Programming Assignment 3

Introduction:

Have developed a web application which enables user to search the content of the programmable web service data stored in the MongoDb no sql database, there are two types of services record which have been stored in the database namely api and mashup user can select the type and based on the type he will be shown different criteria to search for e.g. Updated year, tags, protocols etc. and the results will be search based on the criteria and the text user has entered in the text box.

The application has been developed in Mvc framework using jsp and servlet, and MongoDb No SQL data in the backend. Further the application has been deployed in Apache Tomcat Server 8

Data Preprocessing:

1. Have create 2 collections in mongodb database(test) named **apis** and **mashup** which stores following records of apis/mashup in the given data format

As we have been provided two text files where each line represent the api/mashup record and various fields are separated by $#$, $$$, ### there was need to parse each field so as to make each field separate from each other and store it in the database to ease the process of performing various operations on data, so following steps were performed

1. Split each field by $#$
2. Check whether that field further contains ### if yes then further split that field by ###
   Store it in a list and then store that list as an JSON object
   Eg. Consider, *tag1###tag2###tag3* so in this case tag1, tag2, tag3 are separated by ### so further store them in an array/list and then store. Resultant field become
   Tags: [tag1,tag2,tag3]
3. If the string doesn't contain ### then directly store the string split at the index in appropriate data type Eg. int, string, float {name:"ame of the api"}
4. If the split string is empty store a null object
5. Then further check whether field contains $$$ if yes then again split that field on $$$ and store in the list as mentioned above
   Eg. Consider,

   Flickr$$$http://www.programmableweb.com/api/flickr###GoogleMaps$$$http://www.programmableweb.com/api/google-maps

6. In the following case data is stored as follows,

```
Apis:[
        { api:
            {
                apiName:"Flickr",
```

apiUrl: www.programmableweb.com/api/flickr

}

Api:

{

apiName:" Google Maps"

apiUrl:" http://www.programmableweb.com/api/google-maps"

}

}

]

Additional Functionality:

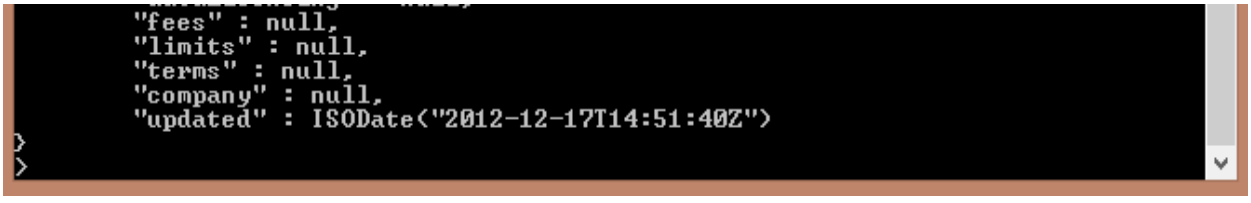
1. The web application designed takes care of sensitivity
2. For the keyword search created an index on fields title, description and summary so as to search all the key words in at least one of these fields

Design Of the Data Structure For Apis (Note:If N/A that means field doesn't exist in the type)

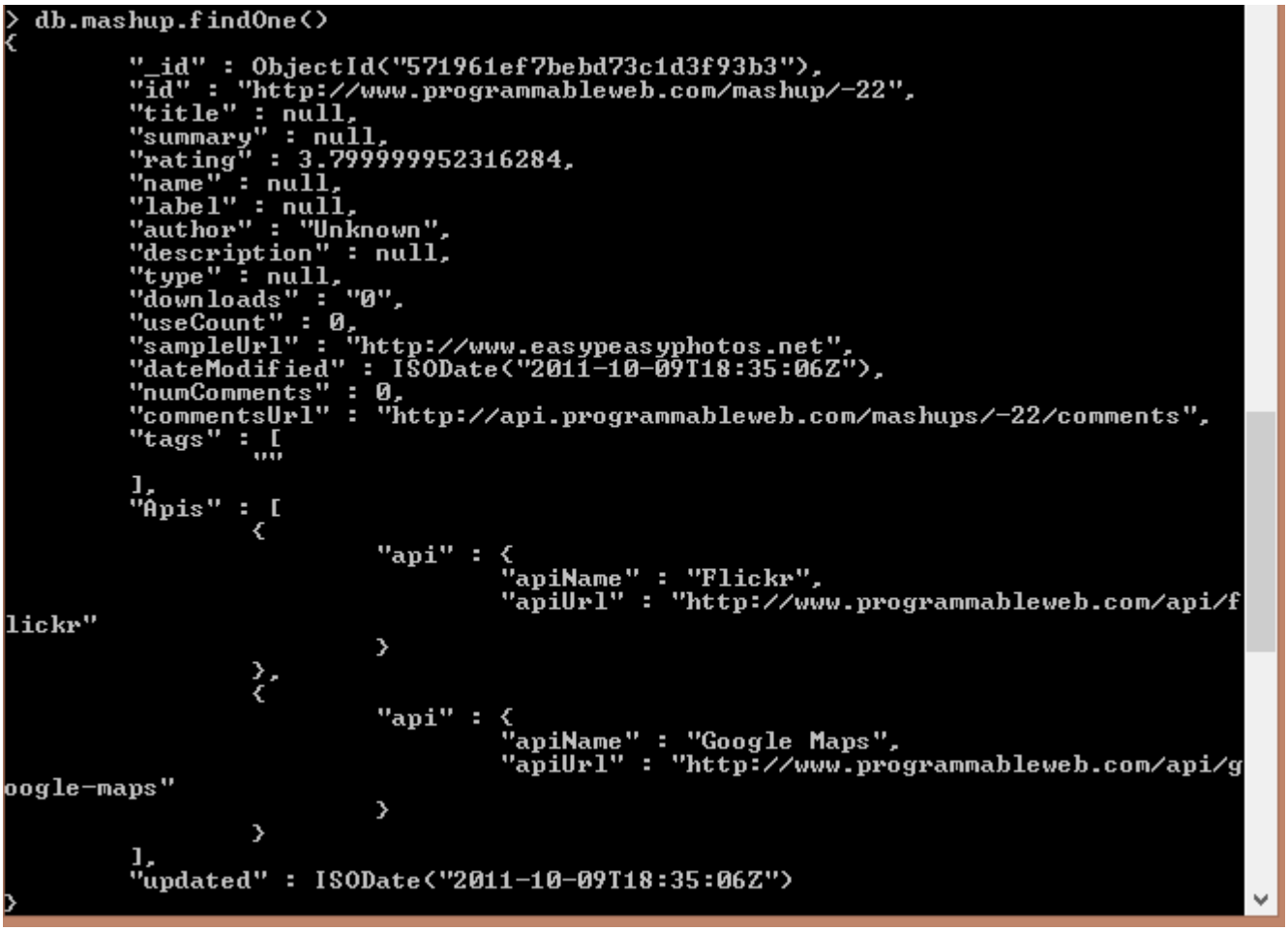
Type---------->	Api	Mashup
Name of Field	**Data Type**	**Data Type**
id	String	String
title	String	String
summary	String	String
rating	Float	Float
name	String	String
label	String	String
author	String	String
description	String	String
type	int	int
downloads	String	String
useCounts	int	int
sampleUrl	String	String
downloadUrl	String	N/A
dateModified	Date	Date
numComments	int	int
remoteFeed	String	N/A
commentsUrl	String	String
Tags	Contains list of string	Contains list of string
category	String	N/A
protocols	String	N/A
serviceEndpoint	String	N/A
version	String	N/A
wsdl	String	N/A
Data Formats	String	N/A

apigroups	String	N/A
example	List of String	N/A
clientInstall	String	N/A
authentication	String	N/A
ssl	String	N/A
readonly	String	N/A
VendorApiKits	String	N/A
CommunityApiKits	String	N/A
blog	String	N/A
forum	String	N/A
support	String	N/A
accountReq	String	N/A
commercial	String	N/A
provider	String	N/A
managedBy	String	N/A
nonCommercial	String	N/A
dataLiscensing	String	N/A
fees	String	N/A
limits	String	N/A
terms	String	N/A
company	String	N/A
updated	Date	N/A
Apis	N/A	List of apis further contains apiname and url(as explained above)

Following is the screenshot of the database data structure for apis records data, in the mongodb shell



Following is the screenshot of the database data structure for mashup data, in the mongodb shell



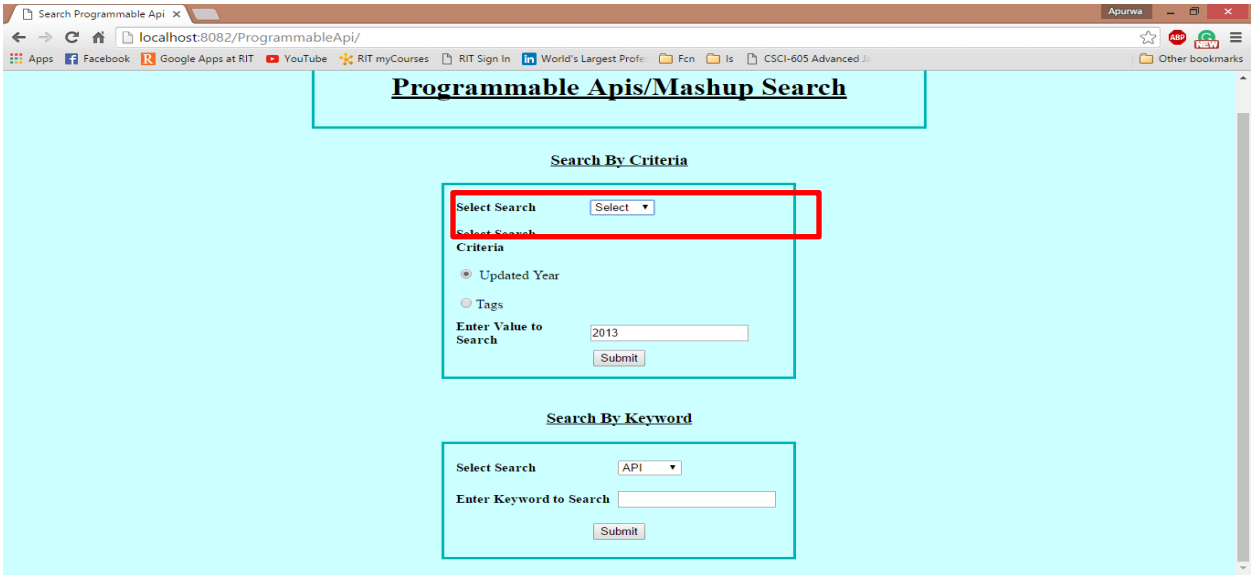
```
> db.mashup.findOne()
{
        "_id" : ObjectId("571961ef7bebd73c1d3f93b3"),
        "id" : "http://www.programmableweb.com/mashup/-22",
        "title" : null,
        "summary" : null,
        "rating" : 3.799999952316284,
        "name" : null,
        "label" : null,
        "author" : "Unknown",
        "description" : null,
        "type" : null,
        "downloads" : "0",
        "useCount" : 0,
        "sampleUrl" : "http://www.easypeasyphotos.net",
        "dateModified" : ISODate("2011-10-09T18:35:06Z"),
        "numComments" : 0,
        "commentsUrl" : "http://api.programmableweb.com/mashups/-22/comments",
        "tags" : [
                ""
        ],
        "Apis" : [
                {
                        "api" : {
                                "apiName" : "Flickr",
                                "apiUrl" : "http://www.programmableweb.com/api/f
lickr"
                        }
                },
                {
                        "api" : {
                                "apiName" : "Google Maps",
                                "apiUrl" : "http://www.programmableweb.com/api/g
oogle-maps"
                        }
                }
        ],
        "updated" : ISODate("2011-10-09T18:35:06Z")
}
```

Testing scenarios

Testing Scenarios for Api Records

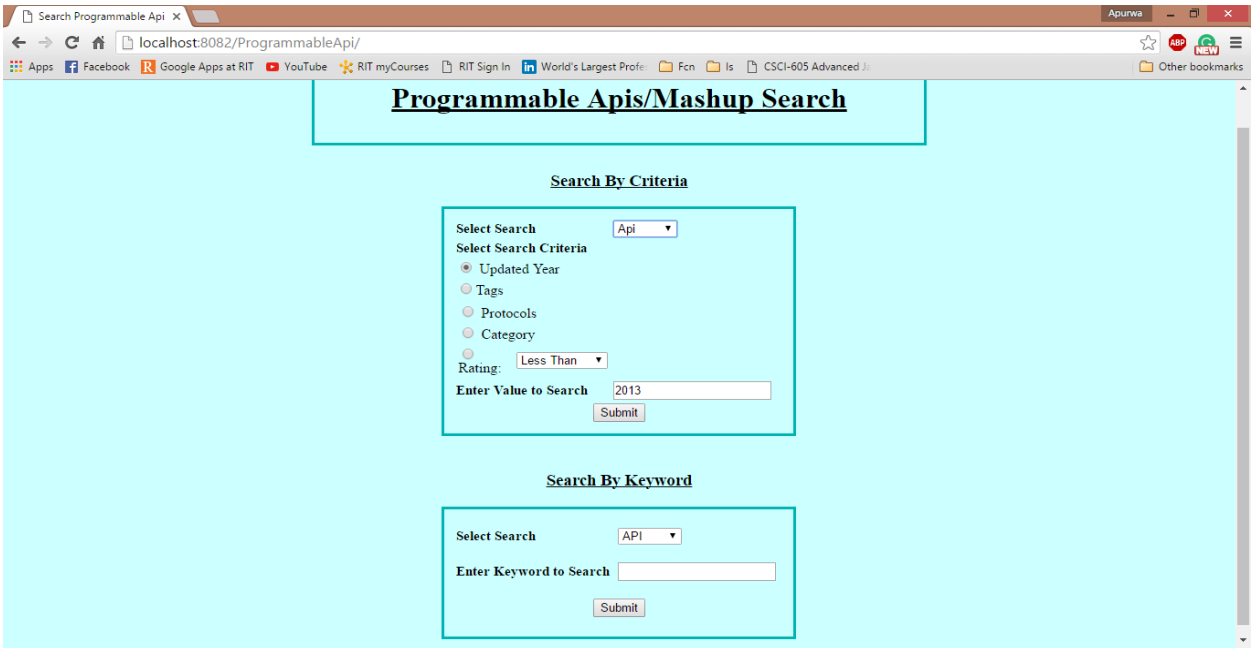
When the application is started,

Initially the options to choose are as follows, which are common to both api and mashup search

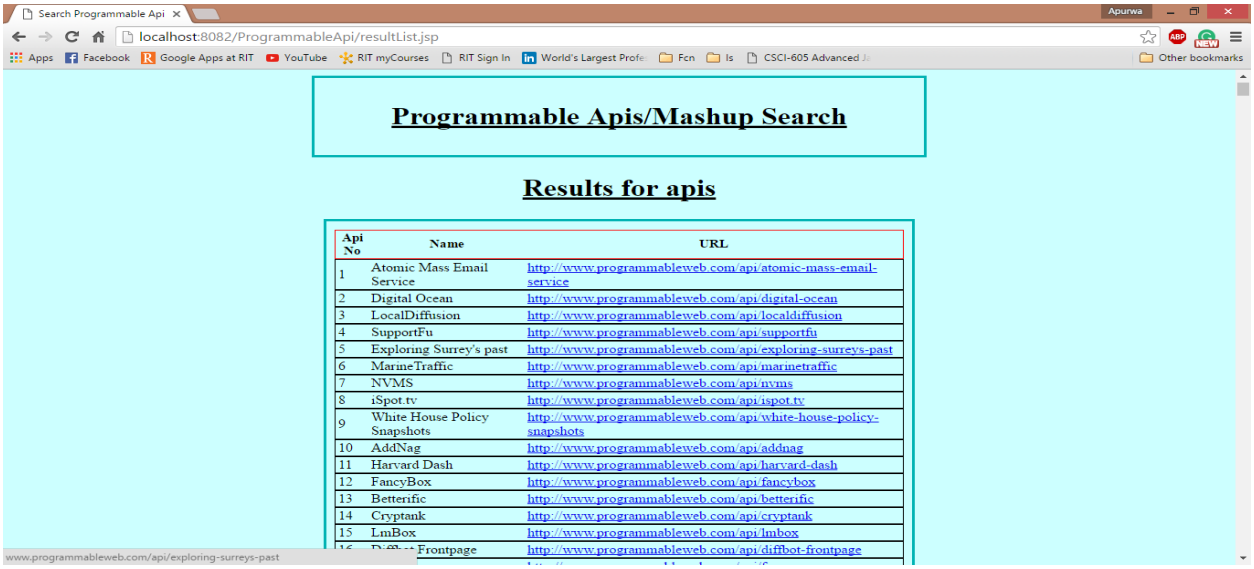


When you select Api in drop down you will get api specific options as follows,

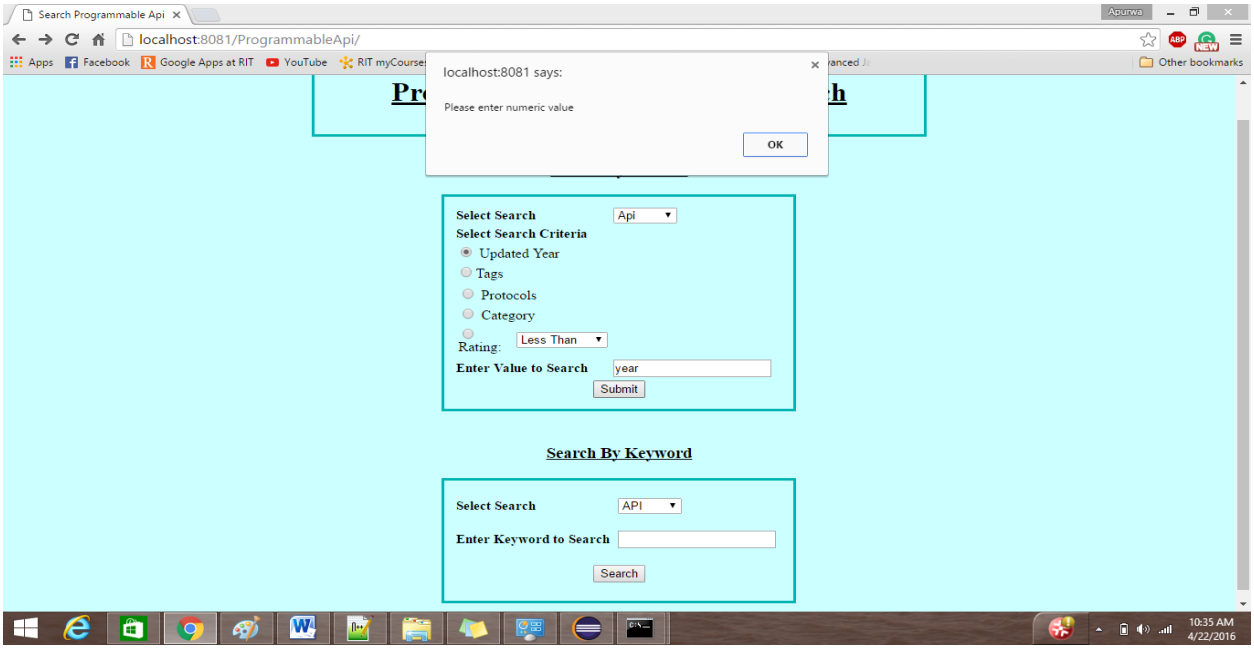
Now let's search for updated year 2013, for that you will have to click on desired radio button to select the criteria which is Updated year as shown below



Results for the updated year 2013 is as follows,

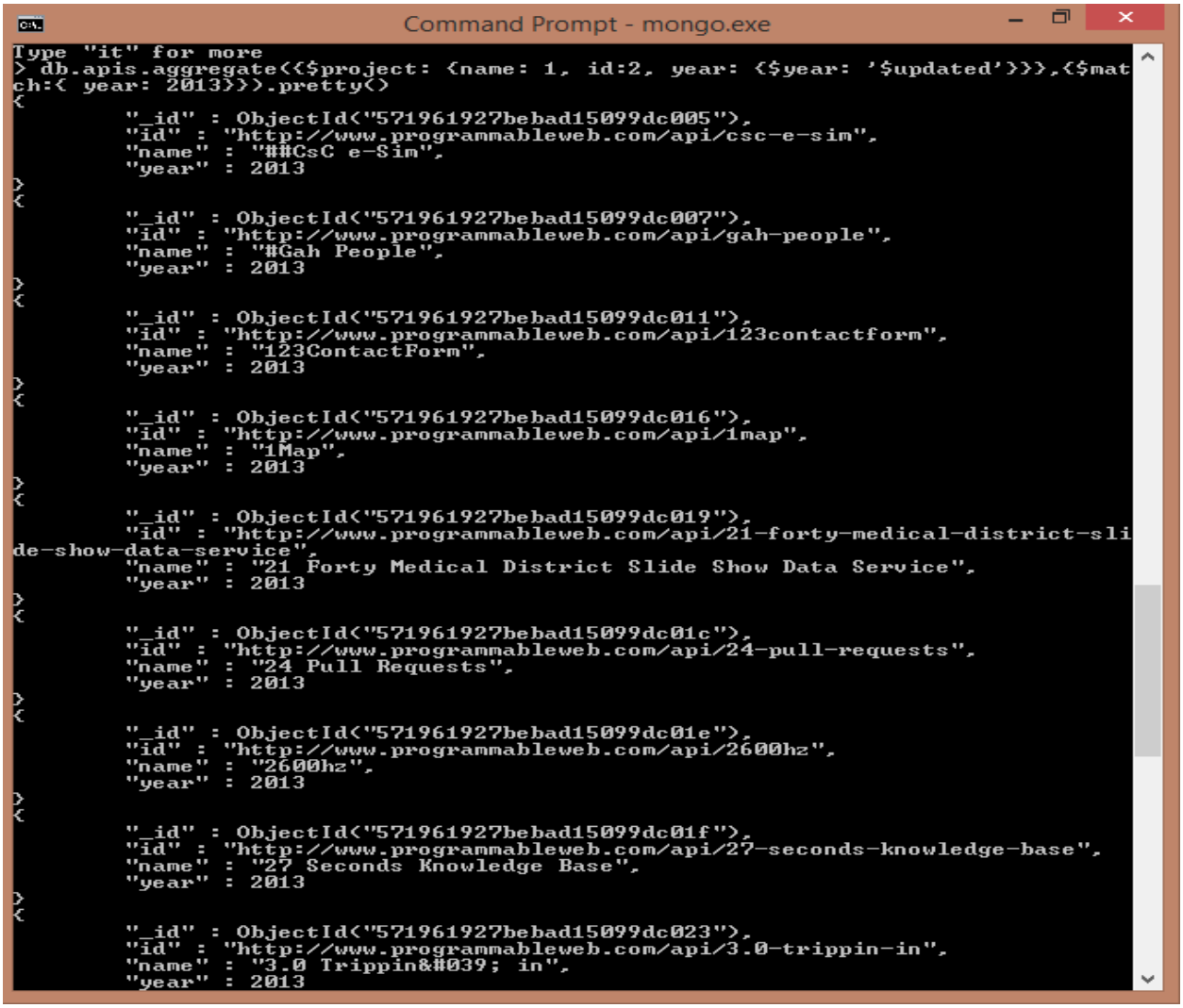


In case the user enters non numeric value for year, the application validates and throws an error "Please enter numeric value"

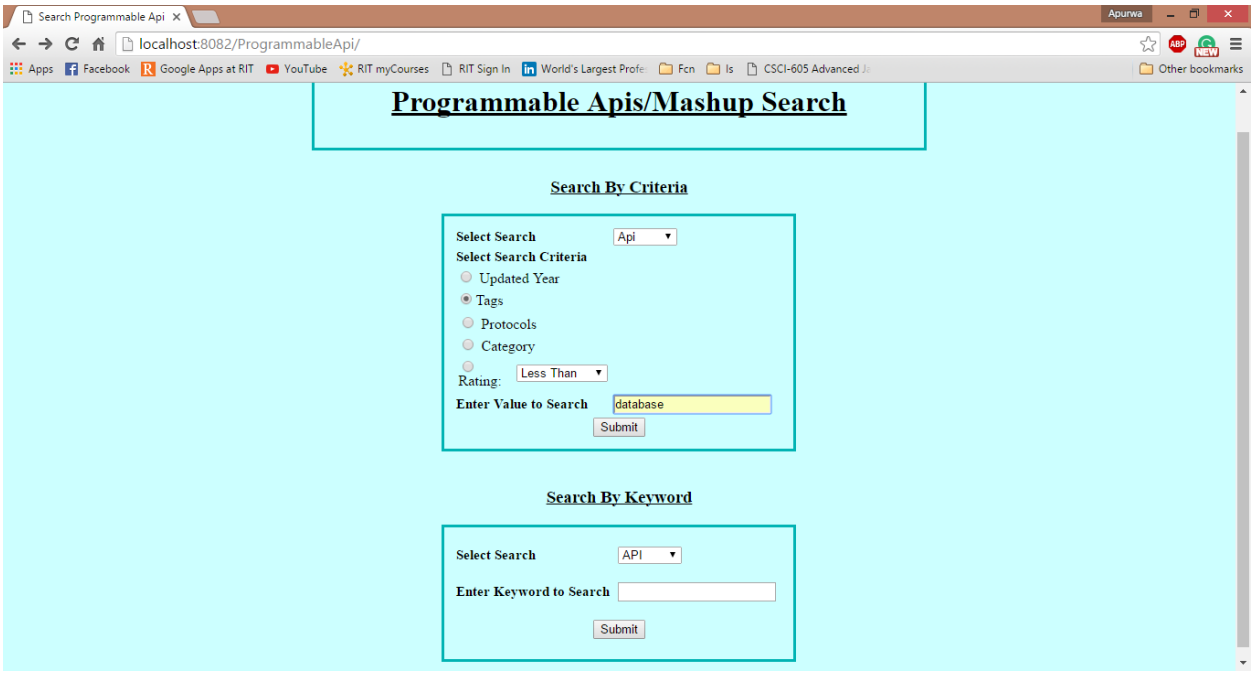


Results For Query to search according to the updated year: in the mongodb shell

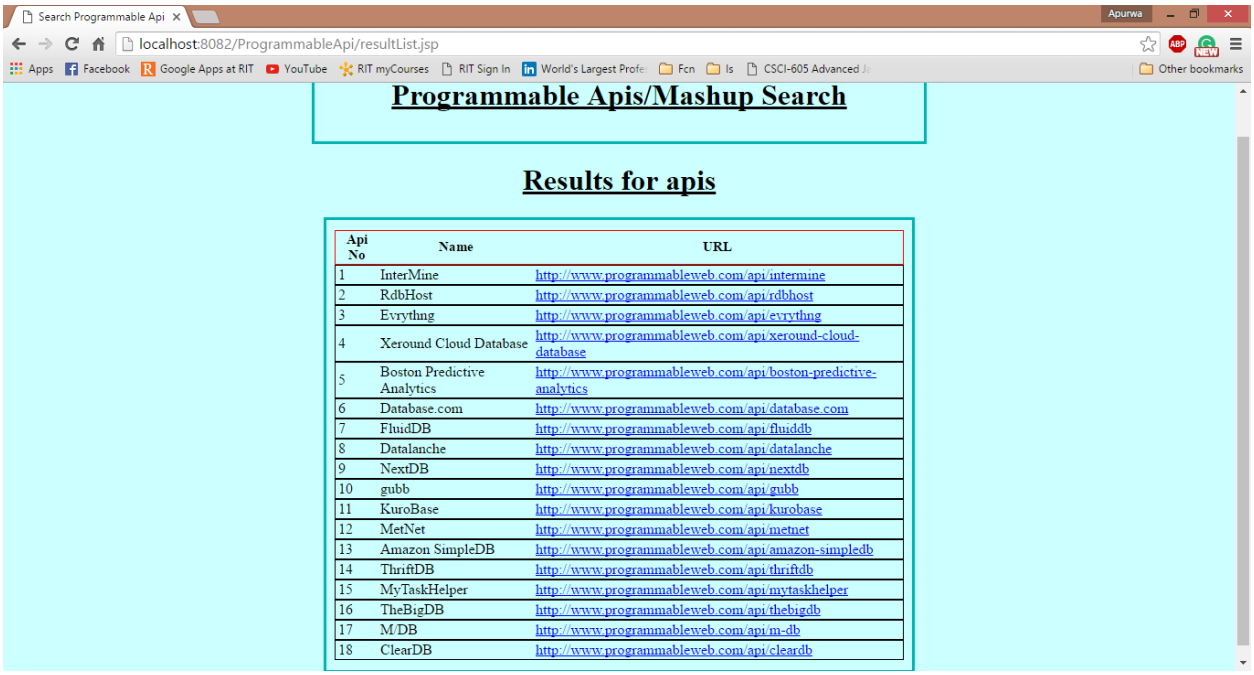
db.apis.aggregate({$project: {name: 1, id:2, year: {$year: '$updated'}}},{$match:{ year: 2013}});



Search For tags: database select tags as a criteria and enter database in the text field



The results are



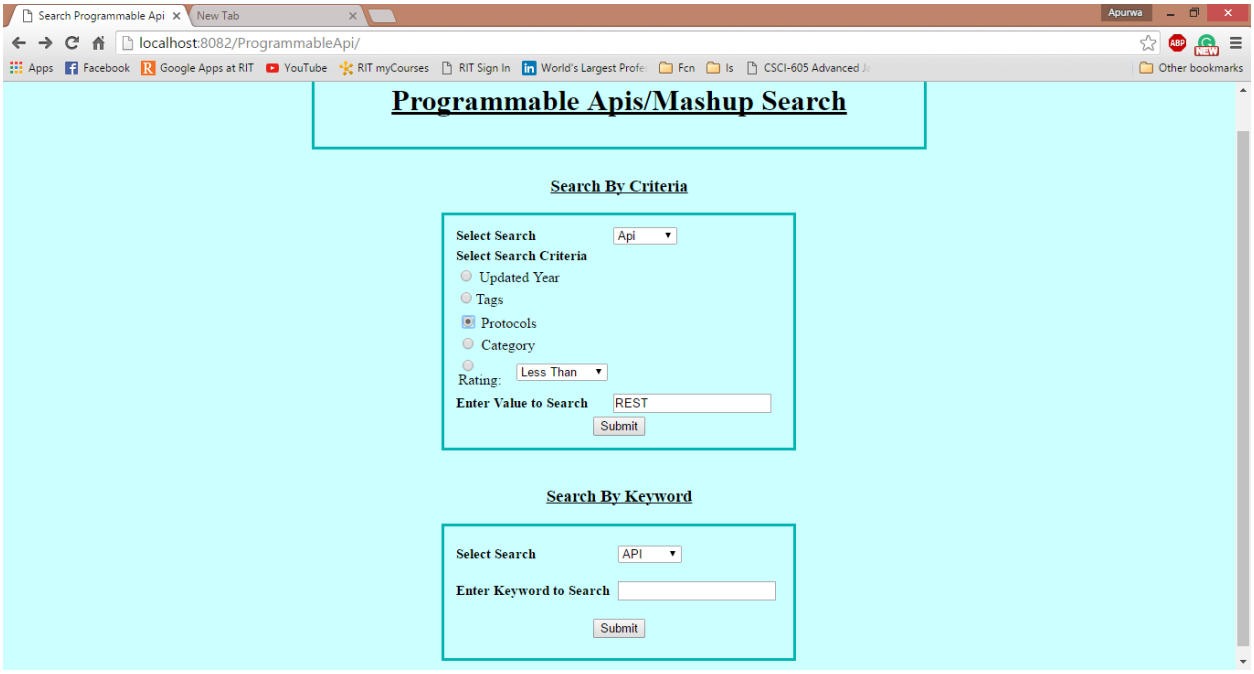
Api No	Name	URL
1	InterMine	http://www.programmableweb.com/api/intermine
2	RdbHost	http://www.programmableweb.com/api/rdbhost
3	Evrything	http://www.programmableweb.com/api/evrything
4	Xeround Cloud Database	http://www.programmableweb.com/api/xeround-cloud-database
5	Boston Predictive Analytics	http://www.programmableweb.com/api/boston-predictive-analytics
6	Database.com	http://www.programmableweb.com/api/database.com
7	FluidDB	http://www.programmableweb.com/api/fluiddb
8	Datalanche	http://www.programmableweb.com/api/datalanche
9	NextDB	http://www.programmableweb.com/api/nextdb
10	gubb	http://www.programmableweb.com/api/gubb
11	KuroBase	http://www.programmableweb.com/api/kurobase
12	MetNet	http://www.programmableweb.com/api/metnet
13	Amazon SimpleDB	http://www.programmableweb.com/api/amazon-simpledb
14	ThriftDB	http://www.programmableweb.com/api/thriftdb
15	MyTaskHelper	http://www.programmableweb.com/api/mytaskhelper
16	TheBigDB	http://www.programmableweb.com/api/thebigdb
17	M/DB	http://www.programmableweb.com/api/m-db
18	ClearDB	http://www.programmableweb.com/api/cleardb

The query to find the database tag in the mongodb:

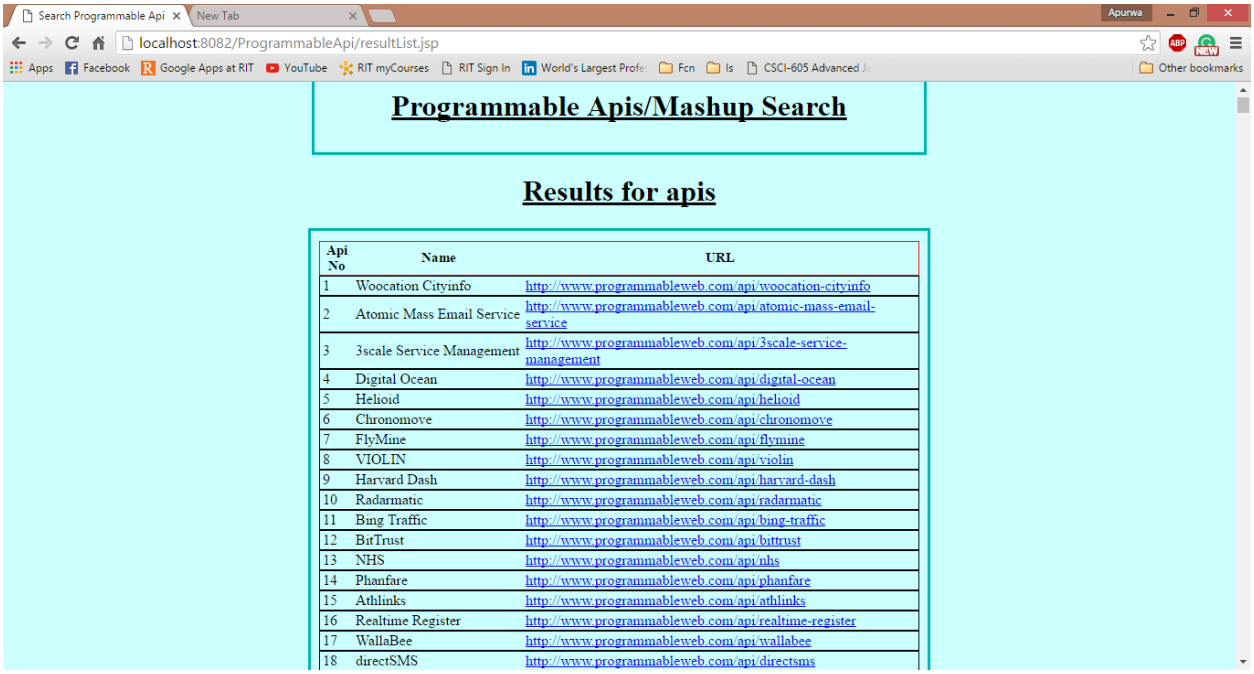
db.apis.find({ "Tags" : { "$regex" : "database" , "$options" : "i"}})

Result:

Search for a protocol for REST, select a search criteria protocols from the radiobutton



Results are as follows



Api No	Name	URL
1	Woocation Cityinfo	http://www.programmableweb.com/api/woocation-cityinfo
2	Atomic Mass Email Service	http://www.programmableweb.com/api/atomic-mass-email-service
3	3scale Service Management	http://www.programmableweb.com/api/3scale-service-management
4	Digital Ocean	http://www.programmableweb.com/api/digital-ocean
5	Helioid	http://www.programmableweb.com/api/helioid
6	Chronomove	http://www.programmableweb.com/api/chronomove
7	FlyMine	http://www.programmableweb.com/api/flymine
8	VIOLIN	http://www.programmableweb.com/api/violin
9	Harvard Dash	http://www.programmableweb.com/api/harvard-dash
10	Radarmatic	http://www.programmableweb.com/api/radarmatic
11	Bing Traffic	http://www.programmableweb.com/api/bing-traffic
12	BitTrust	http://www.programmableweb.com/api/bittrust
13	NHS	http://www.programmableweb.com/api/nhs
14	Phanfare	http://www.programmableweb.com/api/phanfare
15	Athlinks	http://www.programmableweb.com/api/athlinks
16	Realtime Register	http://www.programmableweb.com/api/realtime-register
17	WallaBee	http://www.programmableweb.com/api/wallabee
18	directSMS	http://www.programmableweb.com/api/directsms

Query to find protocols REST in protocols field, **in the mongodb shell**

db.apis.find({ "protocols" : { "$regex" : "REST" , "$options" : "i"}})

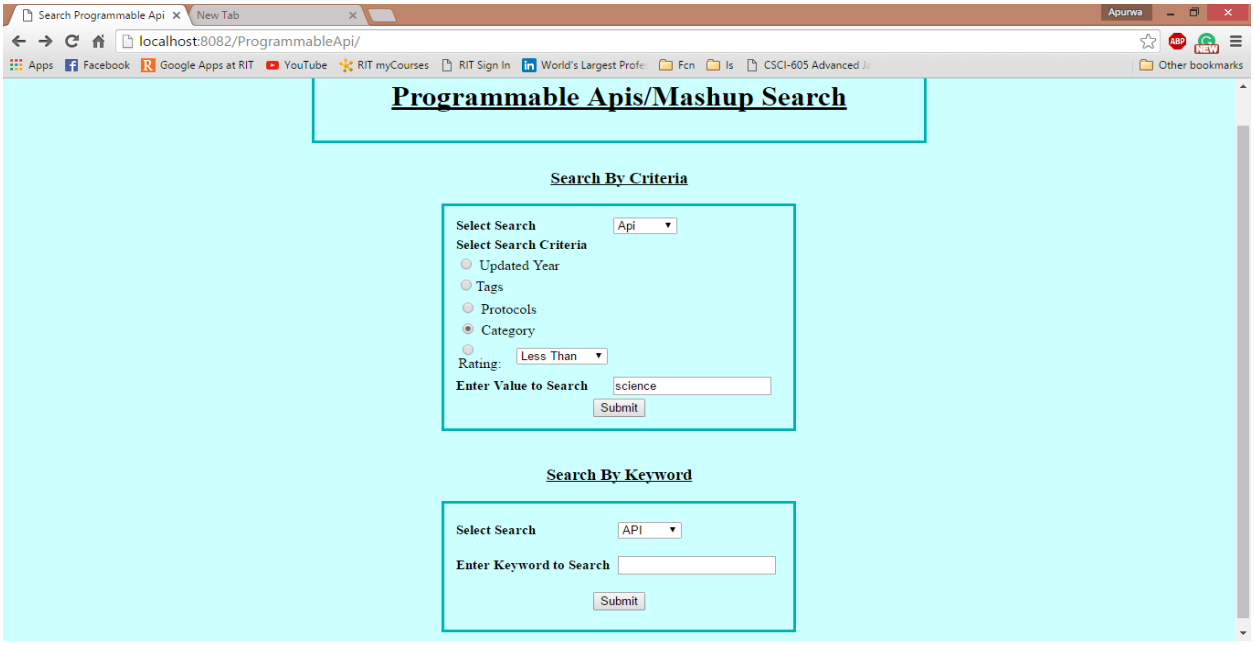
```
}
Type "it" for more
> db.apis.find({ "protocols" : { "$regex" : "REST" , "$options" : "i"}}).pretty(
)
```

Result:

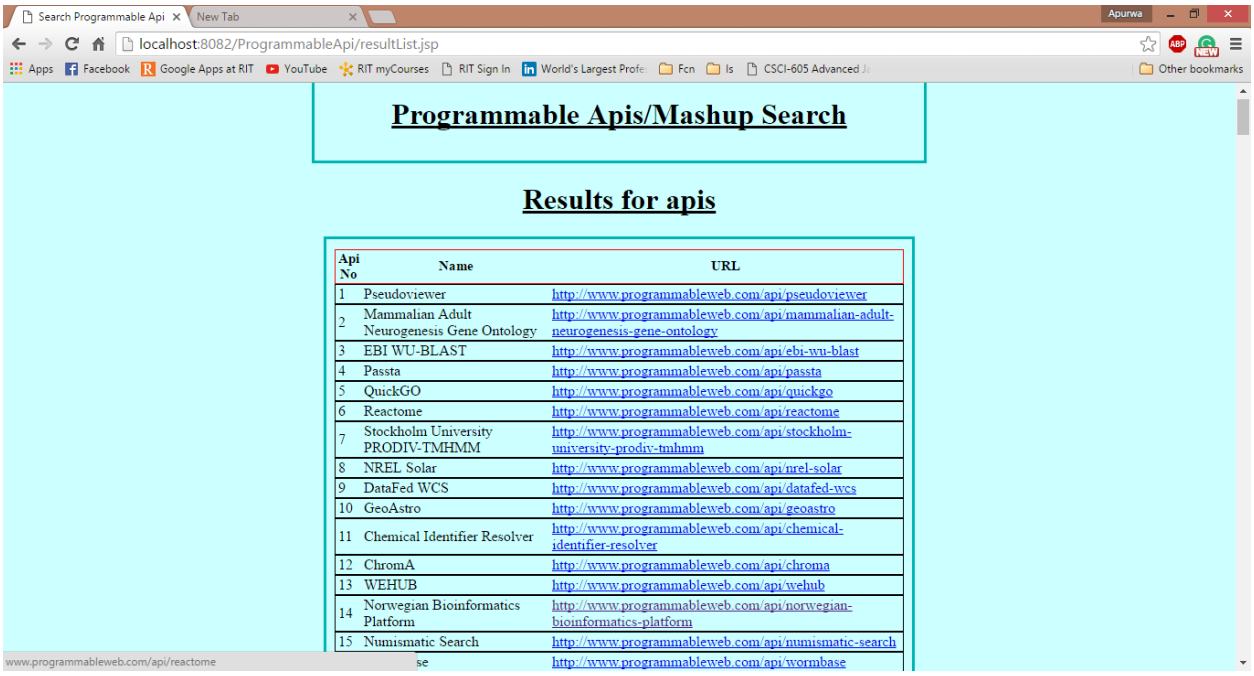
Command Prompt - mongo.exe


```
        "_id" : ObjectId("571961927bebad15099dc022"),
        "id" : "http://www.programmableweb.com/api/2sms",
        "title" : "2sms",
        "summary" : "SMS messaging service",
        "rating" : 3.5,
        "name" : "2sms",
        "label" : "2sms",
        "author" : null,
        "description" : "2sms provides text messaging services for businesses of
 all sizes. Services include one way and two way messaging, pager replacement, l
ong messaging, reporting and more. 2sms offers an API that allows developers to
integrate messaging services into their applications. The API exposes the SMS ga
teway and is available as in both RESTful and SOAP versions.",
        "type" : 1,
        "downloads" : null,
        "useCount" : null,
        "sampleUrl" : "http://www.2sms.com/software.aspx",
        "downloadUrl" : null,
        "dateModified" : ISODate("2012-09-19T11:37:57Z"),
        "remoteFeed" : null,
        "numComments" : null,
        "commentsUrl" : "http://api.programmableweb.com/apis/2sms/comments",
        "Tags" : [
                "sms"
        ],
        "category" : "Messaging",
        "protocols" : "REST, SOAP",
        "serviceEndpoint" : null,
        "version" : null,
        "wsdl" : null,
        "data Format" : "XML",
        "apigroups" : null,
        "example" : [
                "messaging",
                "sms"
        ],
        "clientInstall" : null,
        "authentication" : null,
        "ssl" : null,
        "readonly" : null,
        "VendorApiKits" : null,
        "CommunityApiKits" : null,
        "blog" : null,
        "forum" : null,
        "support" : null,
        "accountReq" : null,
        "commercial" : null,
        "provider" : "http://www.2sms.com/",
        "managedBy" : null,
        "nonCommercial" : null,
        "dataLicensing" : null,
        "fees" : null,
        "limits" : null,
        "terms" : null,
        "company" : null,
        "updated" : ISODate("2012-09-19T11:37:57Z")
}
```

Search for a category **science :** select criteria category and enter science in the text box



Results are



Query to search Category , **in the mongodb shell**

```
}
Type "it" for more
> db.apis.find({ "category" :  { "$regex" : "science" , "$options" : "i"}}).pret
ty()
```

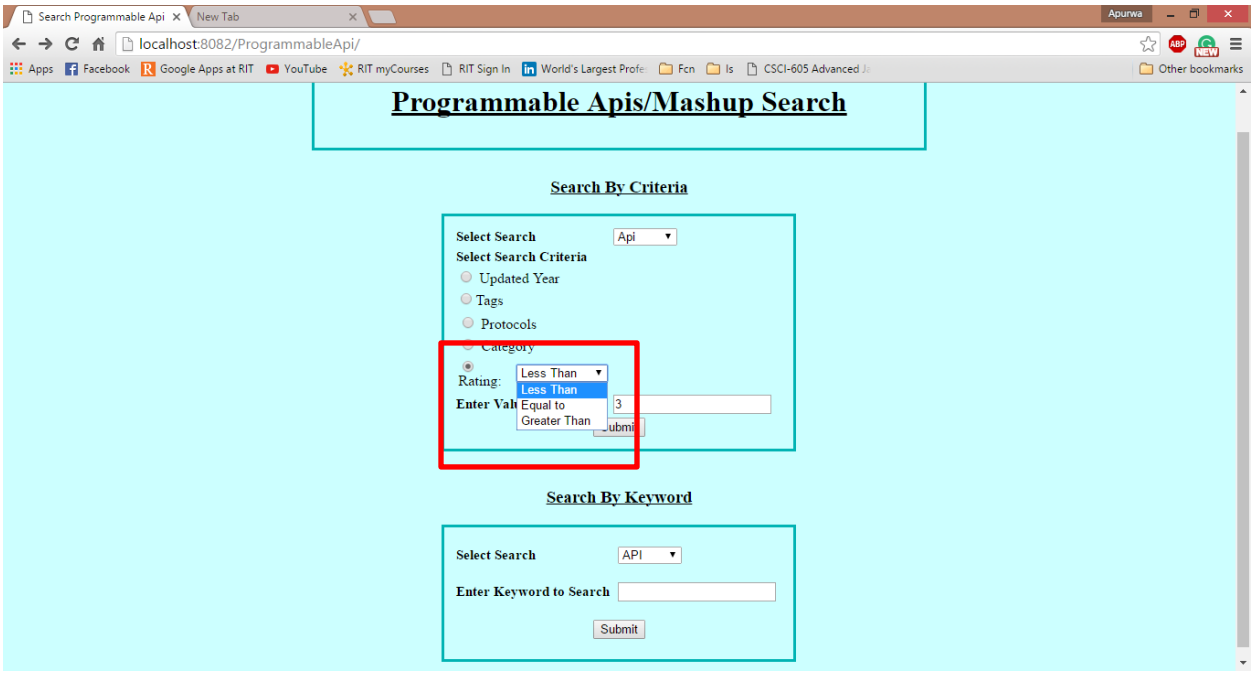
Results:

Command Prompt - mongo.exe

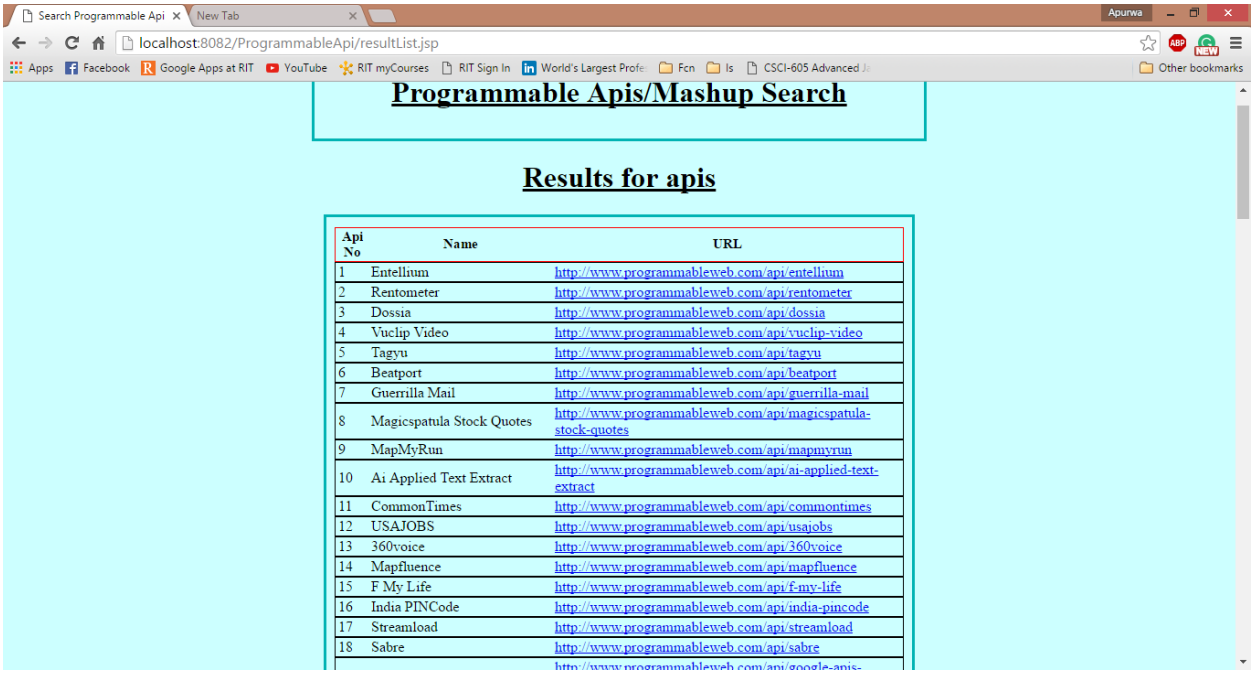

```
        "name" : "BIOBASE",
        "label" : "BIOBASE",
        "author" : null,
        "description" : "BIOBASE is the leading provider of expert-curated biolo
gical databases, software and services for the life sciences. BIOBASE products a
re used to identify relations that aid in drug and biomarker discovery. BIOBASE
helps researchers to identify connections by offering well-structured data, asse
mbled by highly qualified subject-matter experts, organized in an accessible and
 easily searchable manner. This data is accessible via an API. The API uses REST
ful calls and responses are formatted in XML.",
        "type" : 1,
        "downloads" : null,
        "useCount" : null,
        "sampleUrl" : "http://www.biobase-international.com/solutions-for/bioinf
ormaticians",
        "downloadUrl" : null,
        "dateModified" : ISODate("2012-06-11T14:36:37Z"),
        "remoteFeed" : null,
        "numComments" : null,
        "commentsUrl" : "http://api.programmableweb.com/apis/biobase/comments",
        "Tags" : [
                "science"
        ],
        "category" : "Science",
        "protocols" : "REST",
        "serviceEndpoint" : null,
        "version" : null,
        "wsdl" : null,
        "data Format" : "XML",
        "apigroups" : null,
        "example" : [
                "biology",
                "reference",
                "research",
                "science"
        ],
        "clientInstall" : null,
        "authentication" : null,
        "ssl" : null,
        "readonly" : null,
        "VendorApiKits" : null,
        "CommunityApiKits" : null,
        "blog" : null,
        "forum" : null,
        "support" : null,
        "accountReq" : null,
        "commercial" : null,
        "provider" : "http://www.biobase-international.com/",
        "managedBy" : null,
        "nonCommercial" : null,
        "dataLicensing" : null,
        "fees" : null,
        "limits" : null,
        "terms" : null,
        "company" : null,
        "updated" : ISODate("2012-06-11T14:36:37Z")
}
```

Now lets search for all the apis having rating less than 3

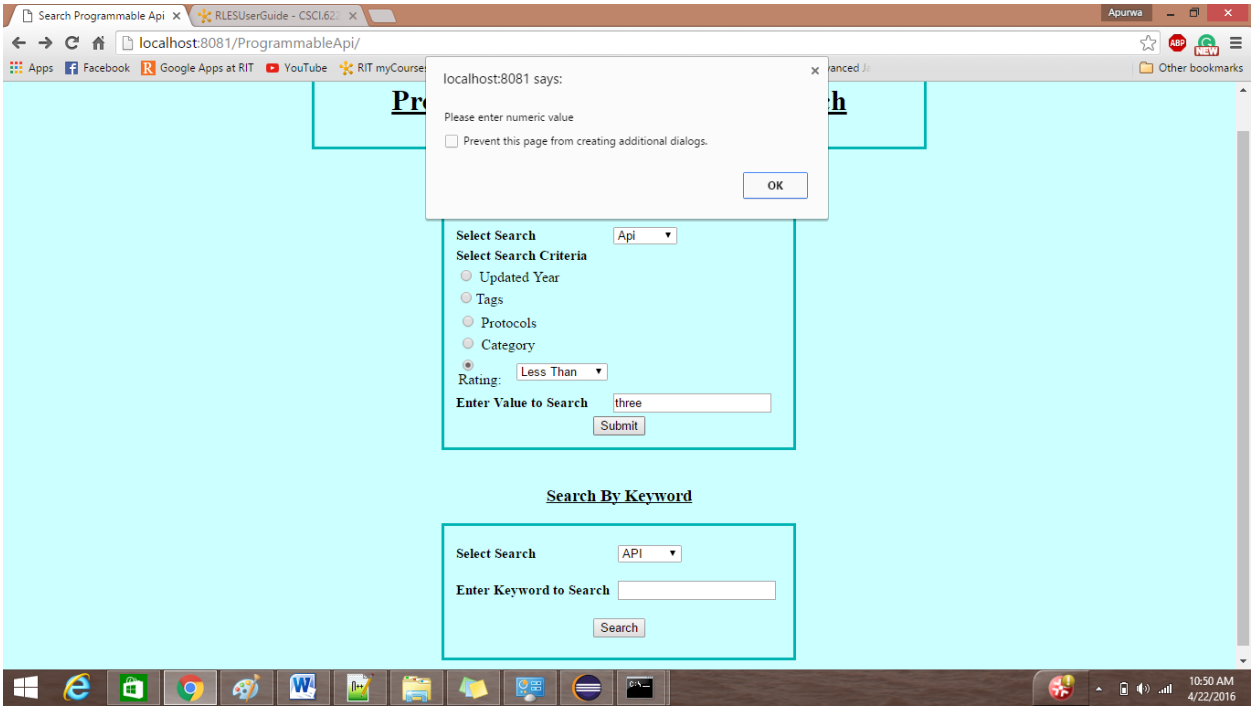
As you can see when you click on rating you will have to select one of the following option less than, equal to , greater than values and the results will be shown accordingly



The results are as follows,



The application throws an error if the user enters non numeric value for rating chosen,"Please enter numeric value"



Query to search for api having rating 3 is as follows,

db.apis.find({ "rating" : { "$lt" : 3.0}}).pretty()

```
>
Type "it" for more
> db.apis.find({ "rating" : { "$lt" : 3.0}}).pretty()
```

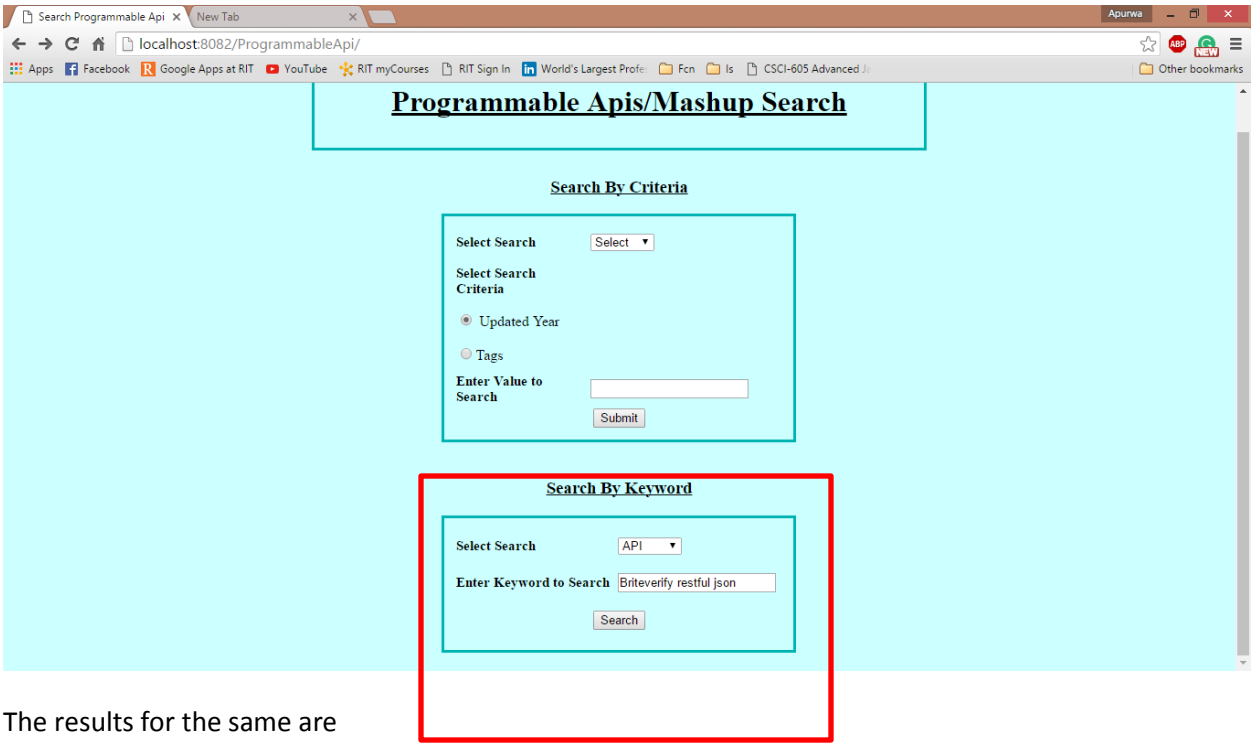
Results are:

Command Prompt - mongo.exe

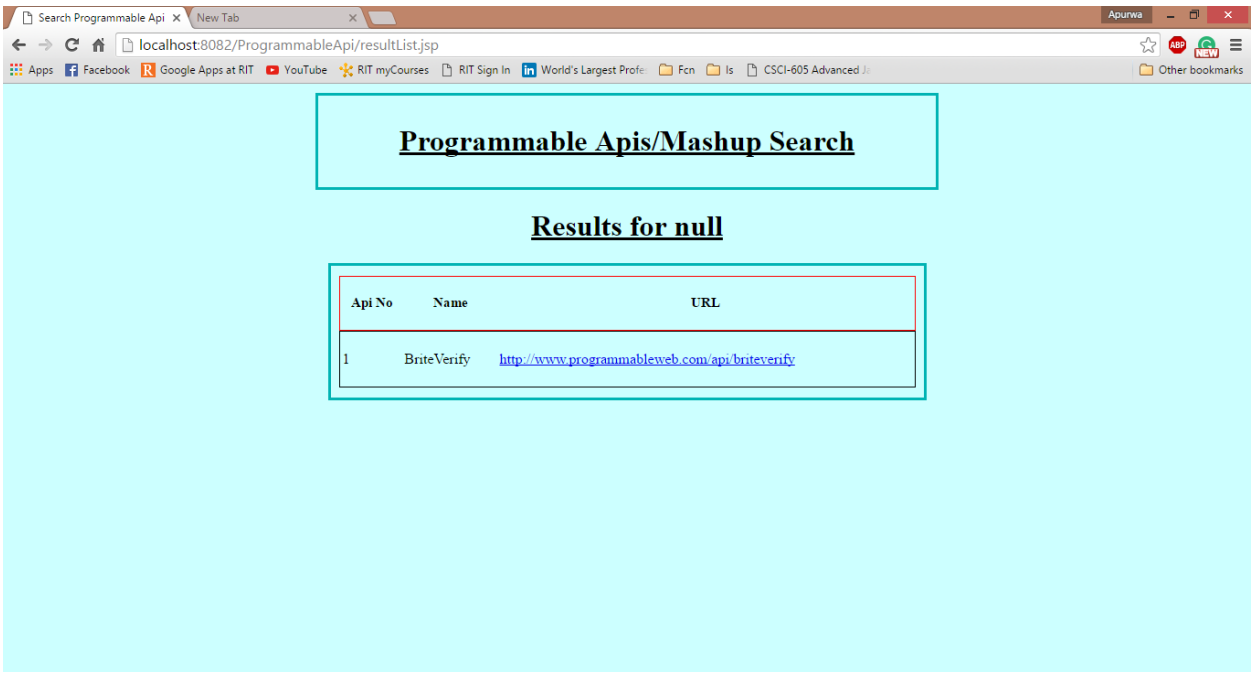

```
        "_id" : ObjectId("571961957bebad15099dc54e"),
        "id" : "http://www.programmableweb.com/api/cafe-press",
        "title" : "Cafe Press",
        "summary" : "Customized retail product service",
        "rating" : 2.700000047683716,
        "name" : "Cafe Press",
        "label" : "Cafe Press",
        "author" : null,
        "description" : "Allows developers to make queries and run searches agai
nst Cafe Press&amp;amp;#039; store and product database.",
        "type" : 1,
        "downloads" : null,
        "useCount" : null,
        "sampleUrl" : "http://api.cafepress.com/",
        "downloadUrl" : null,
        "dateModified" : ISODate("2006-03-06T15:02:38Z"),
        "remoteFeed" : null,
        "numComments" : null,
        "commentsUrl" : "http://api.programmableweb.com/apis/cafe-press/comments
",
        "Tags" : [
                "shopping"
        ],
        "category" : "Shopping",
        "protocols" : "SOAP, REST",
        "serviceEndpoint" : "http://api.cafepress.com/",
        "version" : null,
        "wsdl" : "http://api.cafepress.com/index.asmx?WSDL",
        "data Format" : "XML",
        "apigroups" : null,
        "example" : [
                "personalization",
                "shopping"
        ],
        "clientInstall" : "No",
        "authentication" : null,
        "ssl" : null,
        "readonly" : "No",
        "VendorApiKits" : null,
        "CommunityApiKits" : null,
        "blog" : null,
        "forum" : null,
        "support" : null,
        "accountReq" : null,
        "commercial" : null,
        "provider" : "http://www.cafepress.com",
        "managedBy" : "Mashery",
        "nonCommercial" : null,
        "dataLicensing" : null,
        "fees" : null,
        "limits" : null,
        "terms" : null,
        "company" : null,
        "updated" : ISODate("2006-03-06T15:02:38Z")
}
Type "it" for more
>
```

For the keyword search enter the keywords in the second form:

Let's enter "**Briteverify restful json** " keywords into the text box as shown below and select API from the dropdown options



The results for the same are

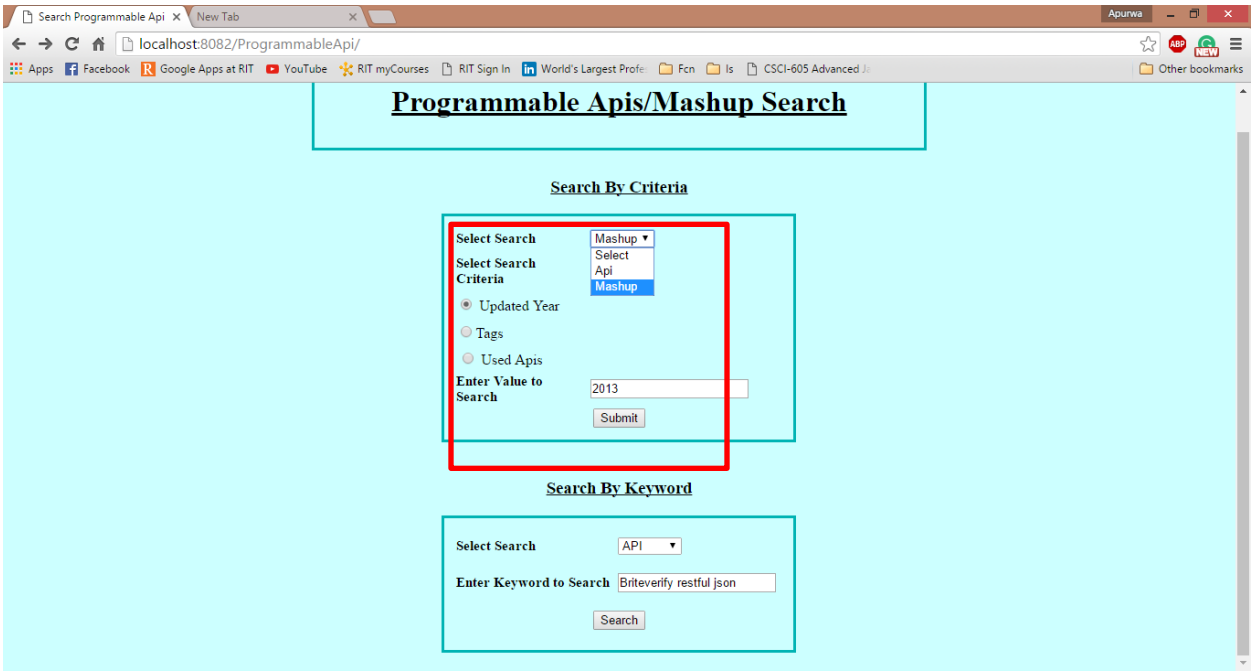


Query for the keyword search is as follows,

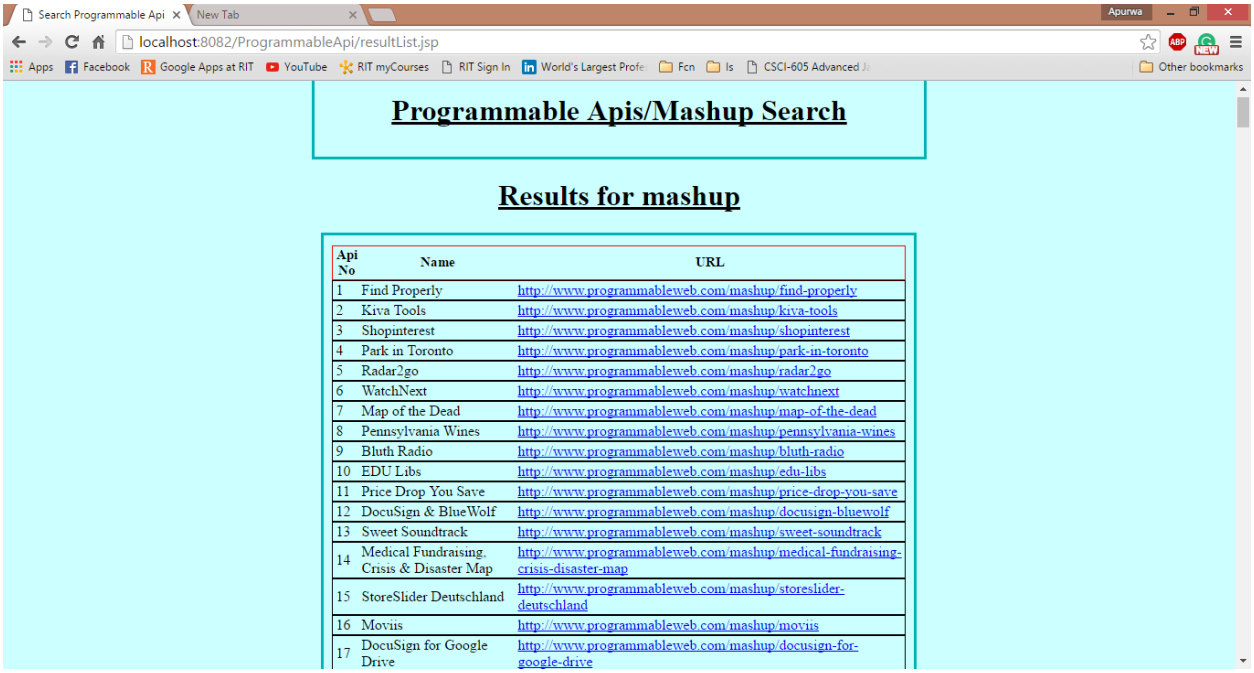
db.apis.find({ "$text" : { "$search" : "\"Briteverify\"\"restful\"\"json\""}})

Search For Mashup records:

Select Mashup from the drop down and select updatedyear and enter year value 2013



The results are as follows



Api No	Name	URL
1	Find Properly	http://www.programmableweb.com/mashup/find-properly
2	Kiva Tools	http://www.programmableweb.com/mashup/kiva-tools
3	Shopinterest	http://www.programmableweb.com/mashup/shopinterest
4	Park in Toronto	http://www.programmableweb.com/mashup/park-in-toronto
5	Radar2go	http://www.programmableweb.com/mashup/radar2go
6	WatchNext	http://www.programmableweb.com/mashup/watchnext
7	Map of the Dead	http://www.programmableweb.com/mashup/map-of-the-dead
8	Pennsylvania Wines	http://www.programmableweb.com/mashup/pennsylvania-wines
9	Bluth Radio	http://www.programmableweb.com/mashup/bluth-radio
10	EDU Libs	http://www.programmableweb.com/mashup/edu-libs
11	Price Drop You Save	http://www.programmableweb.com/mashup/price-drop-you-save
12	DocuSign & BlueWolf	http://www.programmableweb.com/mashup/docusign-bluewolf
13	Sweet Soundtrack	http://www.programmableweb.com/mashup/sweet-soundtrack
14	Medical Fundraising, Crisis & Disaster Map	http://www.programmableweb.com/mashup/medical-fundraising-crisis-disaster-map
15	StoreSlider Deutschland	http://www.programmableweb.com/mashup/storeslider-deutschland
16	Moviis	http://www.programmableweb.com/mashup/moviis
17	DocuSign for Google Drive	http://www.programmableweb.com/mashup/docusign-for-google-drive

The Query for searching api record according to updated year 2013 is

db.mashup.aggregate({$project: {name: 1, id:2, year: {$year: '$updated'}}},{$match:{ year: 2013}})

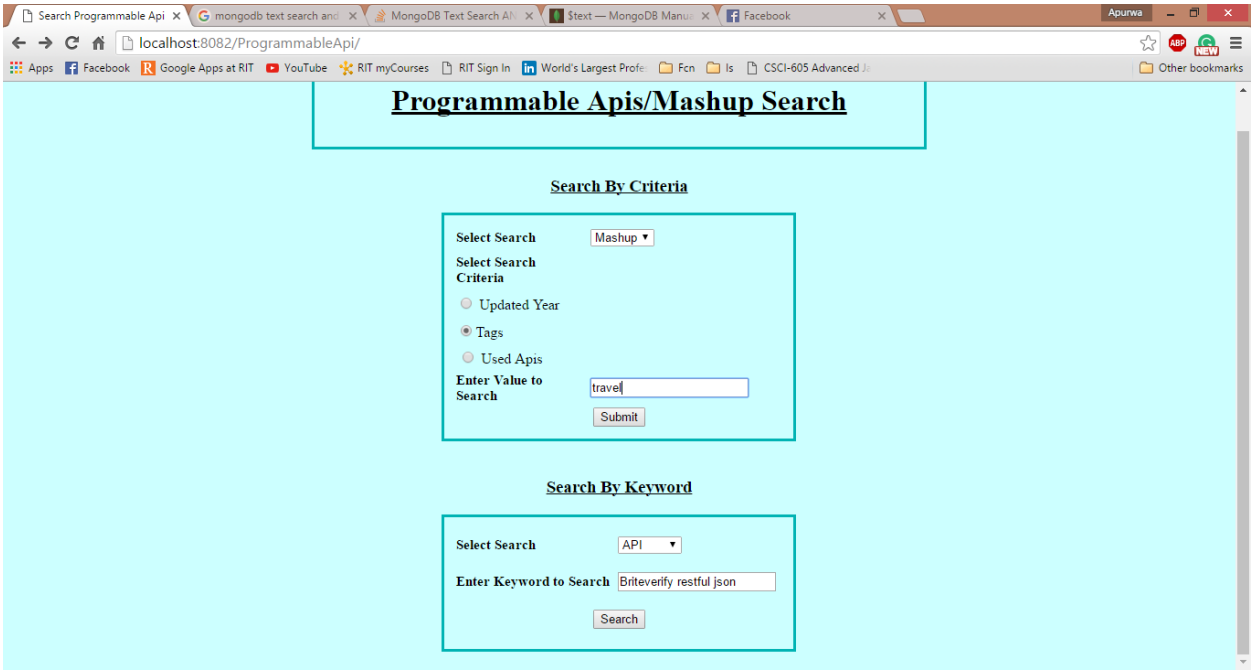
```
Type "it" for more
> db.mashup.aggregate({$project: {name: 1, id:2, year: {$year: '$updated'}}},{$m
atch:{ year: 2013}}).pretty()
```

Results are as follows,

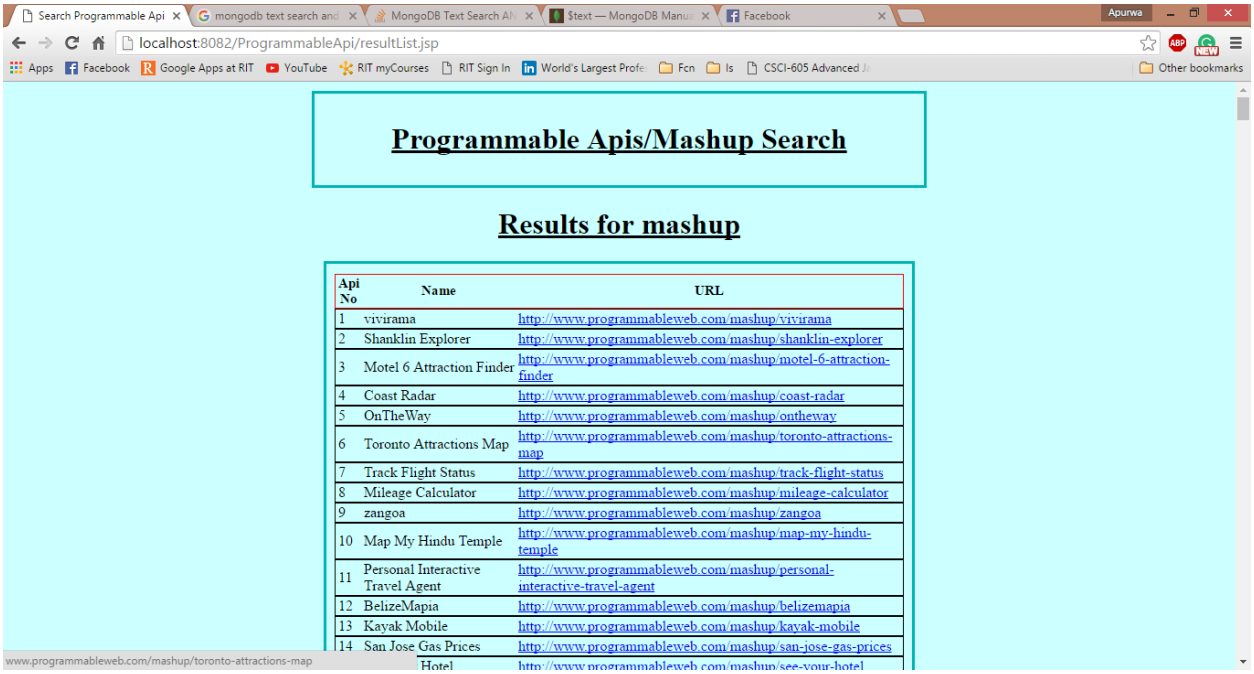
Command Prompt - mongo.exe


```
        "_id" : ObjectId("571961ef7bebd73c1d3f93b5"),
        "id" : "http://www.programmableweb.com/mashup/mobile-emulator",
        "name" : " Mobile Emulator",
        "year" : 2013
}
{
        "_id" : ObjectId("571961ef7bebd73c1d3f93b9"),
        "id" : "http://www.programmableweb.com/mashup/beermap-the-top-2500-beers
-on-twitter",
        "name" : "#BeerMap - The Top 2,500 Beers on Twitter",
        "year" : 2013
}
{
        "_id" : ObjectId("571961ef7bebd73c1d3f93bd"),
        "id" : "http://www.programmableweb.com/mashup/s42at",
        "name" : "#S42AT",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f93c7"),
        "id" : "http://www.programmableweb.com/mashup/100-destinations",
        "name" : "100 Destinations",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f9402"),
        "id" : "http://www.programmableweb.com/mashup/4wheelz-routemate",
        "name" : "4Wheelz RouteMate",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f9417"),
        "id" : "http://www.programmableweb.com/mashup/aanbieding",
        "name" : "Aanbieding",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f941a"),
        "id" : "http://www.programmableweb.com/mashup/about-pune-city",
        "name" : "About Pune City",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f941d"),
        "id" : "http://www.programmableweb.com/mashup/academic-ranking-of-world-
universities-shanghai-ranking",
        "name" : "Academic Ranking of World Universities (Shanghai Ranking)",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f9454"),
        "id" : "http://www.programmableweb.com/mashup/airlift-docusign",
        "name" : "Airlift &amp;amp; DocuSign",
        "year" : 2013
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f9469"),
```

Search For tags: select Tags as the criteria and enter text in the text box



The results are



Query to search records according to the travel as a tag is

db.mashup.find({ tags: { $in: ["travel"] } })

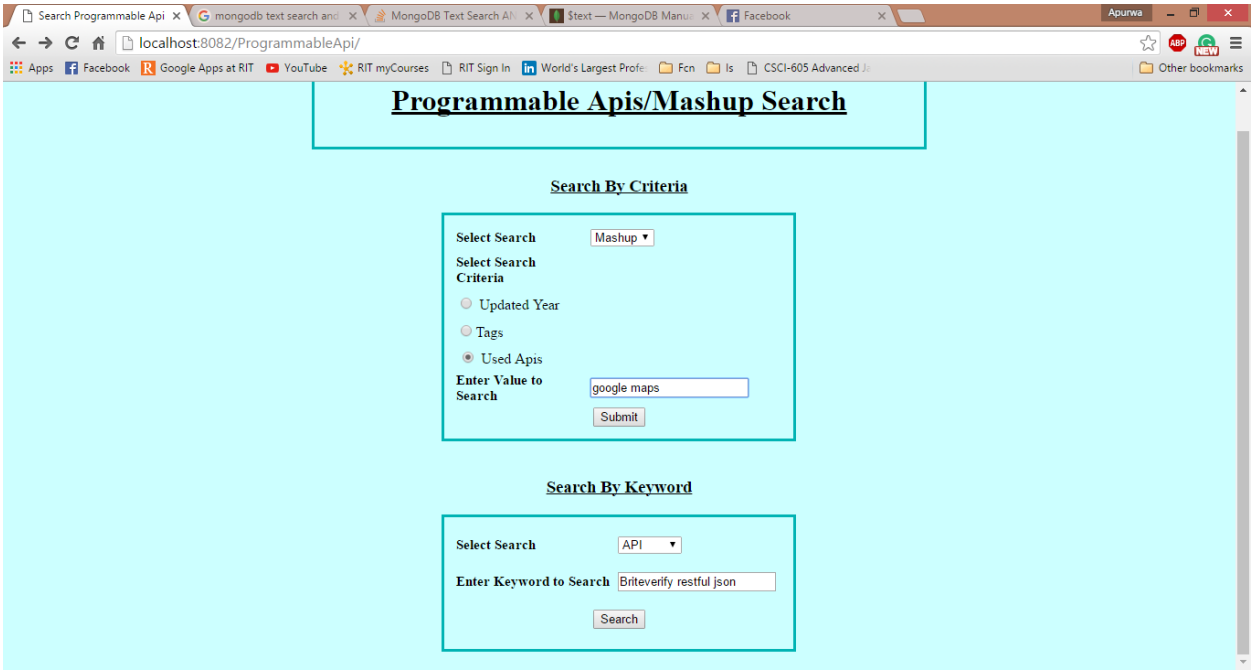
```
}
Type "it" for more
> db.mashup.find({ tags: { $in: [ "travel" ] } }).pretty()
```

Results are

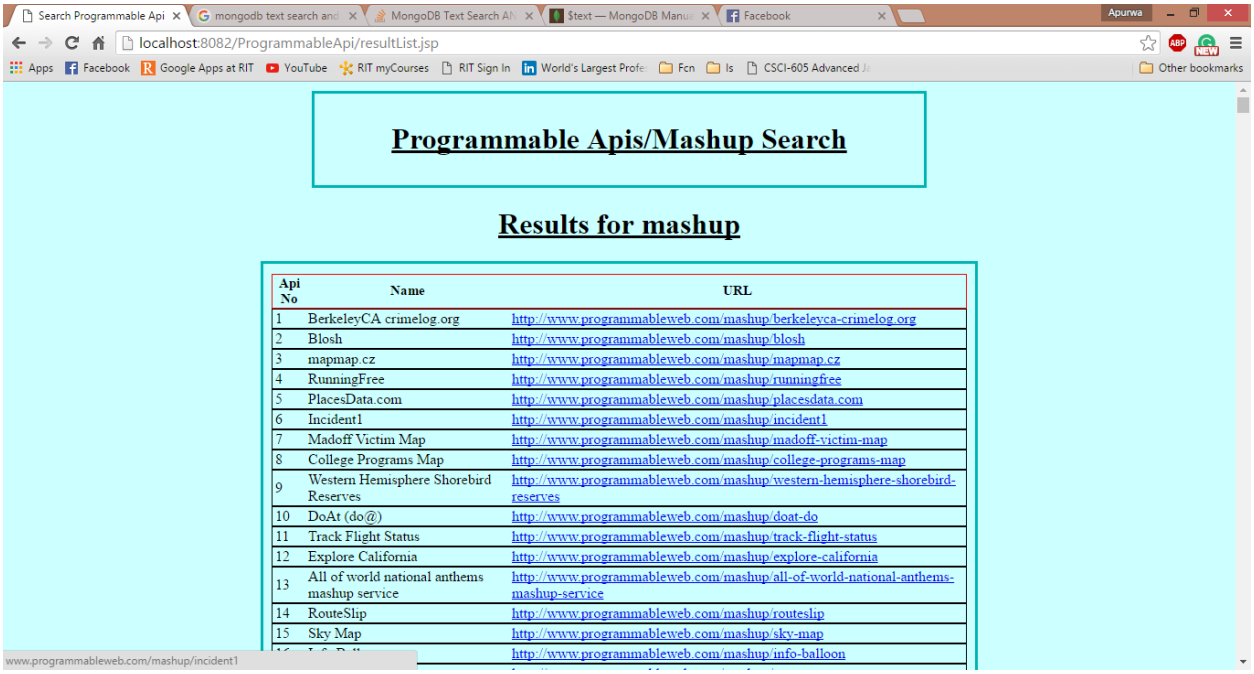
Command Prompt - mongo.exe


```
                "politics",
                "science",
                "travel"
        ],
        "Apis" : [
                {
                        "api" : {
                                "apiName" : "Google Maps",
                                "apiUrl" : "http://www.programmableweb.com/api/g
oogle-maps"
                        }
                }
        ],
        "updated" : ISODate("2006-07-19T01:18:15Z")
}
{
        "_id" : ObjectId("571961f07bebd73c1d3f9453"),
        "id" : "http://www.programmableweb.com/mashup/aircraft-flight-tracking-d
emo",
        "title" : "Aircraft Flight Tracking Demo",
        "summary" : "Demonstration that tracks a private aircraft anywhere in th
e world via  Iridium satellite network.",
        "rating" : 3.0999999046325684,
        "name" : "Aircraft Flight Tracking Demo",
        "label" : "Aircraft Flight Tracking Demo",
        "author" : "Unknown",
        "description" : "Demonstration that tracks a private aircraft anywhere i
n the world via  Iridium satellite network.",
        "type" : null,
        "downloads" : null,
        "useCount" : 6206,
        "sampleUrl" : "http://www.fsinsider.com/SpecialFeatures/BarringtonIrving
/barrington_map_route.aspx",
        "dateModified" : ISODate("2007-04-11T04:45:18Z"),
        "numComments" : 0,
        "commentsUrl" : "http://api.programmableweb.com/mashups/aircraft-flight-
tracking-demo/comments",
        "tags" : [
                "aviation",
                "deadpool",
                "gps",
                "mapping",
                "travel"
        ],
        "Apis" : [
                {
                        "api" : {
                                "apiName" : "Microsoft Bing Maps",
                                "apiUrl" : "http://www.programmableweb.com/api/m
icrosoft-bing-maps"
                        }
                }
        ],
        "updated" : ISODate("2007-04-11T04:45:18Z")
}
Type "it" for more
>
```

Seach based on Used apis select Used Api as the a criteria and enter name of the api in the text



Results

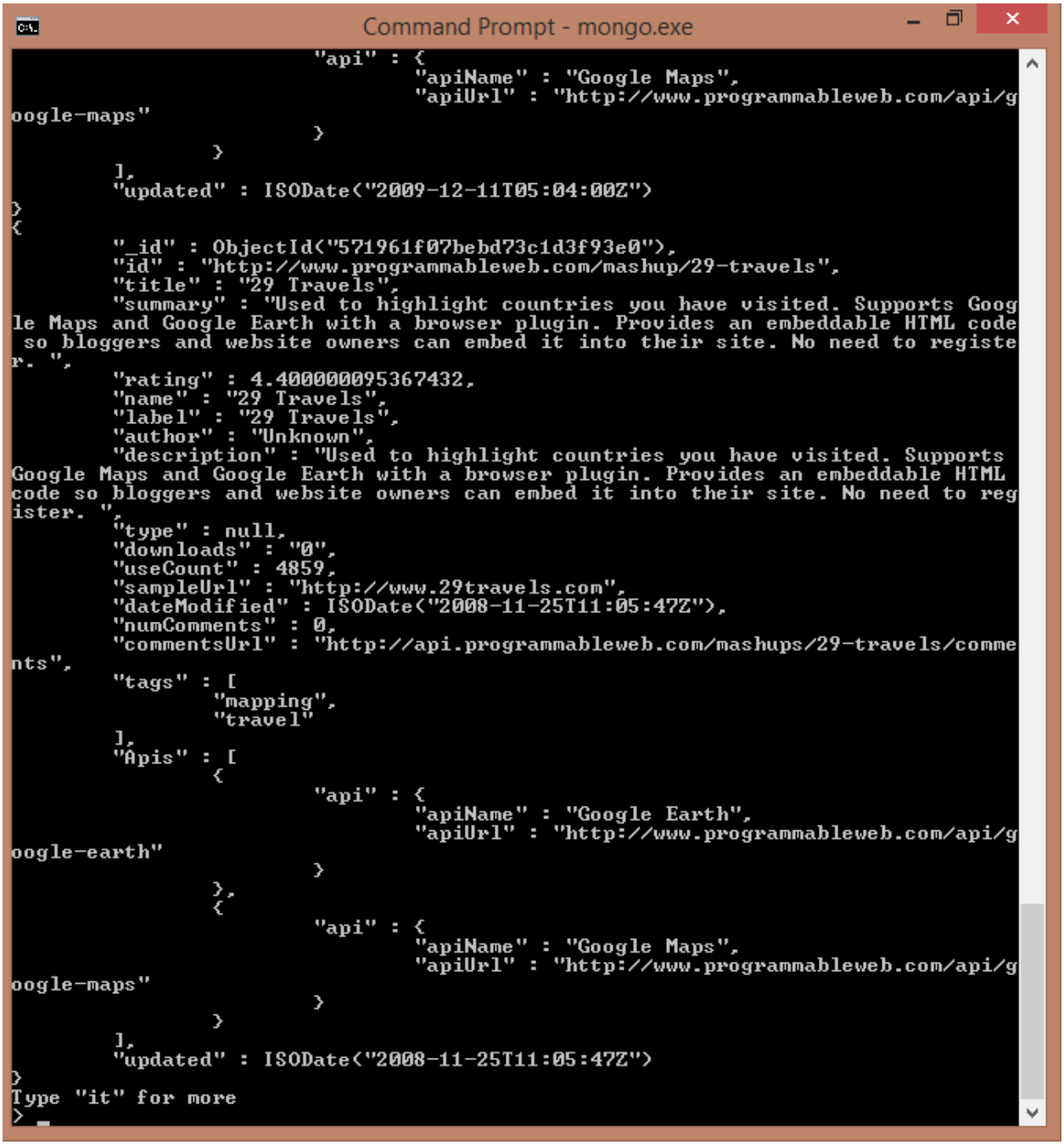


Query to search records having apiname as google maps is as follows,

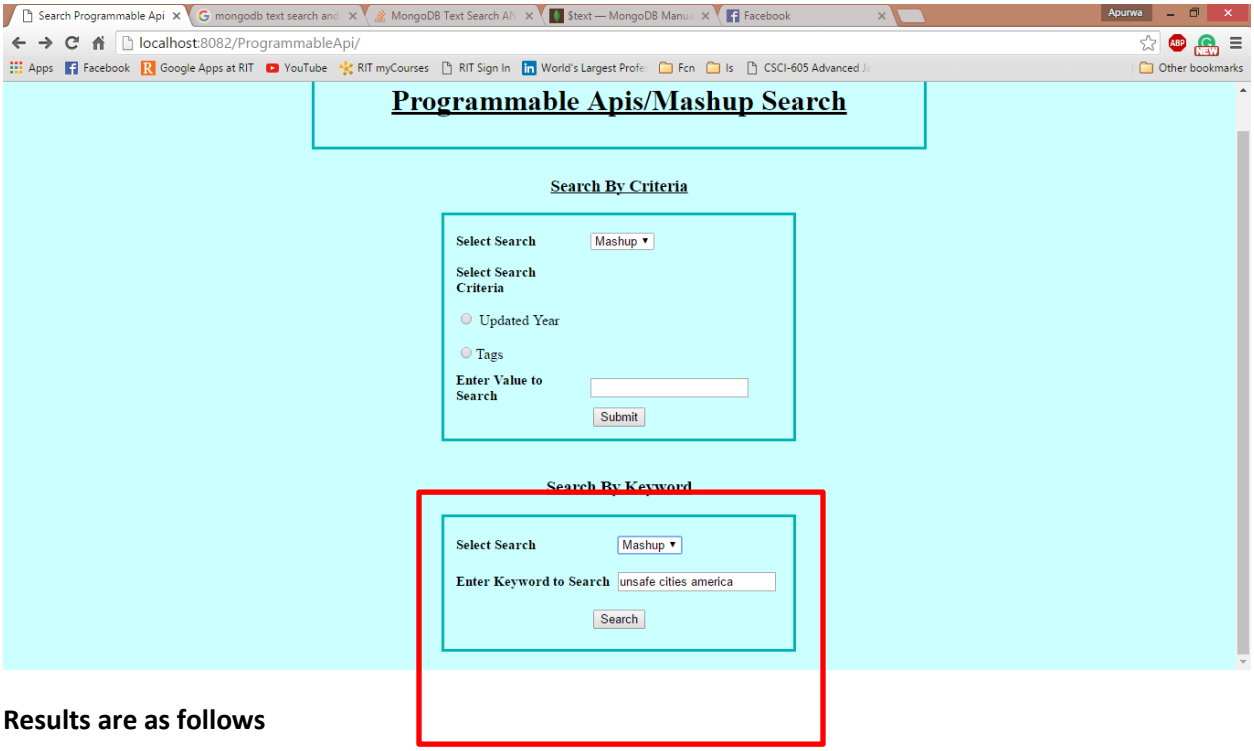
db.mashup.find({ "Apis.api.apiName" : { "$regex" : "google maps" , "$options": "i"}}).

```
Type "it" for more
> db.mashup.find({ "Apis.api.apiName" : { "$regex" : "google maps" , "$options":
 "i"}}).pretty()
```

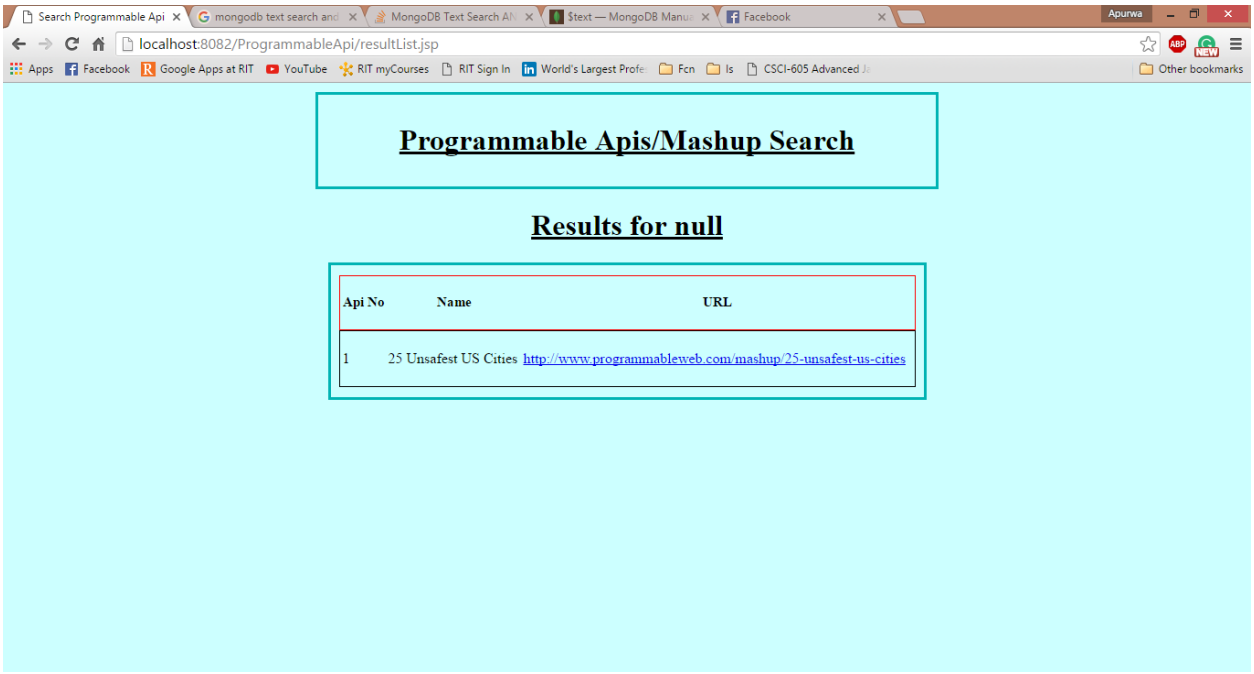
The results are as follows,

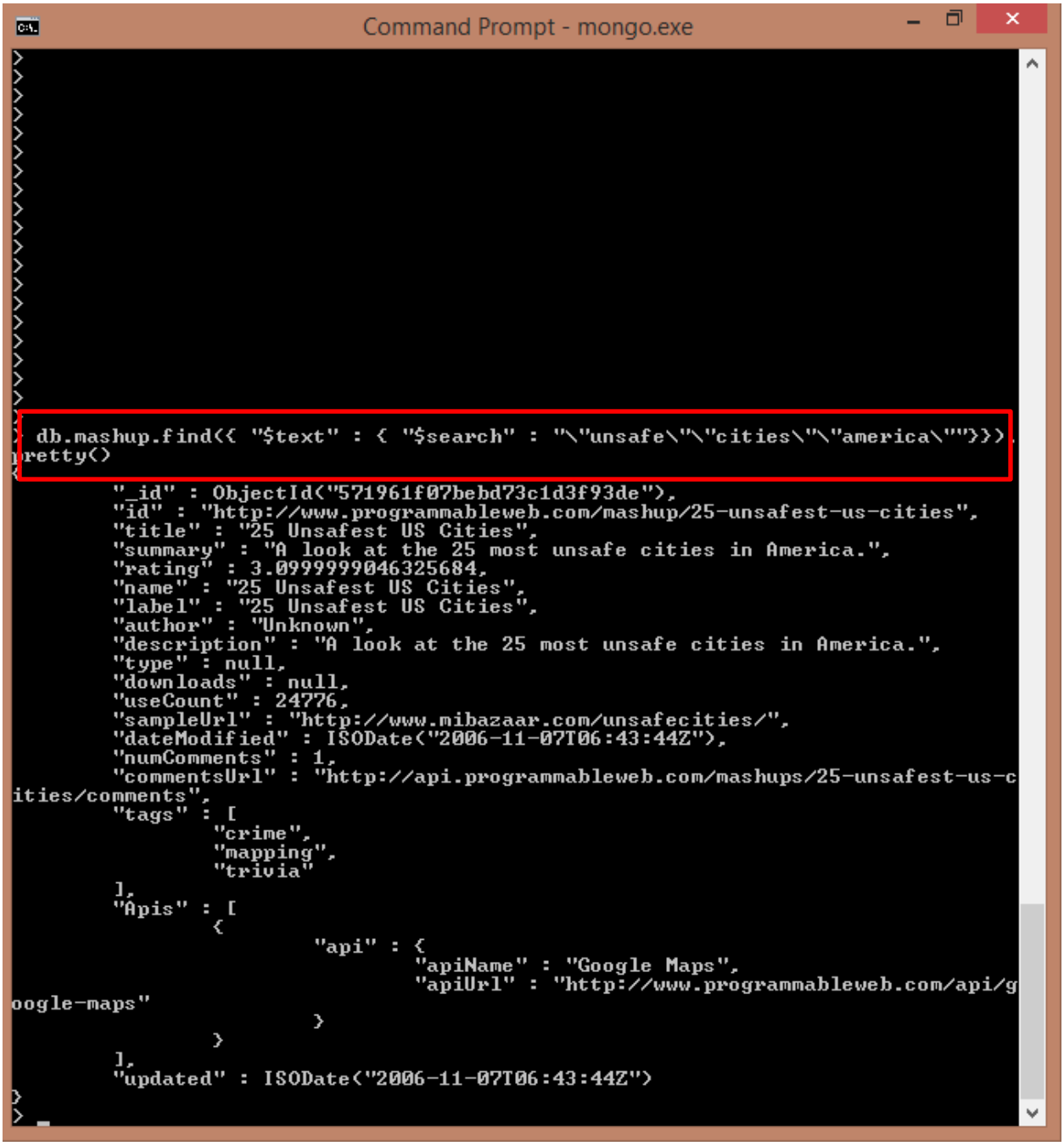


Keyword search in mashup: select Mashup in the dropdown options and enter keyword as unsafe cities america



Results are as follows




Command Prompt - mongo.exe
> db.mashup.find({ "$text" : { "$search" : "\"unsafe\"\"cities\"\"america\"" }}).pretty()
{
"_id" : ObjectId("571961f07bebd73c1d3f93de"),
"id" : "http://www.programmableweb.com/mashup/25-unsafest-us-cities",
"title" : "25 Unsafest US Cities",
"summary" : "A look at the 25 most unsafe cities in America.",
"rating" : 3.0999999046325684,
"name" : "25 Unsafest US Cities",
"label" : "25 Unsafest US Cities",
"author" : "Unknown",
"description" : "A look at the 25 most unsafe cities in America.",
"type" : null,
"downloads" : null,
"useCount" : 24776,
"sampleUrl" : "http://www.mibazaar.com/unsafecities/",
"dateModified" : ISODate("2006-11-07T06:43:44Z"),
"numComments" : 1,
"commentsUrl" : "http://api.programmableweb.com/mashups/25-unsafest-us-cities/comments",
"tags" : [
"crime",
"mapping",
"trivia"
],
"Apis" : [
{
"api" : {
"apiName" : "Google Maps",
"apiUrl" : "http://www.programmableweb.com/api/google-maps"
}
}
],
"updated" : ISODate("2006-11-07T06:43:44Z")
}
>