KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visveswaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

Course Activity Report on

"RPA FOR EASY APPLY IN LINKEDIN"

Submitted in the partial fulfilment for the academic requirement of

6TH *Semester B.E*

In

Information Science Engineering

Submitted by

SL NO.	Batch member Names	USN
1	Adarsh Kumar	2GI20IS002
2	Danesh Naik	2GI20IS011
3	Sahil Faniband	2GI20IS032

Under the Guidance Of
Dr. Sudhindra B Deshpande
ASSISTANT PROFESSOR
Academic Year 2022-2023

KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visveswaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the course project entitled "RPA FOR EASY APPLY IN LINKED IN" is a Bonafede record of the Seminar work done by **Adarsh Kumar, Danesh Naik, Sahil Faniband, Vinayak Nikam** having **USN 2GI20IS002,2GI20IS011, 2GI20IS032, 2GI20IS050** under my supervision and guidance, in partial fulfilment of the requirements for the Outcome Based Education Paradigm in ISE from Gogte Institute of Technology for the academic year 2022-2023.

Faculty In charge

Head of the Department

Rubrics for evaluation of Course Project

Sl. No	Batch No. :					
1.	Project Title: REAL-TIME HUMAN DETECTION AND COUNTING	MARKS RANGE	USN			
			2GI29IS002	2GI20IS011	2GI20IS032	2GI20IS050
2.	Problem statement(PO2)	0-1				
3.	Objectives of Defined Problem(PO1,PO2)	0-2				
4.	Design/Algorithm/ Flowchart/Methodology (PO3)	0-3				
5.	Implementation details/Function/ Procedure/Classes and objects(Language/Tools)	0-4				
6.	Working model of the final solution (PO3,PO12)	0-5				
7.	Report and Oral presentation skill (PO9,PO10)	0-5				
	Total	20				

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.

3. **Design/Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental consideration. **Conduct investigation of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusion.

4. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

5. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

6. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

7. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

8. **Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

9. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as,being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

10. **Project management and finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments

11. **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

RPA FOR EASY APPLY IN LINKED IN

Robotic Process Automation project for Easy apply in Linked In

ABSTRACT

This project investigates and reports benchmarks for users to apply the job application in one go through apply easy button in linked in

The project focuses on implementing the Easy Apply button automation in LinkedIn using the Selenium framework in Python. The Easy Apply button automation aims to simplify and streamline the job application process for LinkedIn users, allowing them to apply to multiple job postings with just a single click.

The project demonstrates the advantages of leveraging the Selenium framework in Python to automate the Easy Apply button in LinkedIn. By automating the repetitive and time-consuming task of job applications, users can save time, increase productivity, and enhance their overall job search experience. The project report provides insights into the implementation process, limitations, and considerations, enabling users to successfully implement the Easy Apply button automation in LinkedIn using Selenium in Python

Table of Contents:

Table of Contents

1. INTRODUCTION

1.1. Problem Statement:

RPA for Easy Apply in Linked In– The code that help LinkedIn users to easily apply for jobs and by clicking on the easy apply button the application is submitted to the recruiters .

1.2. Objectives:

The main objectives of implementing the Easy Apply button automation in LinkedIn using Robotic Process Automation are as follows:

(1) Enhance User Experience: The primary objective is to improve the overall user experience on LinkedIn's job platform. By implementing the Easy Apply button automation, users can save time and effort when applying to job postings, leading to a more streamlined and user-friendly application process.

(2) Simplify the Application Process: The automation aims to simplify the job application process by enabling users to apply to multiple job postings with just a single click. This eliminates the need to navigate through individual job application pages, resulting in a more efficient and convenient experience for users.

(3) Increase Productivity: By automating the repetitive task of applying to job postings, users can focus their time and energy on other crucial aspects of their job search, such as networking, preparing for interviews, and refining their resumes. The automation helps users to be more productive and efficient in their job search efforts.

(4) Error Reduction: Manual data entry during the application process can lead to errors or omissions. The Easy Apply button automation minimizes the chances of such errors by automating the process of filling out application fields accurately and consistently. This ensures that applications are complete and error-free.

(5) Accelerate Application Submission: With the Easy Apply button automation, users can submit job applications more quickly and seamlessly. The automation handles the repetitive tasks of entering personal information, uploading resumes, and, if applicable, submitting cover letters. This accelerates the application submission process, giving users a competitive advantage in their job search.

(6) Improve Efficiency: By leveraging RPA technology, the automation process can efficiently handle a large volume of job applications. It eliminates the need for manual intervention, allowing users to apply to multiple job postings in a shorter amount of time. This increased efficiency translates into higher productivity and a better job search experience.

1.3 Methodology:

The methodology outlined below provides a high-level overview of the steps involved in implementing the Easy Apply button automation in LinkedIn using RPA. However, it's essential to adapt and tailor the methodology based on specific project requirements, the chosen RPA tool, and the nuances of LinkedIn's user interface and application process

1. Define the Scope and Objectives: Clearly define the scope of the automation project, specifying the target processes and functionalities to be automated. Identify the specific objectives and goals of implementing the Easy Apply button automation, such as enhancing user experience, reducing manual effort, and improving application submission efficiency.
2. Select an RPA Tool: Research and select a suitable RPA tool or framework that is compatible with automating web-based applications like LinkedIn. Consider factors such as ease of use, scalability, integration capabilities, and community support while choosing the RPA tool.
3. Set up the RPA Environment: Install and configure the chosen RPA tool on the designated system where the automation will be executed. Ensure that the necessary hardware and software requirements are met.
4. Analyze LinkedIn User Interface: Study the LinkedIn user interface and understand the elements, actions, and workflows involved in the Easy Apply button feature. Identify the web elements, such as search fields, job listings, and the Easy Apply button, that need to be interacted with during the automation process.
5. Develop the Automation Workflow: Using the selected RPA tool, create an automation workflow that replicates the actions required to search for job postings and click the Easy Apply button. Define the sequence of steps, interactions with web elements, and any necessary conditional logic for efficient and accurate automation.
6. User Interaction: Determine the input parameters required for the automation process, such as job title, location, and other filters. Develop a user interface or input mechanism, such as a form or dialog box, to allow users to enter their job preferences and interact with the automation.
7. Implement Automation Logic: Develop the logic to search for job postings based on the user's input parameters. Identify the Easy Apply button element within the search results and create an automation sequence to click on it.
8. Application Submission: Automate the process of filling out the required application fields, such as personal information, resume upload, and cover letter (if applicable). Verify the completion of the application submission and handle any potential errors or exceptions that may occur during the process.
9. Test and Validate: Conduct thorough testing of the automation workflow to ensure its accuracy, robustness, and compatibility with various scenarios and edge cases. Validate the results of the automation by comparing them with manual application submissions to ensure consistency and correctness.
10. Deploy and Monitor: Deploy the Easy Apply button automation in a production environment. Monitor the automation process for performance, efficiency, and

any potential issues or errors. Regularly and maintain the automation workflow to accommodate changes in the LinkedIn user interface or application process.

1.3.1 Project Details and Technology:

Project Name:	Robotic Process Automation for Easy apply in LinkedIn
Language/S Used:	Python
Python Version (Recommended):	3.8/3.9
Database :	None
Type:	Desktop Application
Updates:	0

Table 1.0.2 Project Details

1.3.2 Framework Used:

Selenium:

Selenium is a popular open-source framework used for automating web browsers. It provides a set of tools and libraries that enable developers to interact with web elements, simulate user actions, and automate web-based tasks. Selenium supports multiple programming languages, including Python.

1. Installation: To use Selenium in Python, you need to install the Selenium package. You can install it using pip, the package installer for Python. The command to install Selenium is: pip install selenium.
2. WebDriver: The WebDriver is a key component of Selenium that allows you to control the web browser programmatically. Selenium WebDriver provides APIs to interact with web elements, navigate through web pages, handle forms, and perform various actions like clicking buttons, entering text, and submitting forms.
3. Web Element Interactions: Selenium provides methods to locate and interact with web elements on a web page. You can find elements using different strategies such as ID, class name, CSS selector, XPath, and more. Once you have located an element, you can perform actions like clicking, sending text, retrieving attribute values, or executing JavaScript on the element.
4. Synchronization: Web applications often have dynamic content and asynchronous behavior. Selenium provides mechanisms for handling synchronization and waits. You can wait for specific conditions to be met, such as the presence or visibility of an element, before proceeding with the automation script execution.
5. Cross-Browser Testing: Selenium supports multiple web browsers, including Chrome, Firefox, Safari, and Edge. You can write your automation scripts in Python using Selenium and execute them across different browsers, allowing you to perform cross-browser testing and ensure consistent behavior across various browser environments.
6. Test Framework Integration: Selenium can be integrated with popular test frameworks in Python, such as py test and unit test. This allows you to structure your test cases, perform assertions, and generate test reports using the capabilities provided by these frameworks.
7. Extensibility: Selenium is a highly extensible framework. It provides options to handle complex scenarios through customizations and extensions. You can create custom WebDriver implementations, implement event listeners, and leverage browser-specific extensions or plugins to enhance the capabilities of Selenium.

Selenium with Python is widely used for web automation, testing, and web scraping tasks. It provides a flexible and powerful framework for interacting with web browsers programmatically and automating repetitive web-based tasks.

2. BENIFITS

By leveraging the Easy Apply button automation in LinkedIn using RPA, users can save time, improve their job search efficiency, and enhance their overall experience on the platform. The benefits include increased productivity, accuracy, scalability, and cost savings, ultimately increasing the chances of finding suitable job opportunities.

1. Time Savings: Automation eliminates the need for manual navigation and data entry, allowing users to apply to multiple job postings with just a single click. This saves significant time and effort compared to the traditional manual application process.

2. Enhanced User Experience: By simplifying and streamlining the job application process, the Easy Apply button automation improves the overall user experience on LinkedIn. Users can quickly and easily apply to relevant job postings, reducing frustration and enhancing their engagement with the platform.

3. Increased Productivity: With automation handling repetitive tasks, users can focus their time and energy on other critical aspects of their job search, such as networking, researching companies, and preparing for interviews. This increased productivity allows job seekers to make better use of their time and be more efficient in their job search efforts.

4. Accuracy and Consistency: Automation reduces the risk of errors and ensures consistent data entry. By automating the application form filling process, the Easy Apply button automation minimizes mistakes and maintains consistency across applications, resulting in higher-quality submissions.

5. Higher Application Submission Rate: The Easy Apply button automation enables users to apply to a larger number of job postings in a shorter amount of time. This increased efficiency leads to a higher application submission rate, giving users a competitive advantage in their job search and increasing their chances of being noticed by potential employers.

6. Scalability: RPA tools used for automation can handle a high volume of job applications efficiently. Whether users need to apply to a few job postings or a large number, the automation process can scale to accommodate the demand, ensuring consistent and reliable performance.

7. Cost Savings: Automating the application process reduces the need for manual labor, potentially resulting in cost savings for both job seekers and employers. Companies can also benefit from reduced administrative overhead and increased operational efficiency.

8. Flexibility and Customization: The automation workflow can be customized to cater to specific user preferences and requirements. Users can define their job preferences, filters, and other parameters, allowing the automation to tailor the job search and application process accordingly.

9. Continuous Monitoring and Improvement: RPA tools often provide monitoring capabilities, allowing users to track the progress of their automated job applications. This monitoring data can provide valuable insights for process optimization and continuous improvement of the automation workflow.

3. IMPLEMENTATION

3.1. Source Code:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.common.exceptions import NoSuchElementException, ElementClickInterceptedException,
NoSuchElementException
from selenium.webdriver.common.action_chains import ActionChains
import time
import re
import json

class EasyApplyLinkedin:

  def __init__(self, data):
        """Parameter initialization"""

    self.email = data['email']
    self.password = data['password']
    self.keywords = data['keywords']
    self.location = data['location']
    self.driver = webdriver.Chrome(data['driver_path'])

    def login_linkedin(self):
    """This function logs into your personal LinkedIn profile"""

    # go to the LinkedIn login url
    self.driver.get("https://www.linkedin.com/login")

    # introduce email and password and hit enter
    login_email = self.driver.find_element_by_name('session_key')
    login_email.clear()
    login_email.send_keys(self.email)
    login_pass = self.driver.find_element_by_name('session_password')
    login_pass.clear()
    login_pass.send_keys(self.password)
    login_pass.send_keys(Keys.RETURN)

  def job_search(self):
    """This function goes to the 'Jobs' section a looks for all the jobs that matches the keywords and location"""

    # go to Jobs
    jobs_link = self.driver.find_element_by_link_text('Jobs')
    jobs_link.click()

    # search based on keywords and location and hit enter
    search_keywords = self.driver.find_element_by_css_selector(".jobs-search-box__text-input[aria-
label='Search jobs']")
    search_keywords.clear()
```

```
        search_keywords.send_keys(self.keywords)
        search_location = self.driver.find_element_by_css_selector(".jobs-search-box__text-input[aria-
label='Search location']")
        search_location.clear()
        search_location.send_keys(self.location)
        search_location.send_keys(Keys.RETURN)

    def filter(self):
        """This function filters all the job results by 'Easy Apply'"""

        # select all filters, click on Easy Apply and apply the filter
        all_filters_button = self.driver.find_element_by_xpath("//button[@data-control-name='all_filters']")
        all_filters_button.click()
        time.sleep(1)
        easy_apply_button = self.driver.find_element_by_xpath("//label[@for='f_LF-f_AL']")
        easy_apply_button.click()
        time.sleep(1)
        apply_filter_button = self.driver.find_element_by_xpath("//button[@data-control-
name='all_filters_apply']")
        apply_filter_button.click()

    def find_offers(self):
        """This function finds all the offers through all the pages result of the search and filter"""

        # find the total amount of results (if the results are above 24-more than one page-, we will scroll trhough all available pages)
        total_results = self.driver.find_element_by_class_name("display-flex.t-12.t-black--light.t-normal")
        total_results_int = int(total_results.text.split(' ',1)[0].replace(",",""))
        print(total_results_int)

        time.sleep(2)
        # get results for the first page
        current_page = self.driver.current_url
        results = self.driver.find_elements_by_class_name("occludable-update.artdeco-list__item--offset-4.artdeco-
list__item.p0.ember-view")

        # for each job add, submits application if no questions asked
        for result in results:
            hover = ActionChains(self.driver).move_to_element(result)
            hover.perform()
            titles = result.find_elements_by_class_name('job-card-search__title.artdeco-entity-lockup__title.ember-
view')
            for title in titles:
                self.submit_apply(title)

        # if there is more than one page, find the pages and apply to the results of each page
        if total_results_int > 24:
            time.sleep(2)

            # find the last page and construct url of each page based on the total amount of pages
            find_pages = self.driver.find_elements_by_class_name("artdeco-pagination__indicator.artdeco-
pagination__indicator--number")
            total_pages = find_pages[len(find_pages)-1].text
            total_pages_int = int(re.sub(r"[^\d.]", "", total_pages))
            get_last_page = self.driver.find_element_by_xpath("//button[@aria-label='Page
```

```
"+str(total_pages_int)+"']")
            get_last_page.send_keys(Keys.RETURN)
            time.sleep(2)
            last_page = self.driver.current_url
            total_jobs = int(last_page.split('start=',1)[1])

            # go through all available pages and job offers and apply
            for page_number in range(25,total_jobs+25,25):
                self.driver.get(current_page+'&start='+str(page_number))
                time.sleep(2)
                results_ext = self.driver.find_elements_by_class_name("occludable-update.artdeco-list__item--offset-
4.artdeco-list__item.p0.ember-view")
                for result_ext in results_ext:
                    hover_ext = ActionChains(self.driver).move_to_element(result_ext)
                    hover_ext.perform()
                    titles_ext = result_ext.find_elements_by_class_name('job-card-search__title.artdeco-entity-
lockup__title.ember-view')
                    for title_ext in titles_ext:
                        self.submit_apply(title_ext)
        else:
            self.close_session()

    def submit_apply(self,job_add):
        """This function submits the application for the job add found"""

        print('You are applying to the position of: ', job_add.text)
        job_add.click()
        time.sleep(2)

        # click on the easy apply button, skip if already applied to the position
        try:
            in_apply = self.driver.find_element_by_xpath("//button[@data-control-
name='jobdetails_topcard_inapply']")
            in_apply.click()
        except NoSuchElementException:
            print('You already applied to this job, go to next...')
            pass
        time.sleep(1)

        # try to submit if submit application is available...
        try:
            submit = self.driver.find_element_by_xpath("//button[@data-control-name='submit_unify']")
            submit.send_keys(Keys.RETURN)

        # ... if not available, discard application and go to next
        except NoSuchElementException:
            print('Not direct application, going to next...')
            try:
                discard = self.driver.find_element_by_xpath("//button[@data-test-modal-close-btn]")
                discard.send_keys(Keys.RETURN)
                time.sleep(1)
                discard_confirm = self.driver.find_element_by_xpath("//button[@data-test-dialog-primary-btn]")
                discard_confirm.send_keys(Keys.RETURN)
```

```
                time.sleep(1)
            except NoSuchElementException:
                pass

        time.sleep(1)

    def close_session(self):
        """This function closes the actual session"""

        print('End of the session, see you later!')
        self.driver.close()

    def apply(self):
        """Apply to job offers"""

        self.driver.maximize_window()
        self.login_linkedin()
        time.sleep(5)
        self.job_search()
        time.sleep(5)
        self.filter()
        time.sleep(2)
        self.find_offers()
        time.sleep(2)
        self.close_session()


if __name__ == '__main__':

    with open('config.json') as config_file:
        data = json.load(config_file)

    bot = EasyApplyLinkedin(data)
    bot.apply()
```

4. LIMITATIONS AND CONSIDERATIONS

While implementing the Easy Apply button automation in LinkedIn using RPA offers several benefits, it's important to consider the limitations and take certain considerations into account. These include:

1. Dynamic User Interface: LinkedIn's user interface is subject to frequent updates and changes. These changes can impact the automation workflow as the web elements' locators and structure may be modified. Regular monitoring and maintenance of the automation workflow are required to ensure its compatibility with any user interface changes.

2. Security Measures: LinkedIn employs security measures, such as Captcha, to prevent automated actions. Handling these security measures within the automation workflow can be challenging and may require additional steps or human intervention to handle the Captcha challenge and ensure compliance with security protocols.

3. Data Privacy: Automating the application process involves handling personal and sensitive information of users. It's crucial to ensure compliance with privacy regulations and implement appropriate data protection measures to safeguard user data and maintain the confidentiality of personal information.

4. Application Form Variations: Different job postings on LinkedIn may have variations in their application forms, including different fields, layouts, or requirements. Adapting the automation workflow to handle these variations and dynamically capture the necessary information from the application form can be complex.

5. Limitations of RPA Tools: The chosen RPA tool may have certain limitations in terms of its capabilities or compatibility with specific technologies. It's important to thoroughly understand the capabilities and limitations of the chosen RPA tool to ensure it meets the requirements of the Easy Apply button automation in LinkedIn.

6. Reliability of Automation: While RPA can automate a significant portion of the application process, there may still be instances where human intervention is required. For example, some job postings may require answering additional questions or submitting specific documents. Incorporating logic and error handling mechanisms to handle such scenarios is crucial for maintaining the reliability of the automation.

7. Monitoring and Maintenance: Continuous monitoring and maintenance of the automation workflow are essential to ensure its smooth functioning and adaptability to any changes in the LinkedIn platform. Regular updates and adjustments may be necessary to keep up with the evolving user interface and application process.

8. Ethical Considerations: It's important to use automation responsibly and ethically, adhering to LinkedIn's terms of service and guidelines. Automation should not be used for malicious purposes or in violation of any legal or ethical boundaries

Considering these limitations and considerations helps ensure a successful implementation of the Easy Apply button automation in LinkedIn using RPA. Regular monitoring, maintenance, adherence to security and privacy protocols, and ethical usage are crucial for maintaining the integrity and effectiveness of the automation process.

5. CONCLUSION

In conclusion, implementing the Easy Apply button automation in LinkedIn using RPA offers numerous benefits and opportunities for job seekers. By automating the repetitive and time-consuming task of applying to job postings, users can save valuable time, enhance their productivity, and improve their overall job search experience. The automation streamlines the application process, simplifies data entry, and increases the efficiency of application submission.

However, it is important to consider the limitations and challenges associated with implementing this automation. Dynamic user interfaces, security measures like Captcha, variations in application forms, and data privacy concerns require careful attention and adaptation in the automation workflow. The reliability of automation, regular monitoring, and maintenance are essential to ensure its effectiveness and adaptability to changes.

Considering these limitations and taking appropriate considerations into account, job seekers can leverage the Easy Apply button automation to their advantage, increasing their application submission rate, improving accuracy and consistency, and gaining a competitive edge in their job search efforts.

adopting RPA technology and implementing the Easy Apply button automation in LinkedIn, users can streamline their job application process, focus on more strategic job search activities, and potentially increase their chances of finding suitable employment opportunities.

.

6. REFERENCE

[1] IEEE Xplore (https://ieeexplore.ieee.org/)

[2] ACM Digital Library (https://dl.acm.org/)

[3] ScienceDirect (https://www.sciencedirect.com/)

[4] UiPath Resources (https://www.uipath.com/resources)

[5] Automation Anywhere Resource Library (https://www.automationanywhere.com/resources)