V&V Lab Final

5분마다 자동으로 업데이트됨

Question 1: Selenium Web Automation

Site: www.demoblaze.com

Write a complete Selenium (Java) test script to automate the following workflow on the DemoBlaze e-commerce website.

Tasks to Automate

1. User Registration
   a. Click on the Sign up link.
   b. Register with a unique username and any password.
   c. Handle the signup success alert.

   [5]

2. Login
   a. Log in with the same credentials used in the signup process.
   b. Wait for and confirm successful login (e.g., by checking the presence of "Welcome [username]").

   [5]

   [15]

3. Add Products to Cart
   a. Add the following products to the cart:
   b. 3 laptops of the same product (e.g., "Sony vaio i5").
   c. 2 monitors of different products (e.g., "ASUS Full HD", "Apple monitor 24").
   d. 3 phones of different products (e.g., "Samsung galaxy s6", "Nokia lumia 1520", "HTC One M9").
   e. After each add, handle the "Product added" alert.
   f. Return to home after adding each item.

   [15]

4. Cart and Checkout
   a. Go to the Cart page.
   b. Verify that the selected 8 products are listed.
   c. Click on Place Order.
   d. Fill in all required fields in the order form with dummy data.
   e. Click Purchase.
   f. Save the confirmation message details and print it to the console.

   [10]

5. Send a Contact Message

a. Click on the Contact tab.
b. Fill in the contact form with a name, email, and a short message.
c. Submit the message and handle the alert.

Question 2: JUnit

Project Theme: Library Management System

Scenario

You're managing a small library system. It includes basic operations like adding books, registering members, borrowing/returning books, and checking availability.

Java Project Files

Here are the 5 Java files:

1.

```
public class Book {
    private String title;
    private String author;
    private boolean isBorrowed;

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.isBorrowed = false;
    }

    public String getTitle() { return title; }

    public boolean isBorrowed() { return isBorrowed; }

    public void borrow() {
        if (!isBorrowed) {
            isBorrowed = true;
        } else {
                throw new IllegalStateException("Book
already borrowed");
        }
    }

    public void returnBook() {
```

```
        if (isBorrowed) {
            isBorrowed = false;
        } else {
             throw new IllegalStateException("Book was
not borrowed");
        }
    }
}
```

2.

```
public class Member {
    private String name;
    private int borrowedBooks;

    public Member(String name) {
        this.name = name;
        this.borrowedBooks = 0;
    }

    public String getName() { return name; }

    public int getBorrowedBooks() { return
borrowedBooks; }

    public void borrowBook() {
        if (borrowedBooks >= 3) {
            throw new IllegalStateException("Cannot
borrow more than 3 books");
        }
        borrowedBooks++;
    }

    public void returnBook() {
        if (borrowedBooks == 0) {
            throw new IllegalStateException("No books
to return");
        }
        borrowedBooks--;
    }
}
```

3.

```
import java.util.*;

public class Library {
    private List<Book> books;
    private List<Member> members;

    public Library() {
        books = new ArrayList<>();
        members = new ArrayList<>();
    }

    public void addBook(Book book) { books.add(book);
}

    public void registerMember(Member member) {
members.add(member); }

    public Book findBook(String title) {
        for (Book book : books) {
            if
(book.getTitle().equalsIgnoreCase(title)) return book;
        }
        return null;
    }

    public boolean isBookAvailable(String title) {
        Book book = findBook(title);
        return book != null && !book.isBorrowed();
    }
}
```

4.

```
public class LibraryUtils {
    public static int countAvailableBooks(Library
library) {
        int count = 0;
        for (Book book : library.books) {
            if (!book.isBorrowed()) {
                count++;
            }
        }
        return count;
    }
}
```

5.

```
public class Main {
    public static void main(String[] args) {
        Library library = new Library();
        Member alice = new Member("Alice");
        Book book = new Book("1984", "Orwell");

        library.registerMember(alice);
        library.addBook(book);

        System.out.println("Is '1984' available? " +
```

```
library.isBookAvailable("1984"));
    }
}
```

Task: [11X5 = 55 (5 Extra)]

1. **Borrowing a Book**
   Test that calling the borrow() method on a book changes its state to "borrowed." Ensure the isBorrowed() method returns true after borrowing.

2. **Borrowing an Already Borrowed Book**
   Test that if you try to borrow a book that is already borrowed, an IllegalStateException is thrown.

3. **Returning a Book**
   Test that calling the returnBook() method on a borrowed book changes its state to "not borrowed." After returning, isBorrowed() should return false.

4. **Returning a Book That Was Not Borrowed**
   Test that calling returnBook() on a book that hasn't been borrowed throws an IllegalStateException.

5. **Member Borrowing Limit**
   Test that a Member can successfully borrow books up to the limit of 3. The getBorrowedBooks() method should correctly track the number of borrowed books.

6. **Exceeding Borrow Limit**
   Test that if a member tries to borrow a fourth book, an IllegalStateException is thrown indicating the limit has been reached.

7. **Returning a Book by Member**
   Test that a member who has borrowed books can return a book, reducing the count of borrowed books by one.

8. **Returning a Book When Member Has None Borrowed**
   Test that if a member with zero borrowed books tries to return a book, an IllegalStateException is thrown.

9. **Finding a Book in the Library**
   Test that the findBook(String title) method returns the correct Book object when the title matches (case-insensitive), and

returns null if no book is found.

10. **Checking Book Availability in Library**
    Test that isBookAvailable(String title) returns true if the book exists and is not borrowed, and returns false if the book does not exist or is currently borrowed.

11. **Counting Available Books Using LibraryUtils**
    Test that the method countAvailableBooks(Library library) accurately counts the number of books that are not currently borrowed.