VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A MINI PROJECT

REPORT ON

"Amazon Automation Testing"

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

Submitted by

AYUSH KUMAR SHARAF

(1CR21IS033)

Under the Guidance of

Ms. Saba Tahaseen

Asst. Professor, Department of ISE, CMRIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

AECS LAYOUT, ITPL PARK ROAD, BENGALURU-560037

2023-2024

CMR Institute of Technology

AECS Layout, Bengaluru-560037

Department of Information Science and Engineering

CERTIFICATE

Certified that the Mini Project work in Software Testing Laboratory entitled "**Amazon Automation Testing"** is a bonafide work carried out by **Ayush Kumar Sharaf (1CR21IS033)** in partial fulfillment for the award of Bachelor of Engineering in Information Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2023-2024. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in there port deposited in the department library. The mini-project report has been approved as it is satisfied the academic requirements in respect of project work prescribed for the said Degree.

Ms. SabaTahaseen	**Dr. Sushelamma**	**Dr. Jagadishwari V**
Project Guide	Project Guide	Head of the Department

NAME: Ayush Kumar Sharaf

USN: 1CR21IS033

Examiners Details:

External Examiner (Name & Signature with date)

Internal Examiner (Name & Signature with date)

ABSTRACT

The present report outlines the development and implementation of an automation testing framework using Selenium, Python Web Driver for the popular application like Amazon . The primary objective of this project was to design and execute a comprehensive test suite that ensures the functional correctness and usability of the website. Selenium, a widely-used open-source tool, was employed to automate the testing process, simulating real-user interactions with the website.

The test suite was designed to cover various aspects of the website, including login functionality, scraping data from the website. The tests were executed across multiple browsers and operating systems to ensure cross-browser compatibility and platform independence.

The results of the automation testing revealed several bugs and issues that were previously undetected by manual testing. These defects were reported and tracked, allowing the development team to prioritize and resolve them in a timely manner. The automated test suite has significantly reduced the time and effort required for testing, enabling faster release cycles and improved overall quality of the website. This report presents a comprehensive overview of the automation testing process, highlighting its benefits, challenges, and future directions.

Keywords: Selenium WebDriver, Test Automation, Amazon, User Flow, Web Application Testing, Quality Assurance, Regression Testing, Continuous Integration, Error Handling, Explicit Waits.

ACKNOWLEDGMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible. Success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that I acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success. I would like to thank Dr. Sanjay Jain, Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

I would like to express my gratitude towards Dr. Jagadishwari V, Associate Professor and HOD, Department of Information Science and Engineering CMRIT, Bangalore, who provided guidance and gave valuable suggestions regarding the project.

I consider it a privilege and honor to express my sincere gratitude to our internal guide Ms. Saba Tahaseen , Assistant Professor, Department of Information Science and Engineering, CMRIT, Bangalore, for their valuable guidance throughout the ensure of this project work.

I would like to thank all the faculty members who have always been very cooperative and generous. Conclusively, I also thank all the non-teaching staff and all others who have done immense help directly or in-directly during our project.

Ayush Kumar Sharaf
(1CR21IS033)

Contents

List of Figures

Chapter 1
Introduction

1.1 What is Software Testing?

Software testing is an indispensable process in the software development lifecycle (SDLC) that involves the systematic evaluation of a software product or service to ascertain its adherence to specified requirements, functionalities, and user expectations. This critical practice encompasses a range of techniques and methodologies aimed at identifying defects, errors, or inconsistencies within the software, thereby ensuring its reliability, usability, and overall quality. Testing acts as a safeguard against potential malfunctions, security vulnerabilities, and performance bottlenecks that could adversely impact the user experience. By subjecting the software to rigorous examination through various types of tests, such as functional, non-functional, and regression testing, organizations can significantly reduce the risk of releasing flawed software, enhance customer satisfaction, and build a strong reputation for delivering reliable products.

1.2 Types of Testing

Software testing is a multi-faceted domain, encompassing a wide array of techniques tailored to address specific quality attributes and ensure the overall robustness of software products. Let's delve into some of the most common types of testing employed in the industry:

- **Manual Testing:** This traditional approach relies on human testers to manually execute test cases by interacting with the software, emulating real-world user scenarios. While flexible and adept at uncovering usability issues, it can be time-consuming and prone to human error.
- **Automated Testing:** Leveraging specialized tools and scripts, automated testing enables the execution of predefined test cases without human intervention. This methodology boasts increased efficiency, repeatability, and the ability to cover a wider range of test scenarios.
- **Unit Testing:** A fundamental building block of software testing, unit testing focuses on verifying the correctness of individual code units (functions, methods, or classes) in isolation. This helps pinpoint defects early in the development cycle.
- **Integration Testing:** As the name implies, integration testing checks the interaction between different software modules or components, ensuring seamless collaboration and data exchange.
- **System Testing:** Taking a holistic approach, system testing evaluates the entire integrated system against its requirements, verifying that it behaves as expected as a whole.
- **Acceptance Testing:** Often performed by end-users or stakeholders, acceptance testing determines whether the developed system satisfies the criteria for acceptance and meets the needs and expectations of the intended audience.
- **Performance Testing:** This type of testing assesses the responsiveness, stability, and scalability of the software under different workloads and usage patterns, ensuring optimal performance under real-world conditions.
- **Security Testing:** A critical aspect of software testing, security testing involves identifying vulnerabilities, weaknesses, or potential entry points for unauthorized access, data breaches, or other security threats.

- **Usability Testing:** Focused on the user's perspective, usability testing evaluates how intuitive, user-friendly, and efficient the software is in achieving user goals.
- **Regression Testing:** After modifications or updates to the software, regression testing is conducted to ensure that existing functionalities have not been adversely affected and that new issues have not been introduced.
- **Exploratory Testing:** A less structured approach, exploratory testing involves testers learning about the software through experimentation and discovery, uncovering defects that might not be caught by scripted tests.

Each of these testing types plays a crucial role in the comprehensive evaluation of software, ensuring that it meets stringent quality standards and delivers a seamless user experience. By adopting a combination of manual and automated testing techniques, organizations can optimize their testing efforts, detect defects early, and ultimately release software that is both reliable and user-centric.

1.3 About Automation Tool Used – Selenium WebDriver

Selenium WebDriver is a powerful and versatile open-source framework widely adopted for automating web browser interactions. It empowers developers and testers to create scripts in various programming languages (such as Java, Python, C#, Ruby, etc.) that can simulate user actions on web pages, including clicking buttons, filling forms, navigating through links, and validating expected outcomes. With its cross-browser compatibility, Selenium WebDriver can interact with major browsers like Chrome, Firefox, Safari, Edge, and Internet Explorer, enabling comprehensive testing across different platforms and environments.

One of Selenium WebDriver's core strengths lies in its ability to locate and manipulate web elements through a variety of strategies, including IDs, class names, XPath expressions, and CSS selectors. This flexibility allows testers to interact with even the most complex web page structures. Furthermore, Selenium WebDriver offers implicit and explicit waits, allowing scripts to pause and synchronize with the application under test, ensuring that elements are loaded and ready for interaction before proceeding.

Beyond its core functionality, Selenium WebDriver integrates seamlessly with popular testing frameworks like TestNG and JUnit, providing features like test organization, reporting, and parallelization. This integration facilitates structured test development, enhances maintainability, and optimizes test execution time. Additionally, Selenium WebDriver's extensible architecture allows for the integration of third-party libraries and tools, expanding its capabilities and addressing specific testing needs.

1.4 Problem Statement

Testing the user flow in amazon website from login till scraping product details from automated search queries and then logout.

1.5 Objective of the Project

This project addresses the challenges of manual testing by harnessing the power of Selenium WebDriver to automate a critical user flow on the Amazon website. The primary objectives of this project are as follows:

1. **Enhance Efficiency:** By automating the login, profile selection, search, content preview, and logout steps, we aim to significantly reduce the time and effort required for testing compared to manual execution. This will streamline the testing process and free up valuable resources for other testing activities.
2. **Improve Reliability:** Through automated test scripts, we seek to eliminate human errors that can occur during manual testing, ensuring consistent and reproducible test results. This will enhance the reliability and accuracy of the testing process, leading to greater confidence in the software's quality.
3. **Increase Test Coverage:** The automated test suite will be designed to cover a wide range of user scenarios, including valid and invalid logins, different profile selections, various search terms, and error handling for unexpected events. This increased coverage will help identify potential issues that might be missed in manual testing.
4. **Early Bug Detection:** By integrating the automated tests into the development workflow, we aim to detect and report defects early in the development cycle. This early feedback loop will allow for faster bug fixes, reducing the risk of critical issues reaching production and impacting the user experience.
5. **Continuous Integration:** The project aims to establish a foundation for continuous integration, where the automated test suite can be seamlessly integrated into the development pipeline. This will enable automated testing with every code commit, ensuring that new changes do not introduce regressions or break existing functionalities.

Chapter 2
Literature Survey

2.1 About Manual Testing

Manual testing is a time-tested approach in software quality assurance where human testers meticulously execute test cases, interact with the software, and assess its functionality based on predefined expectations. This hands-on approach involves meticulously following test scripts, simulating user scenarios, and carefully observing the software's behavior.

Advantages of Manual Testing:

1. **Exploratory Testing:** Manual testing allows testers to explore the software intuitively, uncovering unexpected issues or edge cases that might not be covered in scripted test cases. This is particularly valuable for uncovering usability problems or design flaws.
2. **Flexibility and Adaptability:** Manual testers can quickly adapt to changes in requirements, user scenarios, or software updates. They can readily modify their testing approach based on real-time observations.
3. **Cost-Effective for Small Projects:** For smaller projects with limited scope and budget, manual testing can be a more cost-effective option, as it doesn't require the initial investment in automation tools and infrastructure.
4. **User Experience Focus:** Manual testers can assess the software from a user's perspective, providing valuable insights into usability, intuitiveness, and overall user experience.

Disadvantages of Manual Testing:

1. **Time-Consuming:** Manual execution of test cases, especially for large and complex applications, can be extremely time-consuming and resource-intensive.
2. **Prone to Human Error:** Repetitive tasks can lead to fatigue and mistakes, potentially missing critical defects.
3. **Limited Test Coverage:** Due to time and resource constraints, manual testing may not be able to cover all possible combinations of inputs and scenarios, leading to potential gaps in test coverage.
4. **Non-Repeatable:** Manual tests are not easily repeatable, making it challenging to consistently reproduce and diagnose defects.

2.2 About Automation Testing

Automation testing represents a paradigm shift in software testing, where software tools and scripts take center stage in executing test cases, validating results, and generating comprehensive reports. This methodology has gained significant traction due to its ability to increase efficiency, accuracy, and test coverage.

Advantages of Automation Testing:

1. **Efficiency and Speed:** Automated tests can be executed rapidly, significantly reducing the time required for testing cycles. This accelerated feedback loop enables faster identification and resolution of defects.
2. **Increased Accuracy:** By eliminating human error, automated tests provide more consistent and reliable results, ensuring that the software behaves as expected under various conditions.
3. **Expanded Test Coverage:** Automation enables the execution of a vast number of test cases across different environments, platforms, and configurations, leading to broader test coverage and identifying issues that might be missed in manual testing.
4. **Cost-Effective for Large Projects:** While the initial setup of automated tests requires investment, it can be highly cost-effective in the long run, especially for large-scale projects with extensive test suites.
5. **Reusability and Scalability:** Automated test scripts can be reused for different versions of the software, and the test suite can be easily scaled to accommodate new functionalities.
6. **24/7 Availability:** Automated tests can be scheduled to run at any time, even outside of business hours, enabling continuous testing and faster feedback.
7. **Integration with CI/CD:** Automated tests seamlessly integrate into CI/CD pipelines, providing immediate feedback on code changes and ensuring that each build meets quality standards.

Disadvantages of Automation Testing:

1. **Initial Investment:** Setting up and maintaining automated test scripts requires upfront effort, skilled resources, and investment in automation tools.
2. **Limited Flexibility:** Automated tests are less flexible than manual tests and may not be able to adapt to unexpected scenarios or changes in requirements as easily.
3. **Maintenance Overhead:** Automated test scripts need to be updated and maintained as the software evolves, adding to the overall maintenance effort.
4. **False Positives/Negatives:** Automated tests can sometimes produce inaccurate results due to scripting errors, environment issues, or unexpected application behavior.
5. **Not a Replacement for Manual Testing:** Automation testing cannot entirely replace manual testing. Manual testing is still essential for exploratory testing, usability testing, and verifying aspects that are difficult to automate.

Chapter 3
System Architecture and Design

3.1 Architecture/Methodology Used

- **Test Script :** This Python program, utilizing Selenium WebDriver, contains instructions to interact with the Amazon web application.
- **Selenium WebDriver:** A collection of libraries that provides a programming interface to control web browsers. It acts as a bridge between the test script and the browser.
- **ChromeDriver:** A specific WebDriver implementation designed to interact with the Google Chrome browser.
- **Chrome Browser:** The web browser used for executing the automated tests.
- **Amazon Web Application:** The target system being tested.

3.2 Flowchart/Path Flow Diagram

Initialize WebDriver and Navigate to Amazon

- **Initialize WebDriver**: Start a ChromeDriver instance.
- **Navigate to Amazon login page**: Open the amazon login page .

Sign In to Amazon

- **Click "Sign In"**: Locate and click the "Sign In" button.
- **Enter Credentials**: Fill in the username and password fields.
- **Click Login**: Submit the login form.

- **Navigate to Amazon product page**: Open the amazon product pages .

Search through product pages

- **Fetch product cards:** Used to fetch the product details and select important attributes.
- **Convert to HTML:** All collected cards are converted to HTML using BeautifulSoup.
- **Attributes to Excel**: Selected Attributes from HTML are added to the excel sheet

Log Out of Amazon

- **Sign out :** After process is finished , user is signed out of the website

Chapter 4
System Architecture and Design

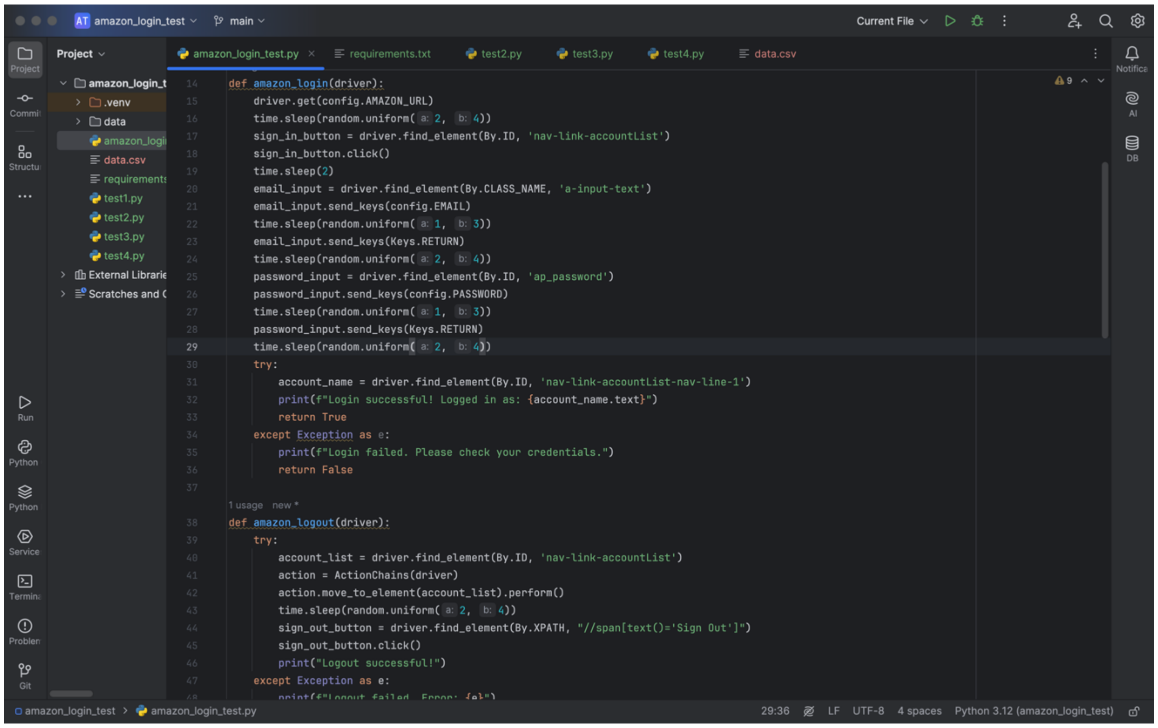
4.1 Test Cases Table

Test Case ID	**Description**	**Expected Result**	**Actual Result**
TC_01	Login with valid credentials	Successful login and redirect to product page	Pass
TC_02	Login with Incorrect password	Login Credentials Wrong	Pass
TC_03	Login with Incorrect email	Login Credentials Wrong	Pass
TC_04	Login with incorrect email and password	Login Credentials Wrong	Pass

4.2 Black Box / White Box Testing

The implemented test is a black-box test, as it focuses on verifying the functionality of the Amazon website without considering its internal code or implementation details.

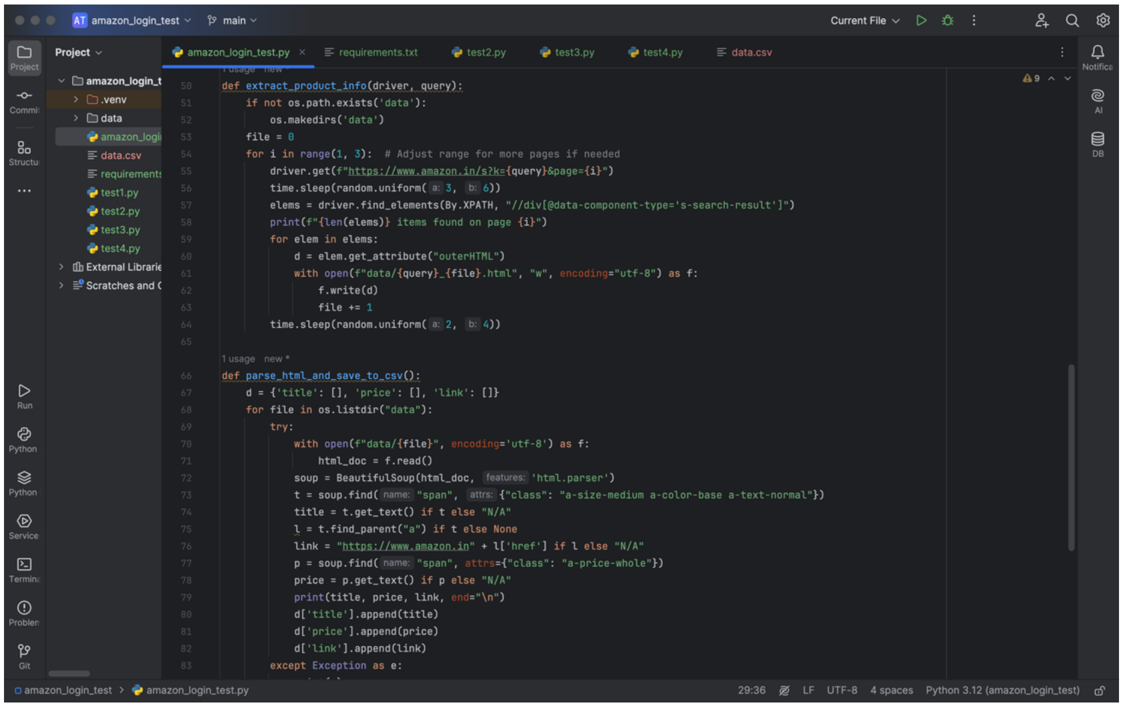
4.3 Code



```python
def amazon_login(driver):
    driver.get(config.AMAZON_URL)
    time.sleep(random.uniform(2, 4))
    sign_in_button = driver.find_element(By.ID, 'nav-link-accountList')
    sign_in_button.click()
    time.sleep(2)
    email_input = driver.find_element(By.CLASS_NAME, 'a-input-text')
    email_input.send_keys(config.EMAIL)
    time.sleep(random.uniform(1, 3))
    email_input.send_keys(Keys.RETURN)
    time.sleep(random.uniform(2, 4))
    password_input = driver.find_element(By.ID, 'ap_password')
    password_input.send_keys(config.PASSWORD)
    time.sleep(random.uniform(1, 3))
    password_input.send_keys(Keys.RETURN)
    time.sleep(random.uniform(2, 4))
    try:
        account_name = driver.find_element(By.ID, 'nav-link-accountList-nav-line-1')
        print(f"Login successful! Logged in as: {account_name.text}")
        return True
    except Exception as e:
        print(f"Login failed. Please check your credentials.")
        return False

def amazon_logout(driver):
    try:
        account_list = driver.find_element(By.ID, 'nav-link-accountList')
        action = ActionChains(driver)
        action.move_to_element(account_list).perform()
        time.sleep(random.uniform(2, 4))
        sign_out_button = driver.find_element(By.XPATH, "//span[text()='Sign Out']")
        sign_out_button.click()
        print("Logout successful!")
    except Exception as e:
        print(f"Logout failed. Error: {e}")
```

Fig 1: Code snippet



```python
def extract_product_info(driver, query):
    if not os.path.exists('data'):
        os.makedirs('data')
    file = 0
    for i in range(1, 3):  # Adjust range for more pages if needed
        driver.get(f"https://www.amazon.in/s?k={query}&page={i}")
        time.sleep(random.uniform(3, 6))
        elems = driver.find_elements(By.XPATH, "//div[@data-component-type='s-search-result']")
        print(f"{len(elems)} items found on page {i}")
        for elem in elems:
            d = elem.get_attribute("outerHTML")
            with open(f"data/{query}_{file}.html", "w", encoding="utf-8") as f:
                f.write(d)
                file += 1
        time.sleep(random.uniform(2, 4))

def parse_html_and_save_to_csv():
    d = {'title': [], 'price': [], 'link': []}
    for file in os.listdir("data"):
        try:
            with open(f"data/{file}", encoding='utf-8') as f:
                html_doc = f.read()
            soup = BeautifulSoup(html_doc, 'html.parser')
            t = soup.find("span", {"class": "a-size-medium a-color-base a-text-normal"})
            title = t.get_text() if t else "N/A"
            l = t.find_parent("a") if t else None
            link = "https://www.amazon.in" + l['href'] if l else "N/A"
            p = soup.find("span", attrs={"class": "a-price-whole"})
            price = p.get_text() if p else "N/A"
            print(title, price, link, end="\n")
            d['title'].append(title)
            d['price'].append(price)
            d['link'].append(link)
        except Exception as e:
```

Fig 2: Code snippet

Chapter 5
Results/Output

5.1 Screenshots

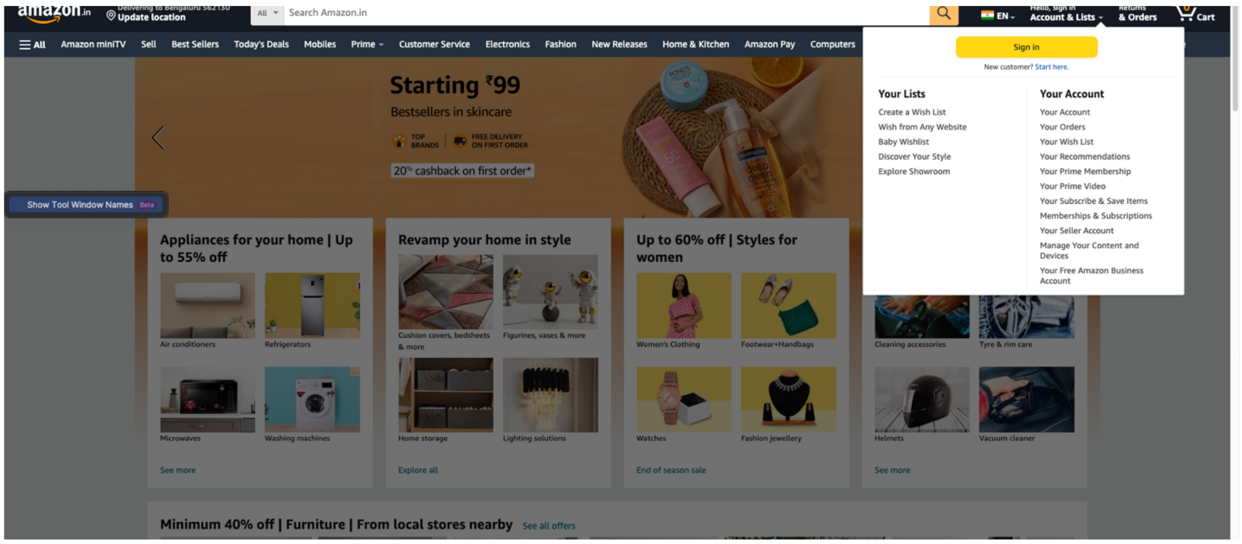
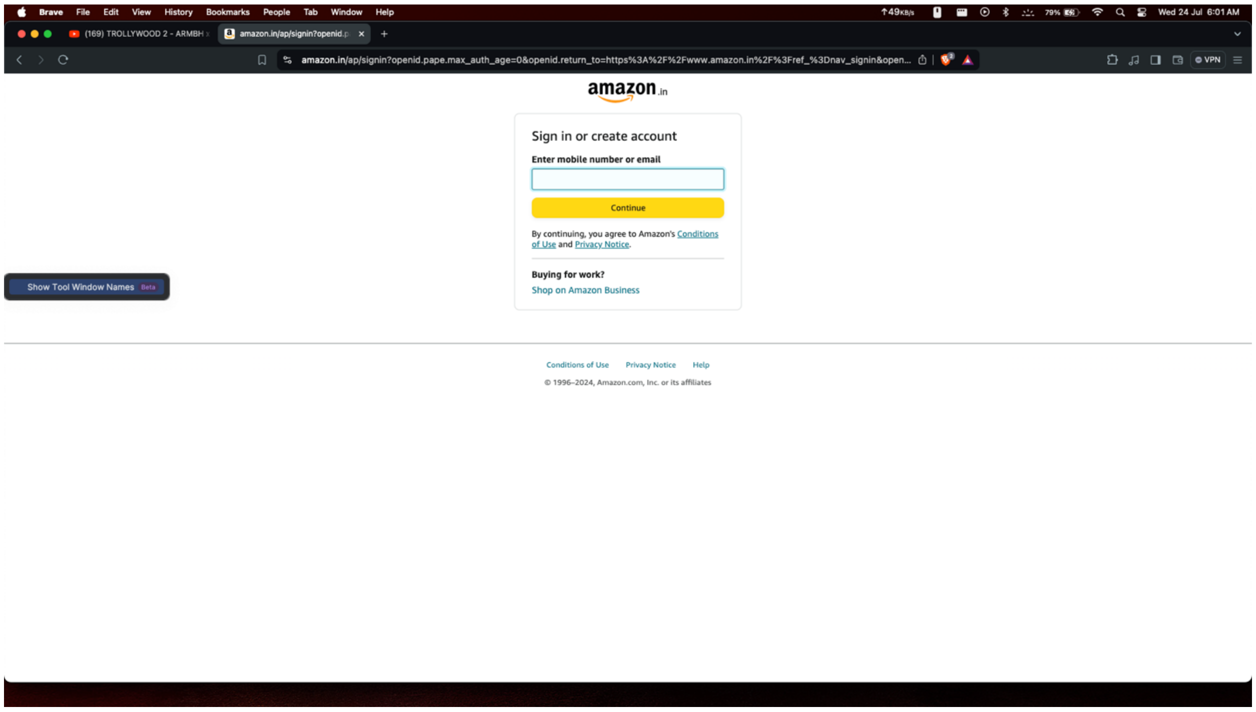


Fig 3: Site Automation


Brave
File
Edit
View
History
Bookmarks
People
Tab
Window
Help
Wed 24 Jul 6:01 AM
(169) TROLLYWOOD 2 - ARMBH
amazon.in/ap/signin?openid.pape.max_auth_age=0&openid.return_to=https%3A%2F%2Fwww.amazon.in%2F%3Fref_%3Dnav_signin&open...
VPN
amazon.in
Show Tool Window Names

Sign in or create account

Enter mobile number or email

Continue

By continuing, you agree to Amazon's Conditions of Use and Privacy Notice.

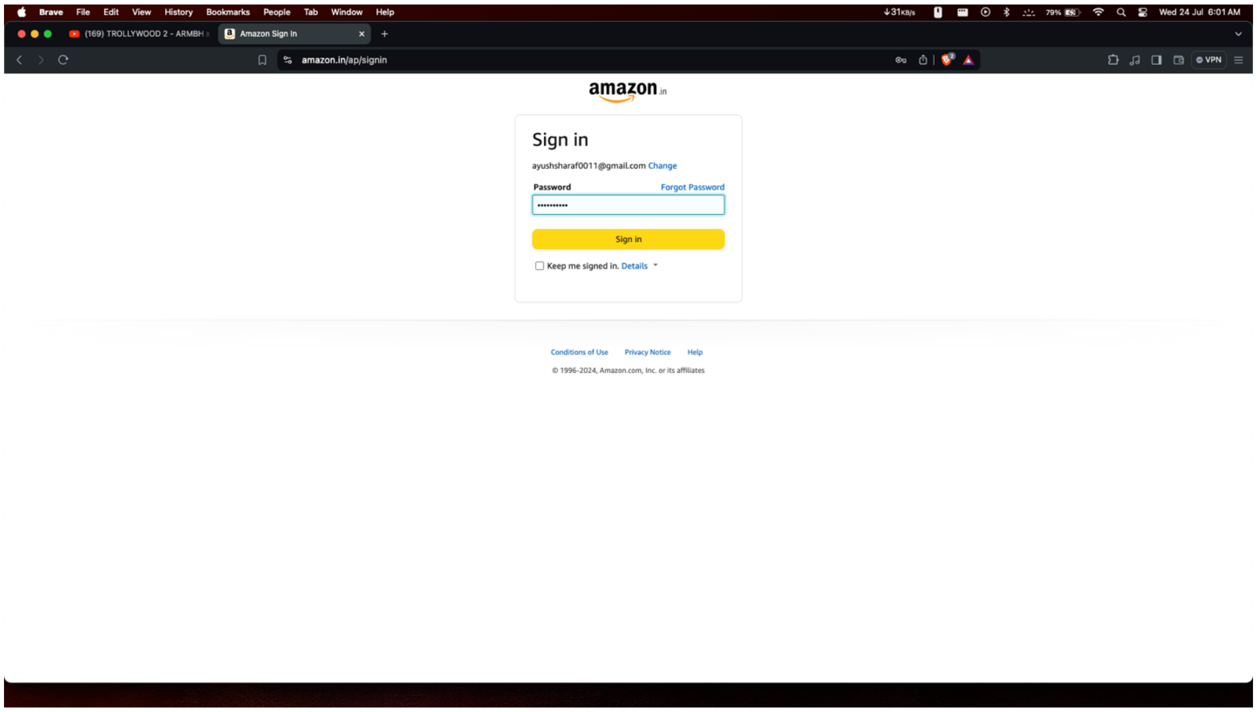
Buying for work?

Shop on Amazon Business

Conditions of Use Privacy Notice Help

© 1996–2024, Amazon.com, Inc. or its affiliates

Fig 4: Login Page



amazon.in

Sign in

ayushsharaf0011@gmail.com Change

Password Forgot Password

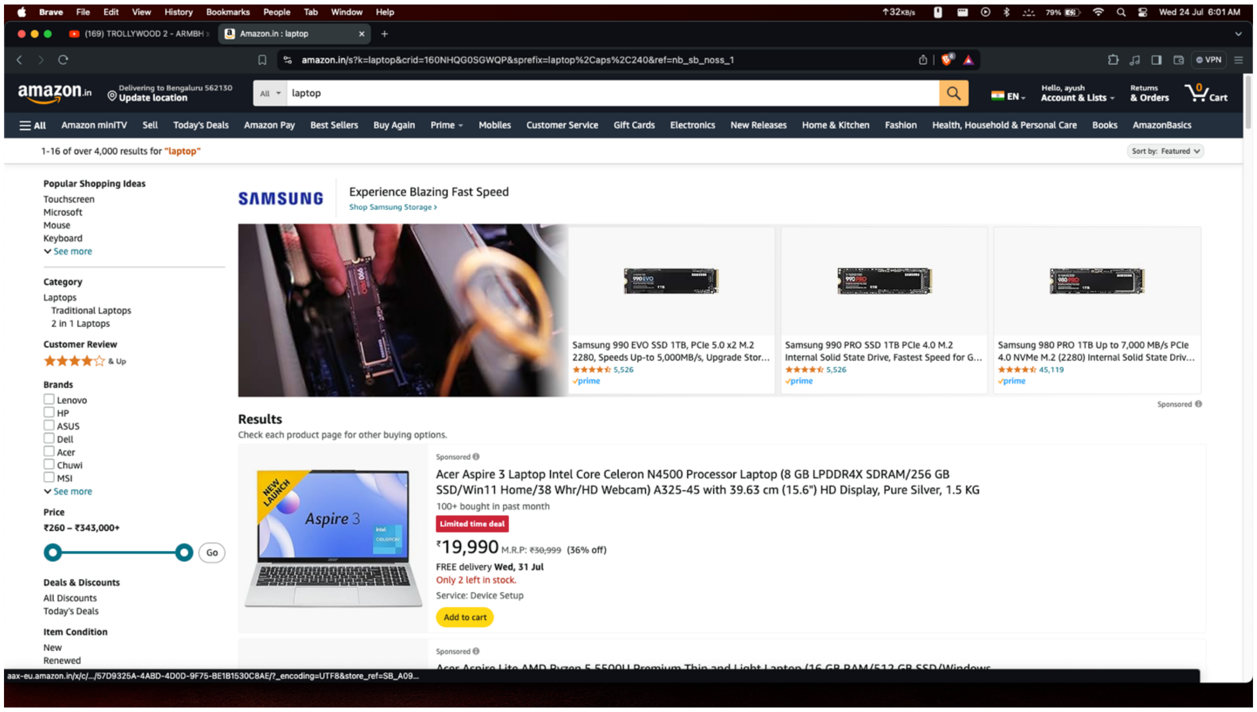
Sign in

Keep me signed in. Details

Conditions of Use Privacy Notice Help

© 1996-2024, Amazon.com, Inc. or its affiliates

Fig 5: Entering Password



amazon.in/s?k=laptop&crid=160NHQG0SGWQP&sprefix=laptop%2Caps%2C240&ref=nb_sb_noss_1

1-16 of over 4,000 results for "laptop"

Samsung 990 EVO SSD 1TB, PCIe 5.0 x2 M.2 2280, Speeds Up-to 5,000MB/s, Upgrade Stor…

Samsung 990 PRO SSD 1TB PCIe 4.0 M.2 Internal Solid State Drive, Fastest Speed for G…

Samsung 980 PRO 1TB Up to 7,000 MB/s PCIe 4.0 NVMe M.2 (2280) Internal Solid State Driv…

Results

Check each product page for other buying options.

Acer Aspire 3 Laptop Intel Core Celeron N4500 Processor Laptop (8 GB LPDDR4X SDRAM/256 GB SSD/Win11 Home/38 Whr/HD Webcam) A325-45 with 39.63 cm (15.6") HD Display, Pure Silver, 1.5 KG

100+ bought in past month

Limited time deal

₹19,990 M.R.P: ~~₹30,999~~ (36% off)

FREE delivery Wed, 31 Jul

Only 2 left in stock.

Service: Device Setup

Add to cart

Fig 6: Search Query Automation

Chapter 6
Conclusion/Future Scope

6.1Conclusion

This project successfully showcases the power and efficiency of Selenium WebDriver in automating a crucial user journey on the Amazon platform. By simulating real user interactions, including website opening, content search, and logout, the automated test suite has successfully verified the functionality and reliability of these core features. The integration of explicit waits and error handling mechanisms further ensures the stability and robustness of the test script, making it a valuable asset for continuous testing and regression testing efforts.

6.2 Future Scope

The current project provides a solid foundation for expanding the scope of automated testing on the Amazon platform. Future enhancements could include:

- **Advanced Search Scenarios:** Test various search filters, sorting options, and error handling for invalid search terms.
- **Content Interaction:** Automate interactions with product reviews, such as liking and disliking reviews. Test product page navigation, including product description and specifications.
- **Profile Management:** Test features related to profile creation, editing, and deletion.
- **Subscription Management:** Automate tests for subscription plans (Amazon Prime), payment methods, and cancellation processes.
- **Recommendation Engine:** Evaluate the accuracy and relevance of Amazon's recommendation system by testing various user profiles and viewing histories.
- **Device Compatibility:** Extend the test suite to cover different devices and screen sizes, ensuring a consistent user experience across platforms.
- **Localization Testing:** Verify the correctness of content translations and UI elements for different regions.
- **Performance Testing:** Assess the performance and responsiveness of the Amazon application under different load conditions.
- **Security Testing:** Identify and mitigate potential security vulnerabilities in the login and profile management features.
- **Accessibility Testing:** Ensure that the platform is accessible to users with disabilities, adhering to accessibility guidelines.

By continuously expanding and refining the automated test suite, Amazon can proactively detect and resolve issues, improve the user experience, and maintain a high level of quality and reliability for its vast global audience.

References

[1] Selenium Documentation: https://www.selenium.dev/documentation/
[2] XPath Tutorial: https://www.w3schools.com/xml/xpath_intro.asp
[4] Agile Testing by Lisa Crispin and Janet Gregory: https://www.amazon.com/Agile-Testing-Practical-Guide-Testers/dp/0321534468
[5] Amazon Technology Blog: https://aws.amazon.com/blogs
[6] Test Automation in Practice by Bas Dijkstra and Dorothy Graham
[7] Complete Guide to Test Automation by Arnon Axelrod
[8] Software Testing: Principles and Practices by Srinivasan Desikan and Gopalaswamy Ramesh
[10] Automation Step by Step (YouTube Channel): Offers step-by-step guides and tutorials for beginners in automation testing.