**DEMOBLAZE PROJECT DOCUMENTATION**

Project Plan: Selenium Maven Project using POM

**Project Overview:**

This project aims to create a Selenium Maven automation framework using the Page Object Model (POM) design pattern. The framework will be used to test a dummy website.

**Tools and Technologies**

* Selenium WebDriver: For browser automation
* Java: Programming language
* TestNG: Test management framework
* Maven: Build automation tool
* IDE: Eclipse

**Project: E-commerce Website Test Automation using Selenium with POM**

**Project Description**

This project aims to automate test cases for an e-commerce website using Selenium WebDriver with the Page Object Model (POM) design pattern. The project will cover functional testing of core functionalities like user registration, login, product search, adding to cart, checkout, and order confirmation. By implementing POM, the test code will be highly maintainable, reusable, and easy to understand.

Test Plan

1. Introduction

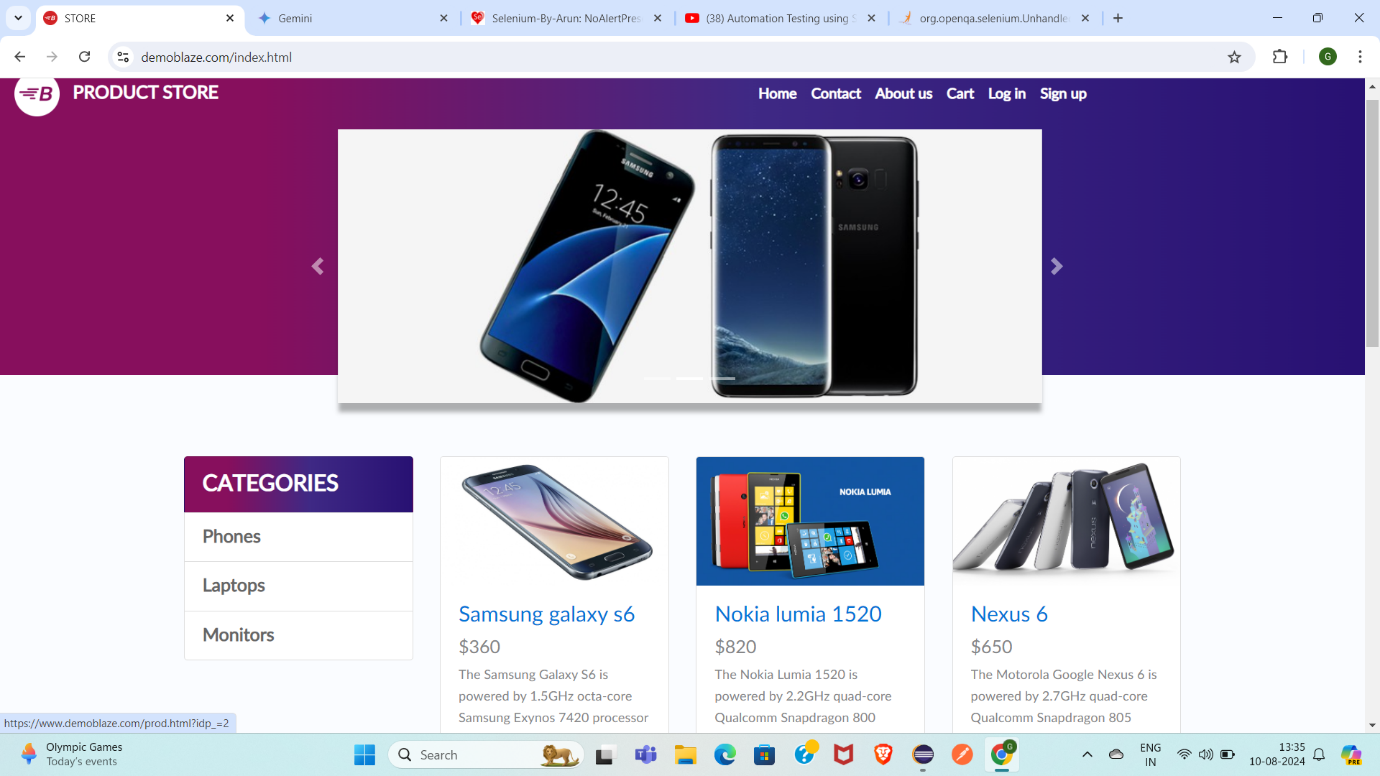
* Project Overview: Testing DemoBlaze using Selenium WebDriver with TestNG and Maven.
* Objective: To automate test cases for critical functionalities of DemoBlaze using POM.
* Scope: Define the specific features and functionalities to be covered in testing.
* Test Environment: Specify hardware, software, and network requirements.

**Test Cases**

Create detailed test cases covering various functionalities of DemoBlaze, following the template outlined in the test plan.

**Introduction**: This section would provide an overview of the test.

**Website:** <https://www.demoblaze.com/index.html>



**Focus Areas**

- Functional correctness of flows

 - UI/navigation

- Performance

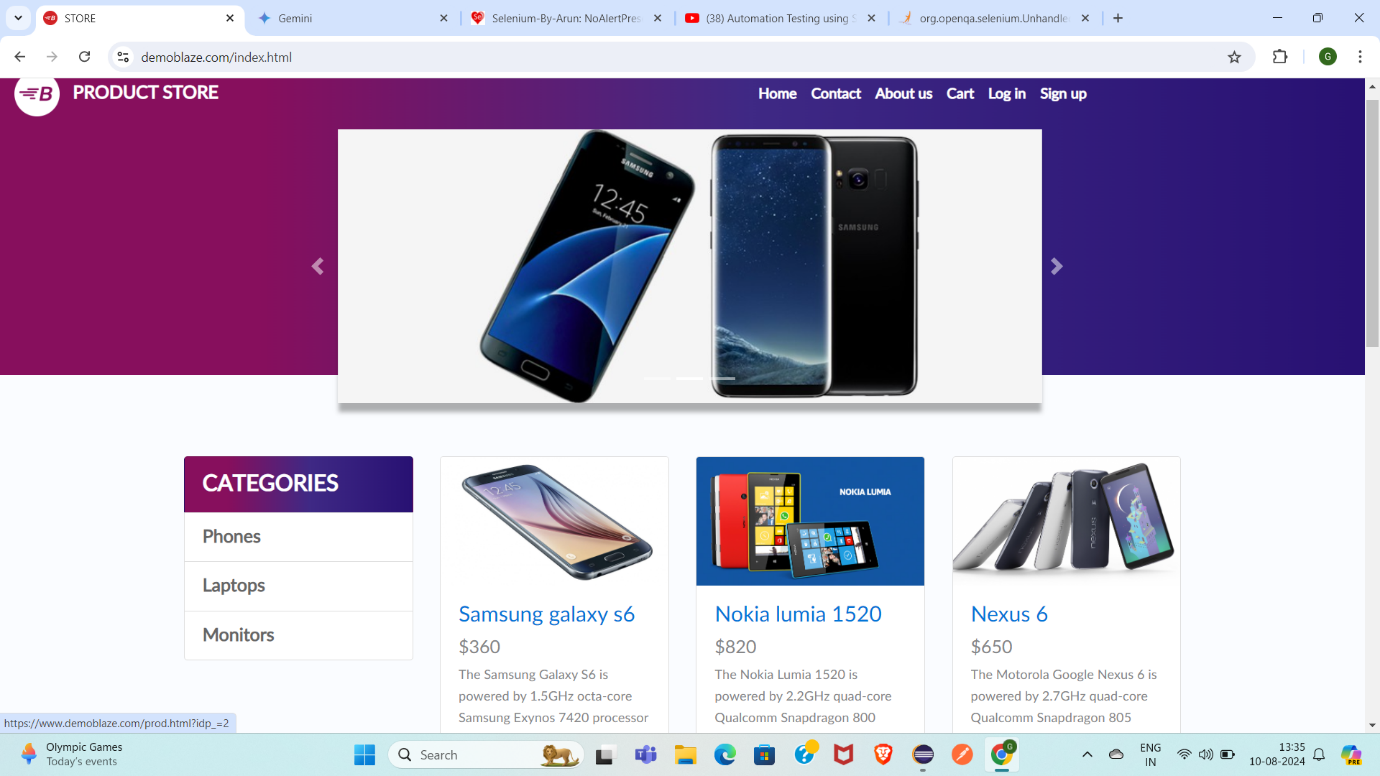
**Project Structure**

The project follows a clear directory structure for improved organization:

* **Gmaven> src/main/java** - Contains source code for the project
  + **pagepackage** - Stores all page object classes representing DemoBlaze pages.
  + **testpackage** - Holds all test classes with automation scripts.
* **resources** - Stores data files like test data or drivers (if needed)

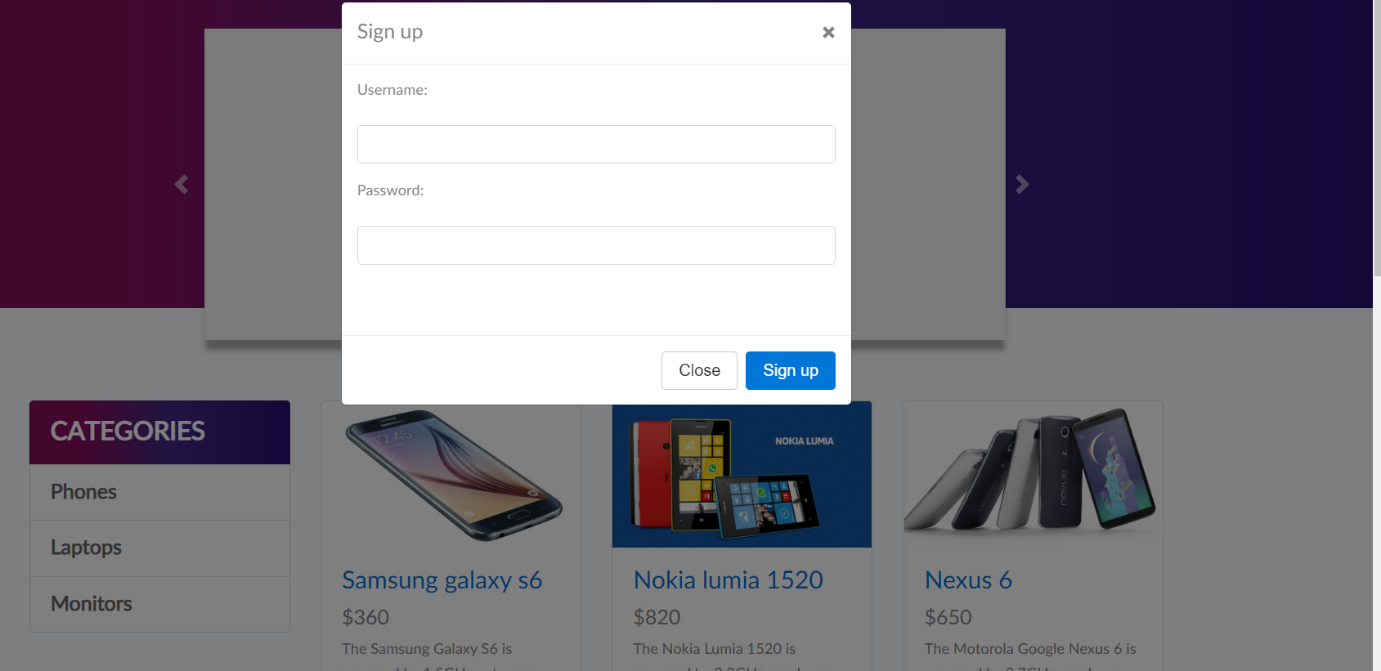
**Pagepackage**

**HomePage**: Handles elements and actions on the DemoBlaze homepage, including navigation links to product categories**.**

****

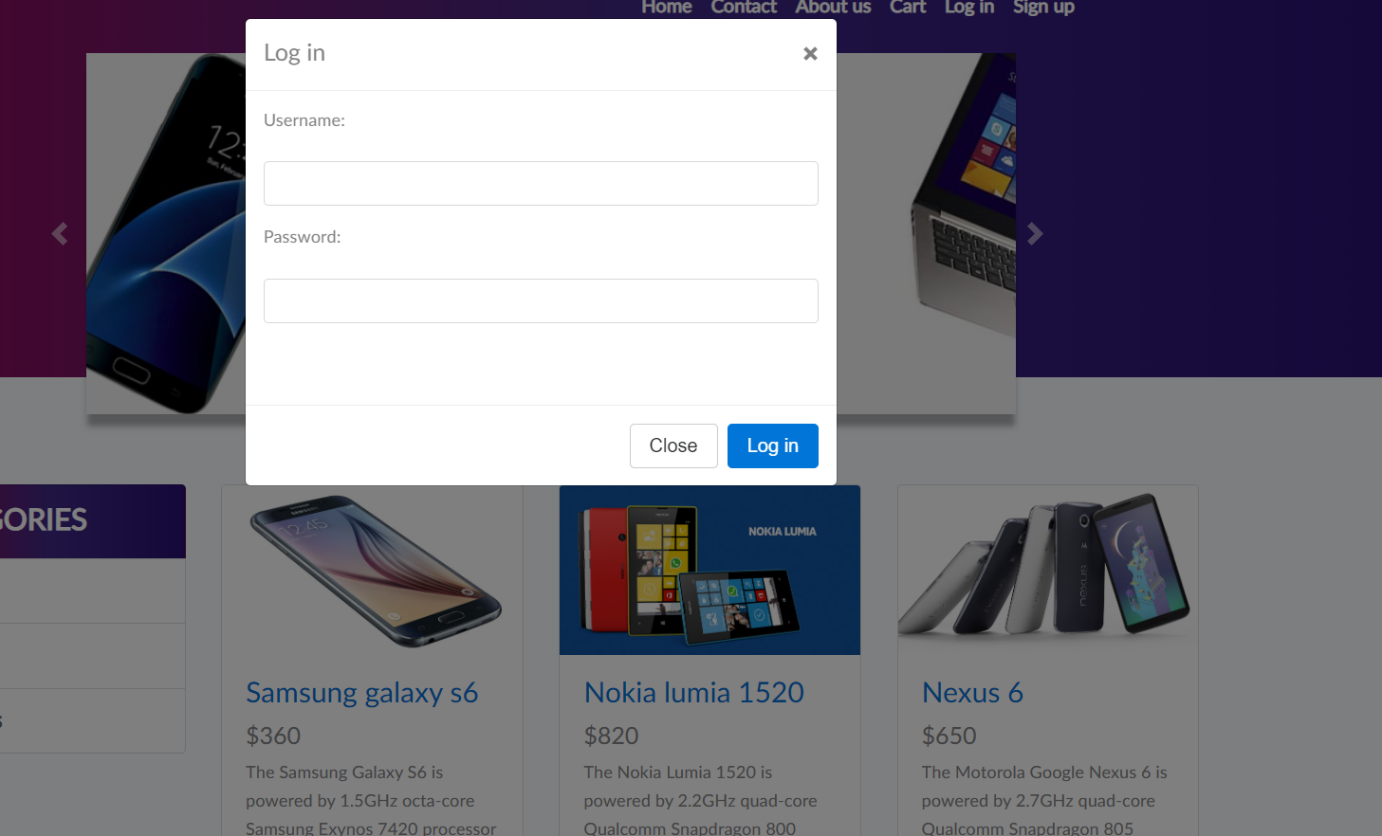
**SignupPage**

Provides method for interacting with the registration form, including entering user details and submission.



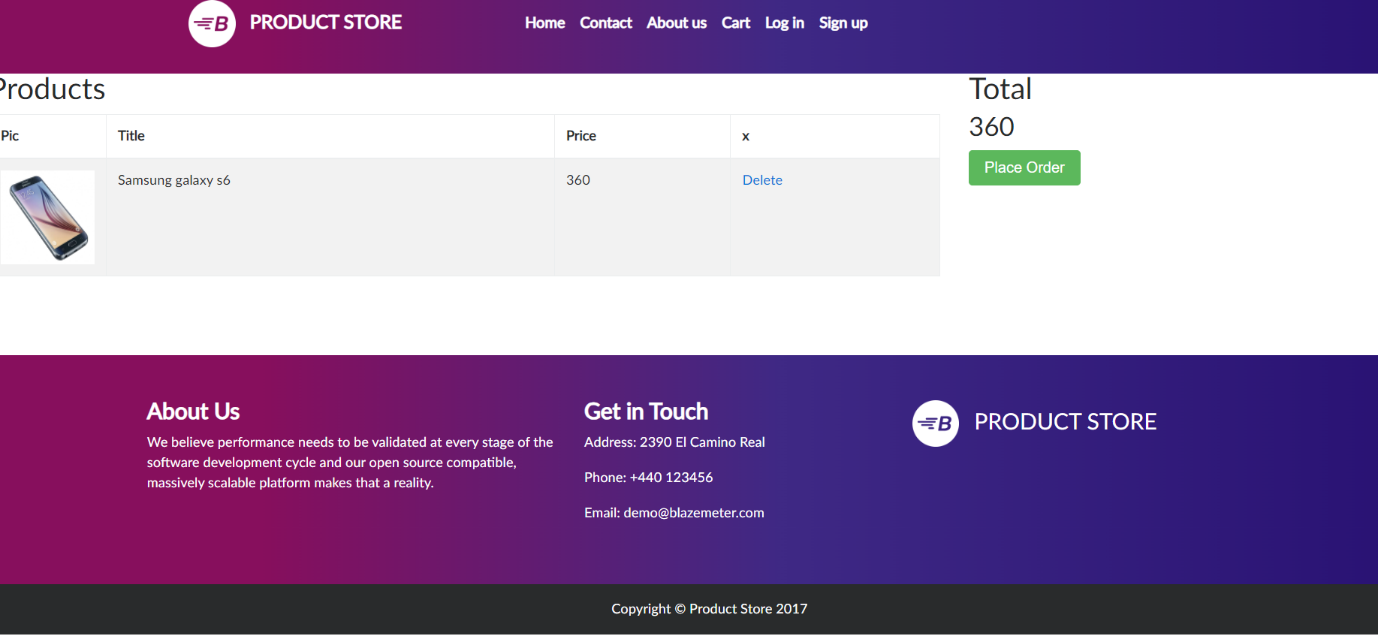
**LoginPage**

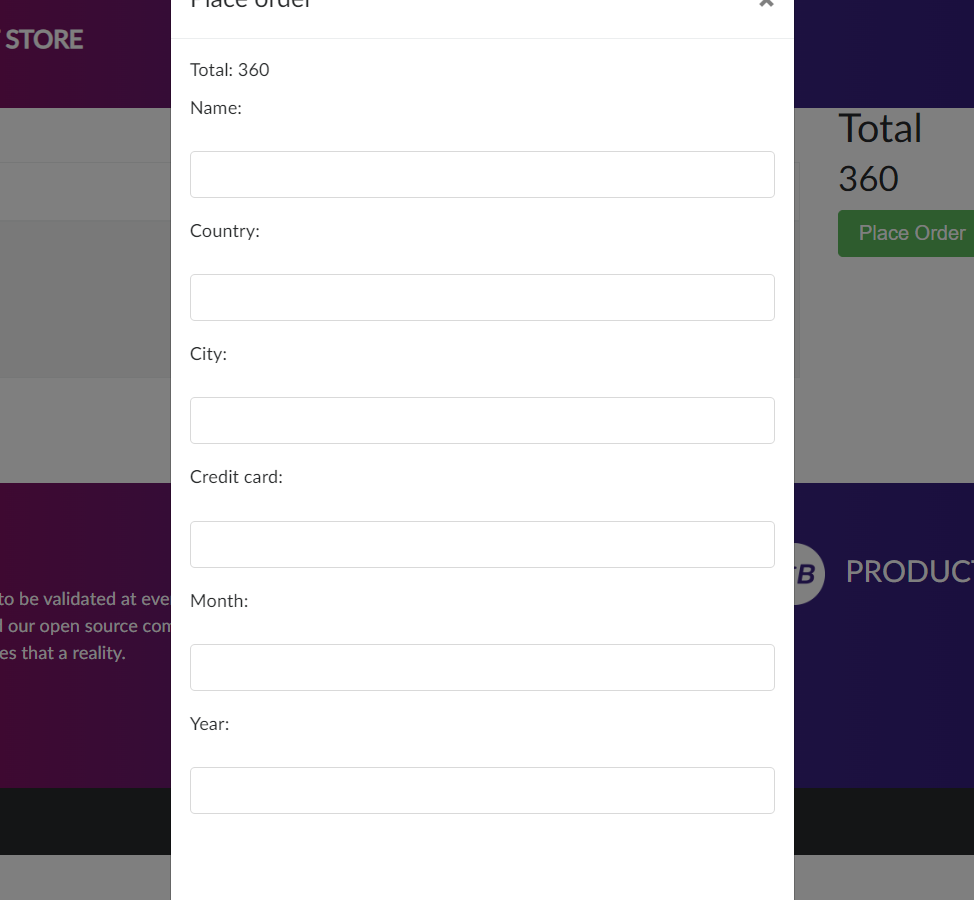
Interacts with the login page elements like username, password, and login button.

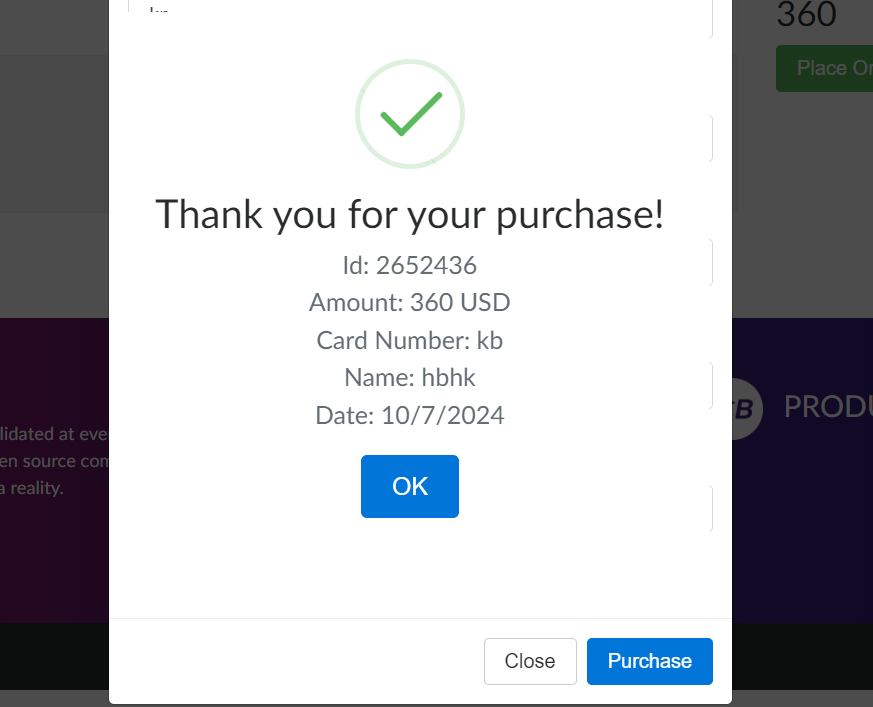


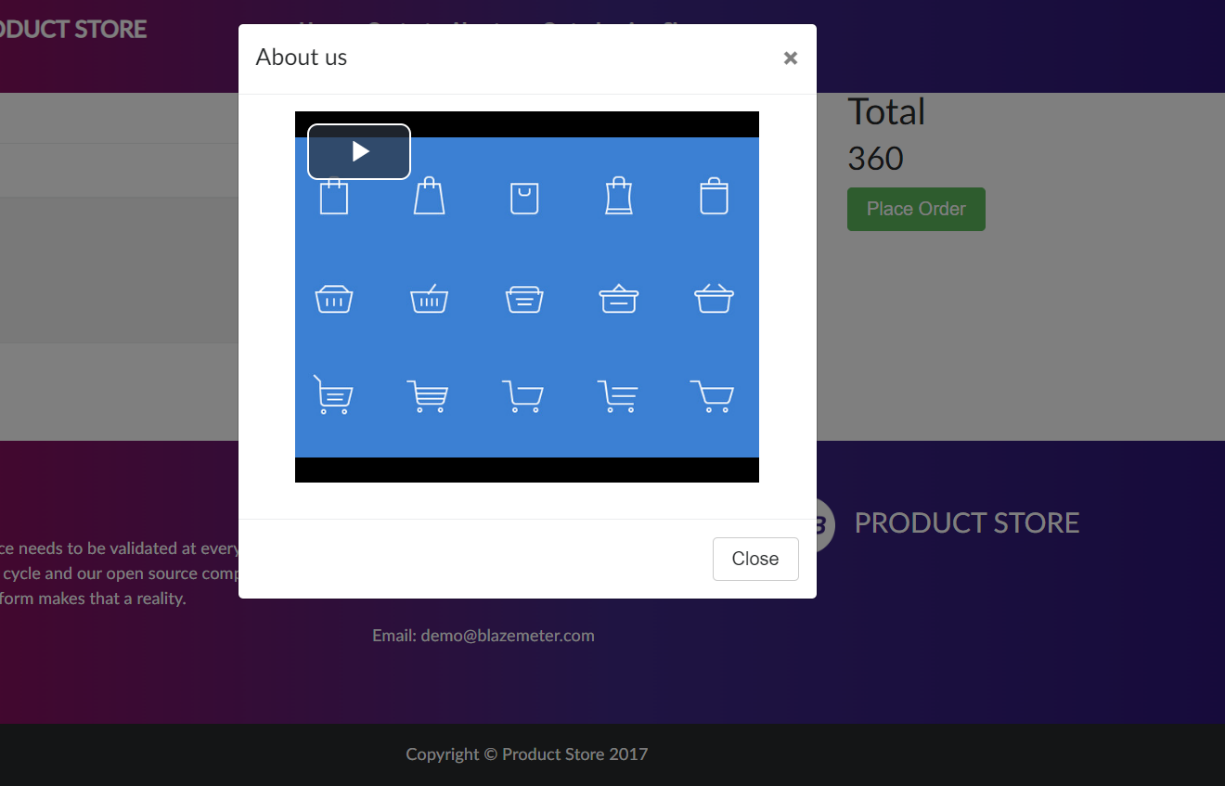
**CartPage**

Represents the shopping cart page, allowing interaction with added items and checkout procedures.

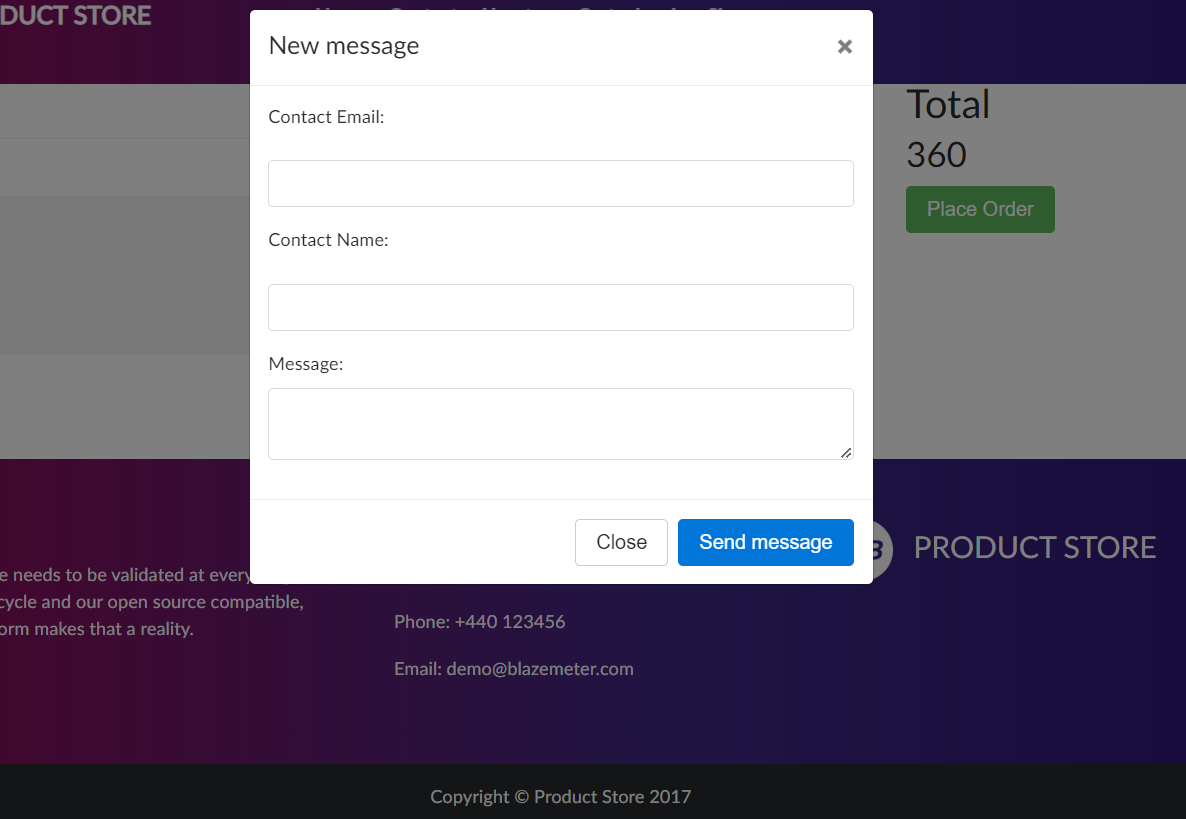






**About us page:** Represents About us page. 

**Contact page:** Represents contact us page



**Challenges**

**Dynamic IDs**: Certain elements on DemoBlaze have dynamically generated IDs, making them difficult to locate directly. This was overcome by using alternative locators like XPath or CSS selectors based on unique attributes**.**

Different **Exceptions** was Encountered during the test run**.**

**Overcoming Challenges**

**Locators Strategy:** Employing a combination of ID, name, class name, and XPath locators provided flexibility for identifying elements.

**Exception Handling:** Wrapping actions prone to JavaScript errors within try-catch blocks with appropriate error handling mechanisms ensured test execution wouldn't break.

**Conclusion**

This documentation provides a high-level overview of the automated testing process for DemoBlaze. By leveraging a layered structure and addressing common challenges, a robust and maintainable test suite was established. The benefits outlined demonstrate the value of automation in ensuring the website's quality and functionality.