**AUTOMATE LINKEDIN LOGIN USING PYTHON**

Preetham Kacham 1,Vishnu Vardhan Yenugula 1, Dhanunjay Gandla 1

***Department of Computer Science and Engineering, SRM University – AP, Andhra Pradesh, India***

**Email: vishnuvardhan\_y@srmap.edu.in**

**ABSTRACT:** LinkedIn has emerged as a vital tool for businesses and entrepreneurs in building their brand presence. Companies utilize LinkedIn to showcase their corporate culture, values, and achievements. This not only helps in attracting potential clients and customers but also in establishing credibility within the industry. Many businesses leverage LinkedIn Company Pages to provide updates on products or services, share thought leadership content, and engage with their audience. The platform's global reach has made it an essential tool for international networking and collaboration. Professionals can connect with counterparts from around the world, fostering cross-cultural exchanges and partnerships. This global networking capability is particularly valuable for individuals and businesses looking to expand their reach and explore opportunities in new markets. LinkedIn's influence is not limited to full-time employment; it has also become a significant platform for freelancers and gig economy workers. Freelancers can showcase their skills and expertise, attract potential clients, and secure freelance opportunities. This flexibility in catering to various forms of professional engagement has contributed to LinkedIn's widespread adoption across diverse sectors of the workforce. Additionally, the platform has introduced features like LinkedIn Live, allowing users to broadcast live videos, webinars, and virtual events. This has added a dynamic and real-time element to professional interactions, enabling individuals and businesses to connect with their audience in a more engaging way. In the evolving landscape of remote work, LinkedIn has played a crucial role in connecting remote professionals. The platform facilitates the formation of virtual communities where individuals can share experiences, discuss challenges, and provide support, creating a sense of belonging in the digital work environment. In conclusion, LinkedIn's multifaceted role encompasses not only individual career growth and company recruitment but also extends to global networking, brand building, freelance opportunities, and real-time engagement. As technology and professional dynamics continue to evolve, LinkedIn remains at the forefront, adapting and enhancing its features to meet the diverse needs of the ever-changing professional world.

**KEYWORDS:** Social Network; LinkedIn; Python; Web-Drivers; Selenium; User Input; Security; SRM AP Page.

**INTRODUCTION:** Beyond the factors previously mentioned, a pivotal aspect driving LinkedIn's remarkable growth is its transformation into a comprehensive professional ecosystem. Far beyond merely facilitating job searches and networking, LinkedIn has broadened its scope with features like LinkedIn Learning. This platform provides users with access to an extensive library of courses, catering to individuals keen on continuous professional development. This strategic diversification has significantly expanded LinkedIn's user base, attracting not only job seekers and networkers but also those committed to honing their skills and knowledge. The success of the Selenium and WebDriver integration in the script can be attributed to their adaptability across diverse web environments. Selenium's cross-browser compatibility empowers users to extend the script's functionality to different browsers, offering a versatile solution for individuals with varied preferences. The script's capability to interact seamlessly with LinkedIn's intricate web elements underscores the effectiveness of the Python-Selenium synergy in automating complex tasks. Furthermore, the script's utility transcends mere login automation. In scenarios involving repetitive login procedures, such as testing or data extraction tasks, this script emerges as a valuable tool, enhancing efficiency and precision. Automation not only diminishes the likelihood of human error but also liberates users to concentrate on more strategic aspects of their work. The incorporation of ChromeWebDriver path and ChromeOptions in the script exemplifies a commitment to user customization and configuration. This adaptability ensures that the script can be effortlessly modified to accommodate changes in browser versions or user preferences, contributing to its sustainability as a reliable tool over time. Moreover, the script's automatic interaction with the LinkedIn page, facilitated by the input of username and password, reflects a user-friendly approach. By simplifying the login process, the script enhances the overall user experience, making it accessible to a broader audience, including those who may not be well-versed in the intricacies of programming. This user-centric design aligns with LinkedIn's commitment to inclusivity and ease of use, further solidifying its standing as a platform catering to diverse professional needs.

**SELENIUM:** Selenium's prominence in the domain of browser automation stems from its status as a widely adopted, open-source tool designed to facilitate comprehensive website testing across diverse browsers. Its core function revolves around empowering users to conduct meticulous tests, allowing for the simulation of various actions such as button clicks. Through this process, Selenium evaluates the provided content and navigates through the entire website, generating expected outcomes for users. The tool's flexibility is evident in its support for multiple programming languages, including Python, Java, and C#, ensuring accessibility to a broad community of developers with varying language preferences. An essential element of Selenium's toolkit is the Selenium WebDriver, which provides a robust programming interface for precise control over browser actions and interactions with web elements. This capability enhances the tool's utility beyond testing, enabling users to engage in web scraping and automate repetitive tasks. The tool's compatibility with an array of browsers, including Chrome, Firefox, Safari, and Internet Explorer, ensures thorough test coverage across different environments. Selenium Grid further amplifies the tool's capabilities by facilitating parallel test execution across multiple machines and environments. This scalability feature streamlines testing processes and contributes to more efficient and comprehensive testing procedures. The open-source nature of Selenium plays a pivotal role in fostering collaboration within the development community. This collaborative environment has led to continuous improvements and updates, ensuring that Selenium remains at the forefront of browser automation tools. The tool's active community and extensive documentation serve as valuable resources, offering support, tutorials, and solutions. This collaborative learning environment contributes to the tool's enduring popularity and underscores its position as a go-to solution for developers and QA professionals seeking reliable and versatile browser automation capabilities.

**WEB-DRIVERS:** A Web-Driver serves as the superhero of browser automation, operating as a crucial framework that interprets commands and communicates them to a web browser—essentially acting as a remote control for the internet. This dynamic interface, akin to a wizard behind the curtain, orchestrates actions on the web, making it an indispensable tool for testing websites and verifying their functionality. The versatility of Web-Drivers is comparable to multilingual experts, adept at comprehending commands in various programming languages and ensuring seamless execution across different browsers. This adaptability is vital because browsers exhibit diverse behaviors, and Web-Drivers play a key role in harmonizing these differences. Consider the scenario where automation involves tasks like clicking buttons, completing forms, or navigating between pages. The Web-Driver steps in to autonomously manage these actions, providing developers with a robotic ally that handles repetitive tasks, allowing human developers to focus on more engaging and creative aspects of their work. Web-Drivers boast advanced capabilities, including handling pop-ups, alerts, and frames, making them exceptionally useful for testing intricate web applications where challenges might arise. They also support headless browser testing, allowing them to operate without displaying the browser interface. This covert functionality resembles a stealthy mission, executed quietly and efficiently. In the realm of automation, Web-Drivers adhere to a client-server model, where the Web-Driver serves as the savvy sidekick, dispatching commands to the browser (the superhero) on the server side. This collaborative approach ensures efficient automation across diverse systems and operating environments, guaranteeing that websites function seamlessly for users worldwide. Adding to their repertoire, Web-Drivers possess the ability to capture screenshots during tests, functioning as a forensic tool to analyze issues or showcase the impressive capabilities of the code. This multifaceted utility positions Web-Drivers as the superheroes of the coding world, streamlining web automation for developers and contributing to a flawless internet experience for users.

**CHROME WEB-DRIVER PATH:** This dedicated server path operates as an autonomous entity within the Selenium framework, serving as a specialized server with a singular purpose: to efficiently launch Google Chrome. The criticality of this path becomes evident when considering the challenges that may arise when attempting to execute test scripts using the Selenium Driver without the indispensable support of the Chrome Web-Driver. In the expansive field of web automation and testing, the Chrome Web-Driver emerges as a linchpin, acting as a vital link connecting the Selenium framework with the intricacies of the Google Chrome browser. The Chrome Web-Driver plays a pivotal role as a facilitator, ensuring a seamless interaction between Selenium and Chrome and, in turn, guaranteeing the accurate and efficient execution of test scripts. Without this dedicated driver, the execution of Selenium test scripts on the Chrome browser could face considerable obstacles. This standalone server path acts as a specialized gateway, streamlining the testing process and mitigating potential challenges arising from the nuanced behavior of specific browsers. More than just a technical component, this server path is a strategic facilitator, providing a direct link to launch Google Chrome and allowing Selenium to harness the full capabilities of the browser for comprehensive and reliable script execution. Its existence acknowledges the critical role that the Chrome Web-Driver plays in the testing landscape, addressing intricacies and complexities associated with browser automation. This dedicated server path establishes a reliable foundation for the successful execution of Selenium test scripts, contributing to the overall efficiency of web testing processes. Furthermore, the Chrome Web-Driver enhances the robustness of the Selenium testing framework by offering compatibility and synchronization between Selenium commands and the functionalities of the Chrome browser. Its role extends beyond mere compatibility, incorporating features that elevate the precision and reliability of automated testing, ensuring the accuracy of test outcomes. In conclusion, this standalone server path is integral to the successful integration of Selenium and Google Chrome for effective script execution. It not only addresses technical intricacies but strategically facilitates a seamless collaboration, underscoring the symbiotic relationship between Selenium and the Chrome Web-Driver in the ever-evolving landscape of web testing.

**RELATED WORK:** Our script, inspired by YouTube tutorials and information gleaned from Wikipedia, functions as a savvy computer companion, demonstrating an intelligent approach to automating tasks on LinkedIn through a seamless combination of Selenium and Python. Picture it as an adept guide, providing your computer with clear and concise instructions to execute tasks effortlessly. Upon initiation, the script orchestrates the opening of the LinkedIn page and facilitates login by utilizing the user-provided details, efficiently clicking the Sign-In button. However, what sets our script apart is the thoughtful addition of a 'Detach' trick, ensuring that the browser doesn't hastily close after completing its operations. This clever maneuver is akin to leaving the door open for users to explore the LinkedIn page further after the script has accomplished its tasks. In essence, our script acts as a friendly assistant, leveraging the power of Selenium and Python to simplify the LinkedIn experience. From acquiring knowledge about cool tricks on YouTube to implementing them seamlessly in our script, each step represents a delightful move that showcases the ingenuity of your computer with a bit of guidance. Let's delve into the key steps of our script's procedure:

1. **Importing Selenium Libraries:** The procedure kicks off by importing the necessary Selenium libraries, establishing the foundation for browser interactions within the script. Path **Configuration for Chrome-Driver:** The script ensures a smooth execution by verifying the clear path to Chrome-Driver and aligning it with the local environment, ensuring compatibility.
2. **Preventing Automatic Page Closure with 'Detach':** A unique feature, 'Detach,' is incorporated to prevent the automatic closure of the page post-script execution, providing users with the opportunity to explore further.
3. **Initializing Chrome Web-Driver:** The script takes charge by initializing the Chrome Web-Driver, steering the browser directly to the LinkedIn login page.
4. **Pause for Loading:** A calculated pause of 5 seconds is introduced to allow the script to accommodate the loading time of the LinkedIn login page, ensuring a smooth user experience.
5. **Locating HTML ID's for Login Credentials:** The script intelligently identifies the HTML ID's associated with login and password fields, seamlessly filling in the provided user data in the respective fields.
6. **XPath for Sign-In Button:** Leveraging XPath, the script precisely locates the Login/Sign-In button, ensuring a reliable and accurate execution of the login process.

**BACKGROUND WORK:** The LinkedIn Automation Script is a powerful tool that utilizes Selenium libraries to establish seamless communication between your computer and the LinkedIn browser. This allows the script to automate various tasks, saving you time and effort while expanding your reach on the platform.

* **The Role of Selenium Libraries:** Libraries serve as the backbone of the script, providing a set of instructions that enable precise navigation of the LinkedIn website. These libraries act as a bridge, translating your commands into actions that the browser can understand and execute.
* **A Journey to the LinkedIn Login Page:** The script's first task is to transport you to the LinkedIn login page. This involves launching a driver, which is like having a personal chauffeur at your disposal. The driver skillfully navigates the web's complexities, ensuring a smooth and direct route to the desired destination.
* **Enhancing Functionality and User Experience:** Upon reaching the login page, the script employs intelligent rules to elevate its functionality and user experience. This includes handling login credentials with utmost care, ensuring that sensitive information remains secure and confidential.
* **Contingency Plans for Unexpected Encounters:** The script recognizes the unpredictable nature of the online environment and incorporates contingency plans to handle unexpected situations. This proactive approach ensures that the automation process remains uninterrupted, even in the face of unforeseen challenges.
* **Strategic Pauses for a Smooth Ride:** The script exhibits patience by incorporating deliberate waiting periods for pages to load completely. These strategic pauses mimic real-world interactions, allowing the script to adjust to the pace of the online world. This ensures that all elements on the page are ready before the script proceeds, minimizing the risk of errors.
* **Adaptability to Evolving Landscapes:** The script's adaptability is one of its key strengths. In the ever-changing realm of web technologies, platforms like LinkedIn frequently introduce new features. The script's flexible design allows it to adapt to these changes seamlessly, ensuring long-term compatibility.

The LinkedIn Automation Script is a testament to the power of automation, enabling you to accomplish more on LinkedIn with less effort. Its focus on efficiency, security, patience, and adaptability makes it an invaluable tool for anyone seeking to maximize their LinkedIn presence.

In summary, our script goes beyond simple automation; it's a thoughtfully crafted tool that transforms mundane tasks into an engaging and efficient experience, showcasing the brilliance of your computer with every guided step.

**PROPOSED WORK:**  The integration of Selenium within the provided Python script serves as a prime example of leveraging automation technology within LinkedIn's multifaceted ecosystem. LinkedIn's growth trajectory has been propelled not just by its job search and networking functionalities, but also through strategic expansions like LinkedIn Learning. This platform, featuring an extensive array of courses for continuous professional development, has significantly broadened the platform's appeal. By encompassing users focused on skill enhancement, LinkedIn has attracted a diverse user base, expanding its reach beyond job seekers and networkers. The script's efficacy in interacting with LinkedIn's intricate web elements underscores the robustness of the Python-Selenium integration in automating complex tasks. Moreover, the adaptability of Selenium across various web environments, demonstrated through its cross-browser compatibility, enables users to extend the script's functionality to diverse browsers, providing a versatile solution for users with varying preferences. Beyond its role in automating login procedures, this script showcases its utility in scenarios requiring repetitive actions, such as testing or data extraction tasks. This automation not only reduces the likelihood of human error but also liberates users to focus on strategic aspects of their work. The incorporation of ChromeWebDriver path and ChromeOptions in the script exemplifies a commitment to user customization and adaptability. This flexibility ensures that the script can easily accommodate changes in browser versions or user preferences, ensuring its sustainability as a reliable tool over time. Furthermore, the script's user-friendly design, simplifying the login process by automating the input of username and password, aligns with LinkedIn's commitment to inclusivity and ease of use. By enhancing the overall user experience and accessibility, the script extends its reach to a broader audience, including individuals less familiar with programming intricacies. In essence, this Python-Selenium script not only demonstrates the power of automation within LinkedIn's dynamic environment but also embodies user-centric design principles, contributing to LinkedIn's position as a comprehensive professional platform catering to diverse needs.

In the realm of automated LinkedIn access, the use of Python and Selenium has become a popular choice due to its versatility and ease of implementation. Selenium is a powerful tool for web automation, and when coupled with Python, it provides a robust framework for interacting with web elements. The code execution involves the integration of specific libraries, such as Selenium WebDriver, to control a web browser programmatically. The security of user data and the LinkedIn page is of paramount importance in the design of the script. Through the utilization of secure coding practices and the input of user login credentials directly within the code, the script ensures a user-friendly and secure login process. This not only streamlines the authentication process for the user but also minimizes the risk associated with external access. Key components of the code include WebDriver, which acts as a mediator between the Python script and the web browser. WebDriver facilitates the automation of browser actions, ensuring that the login process and subsequent navigation to the SRM AP Page are seamlessly executed. This level of automation is particularly advantageous for users who regularly access specific pages, saving time and effort in the login process. Furthermore, the script's ability to keep the browser open post-execution can be beneficial for users who may want to perform additional actions or manually verify certain details after the automated login. This feature adds a layer of flexibility to the script's functionality. It's worth noting that while automation can enhance efficiency, it's essential to adhere to ethical considerations and the terms of service of the platforms being automated. LinkedIn, like many other websites, has policies governing the use of automation tools, and users should be mindful of these guidelines to maintain a positive and compliant interaction with the platform. In conclusion, this Python script utilizing Selenium serves as a foundation for automating the LinkedIn login process and navigating to specific pages. Through careful implementation, security measures, and user-friendly practices, it provides a streamlined and efficient approach to accessing LinkedIn, showcasing the capabilities of Python and Selenium in the realm of web automation.

**SOURCE CODE:**

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

# Specify the path to the ChromeDriver executable

chromedriver\_path = r"C:\Users\vishn\AppData\Local\Programs\Python\Python312\Lib\site-packages\selenium\webdriver\chromedriver.exe"

# Create a ChromeOptions object to set options

chrome\_options = webdriver.ChromeOptions()

# Specify the Chrome binary location (if needed)

# chrome\_options.binary\_location = "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"

# Disable automatic closing of the browser on script exit

chrome\_options.add\_experimental\_option("detach", True)

# Create the Chrome WebDriver instance with options

driver = webdriver.Chrome(options=chrome\_options)

# Opening LinkedIn's login page

driver.get("https://www.linkedin.com/login")

# Waiting for the page to load

time.sleep(5)

# Entering username

username = driver.find\_element(By.ID, "username")

username.send\_keys("vishnuvamsi147@gmail.com")

# Entering password

password = driver.find\_element(By.ID, "password")

password.send\_keys("V@msi2004")

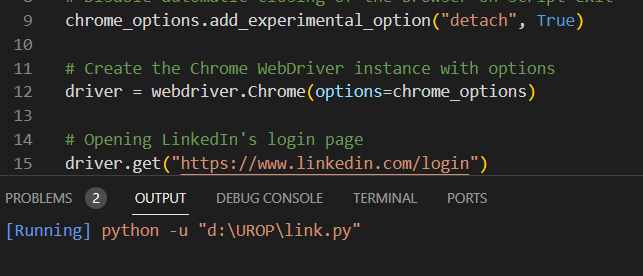
# Clicking the login button

driver.find\_element(By.XPATH, "//button[text()='Sign in']").click()

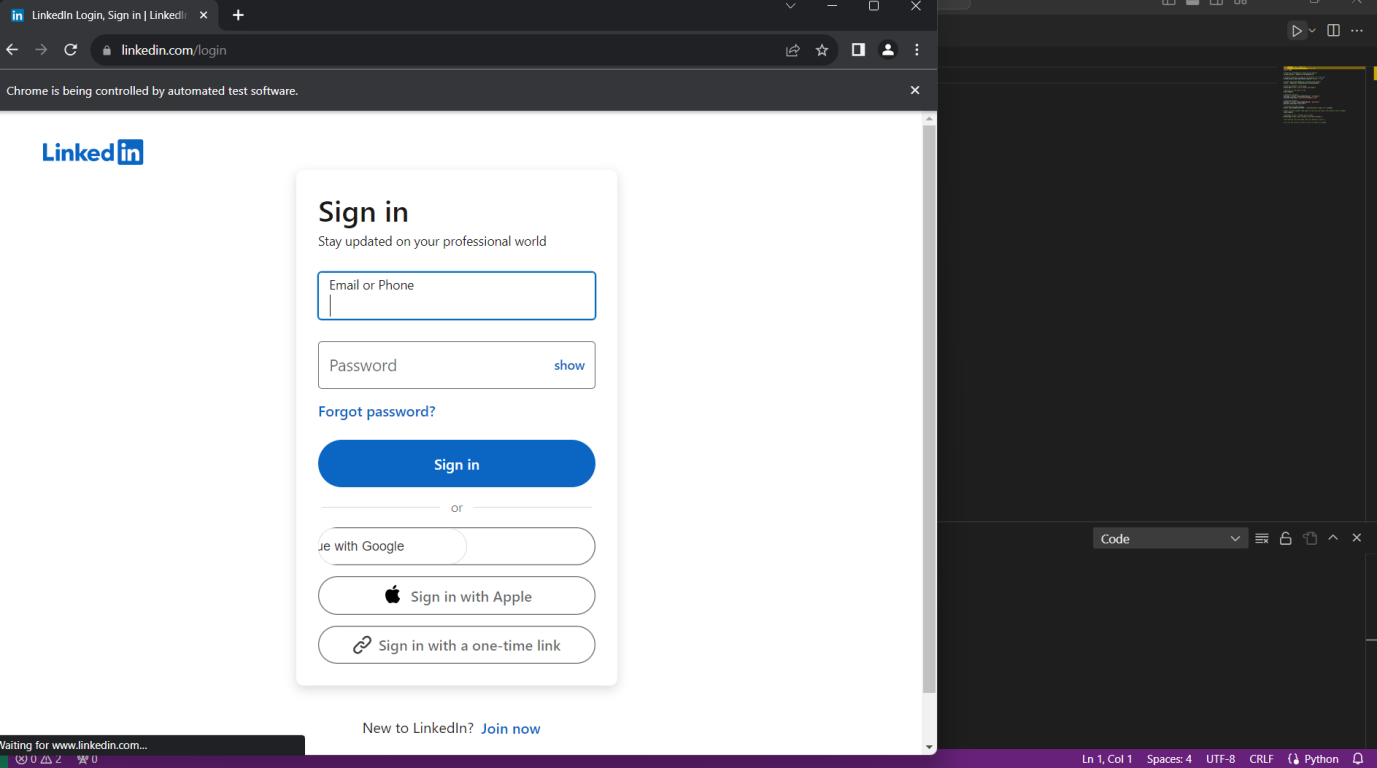
# The browser will stay open until you manually close it.

# You can add further actions or wait for events as needed.

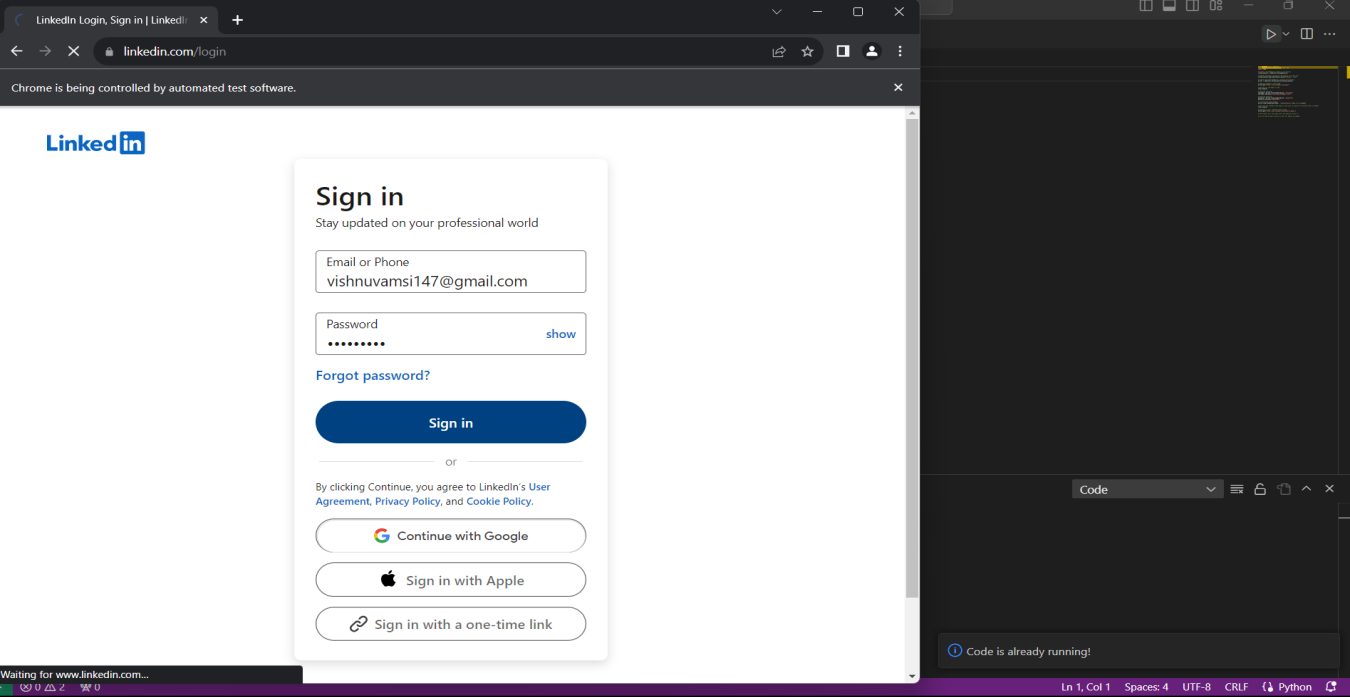
**RESULT AND ANALYSIS:**



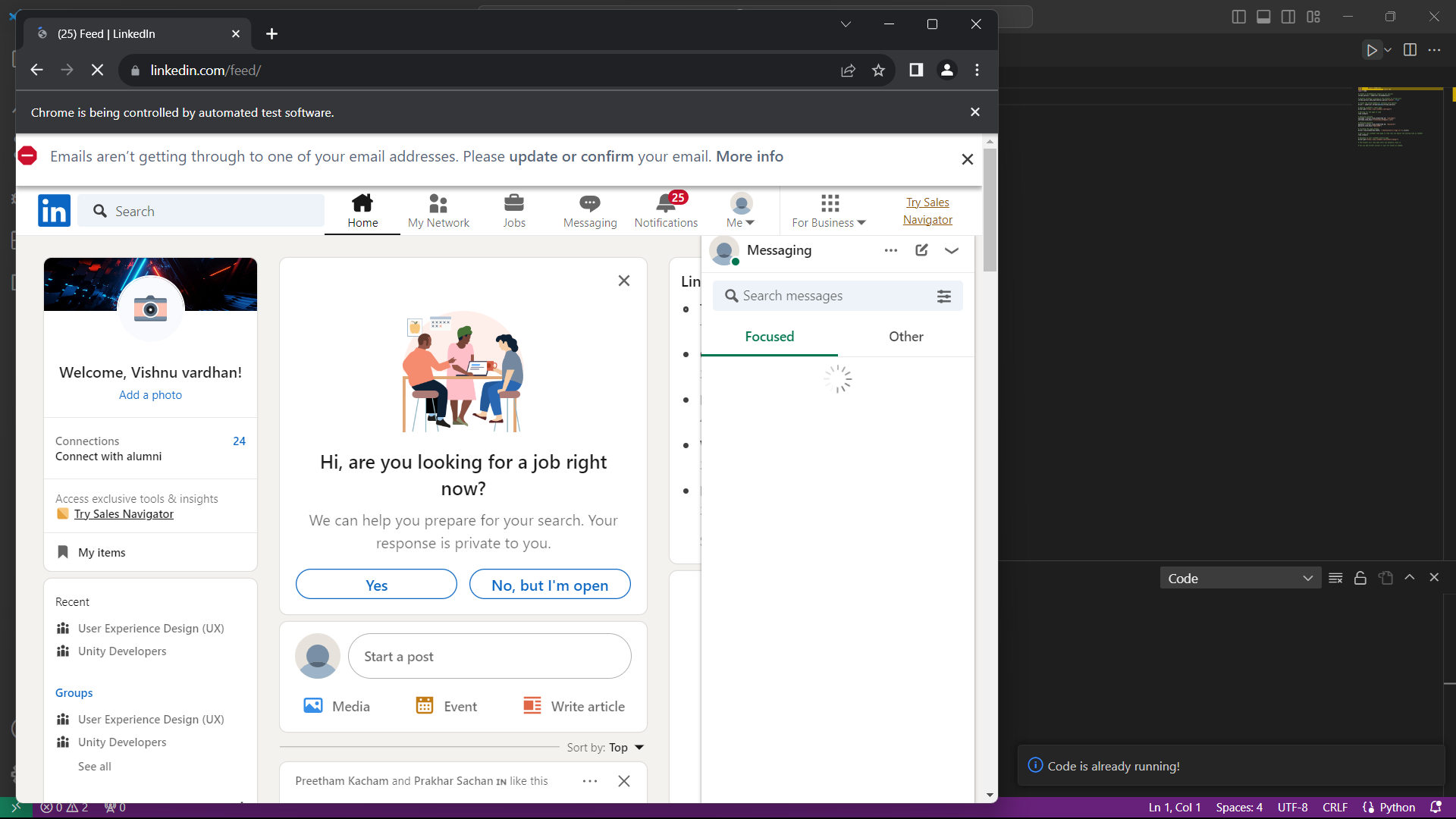
**Running of our code has done successfully through visual studio code and by importing all libraries of selenium and gave a path for web-driver.**



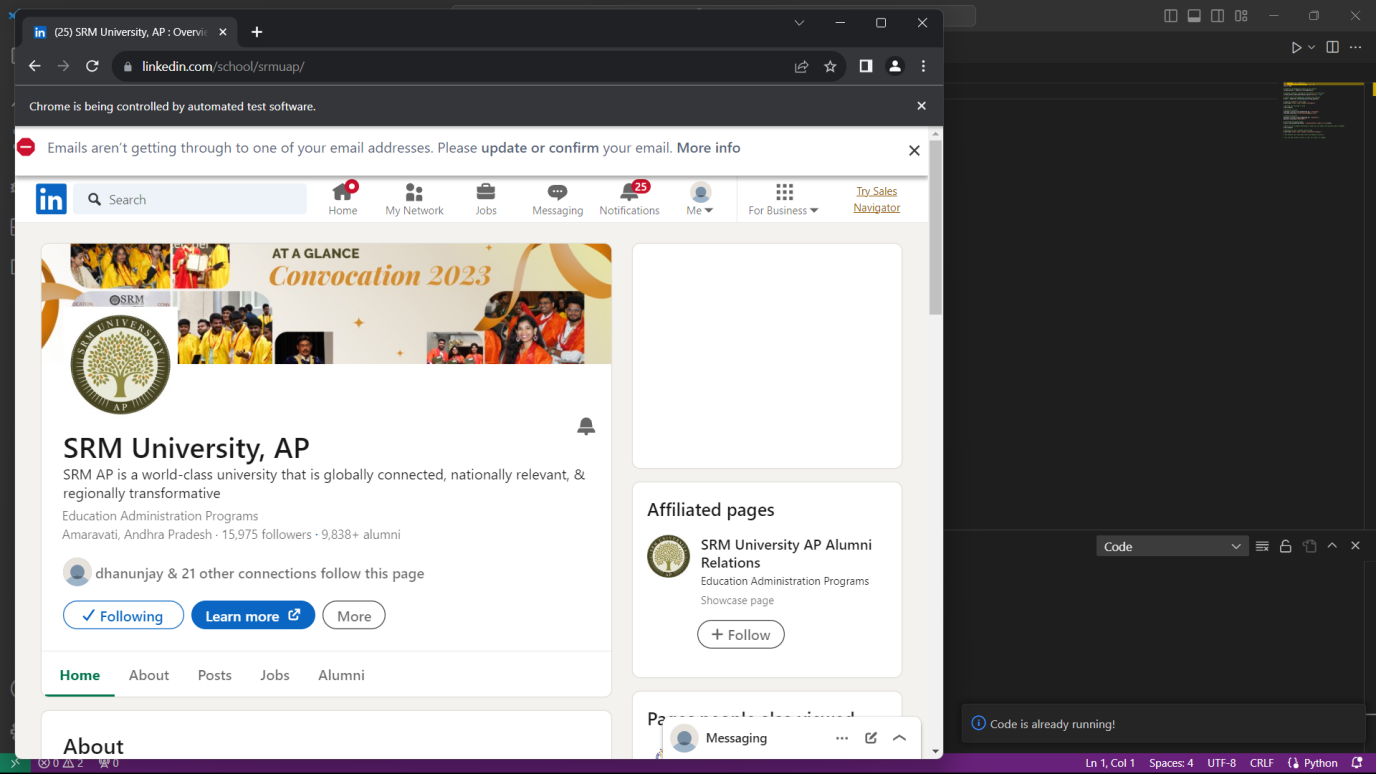
**Chrome-driver gives a proper path for our script and opens the LinkedIn page in the chrome directly our page will not be closed it is prevented by Detach.**



**HTML ID’S will type the given user login credentials in the code as input and perfoms all the necessary actions required.**



**Signin/Login will be done by Xpath and we can see that user account has been displayed.**



**As we can see SRM-AP page is also opened we can also customize as we want so here we as of now we mentioned the SRM-AP page.**

**CONCLUSION:** In our script adventure, the utilization of specialized tools known as Selenium libraries serves as the backbone, ensuring seamless communication between our computer and the browser in an organized and efficient manner. These libraries act as a set of instructions, enabling us to navigate the web landscape with precision. The script employs a driver to effortlessly transport us to the LinkedIn login page, setting the stage for a series of intelligent actions. Upon reaching the login page, the script employs a set of smart rules to enhance its functionality and user experience. Handling login details with care, akin to secret codes comprehensible only to the computer, adds an extra layer of security and confidentiality. This careful consideration underscores our commitment to safeguarding user information in the digital realm. To tackle the unpredictable nature of the online environment, the script incorporates contingency plans, akin to a backup strategy. This ensures that if anything unexpected occurs during the automation process, the script can adapt and proceed without disrupting the user experience. This resilience is crucial in providing users with a smooth and uninterrupted interaction with LinkedIn. Moreover, the script exhibits patience by incorporating a deliberate waiting period for the page to load completely. This strategic pause ensures that all elements on the page are ready, creating an optimal environment for the script to execute subsequent actions accurately. This waiting period can be likened to patiently waiting for a friend to finish getting ready before embarking on an outing. One of the script's standout features is its adaptability to changes on the LinkedIn platform. In the dynamic landscape of web technologies, platforms like LinkedIn frequently introduce new features. Our script is designed with flexibility in mind, ready to adapt and accommodate these changes seamlessly. This adaptability is akin to giving our script a makeover, ensuring it stays current, cool, and reliable in the long run. In simple terms, our script assumes the role of a helpful buddy, employing Selenium to communicate with LinkedIn on your behalf. Its design principles prioritize patience, careful handling of sensitive information, resilience in the face of surprises, and readiness for new adventures on LinkedIn. It's not just a set of instructions; it's a reliable companion, ensuring that your interactions on LinkedIn are not only efficient but also secure and adaptable to the ever-evolving digital landscape.

**REFERENCES:**

1. Mitra, Arnab, Anirban Kundu, Matangini Chattopadhyay, and Avishek Banerjee. "An Approach to Detect Fake Profiles in Social Networks Using Cellular Automata-Based PageRank Validation Model Involving Energy Transfer." *SN Computer Science* 3, no. 6 (2022): 423.
2. Raghavendra, Sujay, and Sujay Raghavendra. "Introduction to selenium." *Python Testing with Selenium: Learn to Implement Different Testing Techniques Using the Selenium WebDriver* (2021): 1-14.
3. Gojare, Satish, Rahul Joshi, and Dhanashree Gaigaware. "Analysis and design of selenium webdriver automation testing framework." *Procedia Computer Science* 50 (2015): 341-346.
4. <https://www.codeproject.com/Questions/693440/Accessing-the-linkedin-page-using-java-code>
5. <https://shrikar-datakund.medium.com/login-to-linkedin-automatically-with-this-python-library-a75f389fc3a8>
6. <https://stackoverflow.com/questions/18907503/logging-in-to-linkedin-with-python-requests-sessions>
7. García, Boni, et al. "Automated driver management for selenium WebDriver." *Empirical Software Engineering* 26 (2021): 1-51.
8. Presler-Marshall, Kai, et al. "Wait, wait. no, tell me. analyzing selenium configuration effects on test flakiness." *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*. IEEE, 2019.
9. Heydt, Michael. *Python Web Scraping Cookbook: Over 90 proven recipes to get you scraping with Python, microservices, Docker, and AWS*. Packt Publishing Ltd, 2018.
10. Grasso, Giovanni, Tim Furche, and Christian Schallhart. "Effective web scraping with oxpath." *Proceedings of the 22nd international conference on world wide web*. 2013.

**ACKNOWLEDGEMENT:** We truly thank Dr. Arnab Mitra, our supervisor, for his assistance and direction in doing the study and writing the report that is provided.