**Login Scenarios**

**GUI**

1. All elements on the UI should display nicely.
2. Max-min length should be set for all textboxes.
3. Password should be masked.
4. Login case sensitives (upper case, lower case).
5. Validation messages for incorrect username/password, fields left blank, wrong input types…
6. Login with correct credentials.
7. Display ‘Forgot my account’ link.
8. Display ‘Create new account’ link.
9. Login credentials should be encrypted in database.

**Security**

1. Logged in user copy a url and paste in another browser, it should redirect to Login page.
2. Can’t copy value in Password textbox.
3. Virtual keyboard applied if application required.
4. Login failed 3 times, account will be blocked in a specific time.
5. Email will be triggered when user logs in on multiple devices.
6. Inject SQL injection, XSS input should be verified.
7. SSL certificate should be implemented.
8. Two way authentication through OTP on mobile/email.
9. Maximum users can log in at the same time on different browser/platform.

**Cookie**

1. Log out and click Back button on the browser, will redirect user to Login page.
2. Another account on another device will be logged out when user logs in to the current device.
3. Session time out.

**Browser**

1. If Cookie session is cleared, and user tried to log in, then ask for credentials again.
2. Validate the login functionality when cookie is turned OFF.

SQL Injection Based on 1=1 is Always True

UserId:  (input value is 105 OR 1 = 1)

Then, the SQL statement will look like this:

SELECT \* FROM Users WHERE UserId = 105 OR 1=1;

## SQL Injection Based on ""="" is Always True

User Name:  
 (input value is “ OR “” = “)

Password:  
 (input value is “ OR “” = “)

The code at the server will create a valid SQL statement like this:

### **Result**

SELECT \* FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""

The SQL statement below will return all rows from the "Users" table, then delete the "Suppliers" table.

User id:  (input value is 105; DROP TABLE Users)

The valid SQL statement would look like this:

### **Result**

SELECT \* FROM Users WHERE UserId = 105; DROP TABLE Suppliers;

XSS example: <script>alert("TEST");</script>)

XPATH

Xpath=//input[@type='text']

Xpath= //label[@id='message23']

Xpath= //input[@value='RESET']

Xpath=//\*[@class='barone']

Xpath=//a[@href='http://demo.guru99.com/']

Xpath= //img[@src='//cdn.guru99.com/images/home/java.png']

Xpath = //\*[contains(@text,’welcome’)]

Xpath = //\*[contains(@href,’www.google.com’)]

Xpath=//input[@type='submit' and @name='btnLogin']

Xpath=//label[starts-with(@id,'message')]

Xpath = //div[@text()=’login’]

Xpath = //\*[@text()=’input’]//following::input[1]

Xpath = //\*[@id()=’userid’]//ancentor::div

Xpath = //div[@type=’class’]//child::li

Xpath=//\*[@type='submit']//preceding::input

xpath=//\*[@type='submit']//following-sibling::input

Xpath=//\*[@id='rt-feature']//parent::div