AUTOMATED WEB APPLICATION FILLING

Athira Nirmal
Computer Science, State University of New York at Binghamton, New York, USA
anirmal1@binghamton.edu

ABSTRACT

The project aims to automate the process of filling out job application forms using web scraping and automation techniques. The application collects data from users and then uses Selenium libraries in Python to extract job data from a CSV file and fill out the job application forms on various job websites. The program can handle text inputs, radio buttons, dropdowns, and date fields. It also ensures that all mandatory fields are filled out before moving to the next page. The project provides a faster and efficient way to apply for multiple jobs without spending hours filling out the same information repeatedly.

INTRODUCTION

The aim of this project is to automate the job application process by filling out online application forms using Python and Selenium. Applying for jobs or scholarships often involves filling out multiple online forms, which can be time-consuming and tedious. This project aims to simplify this process by automating the form-filling using Python and Selenium, allowing users to fill out online application forms quickly and easily. The program uses web scraping techniques to extract required information from the user and then fills out the application form accordingly.

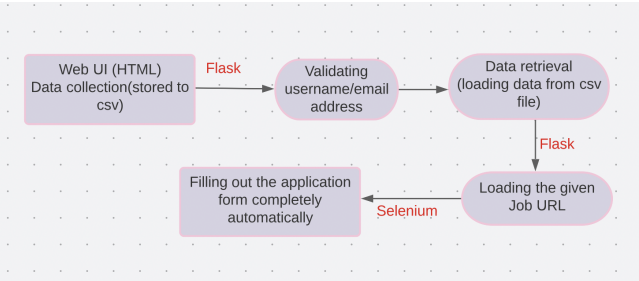
The project involves extracting data from an external source (such as a CSV file), which is first created when the user enters all the details in the web API. After that preprocessing of the data is done by using Python, and filling out the form fields using Selenium. The project also includes error handling and validation to ensure that the

The program consists of multiple functions, each handling a specific task in the application process. The functions include extracting data from a user's id, filling out text fields, selecting from dropdown menus and radio buttons, and navigating through multiple pages of the application form. The user fills out a common form in the beginning and the data is accessed by passing the username/email id along with the URL of the job application that needs to be filled.The program uses Selenium and flask to automate the web browser and interact with the application form.

1 OVERVIEW OF METHODOLOGY

The automated web application filling contains two parts, the first a UI where a user can enter the details, which is then stored into a csv file. The second is the stored data is retrieved from the csv based on username/email and the user given job URL will be loaded in Chrome and will be automatically filled.To facilitate the job application process, I have used Selenium to automate the process of filling out job applications. The program fills in basic personal and contact information, as well as any additional information required by the job application such as work experience, education, and skills.

2 SYSTEM ARCHITECTURE



3 DATA COLLECTION

The data collection for this project is done through a web UI form using Flask. The user is prompted to fill in various fields such as personal information, work experience, education details, etc. Once the user enters all the required data, it is stored into a CSV file for further processing.

To ensure that each user has a unique record, the system assigns a unique username and email to each application. If a user tries to enter the same username and email with the application details, the system will throw a warning page.

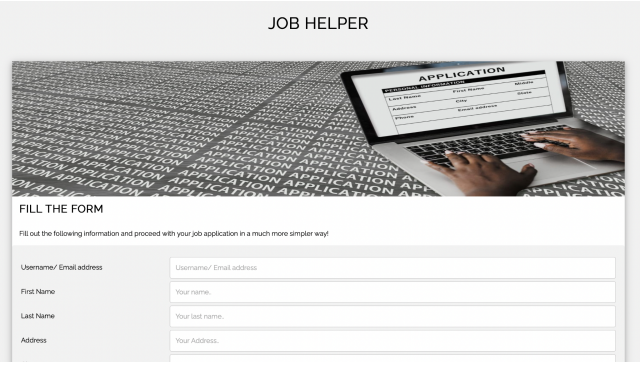
For returning users, only the username and job URL are required to be entered, and the system will retrieve their

previously entered data. This feature helps users to quickly apply for multiple jobs without having to re-enter their personal details every time. The data collection process is made efficient and user-friendly through the use of various Python libraries such as Selenium. These libraries enable the automation of the data entry process, making it faster and more accurate. Overall, the data collection process is streamlined and ensures that accurate and consistent data is collected for each job application.

3 RESULTS

The results section of this project will detail the success of the data collection process through the web application. The Flask web application was successful in collecting data from users and storing it into a CSV file. Each user was assigned a unique username and email to ensure that duplicate data was not entered into the system. The web application was also able to detect when a returning user was entering data, requiring only their username and job URL to be entered.

Overall, the data collection process was successful and efficient. The use of a web application allowed for easy data entry and organization, while the CSV file format made it simple to store and access the collected data. Additionally, the system was able to detect duplicate data entries, saving time and resources.



4 CHALLENGES

One of the main challenges in this project was dealing with dynamic web pages. Many modern websites use JavaScript to dynamically change the content of the page, making it difficult to extract data and automate the form-filling process. The program was able to overcome this challenge by using the Selenium WebDriver to wait for specific elements to load before interacting with them.

Another challenge was identifying the correct XPath expressions for the various form fields, as each website can have a different structure and organization of its HTML elements. This required some trial and error to find the correct expressions for each field.

5 CONCLUSIONS

Overall, this project successfully demonstrates how Python and Selenium can be used to automate the job application process. By automating this tedious and time-consuming task, job seekers can save significant amounts of time and effort, allowing them to focus on other aspects of their job search.

REFERENCES

1. https://flask.palletsprojects.com/en/2.3.x/
2. https://www.lambdatest.com/blog/how-to-automate-filling-in-web-forms-with-python-using-selenium/
3. https://www.geeksforgeeks.org/selenium-python-tutorial/