Assignment Task 2: Web Automation(required form fields)

For this task, I have chosen the practice form at **“https://demoqa.com/automation-practice-form”**. This form includes multiple required fields (First Name, Last Name, Gender, Mobile Number), making it an ideal candidate for demonstrating validation automation.

Approach:

Two-Step Validation
My strategy incorporates both **browser-side and programmatic validation**. A proactive input sanitization and security analysis is performed on the test data pre-submission then I leverage HTML5 validation by attempting form submission with invalid inputs, then capturing the browser's native validationMessage attribute

This involves custom logic to detect excessive length, specific data formats (e.g., mobile numbers), problematic special characters, and common security vulnerabilities Cross-Site Scripting (XSS) and SQL Injection for this problem.

Database-Driven Testing
All test data, containing valid, invalid, empty, and malicious input scenarios, is collected from a database. TestNG's @DataProvider annotation is utilized to dynamically fetch these test cases with **DBUtility** that is already defined during assignment-1.

Page Object Model
The FormsPage class, central to my Page Object Model implementation, encapsulates all relevant form elements and their associated interaction methods. This class provides also contains methods for retrieving validation messages and assessing the validity state of individual form fields that aligns with my goal to apply clean, readable and DRY principles.

Clear Reporting with Screenshots
ExtentReports is integrated for test execution logging, detailing each step with a clear pass or fail status. Upon any validation failure/exception, the program automatically captures a screenshot of the form's state and embeds it directly into the generated HTML report that in turn helps to significantly accelerate the debugging process by providing immediate context for failures.

Test Cases

In order to verify the form input handling, I created a focused set of test cases covering both typical and edge scenarios:

1. **Required Fields**:
   I tested combinations where first name, last name, gender-radio-button and mobile were missing to confirm that the form shows proper validation errors.

2. **Whitespace Inputs**:
   I entered only spaces in fields like first name to check if the form catches empty-looking inputs.

3. **Security Checks**:
   I included script tags (<script>), image injection (<img onerror>), and SQL payloads ('; OR 1=1--) to test for XSS and SQL injection vulnerabilities.

4. **Special Characters & Length**:
   I tested names with special characters and overly long strings to check input restrictions.

5. **Invalid Mobile Numbers**:
   I tried short numbers, alphabets in the mobile field, and incorrect formats to verify validation.

These tests helped confirm that the form correctly handles required fields, rejects unsafe or malformed input, and provides clear feedback to users.

The complete test cases is provided in the table below:

Test Case Code	Description	First Name	Last Name	Mobile	Gender	Expect Error
TC01	All fields empty	*NULL*	*NULL*	*NULL*	Male	True
TC02	Only first name provided	John	*NULL*	*NULL*	Male	True
TC03	Only last name provided	*NULL*	Shrestha	*NULL*	Male	True
TC04	Only mobile number provided	*NULL*	*NULL*	9876543210	Male	True
TC05	First name and last name provided, no mobile	Iron	Shrestha	*NULL*	Male	True
TC06	First name and mobile provided, no last name	Silver	*NULL*	9876543210	Male	True
TC07	Last name and mobile provided, no first name	*NULL*	Rai	9876543210	Male	True
TC08	All required fields provided correctly	Elle	Musk	9876543210	Male	False
TC09	First name has spaces only	" "	Gates	9876543210	Male	True
TC10	First name has script injection	<script>	Test	9876543210	Male	True
TC11	Last name has image error injection	Rock	<img onerror>	9876543210	Male	True
TC12	Mobile has SQL injection attempt	'; OR 1=1--	King	1234abcd	Male	True
TC13	Special characters in first and last name	!@#$	%^&*	()_+	Male	True
TC14	First name exceeds max length	A...A *(100+ characters)*	Thapa	9876543210	Male	True
TC15	First and last names with leading/trailing	" Test "	" Lint "	" 9876543210 "	Male	False

Test Case Code	Description	First Name	Last Name	Mobile	Gender	Expect Error
	spaces					
TC16	Special characters in last name	Jim	!@#$	9876543210	Male	True
TC17	Mobile number too short	Hari	Ram	12345	Male	True
TC18	Mobile number contains alphabets	Geralt	Rivea	9876AB3210	Male	True
TC19	Gender field empty	The	Witcher	9876543210	*NULL*	True

Process Documentation

1) **FormsPage**
   1. I initially identified the required form fields submitting an empty form and located them with their id. The required fields are:
      firstName, lastName, mobile, and gender-buttons
   2. Then I wrote methods to fill the textfields and select radio buttons and submit form.
      1. enterFirstName(), enterLastName(), enterMobile(), selectGenderMale(), submitForm()
      2. Then wrote **JavaScriptUtility** to interact with javascript elements that further encompasses scrollToElementJS(), clickJS() and then leveraged it on the FormsPage with scrollToField()
   3. Next to complete the custom field validation I wrote two more utilities:
      1. **InputValidatorsUtility**
         i. This utility checks for isWhitespaceOnly(), containsXSS(), containsSQLInjection(), containsSpecialCharacters(), exceedsLength(), isValidMobileFormat()
         ii. Next I categorized these methods to combine them into two distinct validation methods namely, validateSanitization(), and validateSecurity()
      2. Then I created **ValidationResult** utility that grabs the failChecks for all the functions in InputValidatorUtility and adds it to a List<String>. This list is then returned to the function in FormsPageTest for robust logging.

2) FormsValidationTest
   1. This part quickly initializes the Screenshot utility and then overrides the getReportName() and getDocument() method for dynamic naming of the report.
   2. The test data is loaded from the testdb sqlite database and passed to the testRequiredFieldValidation() through **@DataProvider**
   3. Then two more functions are used in the test for input validation and the verifyValidationMessage() which checks if HTML element has the validationMessage through getAttribute property.
   4. Since the radio-buttons and the mobileField did not have validation message two more functions are defined in the FormsPage isGenderSelected() and isMobileFieldValid()

3) Reporting
   The process of reporting is similar to that of assignment-1 where the exceptions and validation error are caught and proper screenshots are recorded.
   Here to make debugging process more easier a new method is added in the **JavaScriptUtility:**

highlightElement() that highlights the element where the error is encountered.

4) Config Properties
   This resource now contains the url for two different test cases, the login test and the forms test and this is where we define our database url and the testUrl and leverage the ConfigReader to load respective data values.

Final Result:

The final obtained results is provided in the table below. All the test passed without any errors.

Test Case Code	Description	First Name	Last Name	Mobile	Gender	Expect Error	Actual Error
TC01	All fields empty	*NULL*	*NULL*	*NULL*	Male	True	True
TC02	Only first name provided	John	*NULL*	*NULL*	Male	True	True
TC03	Only last name provided	*NULL*	Shrestha	*NULL*	Male	True	True
TC04	Only mobile number provided	*NULL*	*NULL*	9876543210	Male	True	True
TC05	First name and last name provided, no mobile	Iron	Shrestha	*NULL*	Male	True	True
TC06	First name and mobile provided, no last name	Silver	*NULL*	9876543210	Male	True	True
TC07	Last name and mobile provided, no first name	*NULL*	Rai	9876543210	Male	True	True
TC08	All required fields provided correctly	Elle	Musk	9876543210	Male	False	False
TC09	First name has spaces only	" "	Gates	9876543210	Male	True	False
TC10	First name has script injection	<script>	Test	9876543210	Male	True	False
TC11	Last name has image error injection	Rock	<img onerror>	9876543210	Male	True	False
TC12	Mobile has SQL injection attempt	'; OR 1=1--	King	1234abcd	Male	True	False
TC13	Special characters in first and last name	!@#$	%^&*	()_+	Male	True	False
TC14	First name exceeds max length	A...A *(100+ characters)*	Thapa	9876543210	Male	True	False
TC15	First and last names with leading/trailing spaces	" Test "	" Lint "	" 9876543210 "	Male	False	True
TC16	Special characters in	Jim	!@#$	9876543210	Male	True	False

Test Case Code	Description	First Name	Last Name	Mobile	Gender	Expect Error	Actual Error
	last name						
TC17	Mobile number too short	Hari	Ram	12345	Male	True	True
TC18	Mobile number contains alphabets	Geralt	Rivea	9876AB3210	Male	True	True
TC19	Gender field empty	The	Witcher	9876543210	*NULL*	True	True

Analysis

Test Case	Description	Reasons
TC09	First name has spaces only	Only whitespaces should not be allowed in first name but allowed
TC10	First name has script injection	<script> used in XSS attacks is valid form input which should be disregarded or converted to escapeHTML
TC11	Last name has image error injection	Same reason as TC10
TC12	Mobile has SQL injection attempt	SQL injection command patterns are not validated
TC13	Special characters in first and last name	Special characters in first name and last name must not be allowed but our form allows it as valid input
TC14	First name exceeds max length	There is no max length criterial currently being used in first name which might have maximum length (50 assumption) and is not realistic
TC15	First and last names and number with leading/trailing spaces	The leading and trailing spaces are not trimmed from selenium input (as sendKeys)but omitted in real case so becomes false positive
TC16	Special characters in last name	Same as TC13