**Project Report**



**Project Name:** DemonopCommerce

**Base URL:** [*https://demo.nopcommerce.com*](https://demo.nopcommerce.com)**Version:** 1.0 **Date:** 7-September-2025 **Prepared By:** Ankush

**TABLE OF CONTENTS**

[1. Executive Summary 3](#_Toc208176454)

[2. Project Overview 4](#_Toc208176455)

[3. Objectives & Scope 5](#_Toc208176456)

3.1 [Scope Inclusions 5](#_Toc208176457)

3.2 [Scope Exclusions 6](#_Toc208176458)

[4. Role & Responsibility 6](#_Toc208176459)

[5. Tools & Framework Architecture 6](#_Toc208176460)

[6. Test Environment 11](#_Toc208176461)

[7. Requirement Traceability Matrix (RTM) 11](#_Toc208176462)

[8. Execution Summary 12](#_Toc208176463)

[9. Defect Management 12](#_Toc208176465)

[10. Risks & Mitigation 12](#_Toc208176466)

[11. Benefits of Automation 12](#_Toc208176467)

[12. Challenges Faced 13](#_Toc208176468)

[13. Appendix 13](#_Toc208176469)

# 1. Executive Summary

This report provides a comprehensive overview of the automation testing project conducted on the **nopCommerce demo website**. The project primarily focused on validating essential **customer-facing workflows** such as **User Registration, Login, Product Browsing, Cart Management, and Checkout**. These modules represent the backbone of any e-commerce application and were prioritized to ensure stability, accuracy, and seamless user experience.

The automation framework was implemented using **Java**, **Selenium WebDriver**, **TestNG**, and **ExtentReports**, following the **Page Object Model (POM) design pattern**. This ensured that the framework was **scalable, reusable, and maintainable**, reducing overall test maintenance costs while improving readability and collaboration.

The objective of this project was not only to verify functional correctness but also to establish a **robust regression suite** that can be integrated into a CI/CD pipeline for continuous testing. This helps detect defects early, provides faster feedback to developers, and ultimately accelerates release cycles.

Key highlights of the project include:

* **Automation Coverage**: Comprehensive test coverage for the end-to-end user journey – from creating an account to completing a successful checkout.
* **Data-Driven Testing**: Test scenarios were executed with multiple sets of input data sourced from Excel, enabling thorough validation of edge cases and variations.
* **Reporting & Analytics**: ExtentReports integration provided rich, interactive HTML reports with detailed logs, screenshots, and test execution metrics.
* **Scalability**: The modular POM-based design allows easy addition of new test cases for future enhancements like Wishlist, Payment Gateways, and Admin Panel features.
* **Efficiency**: Manual regression testing that typically takes hours was reduced significantly, enabling faster verification of critical business workflows.

In conclusion, this project has successfully delivered a **highly reliable automation testing solution** for nopCommerce demo workflows. The solution enhances testing efficiency, reduces human error, and lays a strong foundation for scaling automation efforts across more complex modules in the future.

# 2. Project Overview

nopCommerce is a feature-rich, open-source **eCommerce platform** that powers thousands of online stores globally. Its modular architecture provides extensive functionalities such as catalog management, customer profiles, shopping carts, order handling, and integrated checkout flows.

For this project, the **automation scope was confined to the frontend modules** that simulate the real-world actions of a customer. These included registering as a new user, logging into the system, navigating product categories, adding items to the shopping cart, and completing the checkout process. These workflows were identified as **critical paths**, since they directly influence purchase success rates and user satisfaction.

The framework was built on **Java with Selenium WebDriver** for browser automation, **TestNG** for test orchestration, and **ExtentReports** for detailed reporting. To improve maintainability, the **Page Object Model (POM)** design pattern was applied, ensuring that UI elements and business logic were separated from the test scripts. This modularity enables quick adaptability to UI or business logic changes in nopCommerce.

Additionally, the project leveraged **data-driven testing** through Excel integration, allowing the suite to handle multiple datasets efficiently. This made it possible to validate not only the standard workflows but also negative and boundary scenarios without redundant coding efforts.

Another key aspect of this automation suite was its **scalability**. While this phase focused on Registration, Login, Product, Cart, and Checkout, the framework was intentionally designed to accommodate future modules such as Wishlist, Order History, and Payment Gateway validations.

The outcome is a **scalable, reusable, and regression-ready test suite** that provides confidence in nopCommerce’s functionality while reducing manual testing overhead.

# 3. Objectives & Scope

* Validate **core customer workflows** (registration, login, category navigation, product selection, shopping cart).
* Ensure **UI responsiveness and cross-browser compatibility**.
* Verify **product details and configurable specifications**.
* Validate **shopping cart accuracy** (pricing, quantity, subtotal).
* Maintain **regression stability** across releases.

## Scope Inclusions

* Registration and Login workflows.
* Category and subcategory navigation.
* Product details, sorting, filtering, and add-to-cart.
* Shopping cart preview and management.
* UI validation across browsers.
* Edge case handling (empty fields, invalid credentials).

## Scope Exclusions

* Admin panel testing.
* Third-party plugin behavior.
* End-to-end payment gateway flows (beyond cart).
* Performance/load testing at scale.

# 4. Role & Responsibility

|  |  |
| --- | --- |
| Role | Responsibility |
| QA Lead | Strategy, planning, reporting |
| Automation Engineer | Selenium + TestNG/Cucumber script development |
| Manual Tester | Exploratory and edge case testing |
| Developer | Bug fixes and environment support |

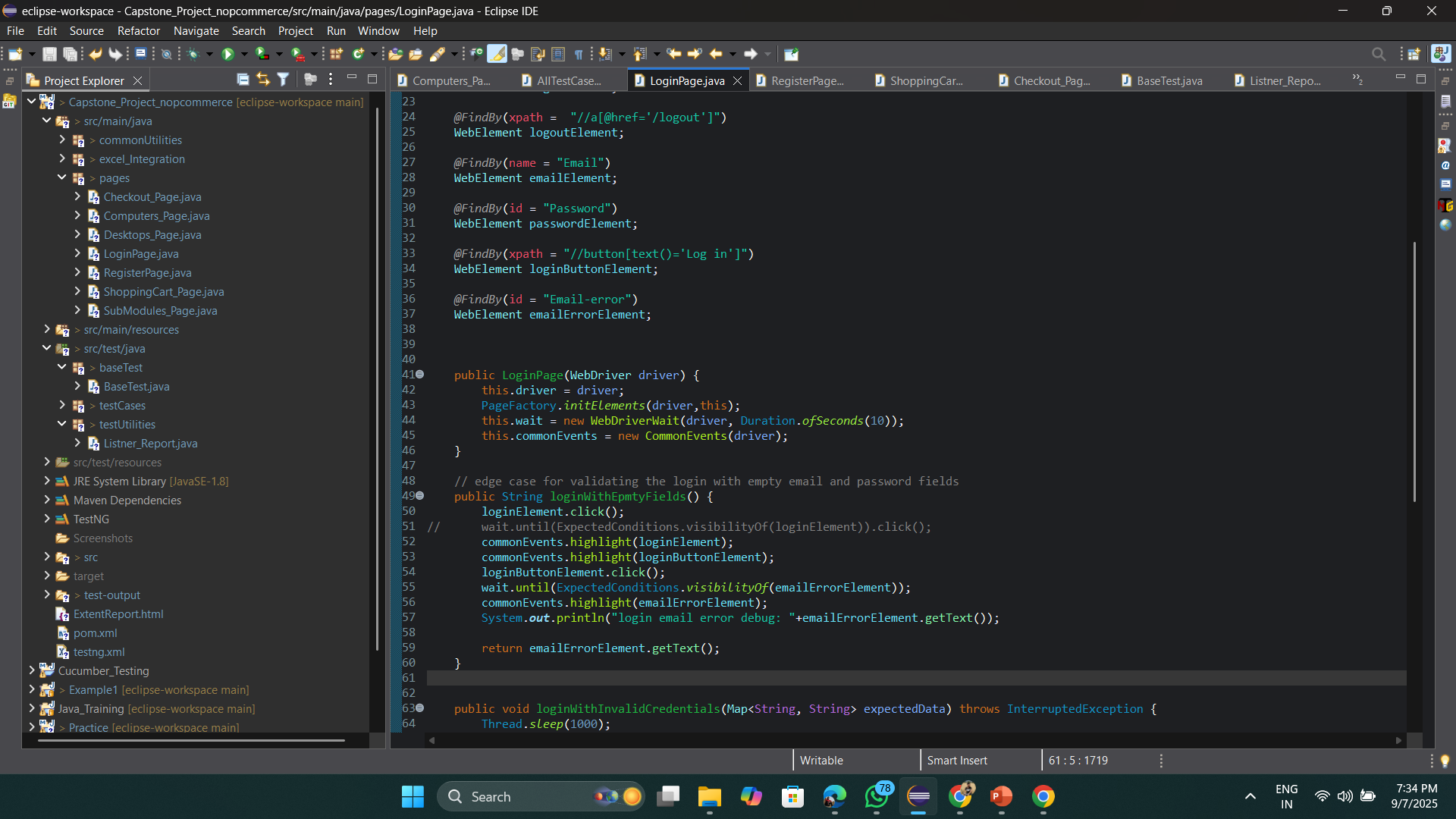
# 5. Tools & Framework Architecture

|  |  |
| --- | --- |
| Tool | Purpose |
| Selenium + TestNG/Cucumber | UI automation |
| Jira | Defect tracking |
| ExtentReports | Execution reporting |
| Eclipse | Test development |

**Framework Structure:**

**5.1 Framework Setup**

* **BaseTest**
  + Initializes WebDriver (ChromeDriver).
  + Maximizes browser window and navigates to base URL (https://demo.nopcommerce.com).
  + Provides getBaseDriver() for other classes.
  + tearDown() can close browser after execution.
* **CommonEvents**
  + Provides utility functions like highlight(WebElement) for debugging/visual confirmation.
* **Excel Integration**
  + Reads test data (Register, Login, Product, Checkout) from Excel.
  + Returns key-value pairs in Map<String, String> for data-driven testing.
* **Listener Report**
  + Implements TestNG ITestListener.
  + Generates **ExtentReports** with:
    - Test start, pass, fail, skip events.
    - Failure screenshots stored in /Screenshots.



**5.2 Test Data Management**

The project uses a centralized Excel file **nopCommerceExpectedData.xlsx** to store all input and expected data for test execution. This enables **data-driven testing** and avoids hardcoding values in test scripts.

The following sheets are maintained:

* **registerData** → Stores user details required for Registration tests.
* **validLoginCredentials** → Contains valid email and password combinations for successful login.
* **invalidLoginCredentials** → Includes invalid or incorrect credentials to test negative login scenarios.
* **productData** → Provides product name, specifications, and configuration details for Add to Cart tests.
* **checkoutData** → Holds billing address, city, zip code, phone number, and other details required for Checkout validation.

This structure ensures flexibility, scalability, and easier updates to test data without modifying the test code.

**5.3 Page Object Model (POM)**

Each web page = separate class with locators and methods.

* **RegisterPage**
  + Handles registration flow.
  + Covers **edge cases**: Empty fields, email already exists, success validation.
* **LoginPage**
  + Valid login, invalid login, and empty field validations.
* **SubModules\_Page**
  + Verifies all top-menu submodules exist.
* **Computers\_Page**
  + Verifies hover actions on submenus (Desktops, Notebooks, Software).
* **Desktops\_Page**
  + Navigates to Desktops.
  + Selects product (Build your own computer).
  + Sets configurations (RAM, HDD, Software, Quantity).
  + Adds product to cart.
* **ShoppingCart\_Page**
  + Opens shopping cart via hover.
  + Proceeds to cart page.
* **Checkout\_Page**
  + Handles checkout process:
    - Accept Terms of Service.
    - Fills Billing Address (country, state, city, etc.).
    - Selects shipping, payment, and confirms order.

**5.4 Test Cases Execution Flow**

AllTestCase.java (TestNG orchestrator)

1. **Register Test Case**
   * Loads data from registerData.
   * Validates registration + handles duplicate email edge case.
2. **Login Test Case**
   * Loads valid + invalid login data.
   * Edge case: Empty credentials.
   * Invalid login attempt.
   * Successful login.
3. **Verify SubModules**
   * Confirms all top navigation modules are visible.
4. **Verify Computers Hover**
   * Tests hover action + submenu navigation under “Computers”.
5. **Verify Desktop Add to Cart**
   * Navigates to Desktops.
   * Validates product name.
   * Selects configurations.
   * Adds to cart.
6. **Verify Shopping Cart**
   * Opens shopping cart via hover.
   * Validates item presence.
7. **Verify Checkout Process**
   * Loads checkoutData.
   * Accepts terms, fills billing address, shipping, payment.
   * Confirms order.

**5.5 Reporting**

* **ExtentReports**
  + HTML report (ExtentReport.html) generated at project root.
  + Includes test details, status, logs, screenshots on failure.
* Console logs track progress (e.g., "Verify Desktop Add To Cart Test Case Started...").

# 6. Test Environment

|  |  |
| --- | --- |
| Category | Specification |
| Hardware | i5+, 8GB RAM (16GB recommended), 256GB SSD, stable internet |
| OS | Windows 10/11, macOS Ventura |
| Browsers | Chrome |
| Data | Demo users, products |
| CI/CD | Jenkins / GitHub Actions (optional) |

# 7. Requirement Traceability Matrix (RTM)

|  |  |  |
| --- | --- | --- |
| Requirement ID | Test Case IDs | Status |
| REQ\_REG\_01 | TC\_REG\_01, TC\_REG\_02 | Automated |
| REQ\_LOGIN\_01 | TC\_LOGIN\_01, TC\_LOGIN\_02, TC\_LOGIN\_03 | Automated |
| REQ\_PRODUCT\_01 | TC\_PRODUCT\_01 | Automated |
| REQ\_CART\_01 | TC\_CART\_01 | Automated |
| REQ\_CHECKOUT\_01 | TC\_CHECKOUT\_01 to TC\_CHECKOUT\_06 | Automated |

# 8. Execution Summary

Regression suite executed successfully across Chrome

# 

# 9. Defect Management

Defects were logged in Jira. Most were UI validation issues and were resolved.

# 10. Risks & Mitigation

* Flaky tests → Robust locators
* Test data dependency → Dynamic generation
* Parallel execution → ThreadLocal driver

# 11. Benefits of Automation

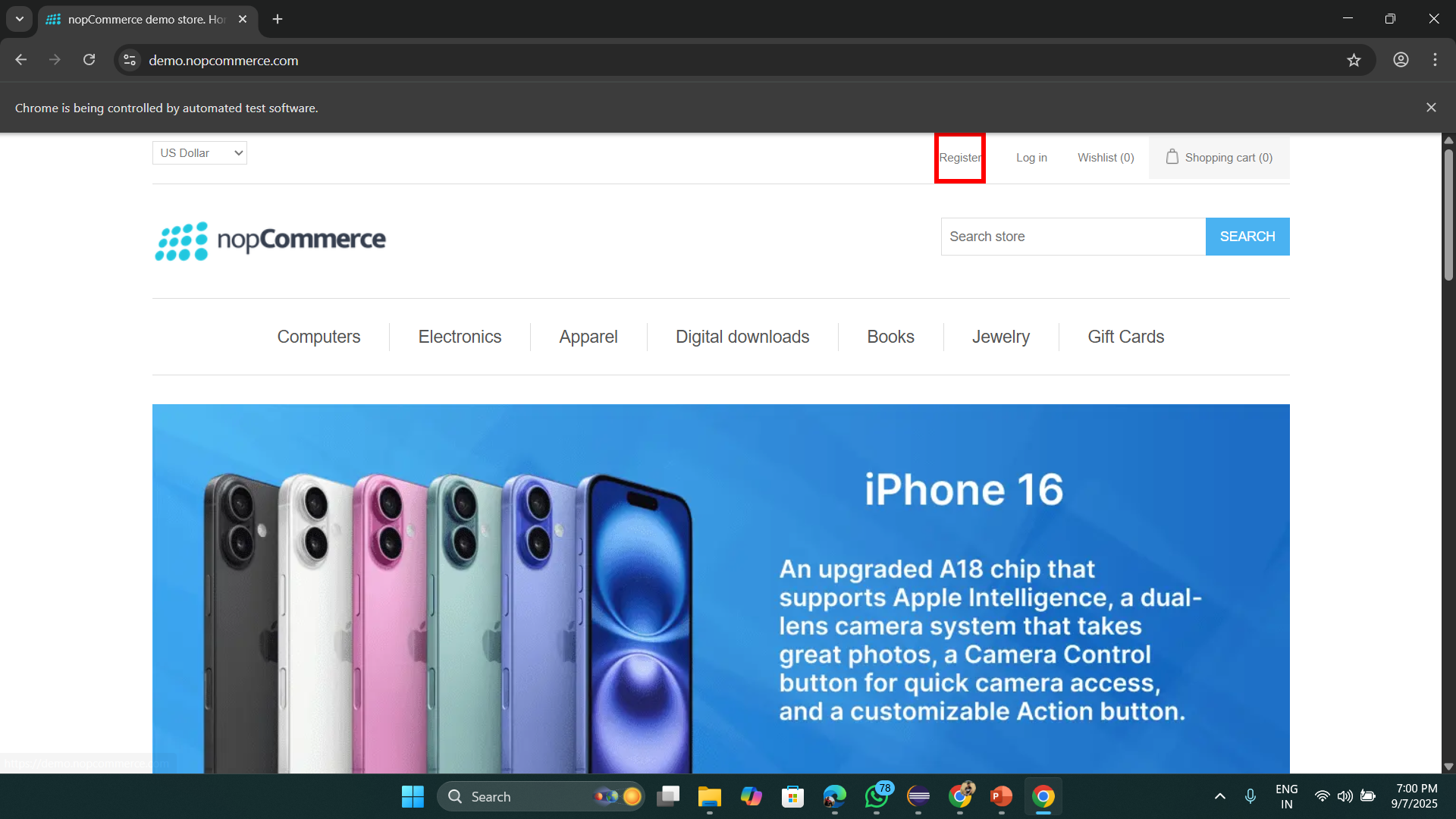
* 70% manual effort reduction
* CI/CD integration
* High regression coverage
* Scalable framework

# 12. Challenges Faced

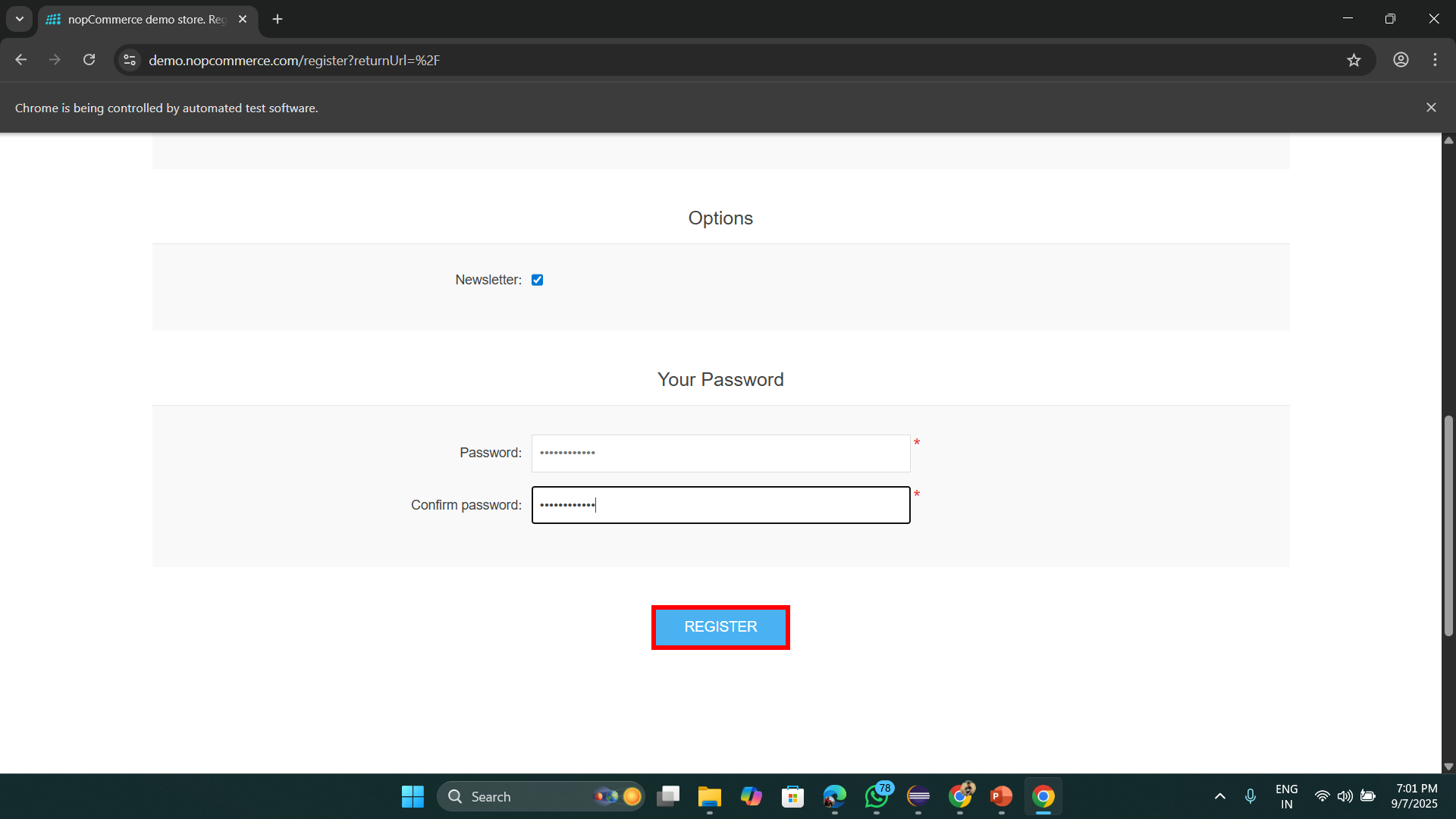
* Dynamic elements
* Checkout synchronization
* Excel-based test data integration

# 13. Appendix

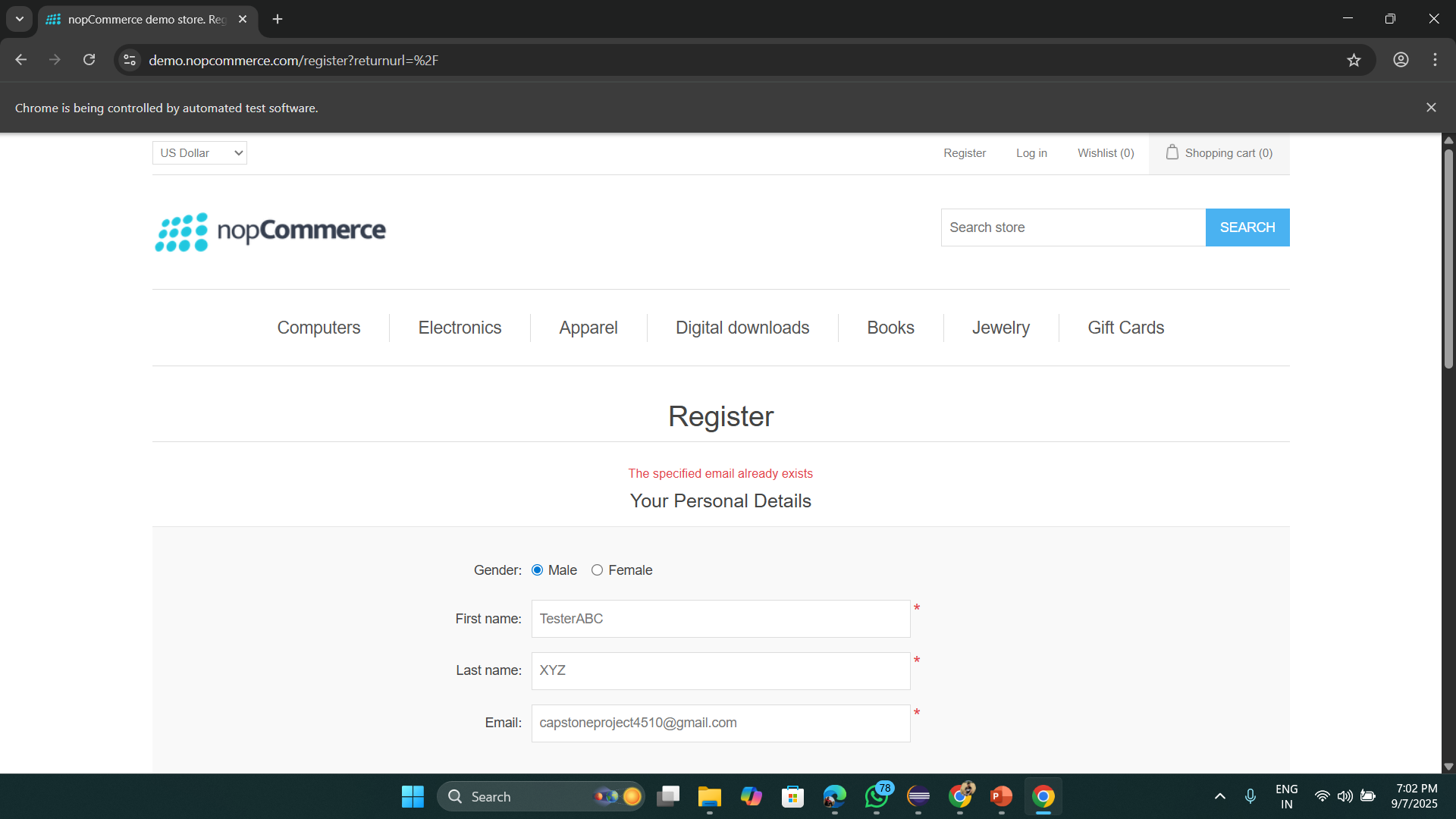
*Validating Click on Register Element*



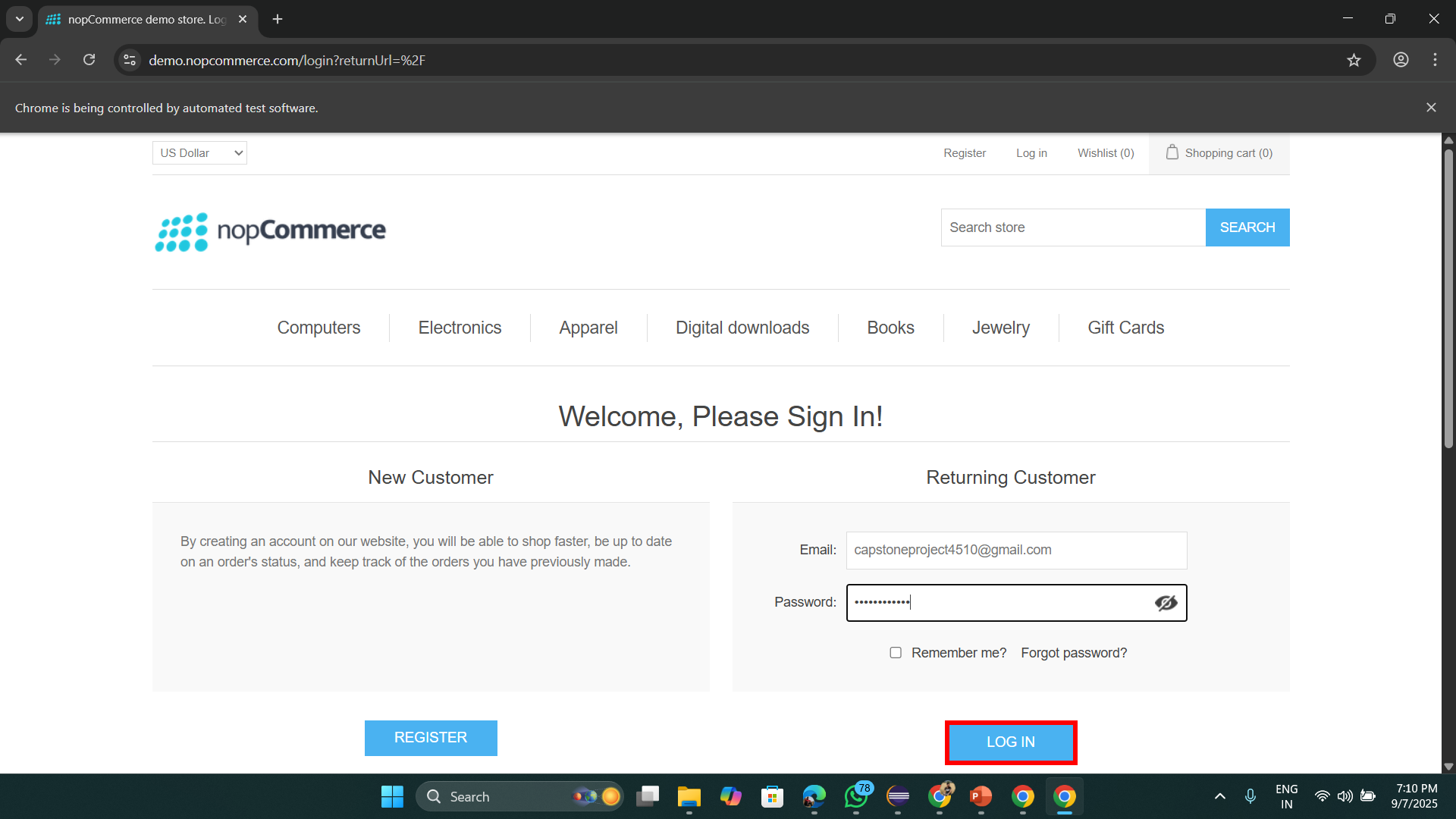
*Validating Register Functionality by entering details from excel file*



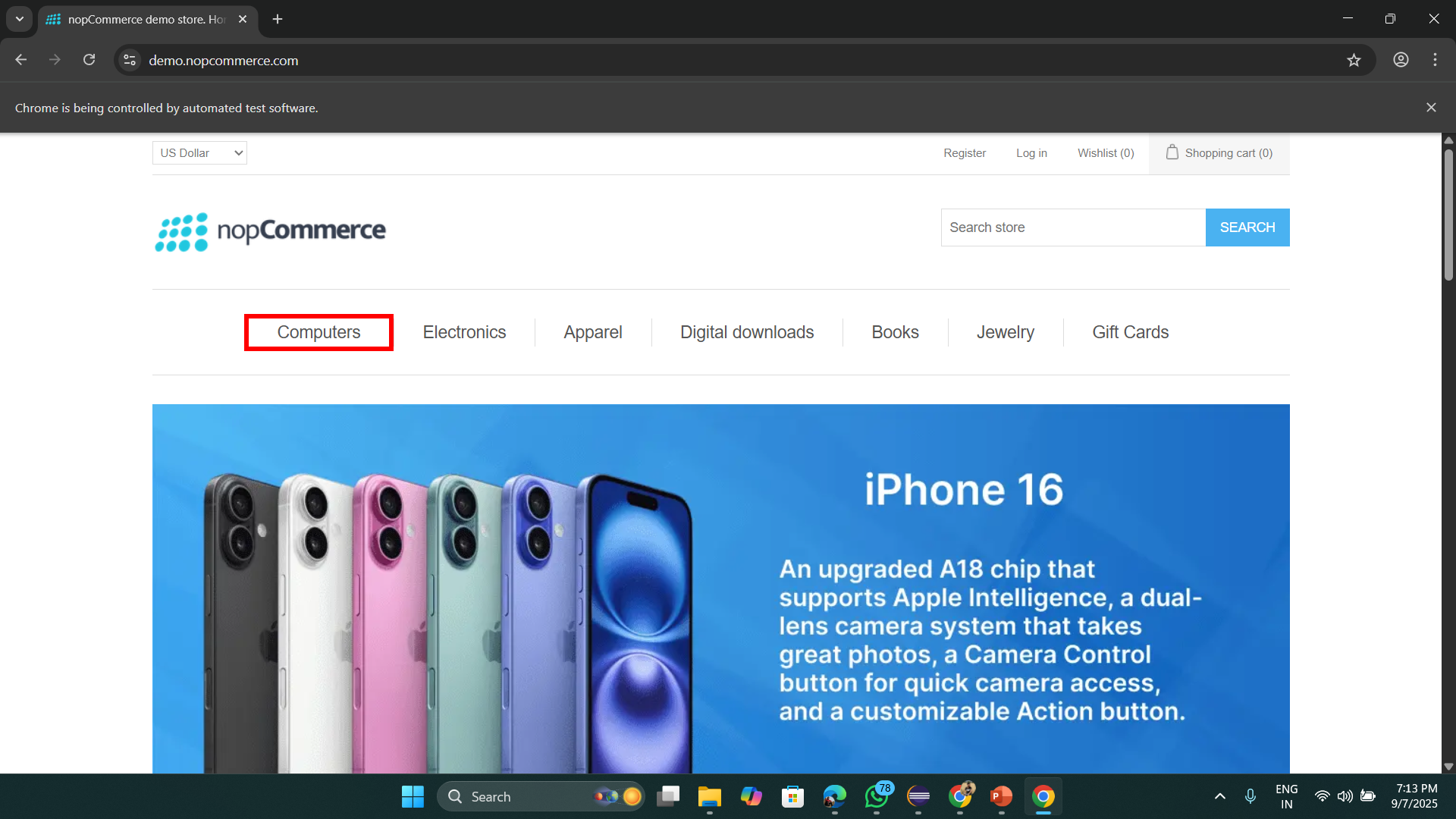
*Validating Edge Case if user email already exists*



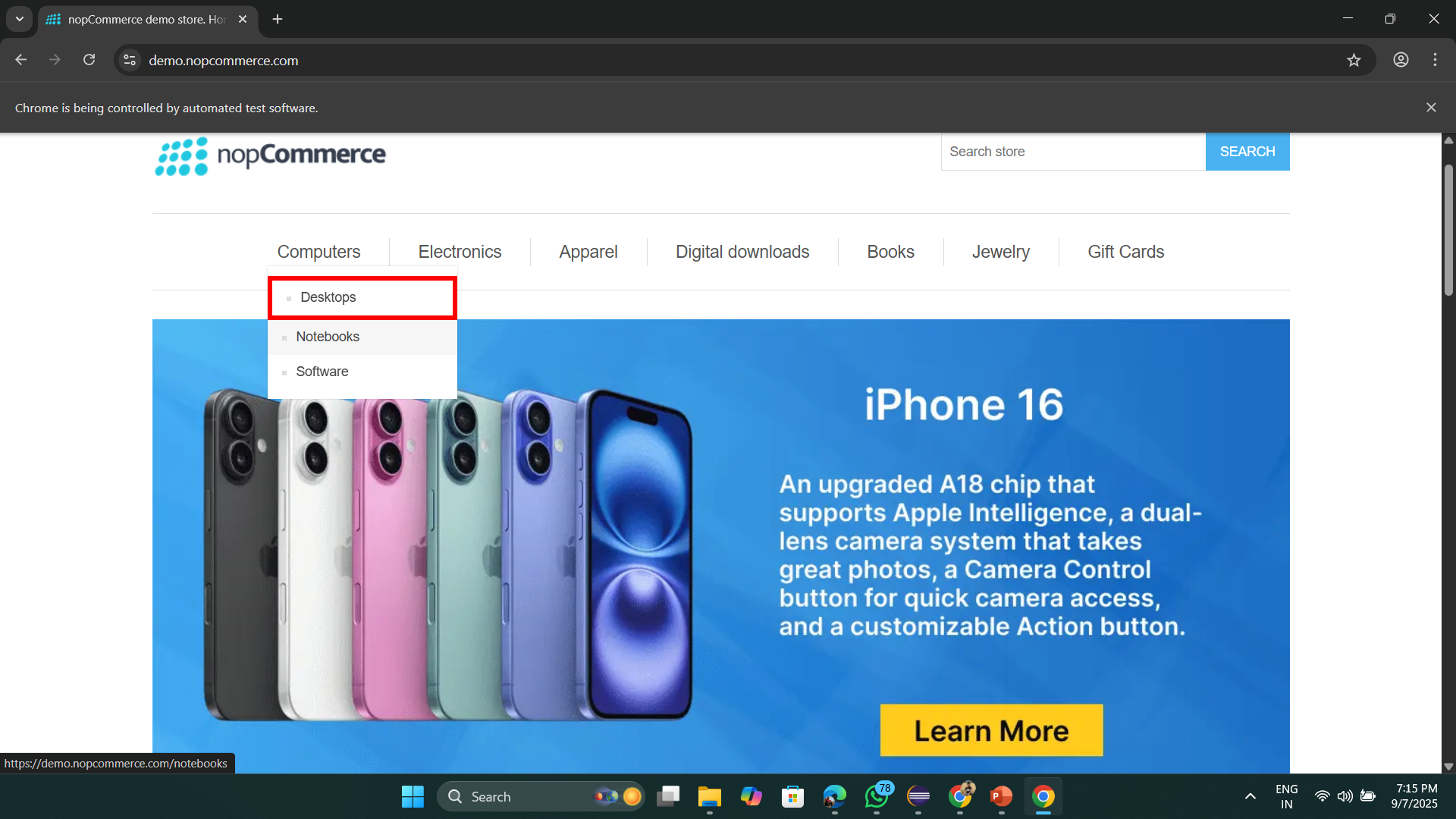
*Validating Login with Empty, Invalid and Valid Credentials from excel*



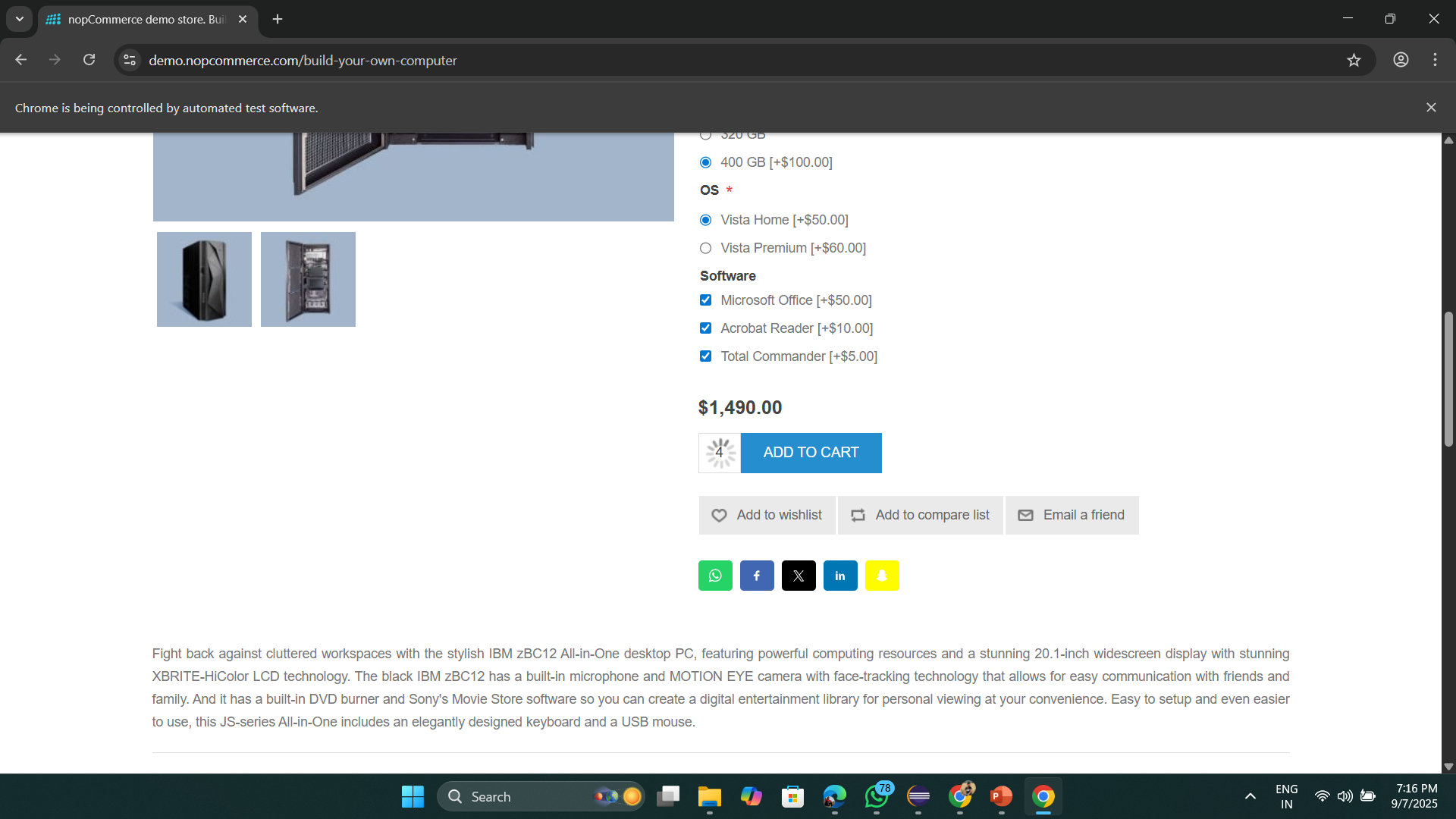
*Validating different Categories of nopCommerce website*



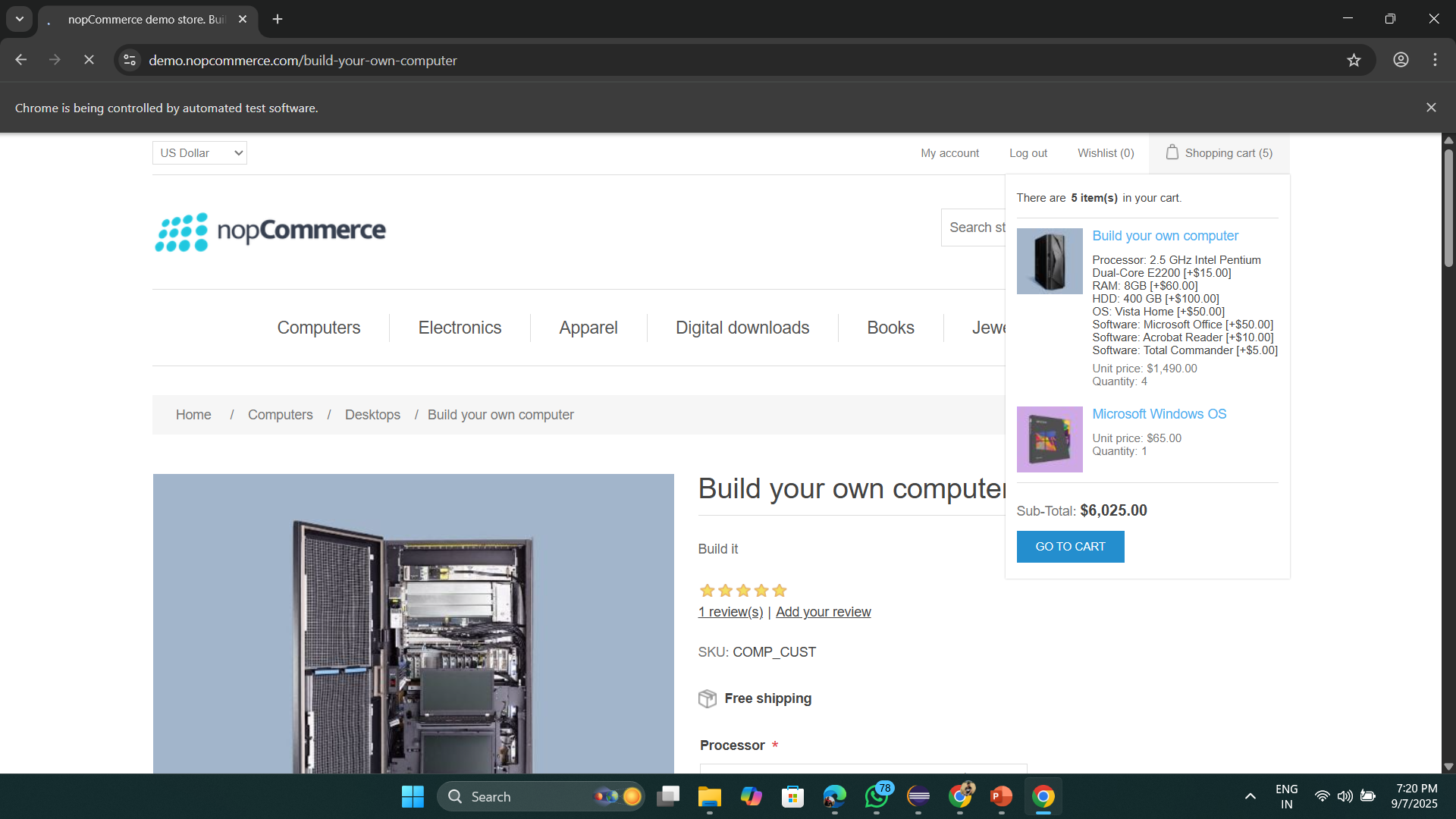
*Validating hover on Categories*



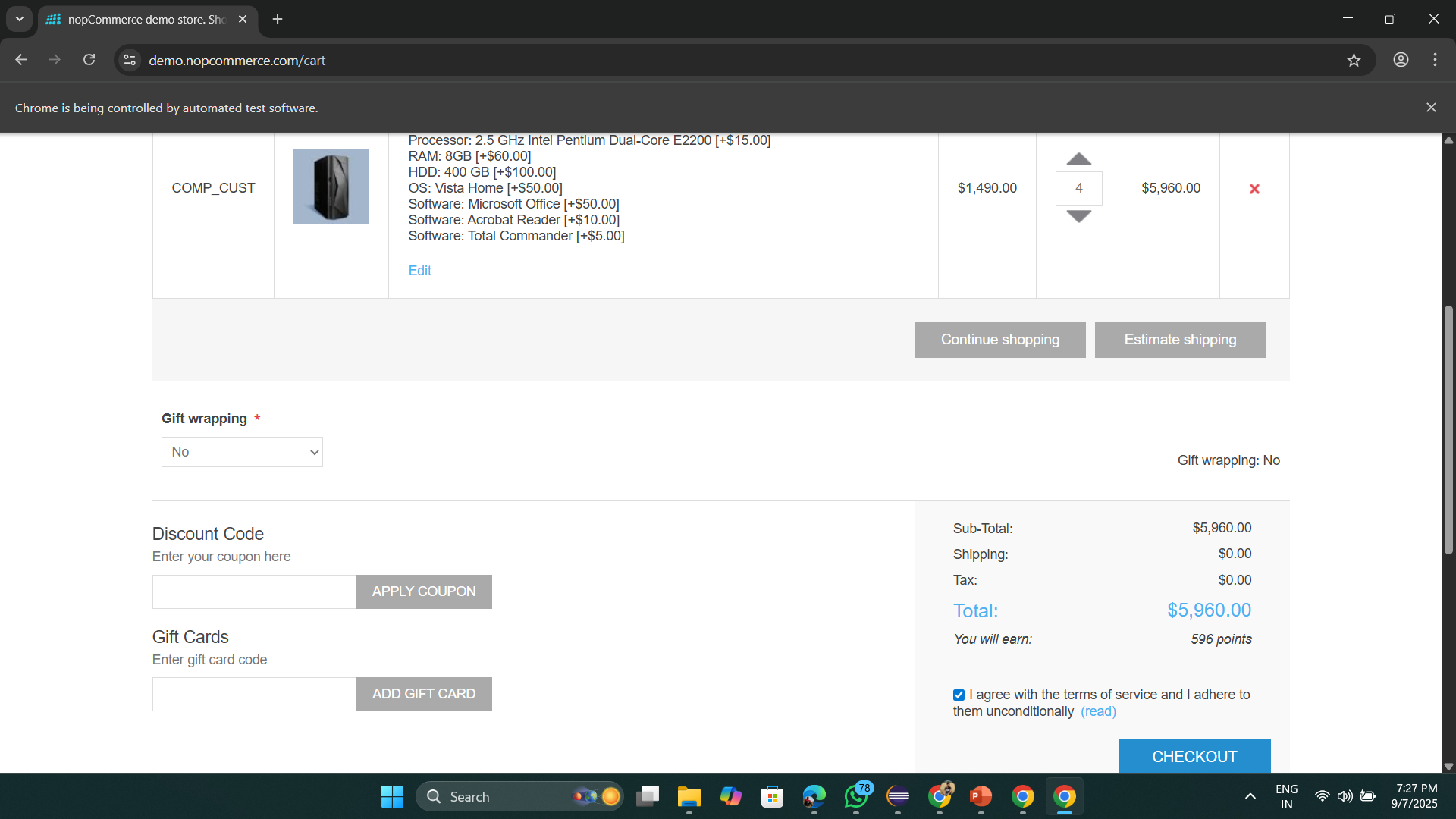
*Validating product details and adding it into cart*



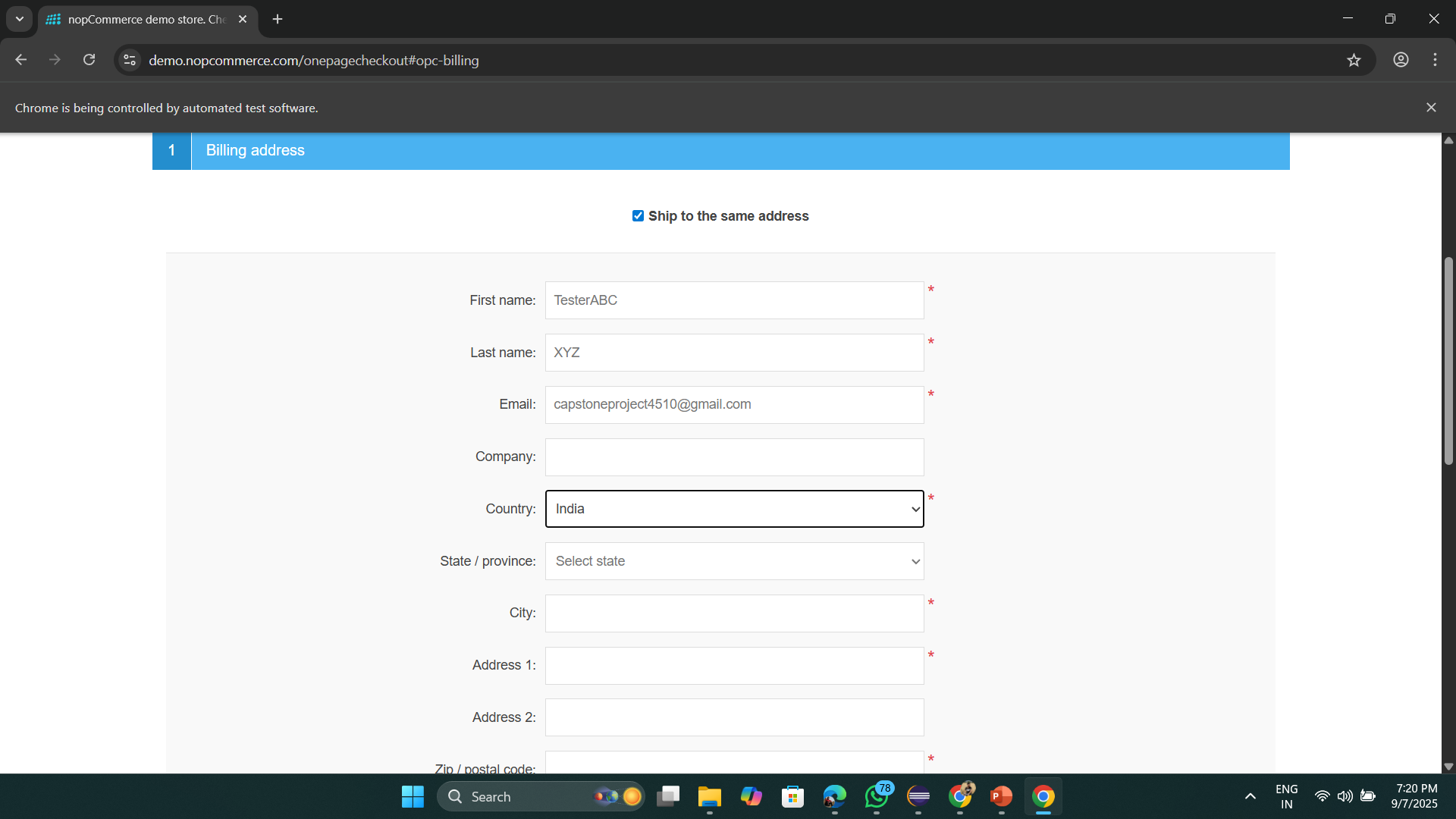
*Validating hover on Shopping Cart*



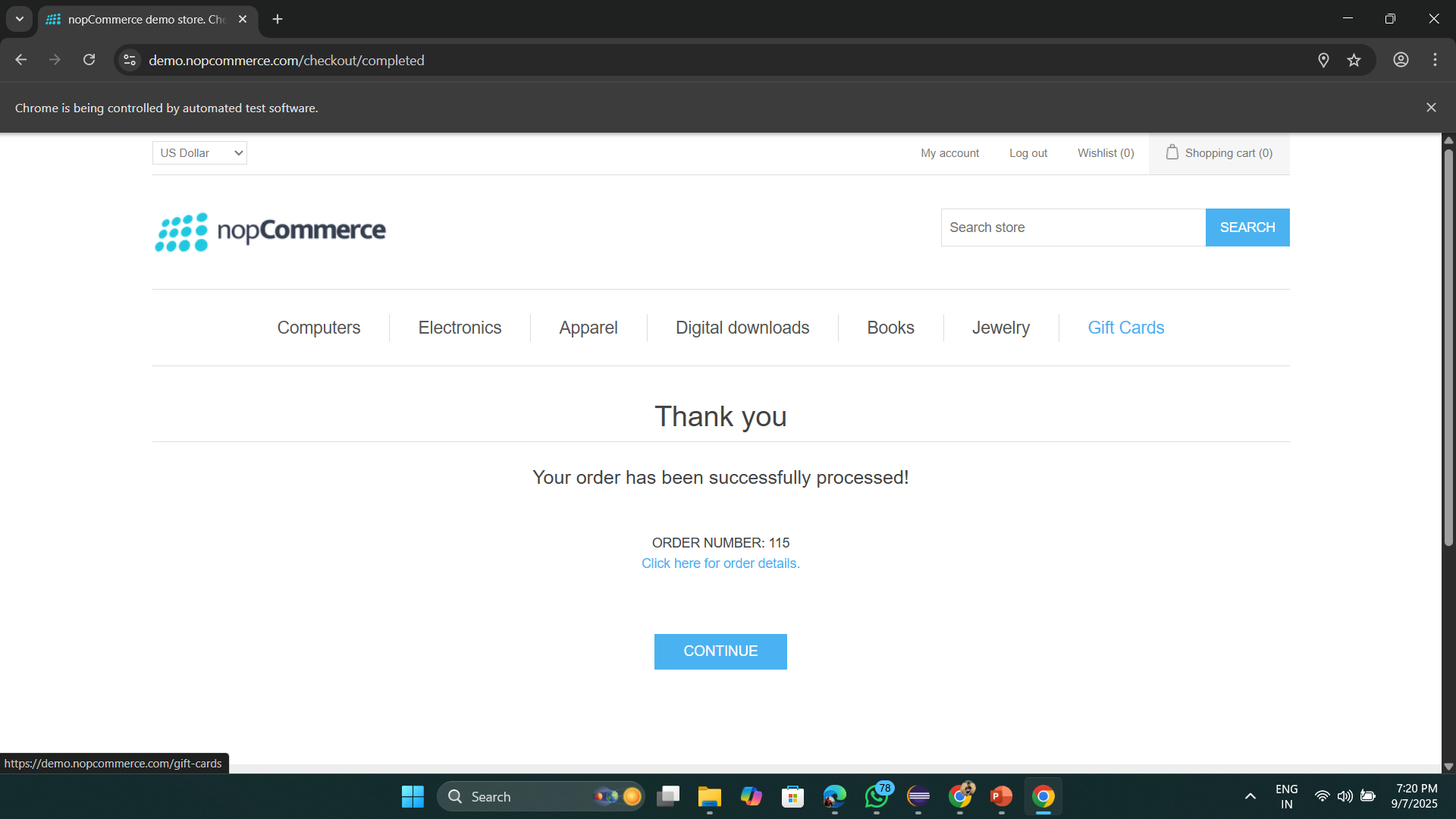
*Validating terms of services and checkout*



*Validating Billing address form by entering details from excel*



*Validating Order has been placed successfully*



*Console Log of Successful Test Cases*

