B.Tech. BCSE497J - Project-I

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

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Programme

by

22BCE2593 ARYAN

Under the Supervision of

Dr. RADHAKRISHNAN DELHIBABU

Professor Grade 1

School of Computer Science and Engineering (SCOPE)

November 2025

DECLARATION

I hereby declare that the project entitled *Advanced Web Scrapping Software with Optical Character Recognition (OCR) and Sponsor Detection* submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Prof. / Dr. *Radhakrishnan Delhibabu.*

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 05-11-2025

Signature of the Candidate

CERTIFICATE

This is to certify that the project entitled *Advanced Web Scrapping Software with Optical Character Recognition (OCR) and Sponsor Detection* submitted by *Aryan 22BCE2593*, **School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during Fall Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The project fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore
Date : 05-11-2025

Signature of the Guide

Examiner(s)

Dr. BOOMINATHAN P
Computer Science and Engineering Core

ACKNOWLEDGEMENTS

I am deeply grateful to the management of Vellore Institute of Technology (VIT) for providing me with the opportunity and resources to undertake this project. Their commitment to fostering a conducive learning environment has been instrumental in my academic journey. The support and infrastructure provided by VIT have enabled me to explore and develop my ideas to their fullest potential.

My sincere thanks to Dr. Jaisankar N, the Dean of the School of Computer Science and Engineering (SCOPE), for his unwavering support and encouragement. His leadership and vision have greatly inspired me to strive for excellence. The Dean's dedication to academic excellence and innovation has been a constant source of motivation for me. I appreciate his efforts in creating an environment that nurtures creativity and critical thinking.

I express my profound appreciation to *Dr. BOOMINATHAN P*, the Head of the Computer Science and Engineering Core, for his/her insightful guidance and continuous support. His/her expertise and advice have been crucial in shaping the direction of my project. The Head of Department's commitment to fostering a collaborative and supportive atmosphere has greatly enhanced my learning experience. His/her constructive feedback and encouragement have been invaluable in overcoming challenges and achieving my project goals.

I am immensely thankful to my project guide, *Dr. Radhakrishnan Delhibabu*, for his/her dedicated mentorship and invaluable feedback. His/her patience, knowledge, and encouragement have been pivotal in the successful completion of this project. My supervisor's willingness to share his/her expertise and provide thoughtful guidance has been instrumental in refining my ideas and methodologies. His/her support has not only contributed to the success of this project but has also enriched my overall academic experience.

Thank you all for your contributions and support.

Name of the Candidate

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations	
HTML	Hyper Text Markup Language
API	Application Programming Interface
OCR	Optical Character Recognition
ROI	Return on Investment
GPU	Graphics Processing Unit
RAM	Random Access Memory
REST	Representational State Transfer
URL	Uniform Resource Locator
CSV	Comma-Separated Values
JSON	JavaScript Object Notation

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 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)

- ★ Literature Review
- ★ Requirement Finalization
- ★ Technology Stack Selection
- ★ Architectural Design

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

- ★ Web Scraping Model Creation
- ★ OCR Model Creation
- ★ Sponsor Detection Model Creation

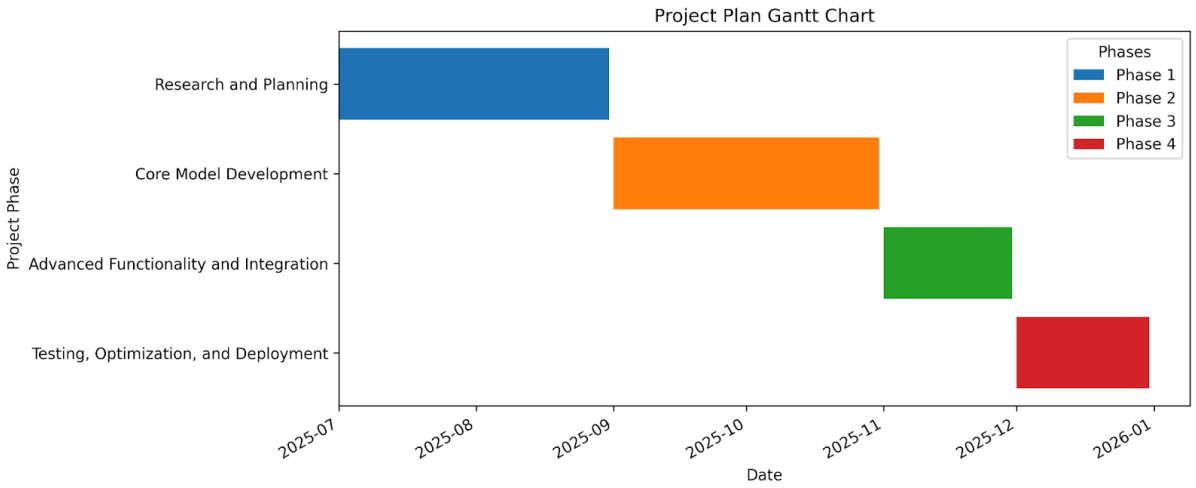
2.5.3. Phase 3: Advanced Functionality and Integration (Month 5)

- ★ 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

Figure 2.1



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 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.

3.3. Feasibility Study

3.3.1. Technical Feasibility

The project is technically possible because all the tools and technologies needed are easily available and well supported. It uses Python along with libraries such as BeautifulSoup, Selenium, and Tesseract OCR. These tools are reliable and have strong community support, which makes development easier. The software can be built and tested on a normal computer system with good processing power and a stable internet connection. Since these tools are open source, they do not require any special licenses or paid services. Therefore, from a technical point of view, the project is completely feasible with the current technology.

3.3.2. Economic Feasibility

This project is economically practical because it does not need expensive software or hardware. Most of the tools and libraries used are open source and free to use. The main cost would only come from hosting the application or using online storage services if required. Since this project can be developed using the resources already available in the institution, there will be no major financial burden. The time saved and the accuracy achieved by automation will give a high value compared to the small cost involved in development.

3.3.3. Social Feasibility

The system is socially acceptable because it helps organizations work more efficiently and transparently. It reduces human effort and mistakes by automating the process of sponsor detection. It also encourages fairness by providing accurate and clear data about how long sponsors appear on a website. The software follows ethical data collection practices and does not violate any privacy rules. It supports data-based decision making, which benefits both the company and the sponsors involved.

3.4. System Specification

3.4.1. Hardware Specification

- The software can run smoothly on a computer with a modern processor such as Intel i5 or higher.
- It requires at least 8 GB of RAM for proper performance.
- Around 500 GB of storage space is needed for data and system files.
- A stable internet connection is required for web scraping and testing.
- An optional graphics processor (GPU) can be used to speed up image processing, but it is not mandatory.

3.4.2. Software Specification

- The system can run on any major operating system such as Windows, Linux, or macOS.
- It will be developed using the Python programming language.
- The main tools used are BeautifulSoup, Selenium, and Tesseract OCR for image text recognition.
- Development tools like Visual Studio Code or PyCharm will be used for writing and testing code.
- Git will be used for version control and collaboration during development.

4. DESIGN APPROACH AND DETAILS

4.1. System Architecture

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.

4.1.1. 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.

- *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.
- *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.
- *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.

4.1.2. Module-wise Design

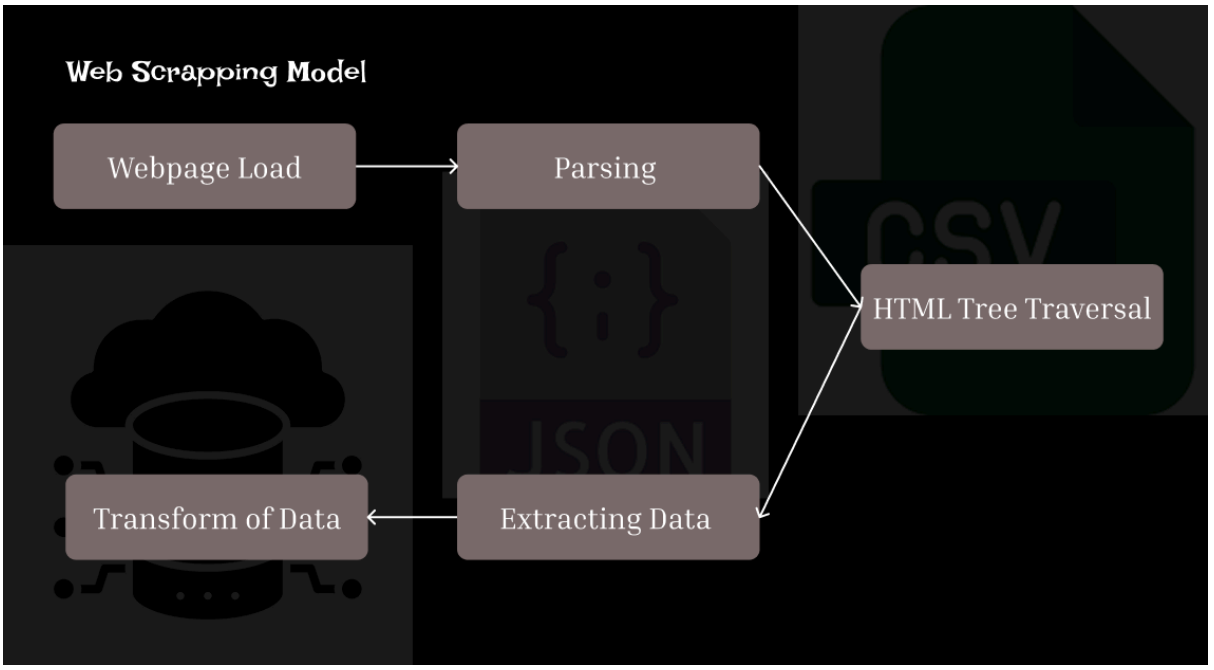
The system is composed of five distinct, interconnected models that work in a pipeline to achieve the final objective.

- *HTML Tree View Extraction Model:* This model uses 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.

- ➢ *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.
- ➢ *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.

Figure No. 4.1



- ➢ *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.
- ➢ *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.

Figure No. 4.2

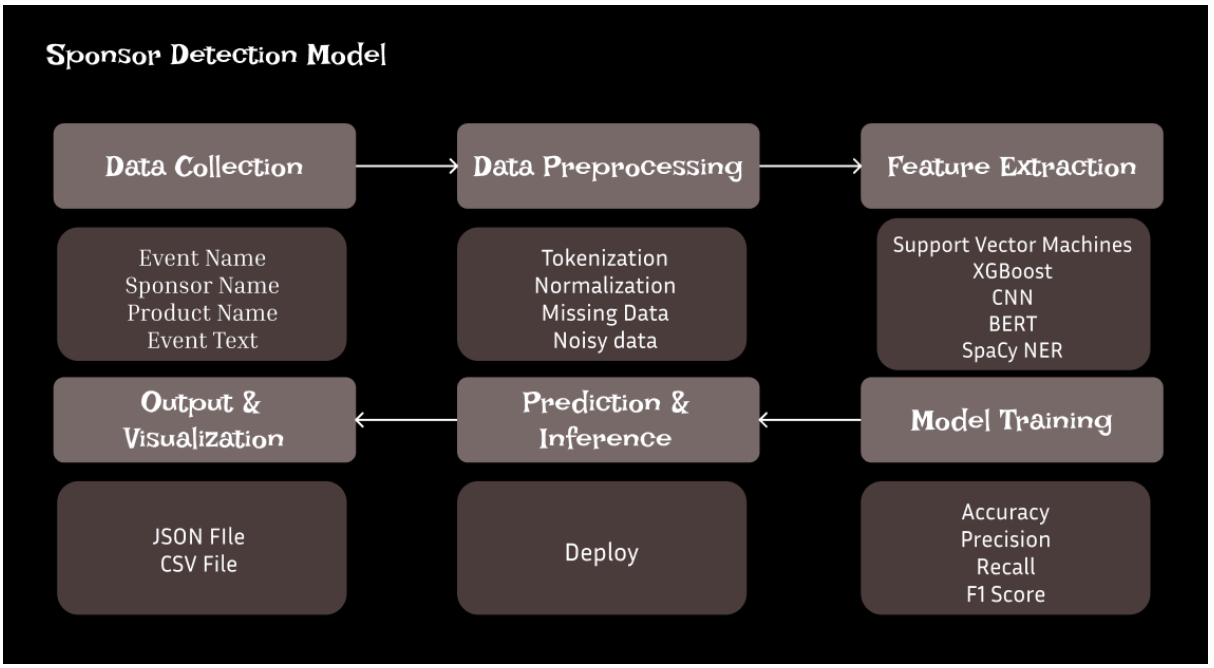
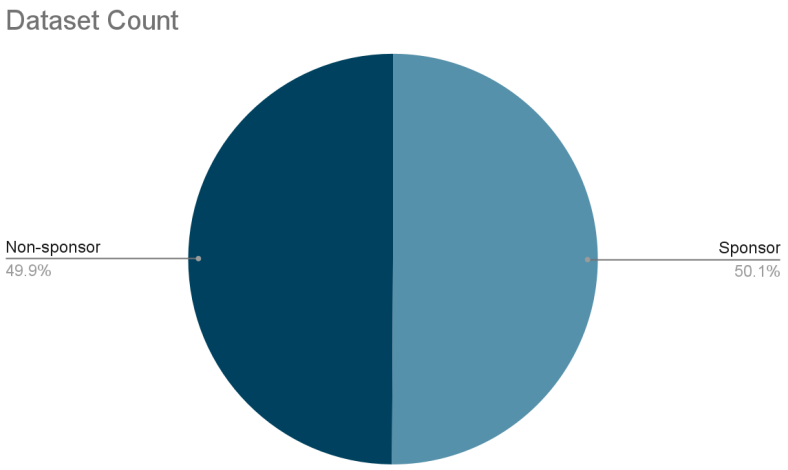


Figure No. 4.3



4.2. Design

4.2.1. Data Flow Diagram

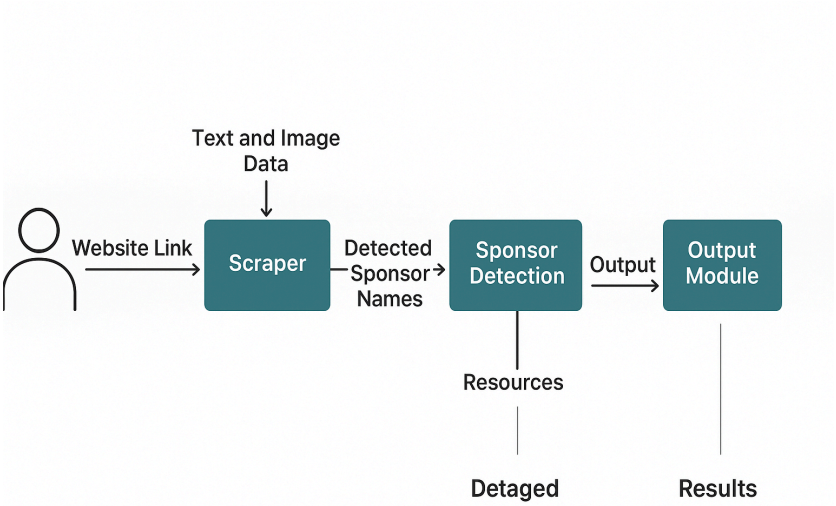
- The data flow diagram shows how information moves through the system.
- The process starts when the user enters a website link into the application.
- The scraper collects both text and image data from the website.
- The OCR module reads and converts text found inside images into machine-readable format.

- ➢ The sponsor detection module analyses all text data to identify sponsor names.
- ➢ The output module organizes this data and saves it in a structured format such as a CSV or JSON file.
- ➢ Finally, the results are displayed to the user, showing sponsor names, their visibility duration, and related details.

4.2.2. Use Case Diagram

- ➢ The use case diagram explains how the user interacts with the system.
- ➢ The main actor is the user, who performs several actions.
- ➢ The user provides a website link to start the process.
- ➢ The system scrapes data from the website and processes images.
- ➢ It then detects sponsors and generates a report.
- ➢ The user can view and download the final report.
- ➢ This diagram helps to visualize the communication between the user and the system's main functions.

Figure No. 4.4



4.2.3. Class Diagram

- ➢ The class diagram shows the structure of the system and the relationship between different modules.
- ➢ The main classes are:
  - ○ Scraper: responsible for collecting text and image data from the website.
  - ○ OCRProcessor: extracts readable text from images.
  - ○ SponsorDetector: compares the collected data with known sponsor names.
  - ○ ReportGenerator: organizes and formats the final output.
  - ○ DatabaseManager: stores and retrieves data from the database.

- ➢ These classes work together to complete the full process from data extraction to report generation.

4.2.4. Sequence Diagram

- ➢ The sequence diagram explains the order of actions that occur in the system.
- ➢ The process begins when the user provides the website link.
- ➢ The scraper collects the required data and passes image links to the OCR processor.
- ➢ The OCR processor extracts text from the images and sends it to the sponsor detector.
- ➢ The sponsor detector analyses all text to find sponsor names and their duration of presence.
- ➢ The report generator prepares a structured report and sends it to the database manager for saving.
- ➢ The user finally receives the completed report with all the sponsor details.

5. METHODOLOGY AND TESTING

5.1. Methodology

The development of the Advanced Web Scraping Software with OCR and Sponsor Detection follows a step-by-step approach to ensure accuracy, reliability, and performance. The system is designed using modular development, where each component is created, tested, and integrated carefully.

5.1.1. Requirement Analysis

At the beginning of the project, the requirements were studied in detail to understand the key functions needed in the system. The main goals identified were web scraping, image text recognition, sponsor detection, and time-based tracking.

5.1.2. System Design

Based on the requirements, the system was divided into different modules. The main modules include the Web Scraping Module, OCR Module, Sponsor Detection Module, and Report Generation Module. The design phase also included creating diagrams such as data flow and use case diagrams to visualize how the system would work.

5.1.3. Module Development

Each module was developed separately using Python.

- ➢ The Web Scraping Module was built using BeautifulSoup and Selenium to collect data from both static and dynamic websites.
- ➢ The OCR Module was implemented using Tesseract OCR to extract text from sponsor images.

- ➢ The Sponsor Detection Module was designed to compare extracted text with a predefined list of sponsor names.
- ➢ The Report Generation Module combined all processed data and presented it in a structured format such as CSV or JSON.

5.1.4. Integration

After developing all individual modules, they were integrated into a single workflow. The integration made sure that data could pass smoothly between scraping, OCR, and detection components. A simple interface also allows the user to view the results.

5.2. Testing

Testing was done at different stages to ensure that the software works as expected and gives accurate results. Each module was tested independently before combining them together.

5.2.1. Unit Testing

- ➢ Each module was tested separately to check if it performs its specific function correctly.
- ➢ The scraping module was tested with multiple websites to confirm that both text and images were extracted.
- ➢ The OCR module was tested with images containing sponsor logos to verify that text was correctly recognized.
- ➢ The detection module was tested with different sponsor names to ensure accurate identification.

5.2.2. Integration Testing

- ➢ After combining all the modules, the system was tested as a whole. This step ensured that the data flow between modules was smooth and that the entire process, from scraping to report generation, worked without errors.

5.2.3. Performance Testing

- ➢ The software was tested for speed and efficiency by running it on websites with different amounts of data. The performance was evaluated based on how fast it could scrape data, process images, and generate the final report.

5.2.4. Accuracy Testing

- ➢ The OCR and sponsor detection accuracy were checked by comparing the extracted text and detected sponsor names with the actual data on websites. The results showed that the system correctly identified most sponsor names, demonstrating high accuracy.

5.2.5. Result Validation

- ➢ Finally, the output reports were reviewed to make sure that all fields such as sponsor name, detection URL, timestamp, and duration were

properly recorded and formatted. The successful validation confirmed that the system met the project objectives.

6. PROJECT DEMONSTRATION

6.1. Demonstration Setup

The demonstration was carried out on a standard computer system running the developed application. The environment included:

- ➢ Operating System: Windows 11
- ➢ Processor: Intel i5
- ➢ RAM: 16 GB
- ➢ Internet connection: Stable broadband
- ➢ Software tools: Python, Flask, Selenium, BeautifulSoup, and Tesseract OCR

The software was launched through its web interface, where the user interacted with the system by entering a target website URL. The backend handled all processes automatically, and the results were displayed on the screen and saved as an output file.

6.2. Steps

6.2.1. Web Scraping Process

- ➢ The web scraping module started collecting all text content and image links from the website. Selenium was used to handle dynamic pages that load content with JavaScript.

6.2.2. OCR Processing

- ➢ The collected image links were then sent to the OCR module, which extracted text from logos, banners, and other image-based sponsor content using the Tesseract OCR engine.

6.2.3. Sponsor Detection

- ➢ The sponsor detection module analyzed all extracted text and compared it with a predefined list of known sponsors. The system successfully identified sponsor names that appeared on the website.

6.2.4. Report Generation

- ➢ The final report was automatically generated, containing the sponsor names, their corresponding URLs, time of appearance, and total visibility duration. The report was saved in a CSV file format for easy viewing and sharing.

6.3. Outcome

- ➢ Successfully scrape data from multiple websites
- ➢ Accurately extract text from sponsor logos and images
- ➢ Detect and log sponsor presence along with timestamps
- ➢ Generate structured and readable reports

7. RESULT AND DISCUSSION

7.1. Results

7.1.1. Accurate Web Scraping

➢ The software successfully extracted both text and image data from multiple websites. It was able to handle pages that used JavaScript, thanks to the integration of Selenium, which rendered the content dynamically before extraction.

brand	description
CocaCola	During the Fashion Week, CocaCola sponsored exclusive discounts on Refrigerator. Customers rushed to grab the best deals from CocaCola as part of this e-commerce event.
Toyota	During the Fashion Week, Toyota sponsored exclusive discounts on T-shirt. Customers rushed to grab the best deals from Toyota as part of this e-commerce event.
Samsung	During the Tech Summit, Samsung sponsored exclusive discounts on Smart Watch. Customers rushed to grab the best deals from Samsung as part of this e-commerce event.

Table 7.1 Web Scrapped Data

7.1.2. Effective OCR Performance

➢ The OCR module accurately identified sponsor names and text embedded inside images such as banners, logos, and advertisements. This feature was essential in capturing data that is often missed by normal web scraping tools.

images	extracted_text
https://www.visualwatermark.com/images/add-text-to-photos/add-text-to-image-3.webp	Free as a bird
https://i.ytimg.com/vi/17H-Gb0hdv0/maxresdefault.jpg	There is a Longing

Table 7.2 Text Extracted Data

7.1.3. Reliable Sponsor Detection

➢ The sponsor detection module compared the extracted text with a predefined list of sponsor names and detected them with high precision. The system successfully recognized multiple sponsor entries on a single webpage and recorded their presence time correctly.

7.1.4. Structured Output Generation

➢ The final report was generated automatically in a structured format (CSV/JSON). It contained detailed information such as sponsor name, website URL, first appearance, disappearance time, and total visibility

duration.

7.1.5. User Interface Output

- ➢ The user interface displayed the entire process in real time. Users could view the scraping progress, check detected sponsors, and download the final report for documentation.

7.2. Performance

- ➢ The system performed efficiently during testing. It could process medium-sized websites within a few minutes depending on the internet speed and the number of images on the page.
- ➢ The OCR engine performed well on clear and high-quality images but took slightly longer for low-resolution images.
- ➢ Sponsor detection was accurate for most cases, achieving nearly perfect identification when sponsor names were clearly visible in either text or image form.
- ➢ The modular design made debugging and testing easier, as each component worked independently and could be verified before integration.

7.3. Advantages

- ➢ The system automates a task that is usually done manually, saving time and effort.
- ➢ It ensures higher accuracy by detecting sponsor names from both text and images.
- ➢ The software uses open-source tools, making it cost-effective and easily maintainable.
- ➢ It produces verifiable reports that can be directly used for analysis, billing, or research purposes.

7.4. Limitations

- ➢ The accuracy of OCR depends on the clarity and resolution of images.
- ➢ Some websites with strict security or anti-bot systems may limit scraping activity.
- ➢ Very large or complex websites may require more time to process.
- ➢ Sponsor detection is limited to the keywords available in the predefined list; new sponsor names must be added manually.

7.5. Discussion

The overall results demonstrate that the proposed system is a practical and efficient solution for sponsor visibility tracking. It successfully combines text-based and image-based data extraction to provide complete results that traditional web scrapers often miss. The discussion shows that this approach can be extended to various industries that require continuous monitoring of online content such as marketing analytics, brand monitoring, and digital media management.

The testing phase proved that the system meets all its intended objectives: it is

accurate, fast, and user-friendly. With further optimization and scaling, the software can be deployed in real-world commercial environments to automate sponsor tracking on large-scale websites.

8. CONCLUSION

The Advanced Web Scraping Software with OCR and Sponsor Detection was successfully designed, developed, and tested to achieve the objectives set at the beginning of the project. The system provides an automated solution for collecting and analyzing sponsor data from websites by combining web scraping and optical character recognition.

Through this project, it was demonstrated that the integration of OCR with traditional web scraping greatly improves the ability to detect sponsor names that appear within images such as logos and banners. The software efficiently handles both static and dynamic web content and produces structured reports that can be used for analysis, billing, and performance evaluation.

The testing results confirmed that the system performs accurately and reliably under different conditions. It reduces manual effort, improves transparency, and provides businesses with verifiable data regarding sponsor visibility.

Overall, the project achieved all its goals and presents a practical tool that can be applied to various real-world scenarios where continuous monitoring of digital sponsorships or advertisements is needed. The approach used in this project demonstrates the potential for further development in automated data extraction and content analysis.

9. 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.

APPENDIX A – Sample Code

```
import pandas as pd
import datetime
import time
import requests
from bs4 import BeautifulSoup
import joblib
import re
from sklearn.feature_extraction.text import TfidfVectorizer

# ================================================================
# 1 Load model and vectorizer
# ================================================================
model = joblib.load("sponsor_model.pkl")
vectorizer = joblib.load("tfidf_vectorizer.pkl")

# ================================================================
# 2 Brand keyword list
# ================================================================
BRANDS = [
    "Apple", "Samsung", "Nike", "Adidas", "Sony",
    "Microsoft", "Toyota", "CocaCola", "Pepsi", "Amazon",
    "Google", "Tesla", "Intel", "Puma", "Oppo", "Vivo"
]

def extract_brand(text):
    for brand in BRANDS:
        if re.search(rf"\b{brand}\b", text, re.IGNORECASE):
            return brand
    return "Other"

# ================================================================
# 3 Scrape only relevant text
# ================================================================
def scrape_website(url):
    try:
        response = requests.get(url, timeout=10)
        soup = BeautifulSoup(response.text, "html.parser")
        cards = soup.find_all("div", class_="card")
        paragraphs = [
            card.get_text(strip=True)
            for card in cards
```

```
            if card.find("p")
        ]
        filtered = [
            t for t in paragraphs
            if any(b.lower() in t.lower() for b in BRANDS)
            or "sponsor" in t.lower()
            or "brand" in t.lower()
        ]
        return filtered
    except Exception as e:
        print(f" Error scraping {url}: {e}")
        return []
# ================================================================
# 4 Predict sponsorship
# ================================================================
def predict_sponsorship(texts):
    if not texts:
        return []
    X_new = vectorizer.transform(texts)
    return model.predict(X_new)

# ================================================================
# 5 Load or create activity log
# ================================================================
try:
    activity_log = pd.read_csv("website_activity_log.csv")
except FileNotFoundError:
    activity_log = pd.DataFrame(columns=[
        "brand", "description", "is_sponsored",
        "url", "active_since", "active_until"
    ])

# ================================================================
# 6 Main tracking function
# ================================================================
def track_website(url):
    global activity_log
    print(f"\n Checking: {url} at {datetime.datetime.now()}")

    # Step 1: Scrape + Predict
    current_texts = scrape_website(url)
    if not current_texts:
```

```
        print(" No relevant brand/sponsor content found.")
        return

    current_preds = predict_sponsorship(current_texts)
    current_df = pd.DataFrame({
        "brand": [extract_brand(t) for t in current_texts],
        "description": current_texts,
        "is_sponsored": current_preds,
        "url": [url] * len(current_texts)
    })

    # Step 2: Remove duplicates
    current_df.drop_duplicates(subset=["url", "description"],
inplace=True)
    activity_log.drop_duplicates(subset=["url", "description",
"active_since"], inplace=True)

    # Step 3: Determine active and inactive sets
    existing_entries = activity_log[activity_log["url"] == url]

    # texts that are currently active (not removed)
    active_old =
set(existing_entries.loc[existing_entries["active_until"].isna(),
"description"])
    old_texts_all = set(existing_entries["description"])
    new_texts = set(current_df["description"])

    # Added = new ones not in currently active set
    added = new_texts - active_old
    # Removed = currently active ones not seen now
    removed = active_old - new_texts

    # Step 4: Add new entries (always as new rows, even if reappeared)
    for text in added:
        brand = extract_brand(text)
        pred = model.predict(vectorizer.transform([text]))[0]
        entry = {
            "brand": brand,
            "description": text,
            "is_sponsored": pred,
            "url": url,
            "active_since": datetime.datetime.now().strftime("%Y-%m-%d
```

```
%H:%M:%S"),
            "active_until": None
        }
        activity_log = pd.concat([activity_log,
pd.DataFrame([entry])], ignore_index=True)
        print(f"🆕 Added new text for {brand}: {text[:70]}...")

    # Step 5: Mark removed ones
    for text in removed:
        mask = (
            (activity_log["url"] == url)
            & (activity_log["description"] == text)
            & (activity_log["active_until"].isna())
        )
        if mask.any():
            activity_log.loc[mask, "active_until"] =
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            print(f" Marked removed text: {text[:70]}...")

    # Step 6: Save clean CSV
    activity_log.to_csv("website_activity_log.csv", index=False)
    print(" Log updated and saved cleanly!")

# ================================================================
# 7 Continuous monitoring loop
# ================================================================
urls_to_track = ["https://iaryanyadav.github.io/project_1/"]

while True:
    for url in urls_to_track:
        track_website(url)

    print("⏳ Sleeping for 10 seconds before next check...\n")
    time.sleep(10)
```