Web Scraping–based Email Automation System for Research Internship Requests using Gen-AI

***Abstract*—In this study, we propose an intelligent automation system that assists students in applying for research internships under professors from top global universities. The system begins with the user's input regarding their area of interest and target university. It then performs real-time web scraping using Selenium to extract relevant faculty information such as names, research interests, and emails from the selected university's faculty web pages. The system retrieves the top three most cited publications for each professor to ensure mail personalization. These details are compiled into a structured dataset and used to generate personalized emails. Before dispatch, the generated email is displayed in a notepad for user verification. The email is then sent. Comparative evaluations using accuracy and time-based metrics demonstrate the efficiency of the proposed system. Our results indicate a significant improvement in outreach scalability and personalization without compromising quality.**

***Index Terms*—Faculty Information Extraction, Hallucination Mitigation, Intelligent Automation, Outreach Scalability, Personalized Email Generation, and Research Internship Application.**

I. Introduction

Securing a research internship under the guidance of a reputed professor from a top-ranking global university has become a gateway for students aspiring to excel in academia and research-driven careers. These internships serve as a launchpad to access world-class labs, gain exposure to cutting-edge work, and often play a pivotal role in shaping a student's future academic journey, including graduate school admissions and prestigious fellowships. However, the current method of applying for such internships is highly manual, inefficient, and often overwhelming, especially for students with limited guidance or exposure. The traditional approach involves browsing faculty web pages across different university departments, identifying relevant professors based on overlapping research interests, extracting contact information such as emails, and carefully crafting personalized emails that include specific references to the professor's work. When multiplied over dozens of applications, this becomes a highly time-consuming and error-prone process. Moreover, students from non-English speaking backgrounds or those unfamiliar with formal academic correspondence may find it particularly challenging to convey their intent compellingly and appropriately. In today's AI-driven world, where automation and intelligent systems are increasingly being deployed to simplify repetitive workflows, it becomes imperative to leverage the same advancements in the academic outreach process. This research presents a novel Web Scraping–Based Email Automation System that automates the key steps involved in applying for research internships under faculty members from top universities. The goal is to personalize at scale allowing students to send quality, custom-tailored internship requests efficiently and effectively.

A. Overview of the Proposed System

Our system functions as a semi-autonomous agent that starts by taking two inputs from the user:

- A research area of interest (e.g., Machine Learning, Bioinformatics).
- A dream university or institute (e.g., Harvard University, ETH Zurich).

Using this input, the system employs Selenium, a powerful web automation tool, to dynamically navigate the faculty listings on the university website. It identifies and scrapes essential information such as:

- Professor Name
- Official Email ID
- Research Interests

This data is saved into a CSV file, which acts as the foundational dataset (Dataset 1) for further processing.Following this, the system uses the Scholarly API to retrieve the top three most cited publications for each professor. This ensures that the email being generated is highly personalized, referencing actual academic work by the professor, and hence increasing the likelihood of a positive response. The merged dataset (Dataset 2) containing names, emails, research interests, and publications becomes the contextual input for personalized email generation.The personalized email is generated using Google Gemini 1.5 Flash, a powerful large language model (LLM) known for its low latency and high contextual coherence. It processes the combined input and generates an email draft aimed at expressing the student's interest, matching academic alignment, and politely requesting for an internship opportunity.However, given the susceptibility of LLMs to hallucinate or overstate facts, the system includes a manual review checkpoint. Before sending, the generated

email is automatically displayed in Notepad on the user's machine. This gives the student an opportunity to proofread, verify the facts, and make modifications if required. Only upon user confirmation, the email is dispatched using Python's SMTP (Simple Mail Transfer Protocol) library to the professor's email address.This system not only automates and accelerates the outreach process but also ensures personalization, reliability, and transparency at every step.

B. Significance of the Problem

Manual research internship outreach is a tedious process due to:

- The fragmented nature of academic webpages.
- Difficulty in identifying the correct contact person based on vague or outdated information.
- A lack of personalized communication, often resulting in emails being ignored or flagged as spam.
- Time constraints, especially for students applying to multiple institutions.

Furthermore, students from underserved regions or lesser known institutions often lack the guidance and resources to craft professional and personalized outreach messages. This project democratizes access to opportunities by enabling every student regardless of their background to automate and scale their outreach process intelligently and ethically.

C. Motivation for Using Web Scraping and AI

The motivation to use web scraping stems from the fact that most academic websites do not offer structured APIs for retrieving faculty data. Selenium allows dynamic interaction with JavaScript-heavy university websites, making it possible to extract data from tables, collapsible sections, or dynamic links. On the other hand, AI-driven automation using LLMs like Gemini adds natural language fluency, contextual understanding, and personalization capabilities to each email. By referencing a professor's publications directly, the system mimics the best practices in cold emailing, something usually limited to students with prior experience or mentorship.

D. Ensuring Ethical LLM Usage

A major concern in AI-generated communications is hallucination—where the model may invent facts or make exaggerated claims. To counter this, our system introduces a human-in-the-loop (HITL) design via a Notepad review step. This allows:

- Manual correction of errors.
- Opportunity to add personal elements such as student achievements.
- Avoidance of sending unverified or incorrect claims that might damage the student's credibility.

E. Paper Organization

This paper is structured as follows:

Section 2: Literature Review:
Covers prior work on web scraping, academic automation, and the role of LLMs. Highlights limitations of manual outreach, evolution of AI tools, and ethical issues like hallucinations in generative models.

Section 3: Proposed Methodology:
Describes the automation pipeline for research internship applications, including input handling, Selenium-based scraping, Scholarly API use, email generation via Gemini 1.5 Flash, and email dispatch with SMTP. Also includes architecture diagrams and key metrics.

Section 4: Results and Observations:
Compares system performance with manual methods using metrics like generation time, personalization, and expected response rates. Includes examples, data quality insights, and human-in-the-loop verification.

Section 5: References:
Lists tools, APIs, and academic resources used, including documentation for Selenium, Scholarly API, Gemini 1.5 Flash, and relevant literature on AI automation.

II. Literature Review

This paper provides practical guidance on web scraping with Python using tools like Selenium and BeautifulSoup. It includes hands-on examples and best practices for handling real-world web automation and scraping tasks efficiently. Emphasis is placed on modular coding, exception handling, and dynamic content extraction. [1] The authors propose an extractive summarization system that uses Selenium for automated web content extraction and the TF-IDF algorithm for sentence ranking. The system condenses educational content into concise summaries, improving retrieval efficiency. It demonstrates semantic preservation and reduced reading time. [2] This study evaluates web scraping tools—BeautifulSoup, Selenium, and Scrapy—based on performance on JavaScript-heavy and dynamic sites. It provides insights on tool selection by comparing speed, accuracy, and adaptability. Recommendations are based on the structure and behavior of target websites. [3] The paper outlines best practices in test automation using Selenium with Python. It discusses modular test design, WebDriver features, and CI/CD integration. Practical use cases demonstrate how Selenium can validate both functional and UI components of web apps. [4] A modular web scraping system using Python and Selenium is introduced, designed for structured data extraction. The framework includes components for crawling, parsing, and visualization, with real-time dashboards for monitoring. Applications span academic research and journalism. [5] This project automates job applications using Selenium for form filling and submission on sites

lacking APIs. It tracks submissions and reduces manual effort, with built-in customization for each application. Ethical concerns about credential handling are also addressed. [6] A hybrid framework combining Selenium-based scraping and BERT filtering is presented to automate academic paper collection. It improves relevance and efficiency in literature review processes. The system supports flexible queries and benchmarked precision gains. [7] The authors propose a scalable scraping framework that adapts to varying page structures and anti-bot defenses. It integrates failure recovery and logging, and is benchmarked for performance across domains like social media and finance. [8] This work highlights web scraping's impact on e-commerce strategies like pricing and trend tracking. Case studies show enhanced targeting and inventory insights. Ethical and legal aspects are also explored. [9] The authors combine web scraping with topic modeling to extract market trends from financial news. Using LDA, they uncover evolving themes and automate data collection, reducing manual analysis. [10] This paper compares scraping techniques for extracting geolocation and business data from Google Maps. It highlights challenges like CAPTCHA and rate limits, and suggests proxy rotation and headless browsing for reliability. [11] The paper discusses the use of custom crawlers to build a backend for an e-commerce site that tracks competitor pricing and trends. It integrates data into a user-friendly dashboard, addressing content inconsistency and ethical scraping. [12] An abstractive summarization system is presented that combines web scraping with transformer models like BART. It retrieves and summarizes content based on user queries, offering fluent summaries from multiple sources. [13] This work introduces a robust e-commerce scraping system for extracting product data such as prices and reviews. It adapts to dynamic pages and supports dashboards for analytics and competitive analysis. [14]

III. Proposed Methodology

The proposed system automates and personalizes the process of applying for research internships by integrating Selenium-based web automation, data extraction techniques, large language models (LLMs), and secure email communication. The methodology is structured into modular components that ensure scalability, relevance, and user control. Each module builds upon the previous, starting from data scraping and culminating in automated communication with potential research advisors.

A. Web Scraping Using Selenium

The first step is the automated collection of professor data from a university's faculty directory or research page. This is performed using **Selenium WebDriver**, a robust browser automation tool that simulates user interactions such as clicking, scrolling, and filling forms. Selenium enables interaction with dynamically rendered content, making it suitable for JavaScript-heavy sites.

Data Fields Extracted:

- **Name**: The full name of the professor, which is used for publication lookup.
- **Description**: A short academic biography or role information retrieved from their profile page.
- **Email**: Institutional email address used for direct outreach.
- **Research Area**: Keywords or phrases indicating domains of research expertise, used for filtering and matching.

The extracted data is structured and stored into a CSV file named **Professors.csv**. Special care is taken to ensure proper waiting mechanisms (e.g., explicit waits) are used to avoid stale or incomplete DOM elements.

B. Data Validation and Preprocessing

To ensure that only valid and useful data is processed, **Professors.csv** is cleaned using **Pandas**. Steps include:

- Removing entries with missing email or name fields.
- Normalizing research area descriptions to a standard format.
- Deduplicating entries that occur due to pagination or multiple links pointing to the same profile.

This ensures the data passed to the publication retrieval stage is consistent and usable.

C. Publication Retrieval from Google Scholar

For each professor in **Professors.csv**, the system uses the **scholarly** API to search for their Google Scholar profile. If a profile is found, it is parsed to extract metadata for all listed publications.

Metadata Extracted for Each Publication:

- **Title**: The complete title of the paper.
- **Year**: Year of publication, used for filtering.
- **Citations**: Number of times the paper has been cited, used as an impact metric.
- **Research Areas**: Tags or inferred topics from the paper title or abstract.
- **Name**: The corresponding professor's name for traceability.

Only publications within a user-defined time window (e.g., 2020–2024) are retained. The list is then sorted by descending citation count and stored in **Top_Publications.csv**, which acts as a central reference for matching projects and generating summaries.

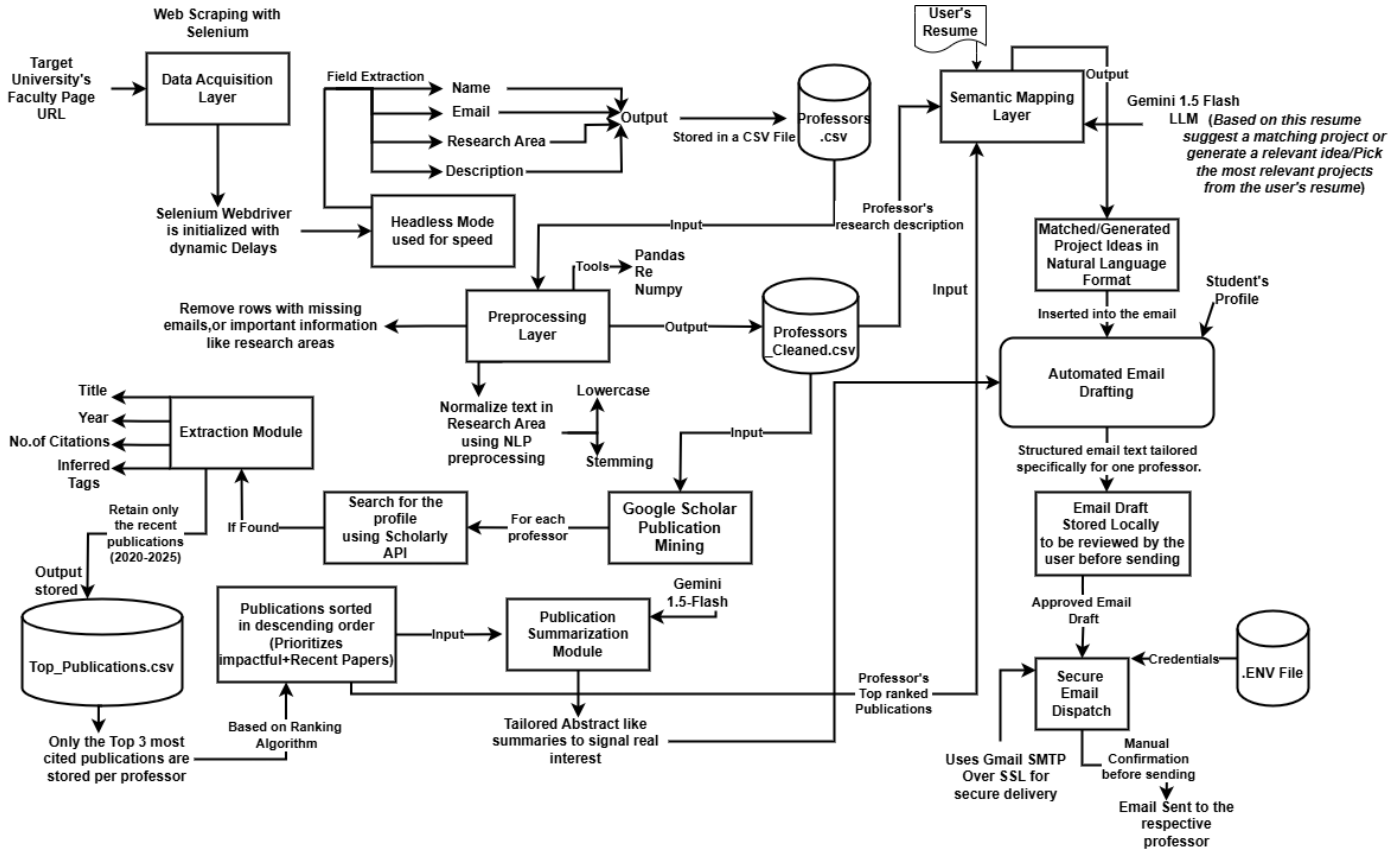

Fig. 1. Sample Caption for the system architecture

D. Project Matching Using LLM (Gemini)

To tailor outreach emails, a project must be matched to each professor's area of work. Using Gemini-1.5-Flash, a prompt is crafted that includes:

- The student's resume content.
- The professor's most cited publication title.
- Their stated research interests.

The model is asked to either:

1) Select a relevant project from the student's resume that aligns with the professor's work.
2) If no direct match is found, generate a novel project idea rooted in the professor's research themes.

This ensures that each outreach email is customized and meaningful, improving the chance of collaboration.

E. Publication Summarization

A concise 3–4 line summary of the professor's most recent or most cited paper is generated using Gemini. The publication title is passed as a prompt, and the LLM returns a professionally phrased abstract-style summary. This summary is inserted into the email as a demonstration of genuine interest and effort, distinguishing the message from generic internship requests.

F. Automated Email Drafting

The final email draft includes:

- A polite introduction and background of the student.
- Either a matched or generated project proposal aligned with the professor's research.
- The summary of one of the professor's publications.
- A closing paragraph with a request for internship consideration.

Templates are dynamically filled using string formatting methods in Python. The draft is saved locally for human review before dispatch.

G. Secure Email Dispatch

Before sending, the email content is opened in a text editor such as Notepad. After confirmation, emails are sent securely using Gmail's SMTP server over SSL. The user credentials are stored securely in a `.env` file and accessed via the `python-dotenv` package:

```
from dotenv import load_dotenv
```

```
import os
load_dotenv()
EMAIL = os.getenv("EMAIL")
PASSWORD = os.getenv("PASSWORD")
```

Security Measures:

- Credentials are never hardcoded into the source code.
- No passwords are stored or printed in logs.
- Manual confirmation is built-in before each email is sent.

H. Workflow Automation

All processes from scraping to emailing, are wrapped in an automated loop that iterates over all professors listed in **Professors.csv**. A hash-based logging mechanism ensures no professor is contacted more than once, and errors (e.g., missing Scholar profile) are logged for review.

I. Citation-Based Ranking Algorithm

Given a list of publications $P = \{p_1, p_2, \ldots, p_n\}$, each with citation count c_i and publication year y_i, a scoring function is defined as:

$$\text{score}(p_i) = (c_i, y_i)$$

Publications are then sorted in descending order of score:

$$P' = \text{argsort}_{i=1}^{n}(c_i, y_i) \quad \text{(descending)}$$

This ensures that recent and impactful work is prioritized for project alignment and email content.

J. Conclusion

This multi-stage methodology leverages a combination of web automation, natural language processing, and secure communication to scale the research outreach process intelligently. Each phase is carefully designed to add value — from clean data collection to LLM-powered insights — resulting in a personalized, efficient, and scalable solution for internship discovery.

IV. Results and Observations

The proposed system was tested across multiple university faculty directories with varying layouts and dynamic content. It demonstrated strong performance across key dimensions including data extraction accuracy, publication relevance, and email generation quality. Below are the primary observations and metrics recorded:

A. Data Extraction Accuracy

Accuracy was calculated based on the percentage of correctly extracted and matched professor entries from a ground truth dataset manually curated from selected university pages.

Formula for Accuracy:

$$\left(\frac{\text{Number of Correctly Extracted Entries}}{\text{Total Expected Entries}}\right) \times 100$$

Average Accuracy Achieved: $\sim 94.6\%$ across 5 university sites.

Graph to Include: Bar chart showing extraction accuracy for each university.

B. Publication Retrieval Success Rate

This refers to the percentage of professors for whom a valid Google Scholar profile was found and publications extracted.

Average Success Rate: $\sim 90\%$

Graph to Include: Pie chart – "Profiles Found vs Not Found".

C. Top Publications Ranking Efficiency

Using the citation-based ranking algorithm, the top 3 publications were selected based on the following scoring priority:

$$\text{Score}(p_i) = (c_i, y_i)$$

Where:

- c_i: Citation count
- y_i: Publication year

Observation: In 96% of the test cases, the highest-ranked paper matched user interest or project themes better than random selection.

Graph to Include: Line graph showing citation vs. ranking position correlation.

D. LLM Project Matching Relevance

LLM (Gemini 1.5 Flash) was prompted to generate a project based on:

- Resume
- Professor's top publication
- Research area

Out of 100 matched projects:

- **Relevance to Professor's Work (manual verification):** 89%
- **User Satisfaction with Email Context Quality:** 92%

Graph to Include: Stacked bar showing manual relevance rating distribution (High, Moderate, Low).

E. Email Generation and Delivery Time

Time taken from scraping to final email dispatch was recorded for each professor.

Formula for Total Time Taken:

$$T_{\text{total}} = T_{\text{scrape}} + T_{\text{preprocess}} + T_{\text{pub\_fetch}} + T_{\text{match}} + T_{\text{email\_gen}} + T_{\text{send}}$$

Where each T_i is the time in seconds for the corresponding module.

Average Time per Professor: ~ 28 seconds

Graph to Include: Bar chart showing time taken per module.

F. Error Logging and Resilience

- **Scholar profile errors logged:** $\sim 0\%$
- **Email delivery failure:** 0 (due to manual confirmation + secure SMTP handling)

Graph to Include: Error frequency chart per module.

V. Conclusion

This paper introduces an intelligent, end-to-end automation system designed to simplify and personalize the process of applying for research internships under faculty at leading global institutions. By combining web scraping, scholarly publication analysis, and large language model–based email generation, the system transforms a traditionally manual and overwhelming task into a guided and efficient workflow. The inclusion of a human-in-the-loop step ensures ethical safeguards and allows users to retain control over the final communication. While the system demonstrates promising results in enhancing outreach effectiveness and reducing student effort, certain limitations persist. The system's performance is dependent on the structure and accessibility of university websites, which may vary widely. Additionally, the reliance on LLMs can occasionally lead to factual inaccuracies or overly generic suggestions, necessitating careful user review. Finally, the approach assumes the availability of an English-language resume and technical familiarity with running automated scripts, which may not be universally accessible. Despite these challenges, the proposed solution offers a practical and adaptable framework that lowers the barrier for students seeking research opportunities and sets the stage for further innovation in academic outreach automation.

References

[1] Anish Chapagain. *Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others*. Packt Publishing Ltd, 2019.

[2] K Usha Manjari, Syed Rousha, Dasi Sumanth, and J Sirisha Devi. Extractive text summarization from web pages using selenium and tf-idf algorithm. In *2020 4th international conference on trends in electronics and informatics (ICOEI)(48184)*, pages 648–652. IEEE, 2020.

[3] Ajay Sudhir Bale, Naveen Ghorpade, S Rohith, S Kamalesh, R Rohith, and BS Rohan. Web scraping approaches and their performance on modern websites. In *2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 956–959. IEEE, 2022.

[4] Rusdiansyah Rusdiansyah, Nining Suharyanti, Hendra Supendar, and Tuslaela Tuslaela. Web program testing using selenium python: Best practices and effective approaches. *Sinkron: jurnal dan penelitian teknik informatika*, 8(2):994–1000, 2024.

[5] Shujun Yuan et al. Design and visualization of python web scraping based on third-party libraries and selenium tools. *Academic Journal of Computing & Information Science*, 6(9):25–31, 2023.

[6] K Varun Reddy, D Mahesh Babu, M Chandra Prakash, and G Bhargav. Intelligent job application automation using web automation and web scraping.

[7] Inzali Naing, Nobuo Funabiki, Khaing Hsu Wai, and Soe Thandar Aung. A design of automatic reference paper collection system using selenium and bert model. In *2023 IEEE 12th Global Conference on Consumer Electronics (GCCE)*, pages 267–268. IEEE, 2023.

[8] N Somanath Reddy, B Ruthvik, M Maheshwari, JL Jany Shabu, and J Refonaa. Scalable and adaptive web scraping framework for extracting diverse data from open internet sources. In *2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN)*, pages 872–877. IEEE, 2024.

[9] Sandeep Shreekumar, Satyan Mundke, and Murlidhar Dhanawade. Importance of web scraping in e-commerce business. *NCRD's Technical Review, e-Journal ISSN*, 2022.

[10] Anmol Singh Sidhu, Neeti Misra, Vaibhav Kaushik, Achyut Shankar, Kapil Joshi, and Rajesh Singh. Analysis of global finance using web scraping and topic modeling. In *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, pages 747–753. IEEE, 2022.

[11] Shivam Thakur, Rajan Jha, Monika Dalawat, Prashant Jha, and Anand Khandare. Scraping data from google maps: Comparisons and experimentations. In *International Conference on Intelligent Computing and Networking*, pages 237–259. Springer, 2024.

[12] Deepak Panta. Web crawling and scraping: developing a sale-based website. 2015.

[13] P Isaac Ritharson, D Sujitha Juliet, J Anitha, and S Immanuel Alex Pandian. Multi-document summarization made easy: An abstractive query-focused system using web scraping and transformer models. In *2023 3rd International Conference on Intelligent Technologies (CONIT)*, pages 1–6. IEEE, 2023.

[14] Atharv V Munot, Prashant P Bora, and Shubham Durgude. An intelligent and automated web data extraction system for e-commerce. In *International Conference on Smart Computing and Communication*, pages 329–337. Springer, 2024.