Ineuron

Assignment-6

1.Explain why selenium is important in web scraping?
Ans:-Benefits of Using Selenium for Web Scraping
Since WebDriver uses an actual web browser to access the website, it is no different from browsing the web by a human or a robot. When you navigate to a web page with Webdriver, the browser loads all the resources of the website (JavaScript files, images, CSS files, etc.) and executes all the JavaScript on the page. It also stores all the cookies created by your websites. This makes it very difficult to determine whether a real person or a robot has accessed the website. With Webdriver this is possible in a few simple steps, however, it is really difficult to emulate all these tasks in a program that sends handcrafted HTTP requests to the server. To extract data from these browsers, Selenium provides a module called WebDriver, which is useful for performing various tasks such as automated testing, cookie retrieval, screenshot retrieval, and much more. Some common Selenium use cases for web scraping are form submission, auto-login, data addition and deletion, and alert handling. For more information about how to use Selenium for web scraping, there are many courses on selenium training online that you can enroll in for your benefit.

However, web scraping is considered to be the most reliable and efficient data acquisition method among all these methods. Web scraping, also known as the extraction of web data, is an automated process of scraping big data from websites.
Drawbacks of using Selenium for Web Scraping
As much as there are advantages to using selenium for web scraping, there must be some drawbacks. Let's see a few of these drawbacks below.
A large network traffic generated: Web browsers download many supplementary files that are of no value to you (such as CSS, JS, and image files). If you only request resources you really need (with different HTTP requirements) this can generate a lot of traffic.
Scraping is easily detected with simple methods like Google Analytics: One of the drawbacks of using selenium for web scraping is, if you explore a lot of pages with Webdriver, you can easily look into JavaScript-based tracking tools (such as Google Analytics). The site owner doesn't even need to install a sophisticated scraping detection mechanism!
Time and Resources Consumption: When you use WebDriver to scrape web pages you load the entire web browser into the system memory. Not only does this take time and consume system resources, but it can also cause your security subsystem to overreact (and even not allow your program to run).
Slow Scraping Process: Since a browser waits for the entire web page to load, and only then allows you to access its elements, the scraping process can take longer than making a simple HTTP request to the webserver.
Types of Web Scraping with Selenium
There are two types of webscraping with Selenium:
Static web scraping
Dynamic web scraping
There is a difference between static websites and dynamic websites. In static pages, the content remains the same unless someone changes them manually. On the other hand, content can move from multiple visitors to dynamic websites. For example, it can be changed according to the user profile. This increases its time complexity as a dynamic website on the client side can

process a static page on the server-side while on the client side.
The content of the static website is downloaded locally, and the corresponding scripts are used to collect the data. In contrast, dynamic website content is generated only for any number of requests during the initial load request. In order to delete the data on the website, Selenium provides some standard locators which help in locating the content of the test page. Locators are nothing more than keywords associated with HTML pages. For further details on why you should use or how to use selenium for web scraping, you can do yourself some good by getting certification on an online selenium training course to widen your horizon.

2.Whats the difference between scraping images and scraping websites?use an example to demonstrate your point?
Ans:- Web Scraping refers to the extraction of data from a website or webpage. Usually, this data is extracted on to a new file format. For example, data from a website can be extracted to an excel spreadsheet.

Web Scraping can also be done manually, although in most cases automated tools will be used to extract the data.

One key aspect of Web Scraping is that it is often done with a focused approach. This means that Web Scraping projects seek to extract specific data sets from a website for further analysis.

For example, a company might extract product details from laptops listed on Amazon in order to figure out how to position their new product in the market.

Want to learn more about web scraping? Check out our in-depth guide on web scraping and what it is used for.
Web Crawling refers to the process of using bots (or spiders) to read and store all of the content on a website for archiving or indexing purposes.

Search engines (such as Bing or Google) use web crawling to extract all the information from a website and index it in their search engines. That’s how Google can tell what pages will have the information you’re looking for.When you demonstrate something, you show what it is or how it works. To demonstrate how your new juicer works, you should have lots of kale, carrots, and beets on hand — and some brave friends to try your concoction.

Demonstrate comes from the Latin word demonstrare, meaning “to point out by argument or deduction.” To demonstrate a point you must make a valid argument and give examples of why you think it's true. Demonstrate can also refer to a public protest. You can demonstrate with your comrades by marching through the streets with homemade protest signs.

Definitions of demonstrate
verb give an exhibition of to an interested audience
synonyms:demo, exhibit, present, show
see more
verb establish the validity of something, as by an example, explanation or experiment
“The experiment demonstrated the instability of the compound”
synonyms:establish, prove, shew, show
see more

verb provide evidence for; stand as proof of; show by one's behavior, attitude, or external attributes
“This decision demonstrates his sense of fairness”
synonyms:attest, certify, evidence, manifest
attest
establish or verify the usage of
see more
verb march in protest; take part in a demonstration
“Thousands demonstrated against globalization during the meeting of the most powerful economic nations in Seattle”
What’s the Difference Between Web Scraping and Web Crawling
At this point, you might already be able to tell the difference between Web Scraping and Web Crawling. Even if both terms refer to the extraction of data from websites.

3.Explain how mongoDB indexes data?
Ans:-Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient and require MongoDB to process a large volume of data.

Indexes are special data structures, that store a small portion of the data set in an easy-to-traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field as specified in the index.

The createIndex() Method
To create an index, you need to use createIndex() method of MongoDB.

Syntax
The basic syntax of createIndex() method is as follows().

```
>db.COLLECTION_NAME.createIndex({KEY:1})
```

Here key is the name of the field on which you want to create index and 1 is for ascending order. To create index in descending order you need to use -1.

Example

```
>db.mycol.createIndex({"title":1})
{
"createdCollectionAutomatically" : false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
"ok" : 1
}
>
```

In createIndex() method you can pass multiple fields, to create index on multiple fields.

```
>db.mycol.createIndex({"title":1,"description":-1})
>
```

This method also accepts list of options (which are optional). Following is the list −

Parameter Type Description

background Boolean Builds the index in the background so that building an index does not block other database activities. Specify true to build in the background. The default value is false.
unique Boolean Creates a unique index so that the collection will not accept insertion of documents where the index key or keys match an existing value in the index. Specify true to create a unique index. The default value is false.
name string The name of the index. If unspecified, MongoDB generates an index name by concatenating the names of the indexed fields and the sort order.
sparse Boolean If true, the index only references documents with the specified field. These indexes use less space but behave differently in some situations (particularly sorts). The default value is false.
expireAfterSeconds integer Specifies a value, in seconds, as a TTL to control how long MongoDB retains documents in this collection.
weights document The weight is a number ranging from 1 to 99,999 and denotes the significance of the field relative to the other indexed fields in terms of the score.
default_language string For a text index, the language that determines the list of stop words and the rules for the stemmer and tokenizer. The default value is English.
language_override string For a text index, specify the name of the field in the document that contains, the language to override the default language. The default value is language.
The dropIndex() method
You can drop a particular index using the dropIndex() method of MongoDB.

Syntax
The basic syntax of DropIndex() method is as follows().

```
>db.COLLECTION_NAME.dropIndex({KEY:1})
```

Here, "key" is the name of the file on which you want to remove an existing index. Instead of the index specification document (above syntax), you can also specify the name of the index directly as:

```
dropIndex("name_of_the_index")
```

Example

```
> db.mycol.dropIndex({"title":1})
{
"ok" : 0,
"errmsg" : "can't find index with key: { title: 1.0 }",
"code" : 27,
"codeName" : "IndexNotFound"
}
```

The dropIndexes() method
This method deletes multiple (specified) indexes on a collection.
Syntax
The basic syntax of DropIndexes() method is as follows() −

```
>db.COLLECTION_NAME.dropIndexes()
```

Example
Assume we have created 2 indexes in the named mycol collection as shown below −

```
> db.mycol.createIndex({"title":1,"description":-1})
```

Following example removes the above created indexes of mycol −

```
>db.mycol.dropIndexes({"title":1,"description":-1})
{ "nIndexesWas" : 2, "ok" : 1 }
>
```

The getIndexes() method
This method returns the description of all the indexes int the collection.

Syntax
Following is the basic syntax od the getIndexes() method −

```
db.COLLECTION_NAME.getIndexes()
```

Example
Assume we have created 2 indexes in the named mycol collection as shown below −

```
> db.mycol.createIndex({"title":1,"description":-1})
```

Following example retrieves all the indexes in the collection mycol .

4.What is the significance of the set modifier?
Ans:- Set modifiers
Set expressions are used to define the scope of a calculation. The central part of the set expression is the set modifier that specifies a selection. This is used to modify the user selection, or the selection in the set identifier, and the result defines a new scope for the calculation.

The set modifier consists of one or more field names, each followed by a selection that should be made on the field. The modifier is enclosed by angled brackets: < >

For example:

```
Sum ( {$<Year = {2015}>} Sales )

Count ( {1<Country = {Germany}>} distinct OrderID )

Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )
```

Element sets
An element set can be defined using the following:

A list of values

A search

A reference to another field

A set function

If the element set definition is omitted, the set modifier will clear any selection in this field. For

example:

Sum({$<Year = >} Sales)

Examples: Chart expressions for set modifiers based on element sets
Examples - chart expressions
Listed values
The most common example of an element set is one that is based on a list of field values enclosed in curly brackets. For example:

{$<Country = {Canada, Germany, Singapore}>}

{$<Year = {2015, 2016}>}

The inner curly brackets define the element set. The individual values are separated by commas.

Quotes and case sensitivity
If the values contain blanks or special characters, the values need to be quoted. Single quotes will be a literal, case-sensitive match with a single field value. Double quotes imply a case-insensitive match with one or several field values. For example:

<Country = {'New Zealand'}>

Matches New Zealand only.

<Country = {"New Zealand"}>

Matches New Zealand, NEW ZEALAND, and new zealand.

Dates must be enclosed in quotes and use the date format of the field in question. For example:

<ISO_Date = {'2021-12-31'}>

<US_Date = {'12/31/2021'}>

<UK_Date = {'31/12/2021'}>

Double quotes can be substituted by square brackets or by grave accents.

Searches
Element sets can also be created through searches. For example:

<Country = {"C*"}>

<Ingredient = {"*garlic*"}>

<Year = {">2015"}>

<Date = {">12/31/2015"}>

Wildcards can be used in a text searches: An asterisk (*) represents any number of characters, and a question mark (?) represents a single character. Relational operators can be used to define numeric searches.

You should always use double quotes for searches. Searches are case-insensitive.

For more information, see Set modifiers with searches.

Dollar expansions
Dollar expansions are needed if you want to use a calculation inside your element set. For example, if you want to look at the last possible year only, you can use:

<Year = {$(=Max(Year))}>

For more information, see Set modifiers with dollar-sign expansions.

Selected values in other fields
Modifiers can be based on the selected values of another field. For example:

<OrderDate = DeliveryDate>

This modifier will take the selected values from DeliveryDate and apply those as a selection on OrderDate. If there are many distinct values – more than a couple of hundred – then this operation is CPU intensive and should be avoided.

Element set functions
The element set can also be based on the set functions P() (possible values) and E() (excluded values).

For example, if you want to select countries where the product Cap has been sold, you can use:

<Country = P({1<Product={Cap}>} Country)>

Similarly, if you want to pick out the countries where the product Cap has not been sold, you can use:
<Country = E({1<Product={Cap}>} Country)>
For more information, see Set modifiers using set functions .
See also
Set analysis
Set identifiers
Set operators

5.Explain the mongo DB aggregation framework?
Ans:- Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of

operations on the grouped data to return a single result. In SQL count(*) and with group by is an equivalent of MongoDB aggregation.

The aggregate() Method
For the aggregation in MongoDB, you should use aggregate() method.

Syntax
Basic syntax of aggregate() method is as follows −

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Example
In the collection you have the following data −

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by_user: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100
},
{
   _id: ObjectId(7df78ad8902d)
   title: 'NoSQL Overview',
   description: 'No sql database is very fast',
   by_user: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 10
},
{
   _id: ObjectId(7df78ad8902e)
   title: 'Neo4j Overview',
   description: 'Neo4j is no sql database',
   by_user: 'Neo4j',
   url: 'http://www.neo4j.com',
   tags: ['neo4j', 'database', 'NoSQL'],
   likes: 750
},
```

Now from the above collection, if you want to display a list stating how many tutorials are written by each user, then you will use the following aggregate() method −

```
> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : 1}}}])
{ "_id" : "tutorials point", "num_tutorial" : 2 }
{ "_id" : "Neo4j", "num_tutorial" : 1 }
>
```

Sql equivalent query for the above use case will be select by_user, count(*) from mycol group

by by_user.

In the above example, we have grouped documents by field by_user and on each occurrence of by user previous value of sum is incremented. Following is a list of available aggregation expressions.

Expression Description Example
$sum Sums up the defined value from all documents in the collection.
db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg Calculates the average of all given values from all documents in the collection.
db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min Gets the minimum of the corresponding values from all documents in the collection.
db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max Gets the maximum of the corresponding values from all documents in the collection.
db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push Inserts the value to an array in the resulting document. db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet Inserts the value to an array in the resulting document but does not create duplicates. db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
$first Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.
db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.
db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])
Pipeline Concept
In UNIX command, shell pipeline means the possibility to execute an operation on some input and use the output as the input for the next command and so on. MongoDB also supports same concept in aggregation framework. There is a set of possible stages and each of those is taken as a set of documents as an input and produces a resulting set of documents (or the final resulting JSON document at the end of the pipeline). This can then in turn be used for the next stage and so on.

Following are the possible stages in aggregation framework −

$project − Used to select some specific fields from a collection.

$match − This is a filtering operation and thus this can reduce the amount of documents that are given as input to the next stage.

$group − This does the actual aggregation as discussed above.

$sort − Sorts the documents.

$skip − With this, it is possible to skip forward in the list of documents for a given amount of documents.

$limit − This limits the amount of documents to look at, by the given number starting from the

current positions.

$unwind − This is used to unwind document that are using arrays. When using an array, the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thus with this stage we will increase the amount of documents for the next stage.