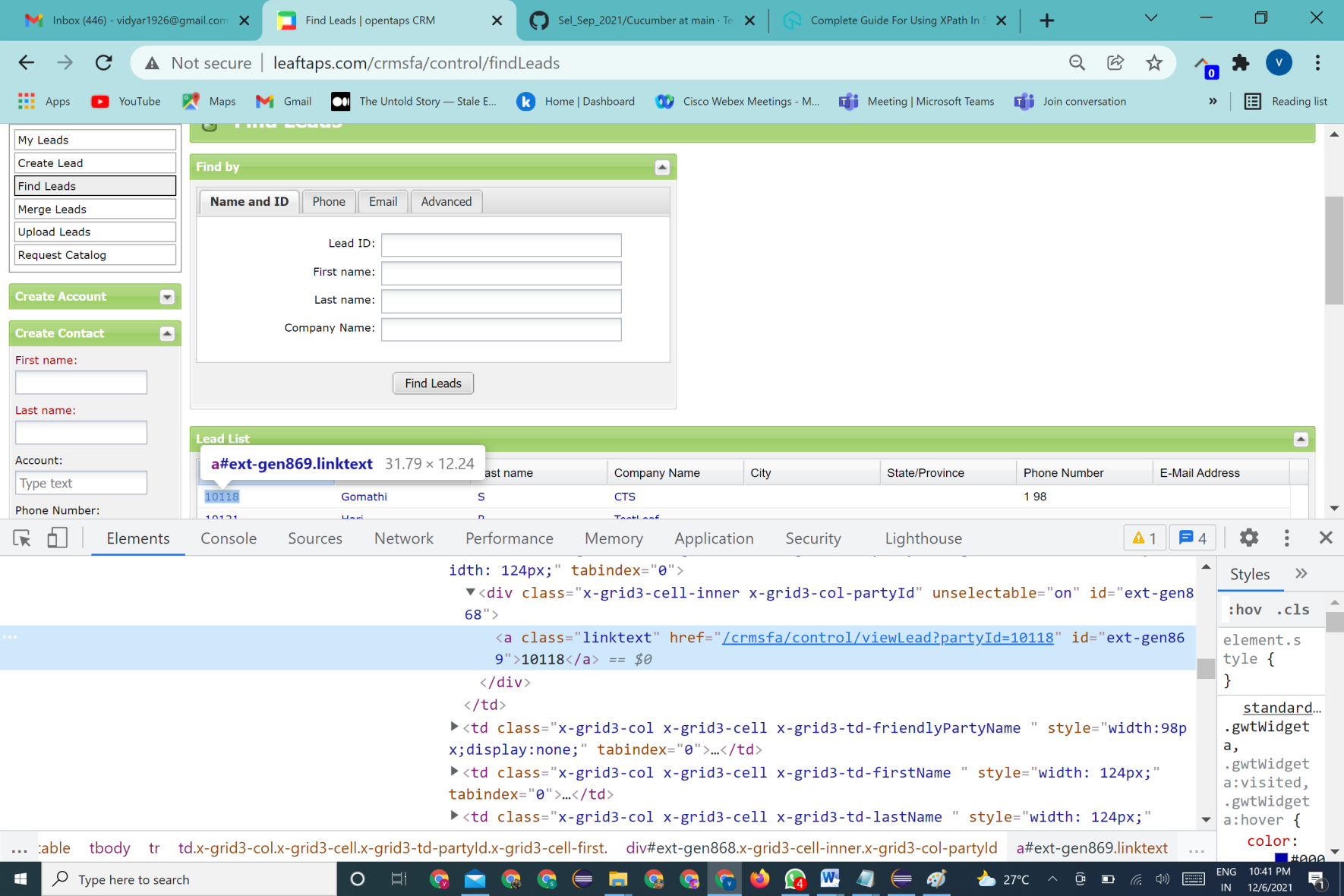
**Locate Element using Xpath:**

Selenium Webdriver is an open source tool which supports to automate the web based application. To interact with the webpage selenium needs HTML document. The Structure of WebElement in the Dom will be like

<tagname>attribute=”attribute value”, text</ tagname >

Selenium uses 2 different ways to locate the web elements. Primary way to locate the element is by using the basic locators id, name, class name, linktext and tagname.

Sometimes the web elements in the dom will not have proper attribute values. For example the attribute id will have auto generated numbers like wise names ,tagname and class names will be duplicated, some time the values will have special characters and white space.



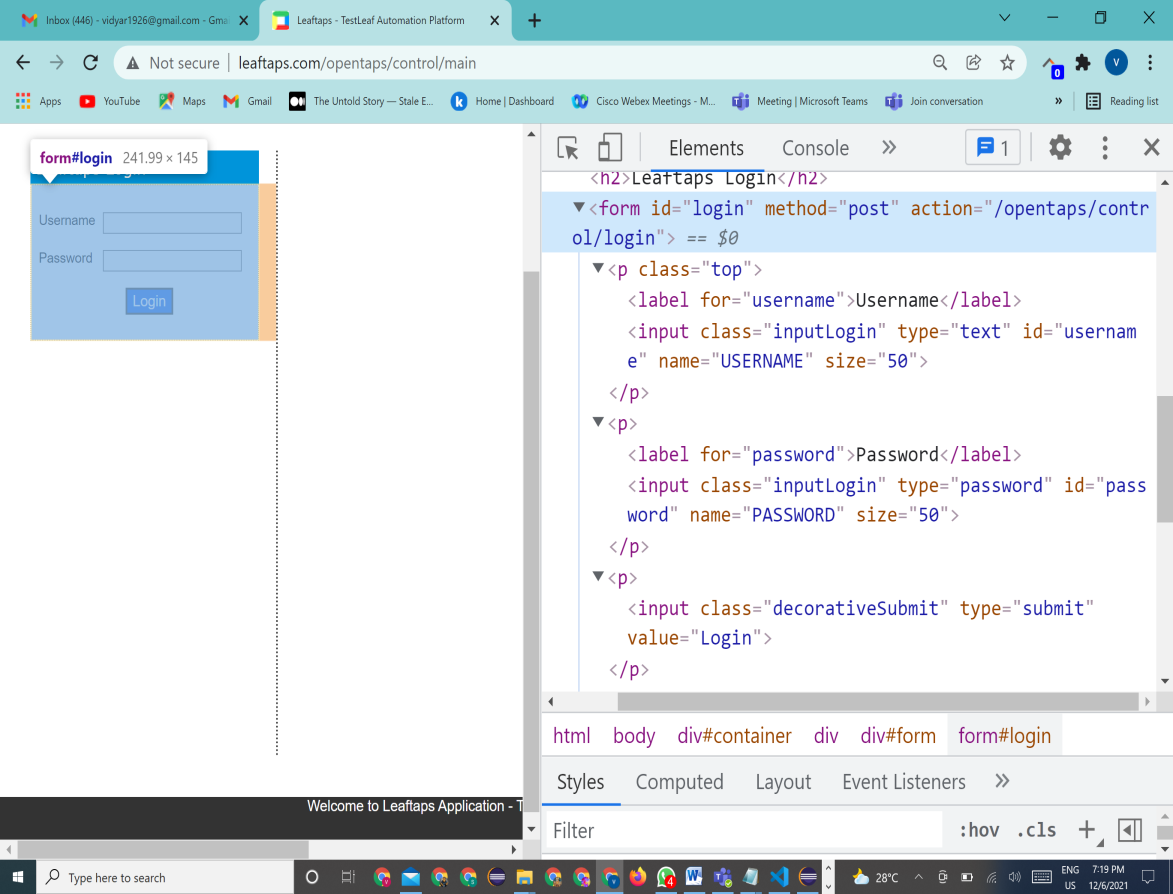
In the above fig, the attributes are identifies with values like

**id="ext-gen869”-** eventhough the id is considered to be reliable locator we cannot use it as it contains number and it is dynamic(subject to change).

**class="x-grid3-col x-grid3-cell x-grid3-td-friendlyPartyName”** -> the class name has special character it also cannot be used as basic locator to identify this particular element . In these cases , Xpath methods are used to locate the webelements accurately.

**Locating Element with Xpath:**

**X-path** refers to xml path. Xml path provides the precise location of the WebElement. It helps to traverse through the HTML structure from the origin to destination of the webpage to find the exact position of the WebElement. Xpath is preferably considered when,

\*There are no other standard locators are available like id, name..

\*The locators are not stable in nature

\*Having multiple values(duplicated)

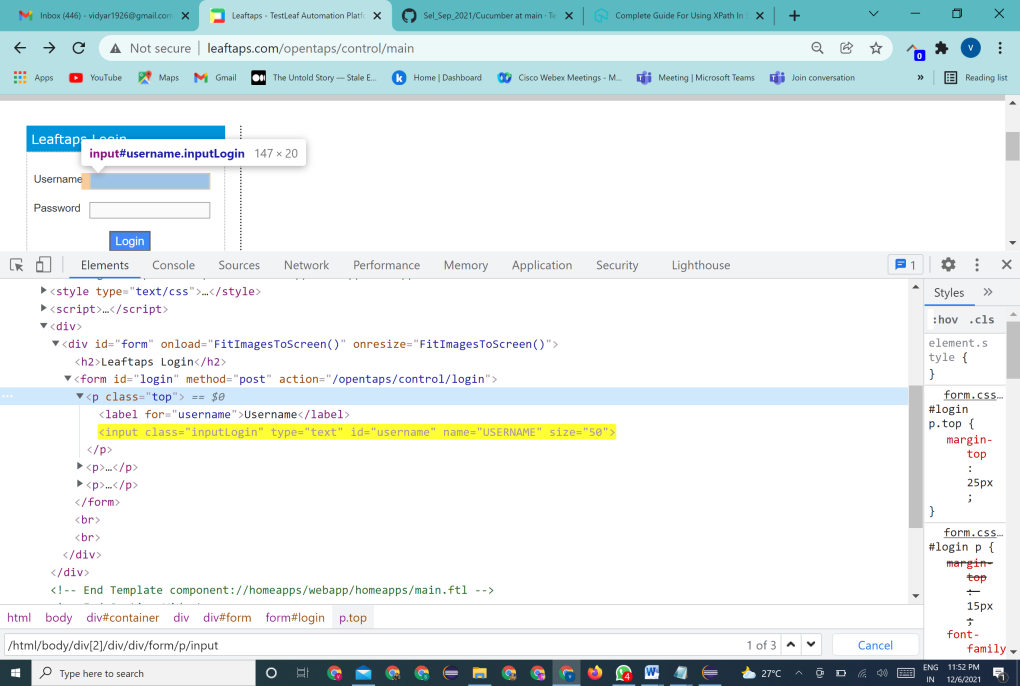
In this fig, class name for both username and password is duplicated and cannot be considered as standard locator to locate the element.

In this case, ***XPath*** comes into picture.

XPath is of 2 types

* Absolute XPath
* Relative XPath

Absolute XPath:

 **Absolute XPath** ->locates the element starting from the root element(HTML) to the destined weblelement. The syntax for the absolute xpath is

**/html/body/div[2]/div/div/form/p/input/…** (Sample to locate username in login page)

* The absolute XPath always starts with / and traverse through each and every element in hierarchy to reach the required WebElement.
* It usually take long path to locate the element. So it not a preferred choice and it is unreliable because a small change in the UI, can change the absolute Xpath .

**Relative XPath:**

**Relative XPath** works in relation with the WebElement location in the dom. Simply it shows the exact address of the WebElememt.

\*Relative XPath starts with the // which locates the inner or middle part of the Dom.

\*Relative XPath is also reliable and it is very handy when the dom is dynamic.

\*It helps to identify the node element with a reference rather than from the root.

***General syntax of the Relative xpath is:***

**//tagname[@attribute='attributeValue] value of the selected attribute**

**Select the attribute type**

**currentnode tagname of current node**

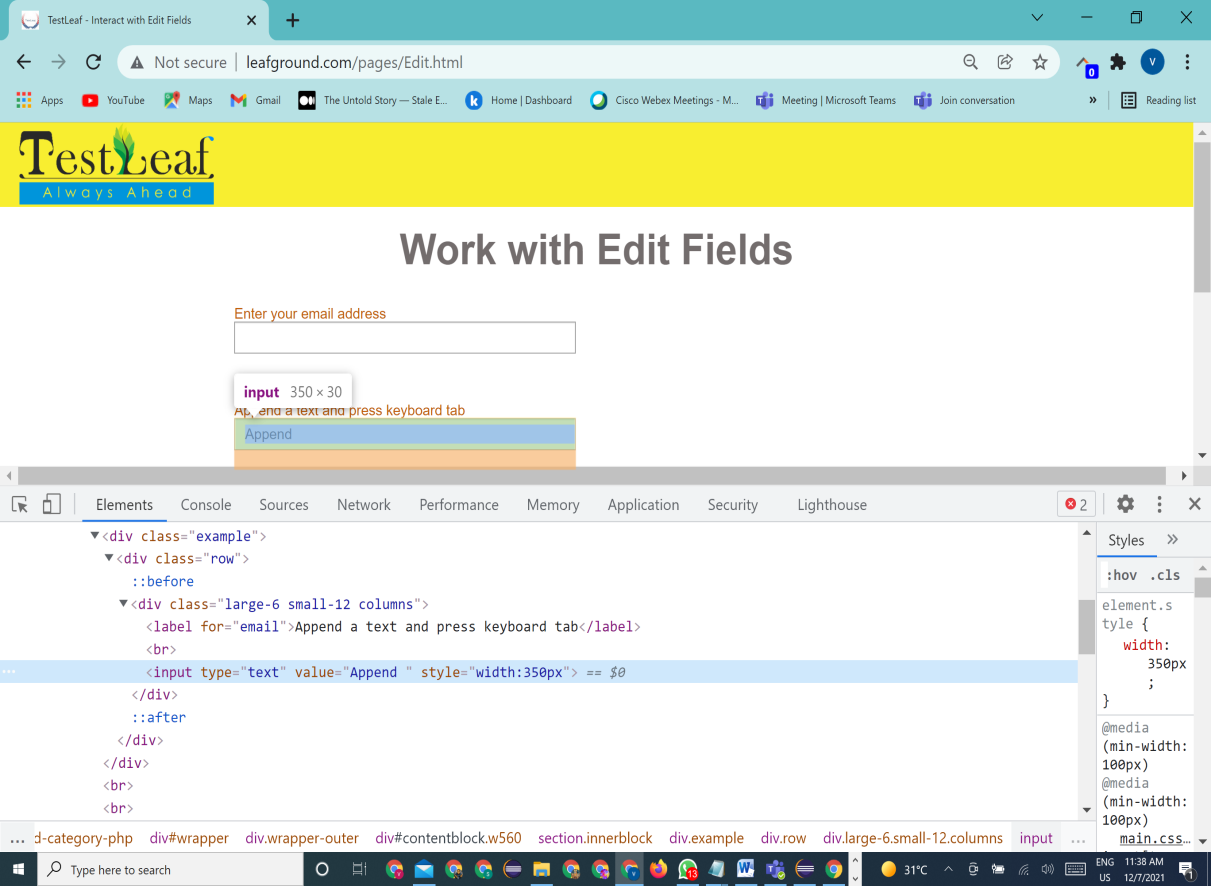
We can write the relative xpath in different ways. Lets look into each type of xpath and how to use this xpath in our automation scripts using Leafground application

**1. Attribute Based XPath:**

Attribute based XPath uses the attribute and attribute of the WebElement in the DOM.

Syntax:

**//tagname[@attribute='attributeValue] Required WebElement**



**Location of the element in the dom**

**To locate the Append textbox using attribute based XPath**

**//input[@value='Append ']**

**2. Text Based Xpath:**

* Xpath is based on the text given to the WebElement

Syntax:

//tagname[text()='text value in DOM']

\*text() is a method, so here we will not use @ to select the attribute

**//label[text()='Append a text and press keyboard tab']**

**3. Partial Match Xpath:**

Partial match xpath is used when the attribute values are dynamic. It uses **Contains()** method to match the partial value of the attribute. Partial match xpath is of two types

1. **Partial Text Based Xpath: =>uses the partial text value of the webElement**

Syntax:

**//tagname[contains(text(),' Partial text in my DOM' ')]**

**// label [contains(text(),'Append a text ')]**

**a) Partial attribute Based Xpath: =>uses the partial attribute value of the WebElement**

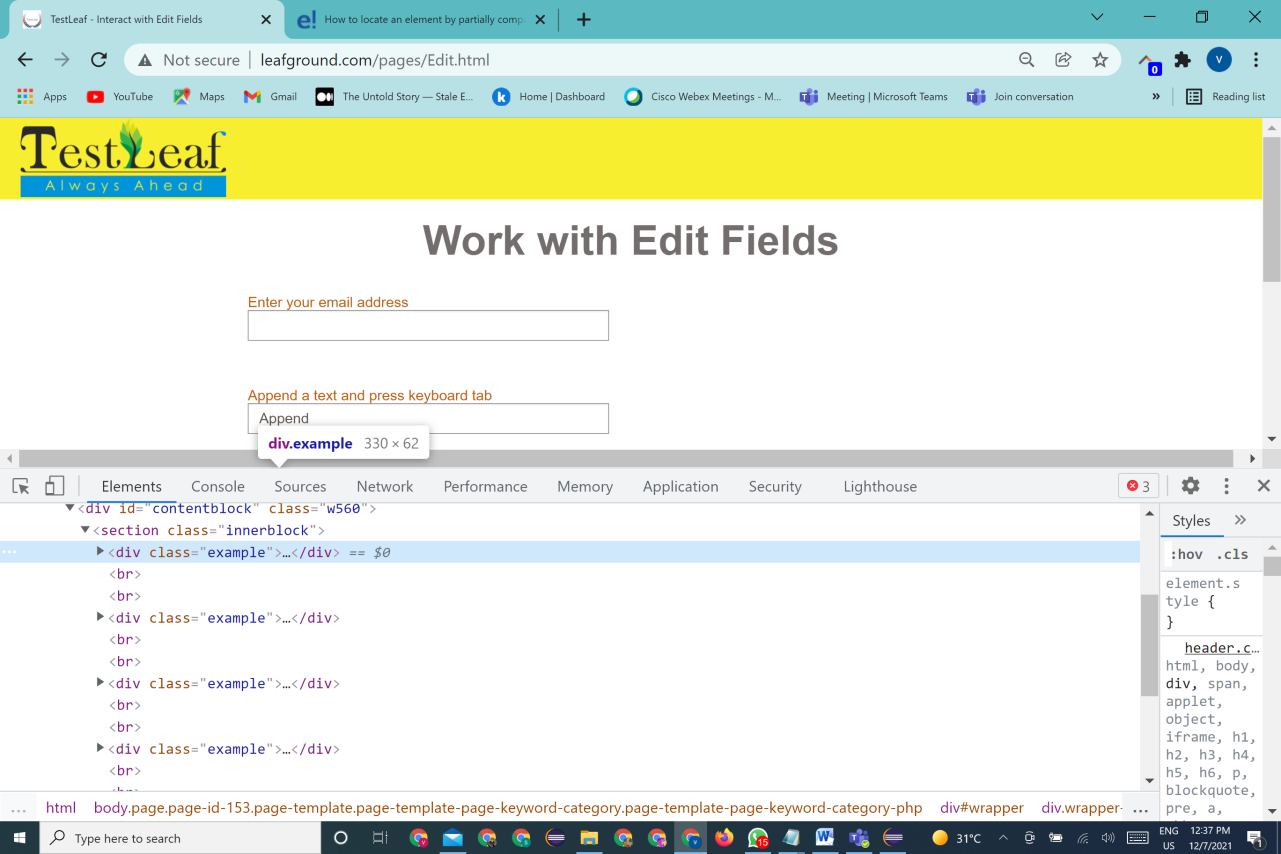
**//tagname[contains(@attribute, 'Partial value of the attribute')]**

**//input[contains(@value='App ')]**

**4. Collections Based Xpath:**

Collection based xpath usually used when there is multiple matches of a WebElement in the dom. It uses index value to locate the particular match. Index of the XPath starts from 1.

Syntax:

**(Any valid relative Xpath)[index]**

In the above dom page, we have multiple entries of **<div class=”example”>**

If we need to identify the second match , then the xpath will be

**(//div[@class='example'])[2]**

**\*Note : if the index is not mentioned then the driver matches the first resulting match of the WebElement.**

**Exceptions**

Exception you will get when the driver could not locate the right WebElement

\*NoSuchException->not a valid xpath  
\*Invalid Selector->refers the syntax error in the xpath

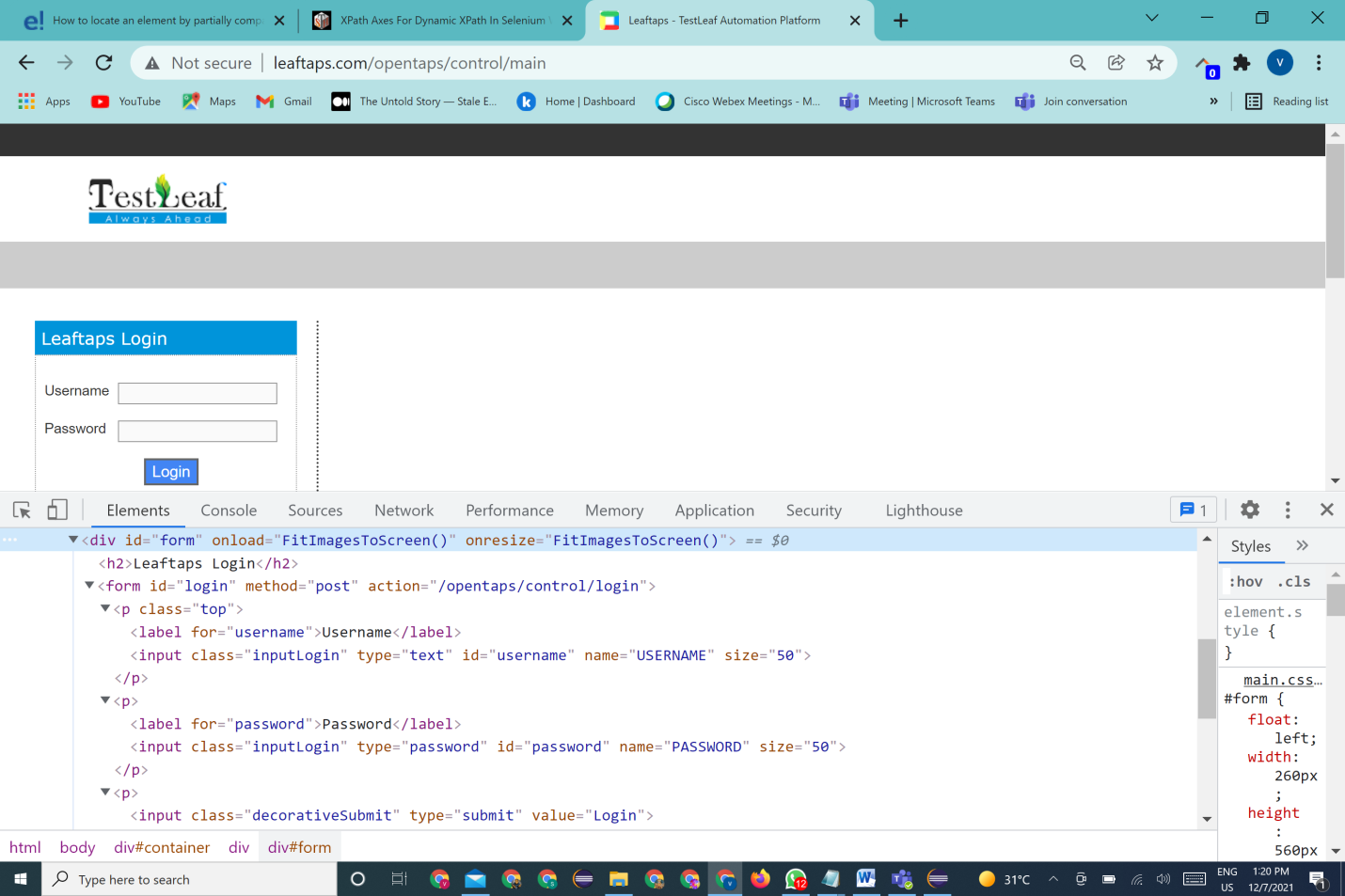
**Advanced Xpath**

If any of the above basic relative XPath is not working to locate the WebElement , the Webdriver throws the exception NoSuchElementException.

To handle this exception, advanced xpath concept comes in. The **Advanced Xpath** is also called as **RelationShip based Xpath/ Axes based Xpath.**

Relationship based XPath by name it refers to the relationship two nodes of the DOM elements. In simple terms, it refers to the current node referred to the hierarchical relationship between the Elements as family structure.

Lets understand the structure here:



We have the dom here as

<div id="form" onload="FitImagesToScreen()" onresize="FitImagesToScreen()">

<h2>Leaftaps Login</h2>

<form id="login" method="post" action="/opentaps/control/login">

<p class="top">

<label for="username">Username</label>

<input class="inputLogin" type="text" id="username" name="USERNAME" size="50">

</p>

<p>

<label for="password">Password</label>

<input class="inputLogin" type="password" id="password" name="PASSWORD" size="50">

</p>

<p>

<input class="decorativeSubmit" type="submit" value="Login">

</p>

</form>

<br><br>

</div>

The **relationship** here as

Parent -child

<**div**>-----><**form**>-

<p> -----------> <label>

Parent -child <p> sibilings -----------> <label> cousin

<p> -----------> <label>

**<div>** represents the ancestor or grandparent)

**<form>** represents parent to **<p>** and child of **<div>**

**<p>** represents child to **<form>**

All the **<p>** represents siblings

<**label**> represents child of <**p**>

All the <**label**> inside the <**p**> represents cousin.

With this we can write 8 types of xpaths

***1 Parent To child*** *:*

To reach the WebElement from <form>(**parent tag**) to <p> (**child**)

Syntax: **parent realtive xpath/tagName of child**

**//form[@id='login']/p**

***2 GrandParent to child Relation :***

To reach the WebElement from <form>(**grandparent tag**) to <label> (**grand**-**child**)

Syntax: **GrandParent relative xpath//tagName of grand child**

**(//form[@id='login']//label)**

***3 Child to parent*** *:*

To reach the WebElement from <**p**>(**child**) to <form> (**parent**)

**Child relative xpath/parent::parent tagName**

**//input[@id='username']/parent::p**

***4 GrandChild to Grand parent*** *:*

To reach the WebElement from <**label**>(**grand**-**child**) to <form> (**grand-parent**) , we have the keyword **ancestor**

**GrandChild relative xpath/ancestor::tagName of grand parent**

**//input[@id='password']/ancestor::form**

***5 Elder Sibling toYounger Sibling :***

To reach the WebElement from <**label> (child)** to <**label**> (**child**) of same parent we have the keyword

**following-sibling ( finding from top to bottom )**

<p class="top">

**<label for="username">Username</label>**

**<input class="inputLogin" type="text" id="username" name="USERNAME" size="50">**

</p>

**Elder sibling relative xpath/following-sibling::tagName of younger sibling**

**//label[text()='Username']/following-sibling::input**

***6 Younger Sibling to elder Sibling:***

To reach the WebElement from Second <**label> (child)** to first<**label**> (**child**) of same parent we have the keyword **preceding-sibling( finding from bottom to top )**

**Younger sibling relative xpath/preceding-sibling::tagName of Elder sibling**

**//input[@id='username']/preceding-sibling::label**

***7 Elder cousin to Younger cousin :***

To reach the WebElement from first parent child<**label> (child)** to second parent child <**label**> (**child**) we have the keyword **following (finding from top to bottom)**

**Elder cousing relative xpath/following::tagName of younger Cousin**

**//input[@id='username']/following::label**

***8 Younger cousin to Elder cousin :***

To reach the WebElement from second parent child<**label> (child)** to first parent child <**label**> (**child**) we have the keyword **preceding( finding from bottom to top )**

**Younger cousin xpath/preceding::tagName of Elder cousin**

**//label[text()='Password']/preceding::input**