

Selenium Automation

A File Submitted
In Partial Fulfilment of the Requirements
for the Degree of

Bachelor of Technology

(COMPUTER SCIENCE)

By

Tanu Sharma, Roll NO. 2207990109008

Devendra Pratap Singh, Roll No. 2207990109001

Pushpendra, Roll NO. 2207990109005



**ACE COLLEGE OF ENGINEERING AND
MANAGEMENT, AGRA
(College Code-799)**

AFFILIATED TO
**Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW, UTTAR PRADESH
2024-2025**

Declaration

I hereby declare that this SYNOPSIS report entitled, "**Selenium Automation**", being submitted by Tanu, Devendra Pratap Singh and Pushpendra to the **DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW** for the award of the degree of Bachelor of Technology (CS) is a record of Bonafide work carried by Tanu Sharma (Roll no-2207990109008), Devendra Pratap Singh (Roll no- 2207990109001) & Pushpendra (Roll No. 2207990109005).

The matter embodied in this report has not been submitted to any other university or institute earlier for the award of any degree or diploma

Tanu Sharma, Roll NO. 2207990109008

Devendra Pratap Singh, Roll No. 2207990109005

Pushpendra, Roll NO. 2207990109005

B.Tech 8th SEM (2024-2025)

1.Introduction

In today's fast-paced digital world, automation has become an essential part of software testing and web interactions. This project is focused on automating common user actions on the popular e-commerce platform **Amazon.in** using **Selenium WebDriver** and **Java**.

The main objective of this project is to simulate real-time user actions like opening the browser, logging into an Amazon account, searching for a product, extracting relevant details like product name, price, and rating, adding an item to the shopping cart, and taking a screenshot of the cart page. By automating these steps, the project demonstrates how web-based tasks can be streamlined to save time and reduce manual effort.

This automation script not only helps understand Selenium's capabilities but also builds a solid foundation for more complex testing frameworks. It can be a stepping stone toward developing robust automated testing systems for real-world applications.

2.Objective



The main objective of this project is to automate key user interactions on the Amazon India website using **Selenium WebDriver** with **Java**. It aims to simulate the real-world behavior of an online shopper and perform the following tasks automatically:

- Launch the Chrome browser and open the Amazon.in website.
- Log in to a registered Amazon account using valid credentials.
- Search for a specific product (e.g., Android phone with 8GB RAM, 128GB storage, and 6000mAh battery).
- Extract and display product details such as name, price, and rating of the top 5 results.
- Add a selected product to the shopping cart.
- Navigate to the cart page and take a screenshot of the cart.
- Save the screenshot locally for verification.



By achieving these goals, the project demonstrates how repetitive tasks on a web application can be automated efficiently using Selenium, which is especially useful in software testing and data scraping.

3. Technologies Used



1. Programming Language: Java

-  Used to develop the automation script.
-  Supports object-oriented concepts, exception handling, and easy integration with Selenium.



2. Automation Tool: Selenium WebDriver

-  Automates browser interactions like clicks, form submissions, and data extraction.
-  Works across different browsers (Chrome, Firefox, Edge, etc.).



3. Browser Driver: ChromeDriver

-  Acts as a connector between Selenium and the Chrome browser.
-  Required to run automation scripts on Google Chrome.



4. Browser: Google Chrome

-  The browser in which automation tasks are performed.
-  Widely supported and commonly used in real-world automation testing.



5. Library: Apache Commons IO

-  Used for file operations, especially saving screenshots.
-  Method: `FileUtils.copyFile()` is used to store the cart screenshot as an image file.



6. IDE (Integrated Development Environment):

-  Eclipse or IntelliJ IDEA
-  Makes coding, debugging, and managing the project easier.

7. Build Tool (Optional): Maven / Gradle

-  Can be used to manage dependencies like Selenium and Apache Commons IO.
 -  Useful in larger projects for clean builds and automation.
-

☐ 8. Operating System: Windows 10 / 11

-  Development and execution were done on a Windows environment.
-  Paths for ChromeDriver and screenshot folders are Windows-based.

4. System Requirements

Hardware Requirements

- **Processor:** Intel i3 or higher (Recommended i5/i7 for faster execution)
 - **RAM:** Minimum 4 GB (8 GB recommended for smoother performance)
 - **Storage:** At least 1 GB free space for project files and screenshots
 - **Display:** 1366 x 768 resolution or higher for better UI visibility
-

Software Requirements

Component	Version/Specification
Operating System	Windows 10 / 11 (64-bit)
Java Development Kit (JDK)	Version 8 or above
Browser	Google Chrome (Latest version recommended)
ChromeDriver	Compatible with the installed Chrome version
IDE	Eclipse IDE / IntelliJ IDEA
Selenium WebDriver	Version 4.x (Latest stable release)
Apache Commons IO	Version 2.x (for file handling)

Network Requirements

- Stable internet connection to access Amazon website during automation.
 - Firewall/Antivirus should allow ChromeDriver and Selenium scripts to run without blocking.
-

Other Requirements

- Correct system path set for ChromeDriver executable in the script.
- Necessary permissions to save screenshots in the specified directory (e.g., Desktop folder).

5.Modules:

1. Browser Initialization Module

- **Purpose:** To launch the Chrome browser and open the Amazon India website.
- **Function Name:** `open_browser()`
- **Key Tasks:**
 - Set the path for the ChromeDriver.
 - Initialize the Chrome browser.
 - Maximize the browser window.
 - Navigate to the Amazon website.
 - Set implicit waits for synchronization.

2. Login Module

- **Purpose:** To log in to the user's Amazon account securely.
- **Function Name:** `login (WebDriver driver)`
- **Key Tasks:**
 - Click on the "Sign In" button.
 - Enter valid email/phone and password.
 - Click the login/submit button to authenticate the user.

3. Product Search and Data Extraction Module

- **Purpose:** To search for a product and extract details from search results.
- **Function Name:** `searchItem(WebDriver driver)`
- **Key Tasks:**
 - Search for a product using the Amazon search bar.
 - Retrieve product details such as:
 - Product title/brand
 - Price (if available)
 - Customer rating (if available)
 - Display the information for the top 5 products.

4. Add to Cart Module

- **Purpose:** To add one of the listed products to the shopping cart.
- **Function Location:** Inside `searchItem()`

- **Key Tasks:**
 - Click the “Add to Cart” button.
 - Wait for confirmation that the product has been added.

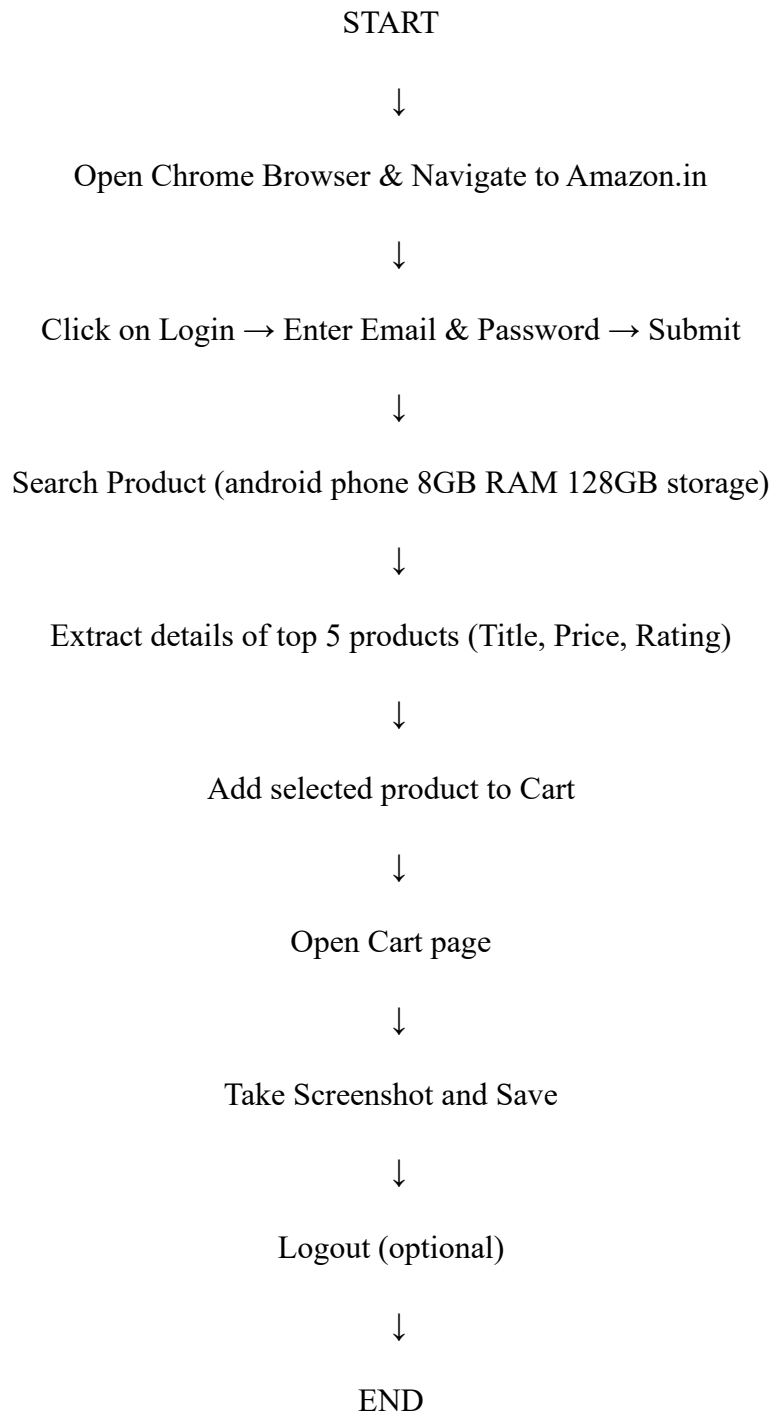
5. Cart Navigation and Screenshot Module

- **Purpose:** To open the shopping cart and take a screenshot of the cart contents.
- **Function Location:** Also handled inside searchItem()
- **Key Tasks:**
 - Click on the cart icon.
 - Capture a screenshot of the cart page.
 - Save the screenshot to a specified directory.

6. Logout Module

- **Purpose:** To log the user out of their Amazon account.
- **Function Name:** logOut(WebDriver driver)
- **Key Tasks:**
 - Hover or click on the account menu.
 - Select and click the “Sign Out” option.
 - Handle waits to ensure smooth navigation.

6. Project Workflow



7. 📄 ✨ Code Explanation

🚀 1. open_browser()

- **Purpose:**

Sets up and launches the Chrome browser, then opens Amazon India's homepage.

- **Details:**

- System.setProperty() ➡ Sets the path for ChromeDriver.
 - new ChromeDriver() 🖥 Starts the Chrome browser.
 - driver.get() 🌐 Opens the Amazon website.
 - Implicit wait ⌚ allows page elements to load properly.
 - Calls the login process (login()).
-

🔑 2. login(WebDriver driver)

- **Purpose:**

Automates logging into your Amazon account.

- **Steps:**

- Clicks login button 👉.
 - Enters email/phone 📧 and password 🔒.
 - Clicks continue and sign-in buttons ✅.
 - Calls product search method (searchItem()).
-

🔍 3. searchItem(WebDriver driver)

- **Purpose:**

Searches for products, extracts details, adds to cart, and takes screenshot.

- **How it works:**

- Finds search box 🔍 and types product query.
 - Clicks search button 🔍.
 - Gets list of products 📄 from search results.
 - For top 5 products:
 - Extracts product title 🏷
-

- Tries to get price 💰 (if available)
- Tries to get rating ★ (if available)
- Prints details to console 🖨️.
- Adds first product to cart 🛒.
- Opens cart page 📁.
- Takes screenshot 📷 and saves locally.

📁 4. logOut(WebDriver driver)

- **Purpose:**

Logs out of the Amazon account (optional).

- **Details:**

- Finds and clicks logout link 📁.
- Waits a few seconds for logout to complete ⌚.

🎯 5. main(String[] args)

- **Purpose:**

The starting point of the program.

- **Action:**

- Calls open_browser() to kick off the automation flow 🚩.

⚠️ Exception Handling

- Uses try-catch blocks to:

- Prevent crashes ❌ if elements (price, rating) are missing.
- Handle unexpected errors smoothly 🧡.

7. 📸 Screenshots

Step 1: 🚀 Open Browser and Navigate to Amazon

```
// Opening Chrome browser and navigating to Amazon homepage

System.setProperty("WebDriver.chrome.driver","E:\\Selenium webDriver\\chromedriver-win64\\chromedriver.exe");

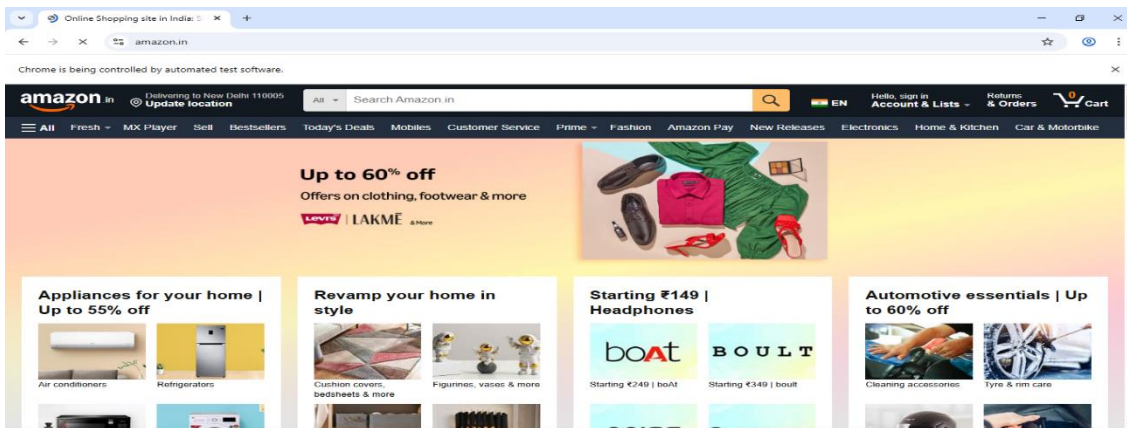
WebDriver driver = new ChromeDriver ();

driver.manage().window().maximize();

driver.get("https://www.amazon.in/");

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Output Screenshot:



Step 2: 🔑 Login to Amazon Account

```
// Click login and enter credentials

driver.findElement(By.id("nav-link-accountList-nav-line-1")).click();

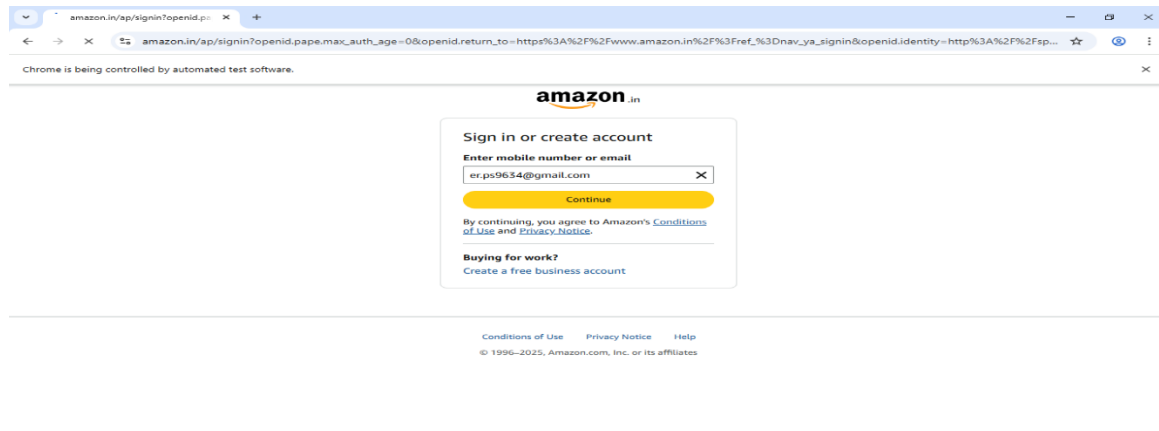
driver.findElement(By.id("ap_email_login")).sendKeys("your_email@example.com");

driver.findElement(By.className("a-button-input")).click();

driver.findElement(By.id("ap_password")).sendKeys("your_password");

driver.findElement(By.className("a-button-input")).click();
```

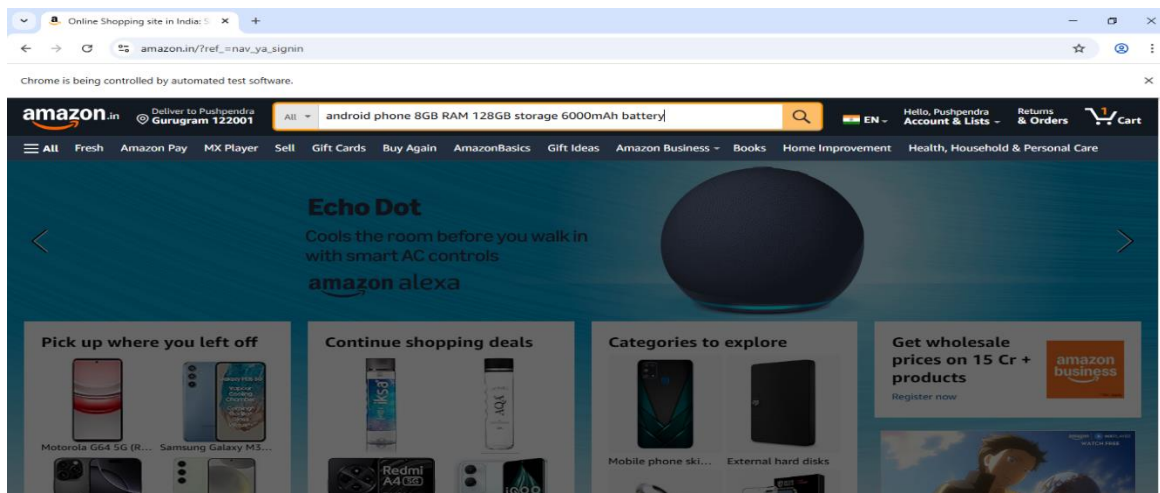
Output Screenshot:



Step 3: 🔍 Search for Product and Extract the data

```
// Search for android phone with specified features  
WebElement searchBox = driver.findElement(By.id("twotabsearchtextbox"));  
searchBox.sendKeys("android phone 8GB RAM 128GB storage 6000mAh battery");  
driver.findElement(By.id("nav-search-submit-button")).click();  
Thread.sleep(3000);
```

Output



```

SeleniumClasses > src > main > java > Amazon_project > open_browser
Run Amazon_project
May 24, 2023 8:49:44 AM org.openqa.selenium.devtools.cdp.log.Logger warn
WARNING: Unable to find CDP implementation matching 136
May 24, 2023 8:49:44 AM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 136.0.7103.116. You may need to include a dependency on a specific version of the CDP using
Brand/Model: Samsung Galaxy M35 5G (Moonlight Blue,8GB RAM,128GB Storage)| Corning Gorilla Glass Victus+| AnTuTu Score 595K+ | Vapour Cooling
Price: ₹10,998
Rating:
-----
Brand/Model: Redmi Note 14 5G (Phantom Purple, 8GB RAM 128GB Storage) | Global Debut MTK Dimensity 7025 Ultra | 2100 nits Segment Brightest 12
Price: ₹18,998
Rating:
-----
Brand/Model: Samsung Galaxy M35 5G (Moonlight Blue,8GB RAM,128GB Storage)| Corning Gorilla Glass Victus+| AnTuTu Score 595K+ | Vapour Cooling
Price: ₹10,498
Rating:
-----
Brand/Model: Samsung Galaxy M35 5G (Daybreak Blue,8GB RAM,128GB Storage)| Corning Gorilla Glass Victus+| AnTuTu Score 595K+ | Vapour Cooling C
Price: ₹10,498
Rating:
-----
Brand/Model: realme NARZO 80x 5G (Deep Ocean,8GB+128GB) | | Dimensity 6400 5G Chipset | 6000mAh Long-Lasting Battery | 45W Fast Charge | 120Hz
Price: ₹13,998
Rating:
-----
Product added to cart successfully.
Process finished with exit code 0

```

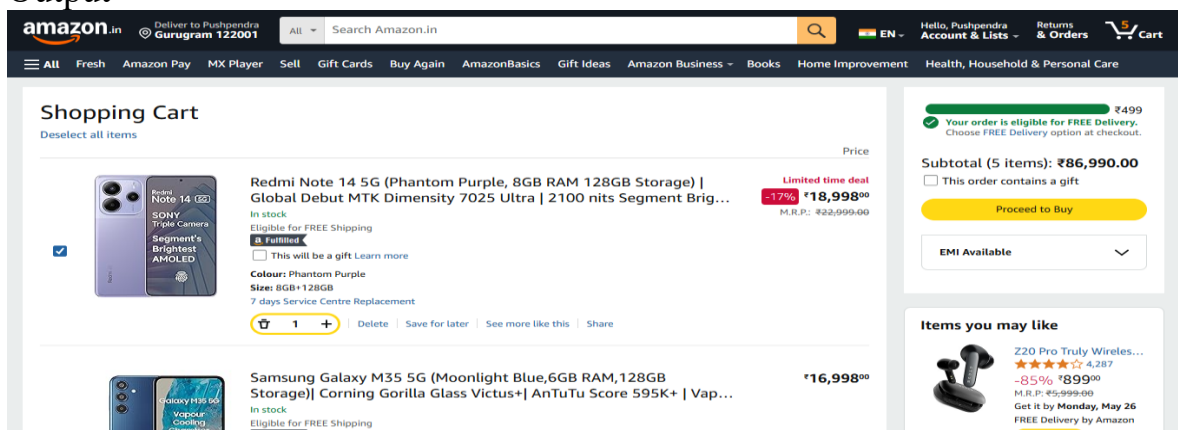
Step 4: 🛒 Add Product to Cart and 📸 Take Screenshot

```

// Add first product to cart
driver.findElement(By.id("a-autoid-1-announce")).click();
Thread.sleep(3000);
// Take screenshot
TakesScreenshot screenshot = (TakesScreenshot) driver;
File src = screenshot.getScreenshotAs(OutputType.FILE);
File dest = new File("C:\\Users\\Pcz\\Desktop\\amazon screenshot\\AddToCart.png");
FileUtils.copyFile(src, dest);

```

Output



8. ⚠️🔧 Challenges Faced

1. 🌐 Dynamic Elements:

Amazon page elements (IDs/classes) change often, so flexible locators like CSS selectors were used.

2. ⌚ Wait Issues:

Elements load slowly sometimes, causing errors; solved with implicit and explicit waits.

3. ❓ Missing Data:

Some products lacked price 💰 or rating ★; try-catch blocks used to handle missing info gracefully.

4. 🖥️ CAPTCHA Challenge:

Amazon detected automation and showed CAPTCHA frequently, requiring manual intervention to continue.

5. 📷 Screenshots:

Needed proper casting and file handling to save screenshots without errors.

6. 🛒 Add to Cart Button:

Button locators changed or were nested, requiring careful element inspection.

7. 🔒 Login Stability:

Sessions timed out or failed sometimes; added waits and checks to keep login smooth.