2.1. LITERATURE REVIEW

The field of web data extraction has evolved significantly, driven by the need to transform the vast, often unstructured data available on the internet into a format suitable for analysis. Traditional methods of web scraping have long been valuable tools for scientific research and business intelligence (Barzin et al., 2023). However, modern websites, with their dynamic, JavaScript-rendered content, present a challenge to conventional scrapers that rely on static HTML parsing (Wahed et al., 2024).

Early research, such as that by Singrodia et al. (2019), focused on foundational scraping techniques, noting the immense volume of disorganized web data. Glez-Peña et al. (2013) reinforced the ongoing relevance of web scraping, even with the rise of APIs, for data sources that lack programmatic interfaces. This foundational work established web scraping as a core data acquisition technology (Mutlu et al., 2024).

A key limitation of early scraping was its inability to handle image-based information. Patnaik et al. (2021) addressed this by developing a system that uses deep learning and OCR to extract product details from images on e-commerce sites. Similarly, Oussaleh and Taoufik (2024) leveraged web scraping with OCR to extract structured data from unstructured documents, proving the value of integrating image processing into data collection.

2.2. GAPS IDENTIFIED

Based on a thorough review of existing literature, our project is positioned to fill several important gaps in the current state of web data extraction. While other research has made significant strides, no single solution addresses the unique and specific business problem of comprehensively analyzing sponsor presence.

2.2.1. *Lack of Temporal Analysis for Specific Elements:* While studies such as that by Barzin et al. (2023) discuss "real-time" scraping for market monitoring, their focus is on continuously updating a dataset rather than on precisely logging the duration of a single element's visibility. Our project introduces a novel layer of analysis by meticulously recording the presence and absence times of sponsors. This is a crucial distinction that directly addresses the core business need for detailed data, a capability not found in the reviewed academic work.

2.2.2. *Isolated vs. Integrated Solutions:* A clear gap exists in the form of fully integrated, multi-module solutions. The research tends to be isolated, with different studies tackling individual problems. For instance, some papers concentrate on the challenges posed by dynamic website content (Wahed et al., 2024), while others specialize in using OCR to extract data from images (Patnaik et al., 2021; Oussaleh and Taoufik,

2024). There is a notable absence of a cohesive framework that combines robust dynamic content scraping, image-based OCR, and a temporal analysis model into a single system designed to solve a complex, real-world business problem.

2.2.3. *Application-Specific Frameworks:* While web scraping has been successfully applied to diverse fields such as real estate (Barzin et al., 2023), academic research, and public procurement (Oussaleh and Taoufik, 2024), the specific domain of digital advertising and sponsor monitoring for the purpose of accurate data and analysis remains largely unexplored in the academic literature. This project aims to bridge this gap by creating a tool specifically tailored to the unique requirements of this industry, providing a level of precision and verifiability not currently available.

2.3. OBJECTIVES

Based on the identified gaps and motivations, the primary objectives of this project are to develop a comprehensive software framework that can accurately and efficiently monitor sponsor presence on websites. These objectives are designed to directly address the limitations of existing methods and provide a valuable business solution. The key objectives are:

- To develop a multi-module web scraping system capable of handling dynamic web content.
- To integrate an Optical Character Recognition (OCR) engine for image-based data extraction.
- To design and implement a sponsor detection model that combines text and image data.
- To create a temporal analysis component that tracks sponsor visibility.
- To generate a structured and detailed output file.

2.4. PROBLEM STATEMENT

In the modern digital landscape, businesses rely heavily on web presence to generate revenue through sponsorships and advertisements. However, a critical and unresolved challenge exists in accurately monitoring the visibility and duration of these sponsorships. The traditional method of manual tracking is inefficient, costly, and prone to human error, leading to inaccurate billing and disputes. Furthermore, existing automated web scraping tools are often inadequate, as they struggle to handle the dynamic content of modern websites and are unable to extract vital information from images, such as company logos or branded banners. This technological gap results in businesses lacking the precise, verifiable data necessary for accurate analysis and transparently demonstrating value to their sponsors. Therefore, a comprehensive and intelligent solution is needed to automatically identify, track, and log the presence of sponsors on websites by effectively handling dynamic content and extracting information from both text and images.

2.5. PROJECT PLAN

2.5.1. Phase 1: Research and Planning (Months 1-2)

2.5.1.1. Literature Review
2.5.1.2. Requirement Finalization
2.5.1.3. Technology Stack Selection
2.5.1.4. Architectural Design

2.5.2. Phase 2: Core Model Development (Months 3-4)

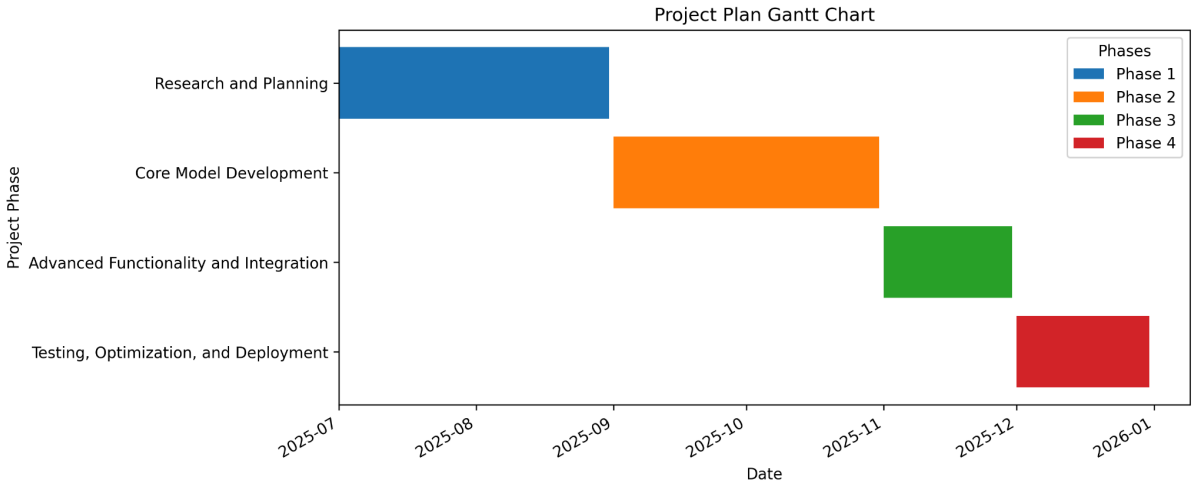
2.5.2.1. Web Scraping Model Creation
2.5.2.2. OCR Model Creation
2.5.2.3. Sponsor Detection Model Creation

2.5.3. Phase 3: Advanced Functionality and Integration (Month 5)

2.5.3.1. Database Connectivity
2.5.3.2. Module Integration

2.5.4. Phase 4: Testing, Optimization, and Deployment (Month 6)

- Unit and End-to-End Testing
- Performance Optimization
- Web Page Interface Creation
- Back-end Deployment



3. REQUIREMENT ANALYSIS

The successful development and deployment of the proposed "Advanced Web Scraping Software with OCR and Sponsor Detection" will depend on fulfilling a set of clear functional and non-functional requirements. These requirements are derived directly from the project's objectives and the gaps identified in current web data extraction technologies.

3.1. Functional Requirements

3.1.1. *URL Input and Navigation:* The system must provide a user-friendly interface to accept a target website URL. It must then be able to navigate and crawl the entire website, or a specified portion of it, to collect data.

3.1.2. *Dynamic Content Scraping:* The system must be capable of interacting with and scraping content from dynamic, JavaScript-heavy websites.

This requires the use of a tool like Selenium to render the page fully before data extraction.

3.1.3. *Text and Image URL Extraction:* The system must be able to extract all textual data and all image URLs from a given webpage, including elements that are dynamically loaded.

3.1.4. *OCR Integration:* The system must include a module that can process the extracted image URLs. This module must utilize an OCR engine to accurately convert any embedded text (such as company names or logos) from images into a machine-readable string.

3.1.5. *Sponsor Detection Logic:* The system must implement a detection algorithm that compares the extracted text (from both text scraping and OCR) against a predefined list of sponsor names.

3.1.6. *Temporal Analysis and Logging:* The system must be able to log the precise timestamp of a sponsor's first appearance and a subsequent "last seen" time. This data will be used to calculate the total duration of the sponsor's visibility on the page.

3.1.7. *Structured Output Generation:* The final system must produce a structured output file (e.g., CSV, JSON) containing key data points: sponsor name, URLs of detection, presence timestamps, and calculated duration.

3.2. Non-Functional Requirements

3.2.1. *Performance:* The system should be optimized for speed and efficiency. The entire process of scraping, processing, and generating the output should be completed in a timely manner to provide near real-time insights.

3.2.2. *Accuracy:* The OCR and sponsor detection models must be highly accurate. The output data must be reliable enough to be used for commercial purposes like billing and reporting.

3.2.3. *Reliability:* The system must be robust and resilient to common web scraping issues, such as broken links, changes in a website's structure, or a website's anti-bot measures. It should fail gracefully and log errors without crashing.

3.2.4. *Scalability:* The architecture should be scalable, capable of handling an increasing number of websites, pages, and concurrent requests without significant degradation in performance.

3.2.5. *Maintainability:* The code should be modular, well-commented, and easy to maintain. This will allow for straightforward updates to the sponsor list, changes to website scraping logic, or integration of new technologies in the future.

4. SYSTEM DESIGN

The system design for the "Advanced Web Scraping Software" is a layered, modular architecture built for both manual and automated operations. The design separates the front-end user interface from the back-end application logic, ensuring scalability and maintainability.

4.1. High-Level Architecture

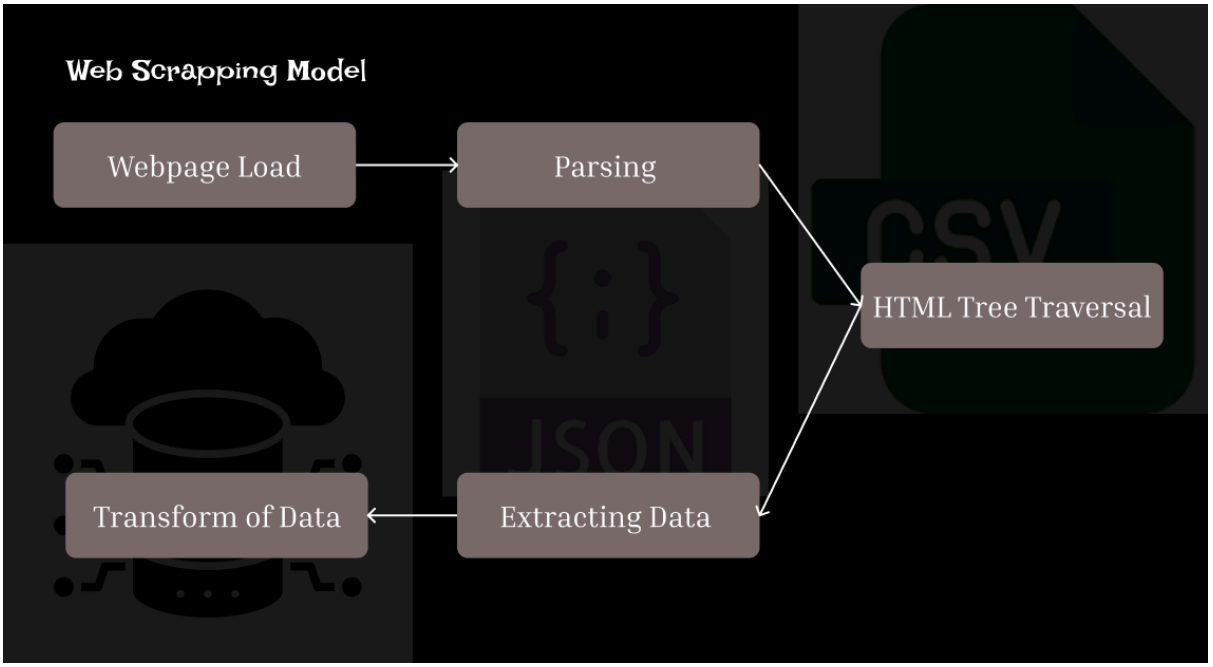
The system operates across three primary layers: a user-facing front-end, a Python-based back-end application, and a data layer for persistent storage.

- 4.1.1. *Front-End Layer:* This will be a single-page web application built with HTML, CSS, and JavaScript. It serves as the user's control panel, offering two distinct interfaces: one for manual, on-demand operations and another for configuring automated tasks. It communicates with the back-end via a RESTful API.
- 4.1.2. *Back-End Layer:* This is the core of the system, implemented using a Python web framework such as Flask or FastAPI. It houses all the business logic and orchestrates the execution of the five core models. The back-end receives requests from the front-end, processes them, and returns the results. For automation, it will integrate a task scheduler.
- 4.1.3. *Data Layer:* A database, such as PostgreSQL or MongoDB, will be used to store configuration data for the automation model (e.g., schedule, run count) and to persist the final results from each run.

4.2. Module-wise Design

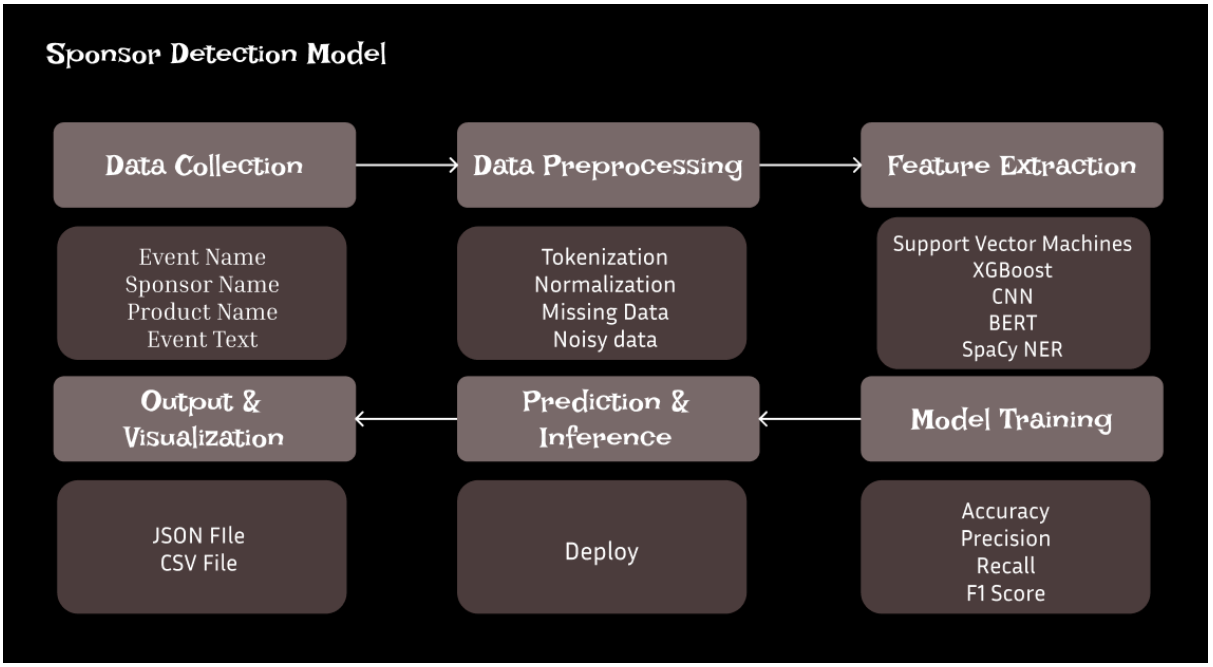
The system is composed of five distinct, interconnected models that work in a pipeline to achieve the final objective.

- 4.2.1. *HTML Tree View Extraction Model:* This model uses the lxml or BeautifulSoup library to parse the raw HTML of a web page into a traversable tree structure. This provides a detailed, hierarchical representation of the page's content, allowing for precise navigation and targeting of specific elements.
- 4.2.2. *Static Web Scraping Model:* This model uses a lightweight library like Requests to fetch the initial HTML content of a URL. It then uses the HTML tree from the previous model to extract all visible text content, such as headings, paragraphs, and list items. This model is highly efficient for websites with minimal dynamic content.
- 4.2.3. *Image Web Scraping Model:* This model, working in tandem with a headless browser like Selenium, handles the complexities of modern, dynamic websites. It waits for JavaScript to render the page fully and then extracts the URLs of all images by identifying <img> tags and other image sources. This is crucial for capturing sponsor logos and banners.



4.2.4. *OCR Model:* This module is built around an OCR engine like Tesseract or PaddleOCR. It takes the image URLs extracted by the previous model, downloads each image, and processes it to convert any embedded text into machine-readable strings. The module includes pre-processing steps like image resizing and noise reduction to improve OCR accuracy.

4.2.5. *Sponsor Detection Model:* This is the final analysis module. It combines the text data from the Static Web Scraping Model and the OCR Model. It uses a flexible, rule-based approach to compare this combined text against a predefined list of sponsor names. It's designed to handle variations in spelling, capitalization, and formatting to ensure accurate matches.



4.3. Deployment Model Design

The system's design accommodates two distinct modes of operation to meet diverse user needs.

4.3.1. Manual Model

4.3.1.1. *Deployment:* Each of the five models will be exposed on the back-end as a separate REST API endpoint.

4.3.1.2. *Interface:* The web page will feature a distinct section with buttons or a dropdown menu for each model. The user can select an operation, input a URL, and trigger a single model run.

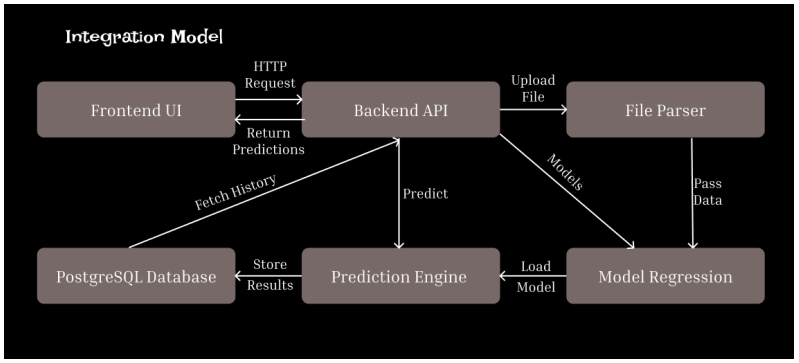
4.3.1.3. *Operation:* When a button is clicked, a request is sent to the corresponding back-end endpoint. The back-end executes the model, and the results are returned and displayed on the front-end. This provides a step-by-step, interactive way to use the system and allows the user to stop the process by simply not proceeding to the next step.

4.3.2. Automation Model

4.3.2.1. *Deployment:* This model leverages a task scheduler like APScheduler (for simplicity) or Celery (for scalability). The entire workflow (Web Scraping -> OCR -> Sponsor Detection) is encapsulated in a single background job.

4.3.2.2. *Interface:* The front-end will have a configuration panel where the user can define a website URL and set a time interval for the recurring task. The UI will display the current status of the automation (Scheduled, Running, Completed) and a counter for the number of runs.

4.3.2.3. *Operation:* When the user initiates the automation, the back-end creates a scheduled task in the scheduler. This task is configured to run the full pipeline at the specified interval. The back-end periodically updates the database with the run status and the number of runs. The front-end, through a simple polling mechanism or WebSockets, fetches this status data and displays it in real-time, providing the user with an overview of the automated process without requiring constant manual oversight.



5. REFERENCE

1. Lotfi, C., Srinivasan, S., Ertz, M., & Latrous, I. (2021). Web Scraping Techniques and Applications: A Literature Review. *Chapter in a book*. ResearchGate.
2. Singrodia, V., Mitra, A., & Paul, S. (2019). A Review on Web Scrapping and its Applications. In *2019 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1–4). IEEE.
3. Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2013). Web scraping technologies in an API world. *Briefings in Bioinformatics, 15*(5), 788–797.
4. Mutlu, M. A., Ulku, E. E., & Yildiz, K. (2024). A web scraping app for smart literature search of the keywords. *PeerJ Computer Science*, *10*, e2384.
5. Baumgartner, J., Zannettou, S., Keegan, B., Squire, M., & Blackburn, J. (2020). The Pushshift Reddit Dataset. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media (ICWSM 2020)* (pp. 53–60). AAAI Press.
6. Patnaik, S. K., Babu, C. N., & Bhave, M. (2021). Intelligent and Adaptive Web Data Extraction System Using Convolutional and Long Short-Term Memory Deep Learning Networks. *Big Data Mining and Analytics*, *4*(4), 279–297.
7. Oussaleh, A., & Taoufik, T. (2024). AI-Enhanced Techniques for Extracting Structured Data from Unstructured Public Procurement Documents. In *2024 8th International Symposium on Innovative Approaches in Smart Technologies (ISAS)* (pp. 1–6). IEEE.
8. Dogra, K. S., & Nirwan, N. (2023). Unlocking the Market Insight Potential of Data Extraction Using Python-Based Web Scraping on Flipkart. In *2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology (ICSEIET)* (pp. 1–5). IEEE.
9. Wahed, M. A., Ayman, J., Alzboon, M. S., & Al-Batah, M. (2024). Automating Web Data Collection: Challenges, Solutions, and Python-Based Strategies for Effective Web Scraping. In *2024 7th International Conference on Internet Applications, Protocols, and Services (NETAPPS)* (pp. 1–6). IEEE.
10. Barzin, F., Yernaux, G., & Vanhoof, W. (2023). SCRIMMO: A Real-time Web Scraper Monitoring the Belgian Real Estate Market. In *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)* (pp. 1–8). IEEE.
11. Mydyti, H., & Ware, A. (2025). Integrating Intelligent Web Scraping Techniques in Internship Management Systems: Enhancing Internship Matching. *Annals of Emerging Technologies in Computing (AETIC)*, *9*(1), 1–8.
12. Yu, M., Secondi, L., Laureti, T., & Palumbo, L. (2025). Saving food surplus and developing new business models: Exploring the potential of Too Good To Go at territorial level using web-scraped data. *Big Data Research*, *40*, 100536.