Test Scenario ID	Test Scenario	Description
BA_01	Valid login with correct credentials	Automate login flow using valid email/password and verify successful login
BA_02	InValid login with correct credentials	Automate login attempt with incorrect email/password and validate error message
BA_03	Login with empty fields	Automate login with blank email/password fields and verify validation messages
BA_04	Create new book	Automate navigation to “Create New book” selecting a theme, entering title/author and publishing the book

BriBooks Automation Assignment

Platform Under Test: https://www.bribooks.com

Test Automation Framework Setup

Recommended Tools:

- **Selenium WebDriver** with Java/Python
- **TestNG/JUnit** for test organization
- **Page Object Model** design pattern
- **Maven/Gradle** for dependency management
- **ExtentReports** for test reporting

Test Scenarios to Automate

1. Login Automation Test Cases

Valid Login Test:`java`

```
@Test(priority = 1)
public void testValidLogin() {

// Navigate to BriBooks login page

driver.get("https://www.bribooks.com/login");


// Enter valid credentials
loginPage.enterEmail("valid@email.com");

loginPage.enterPassword("validPassword");

loginPage.clickLoginButton();


// Verify successful login

Assert.assertTrue(dashboardPage.isUserLoggedIn());
}
```

Invalid Login Test:

java

```
@Test(priority = 2)
public void testInvalidLogin() {

// Navigate to login page

driver.get("https://www.bribooks.com/login");


// Enter invalid credentials

loginPage.enterEmail("invalid@email.com");

loginPage.enterPassword("wrongPassword");

loginPage.clickLoginButton();


// Verify error message

Assert.assertTrue(loginPage.getErrorMessage().contains("Invalid credentials"));
}
```

Empty Fields Login Test:

java

```
@Test(priority = 3)
public void testEmptyFieldsLogin() {

// Navigate to login page
driver.get("https://www.bribooks.com/login");


// Click login without entering credentials

loginPage.clickLoginButton();
// Verify validation messages

Assert.assertTrue(loginPage.isEmailValidationDisplayed());

Assert.assertTrue(loginPage.isPasswordValidationDisplayed());
}
```

2. Book Publishing Automation

Complete Book Creation Flow:

```java
@Test(priority = 4, dependsOnMethods = "testValidLogin")
public void testCompleteBookPublishing() {

// Navigate to create new book section

dashboardPage.clickCreateNewBook();


// Select theme

bookCreationPage.selectTheme("Modern");


// Fill mandatory fields

bookCreationPage.enterBookTitle("Automated Test Book");

bookCreationPage.enterAuthorName("Test Author");

bookCreationPage.enterDescription("This is a test book created through
automation");

bookCreationPage.selectGenre("Fiction");


// Add content (if required)

bookCreationPage.addChapter("Chapter 1", "This is the first chapter
content");


// Publish the book

bookCreationPage.clickPublishButton();


// Verify successful publication

Assert.assertTrue(confirmationPage.isBookPublished());
Assert.assertTrue(confirmationPage.getSuccessMessage().contains("successfully
published"));
}
```

Page Object Model Implementation

Login Page Object:

java

```
public class LoginPage {

private WebDriver driver;


@FindBy(id       = "email")

private WebElement emailField;


@FindBy(id       = "password")

private WebElement passwordField;


@FindBy(xpath = "//button[text()='Login']")

private WebElement loginButton;


@FindBy(className     = "error-message")

private WebElement errorMessage;


public void enterEmail(String email) {

emailField.clear();

emailField.sendKeys(email);

}


public void enterPassword(String password) {

passwordField.clear();

passwordField.sendKeys(password);

}


public void clickLoginButton() {
```

```
loginButton.click();

}


public String getErrorMessage() {

return errorMessage.getText();
}
}
```

Book Creation Page Object:

java

```
public class BookCreationPage {

private WebDriver driver;


@FindBy(id    = "book-title")

private WebElement bookTitleField;


@FindBy(id    = "author-name")

private WebElement authorNameField;


@FindBy(xpath = "//select[@name='theme']")

private WebElement themeDropdown;


@FindBy(id    = "publish-btn")

private WebElement publishButton;


public void enterBookTitle(String title) {

bookTitleField.clear();

bookTitleField.sendKeys(title);

}


public void enterAuthorName(String author) {

authorNameField.clear();
authorNameField.sendKeys(author);

}


public void selectTheme(String theme) {

Select dropdown           = new Select(themeDropdown);

dropdown.selectByVisibleText(theme);

}


public void clickPublishButton() {
publishButton.click();

}
}
```

Test Data Management

Test Data Properties File:

```
properties
#       Valid    Test     Data
valid.email=test@bribooks.com
valid.password=TestPassword123
#       Invalid  Test     Data
invalid.email=invalid@test.com
invalid.password=wrongpass
#       Book     Data
book.title=Automation     Test     Book
book.author=Test          Author
book.genre=Fiction
book.theme=Modern
```

Test Execution Strategy

1. **Test Environment Setup:**

- Configure WebDriver for multiple browsers
- Set up test data and configuration files
- Implement logging and reporting mechanisms

2. **Test Execution Order:**

- Login tests (positive and negative scenarios)
- Book creation and publishing workflow
- Logout and cleanup

3. **Error Handling:**

- Implement try-catch blocks for robust error handling
- Take screenshots on test failures
- Implement retry mechanisms for flaky tests

4. **Reporting and Analytics:**

- Generate detailed test reports with ExtentReports
- Include screenshots and test execution videos
- Track test execution metrics and success rates