Introduction

In this assignment, we are required to work with 2 files namely API.txt and Mashup.txt. These files are service data. Both of these files contain data retrieved through ProgrammableWeb APIs and we are required to parse, store, manage, and analyze the data from these files. Post that we need to develop a web query system that can allow a user to obtain information about APIs, Perform a search based on keywords as well as retrieve the top APIs used.

Design of the data structure and the query system

Technologies/components used

Technology	Usage
Fast API	For creating a full stack application with support for both the backend and front end.
MongoDB	For storing API and Mashup file data
Python	For general-purpose programming
HTML/CSS	For User Interface

Web service Design

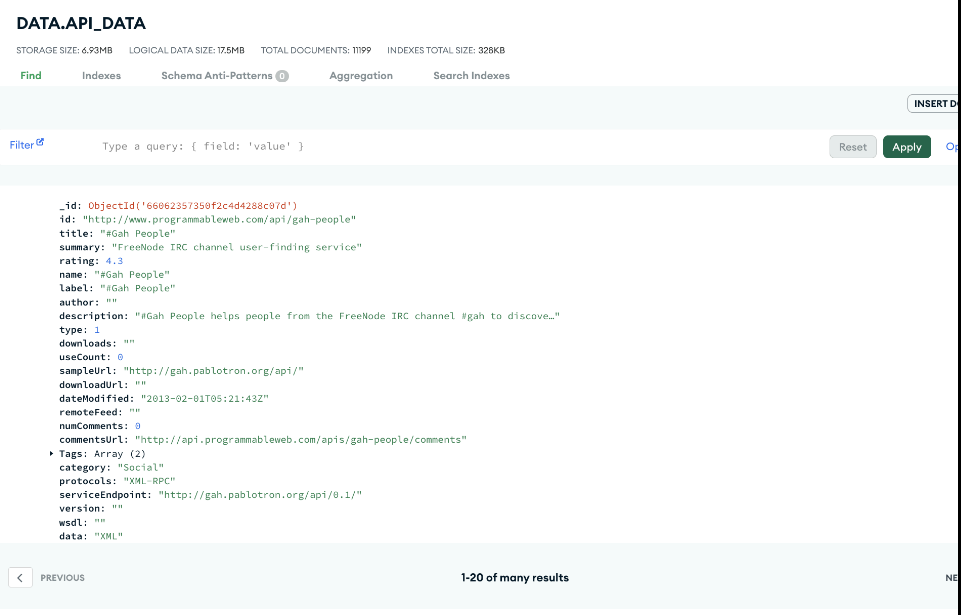
Endpoints

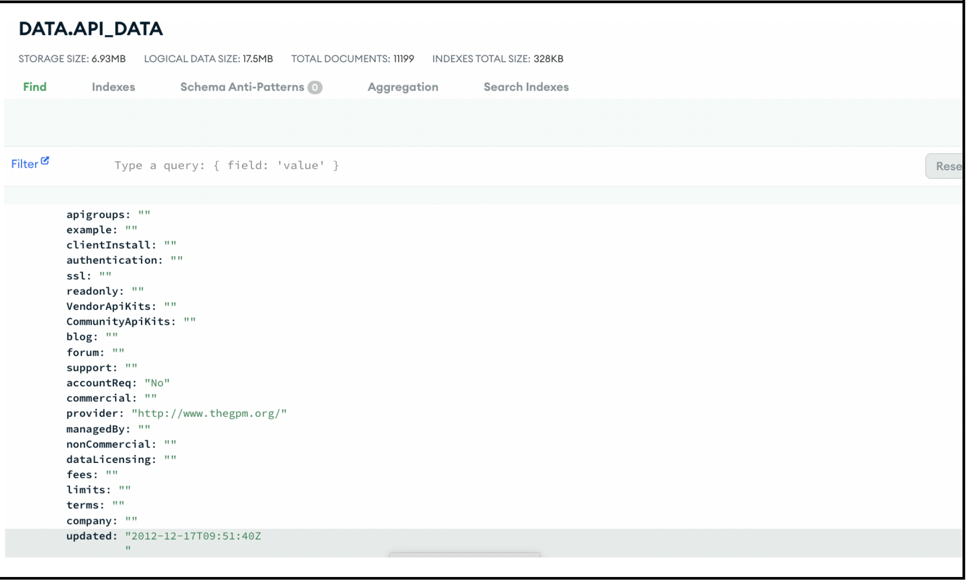
Description	Endpoint	HTML File rendered	Endpoints for queries
Displays the home page	/index	index.html	-
Displays the front-end for queries 3 and 4 related to keywords-based search in API and Mashup	/submitKeywords	searchtype1.html	**query3** - Searches keywords in Mashup **query4** - Searches keywords in API
Displays the front-end for top k API names and Mashup names for queries 5 and 6	/apinames	apiname.html	query5 - top k API names for Mashup query6 - top k mashup names for greatest no. API
Displays the front-end for criteria-based search in API data	/resultsapi	apiquery.html	query1 - matches the user's input against API tables
Displays the front-end for criteria-based search in Mashup data	/resultsmashup	mashupquery.html	query2 - matches the user's input against Mashup tables

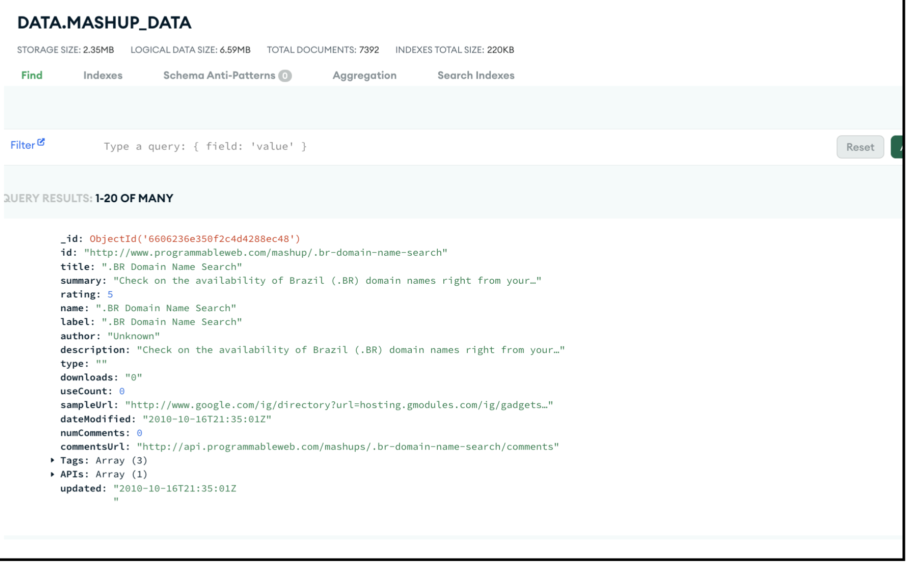
Backend Database

It seems that MongoDB is a good choice of database for storing such detailed information about APIs. Hence we are told to use MongoDB for storing API and Mashup file data. There are two collections namely API_DATA and MASHUP_DATA under the DATA Collection. To load the documents into these collections, I have used Python programming along with libraries like Pymongo.

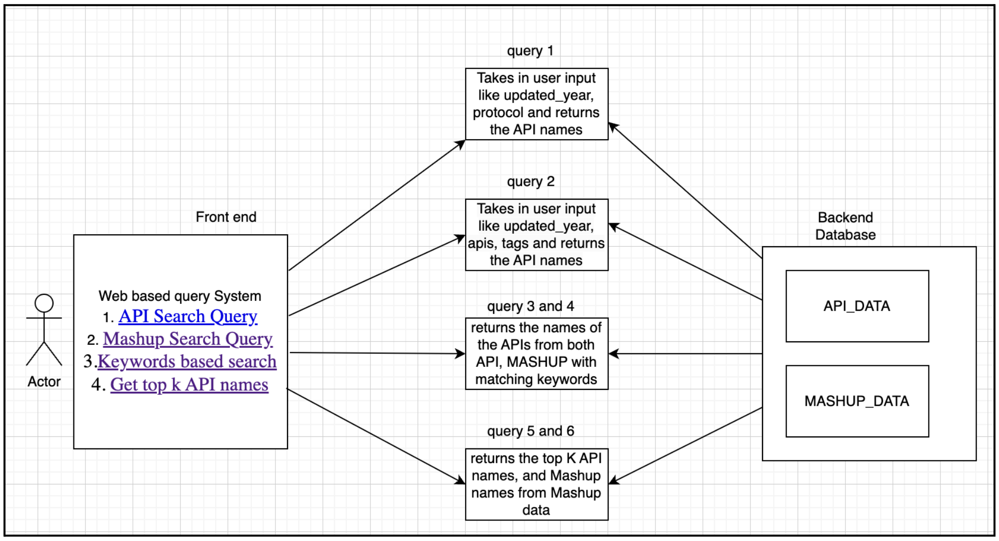
API_DATA	Database

This collection stores many attributes like title, name, summary, description, number of comments, tags, categories, and many more. I have used this table for performing 3 queries.	 Find
	Indexes
	Schema Anti-Patterns 0
	Aggregation
	Search Indexes
	INSERT D
	Filter
	Type a query: { field: 'value' }
	Reset
	Apply
	Op
	PREVIOUS
	NEDATA.API_DATA
	STORAGE SIZE: 6.93MB LOGICAL DATA SIZE: 17.5MB TOTAL DOCUMENTS: 11199 INDEXES TOTAL SIZE: 328KB
	`_id: ObjectId('66062357350f2c4d4288c07d')`
	`id: "http://www.programmableweb.com/api/gah-people"`
	`title: "#Gah People"`
	`summary: "FreeNode IRC channel user-finding service"`
	`rating: 4.3`
	`name: "#Gah People"`
	`label: "#Gah People"`
	`author: ""`
	`description: "#Gah People helps people from the FreeNode IRC channel #gah to discove…"`
	`type: 1`
	`downloads: ""`
	`useCount: 0`
	`sampleUrl: "http://gah.pablotron.org/api/"`
	`downloadUrl: ""`
	`dateModified: "2013-02-01T05:21:43Z"`
	`remoteFeed: ""`
	`numComments: 0`
	`commentsUrl: "http://api.programmableweb.com/apis/gah-people/comments"`
	`▸ Tags: Array (2)`
	`category: "Social"`
	`protocols: "XML-RPC"`
	`serviceEndpoint: "http://gah.pablotron.org/api/0.1/"`
	`version: ""`
	`wsdl: ""`
	`data: "XML"`
	1-20 of many results

	 Find
	Indexes
	Schema Anti-Patterns 0
	Aggregation
	Search Indexes
	Filter
	Type a query: { field: 'value' }
	ReseDATA.API_DATA
	STORAGE SIZE: 6.93MB LOGICAL DATA SIZE: 17.5MB TOTAL DOCUMENTS: 11199 INDEXES TOTAL SIZE: 328KB
	`apigroups: ""`
	`example: ""`
	`clientInstall: ""`
	`authentication: ""`
	`ssl: ""`
	`readonly: ""`
	`VendorApiKits: ""`
	`CommunityApiKits: ""`
	`blog: ""`
	`forum: ""`
	`support: ""`
	`accountReq: "No"`
	`commercial: ""`
	`provider: "http://www.thegpm.org/"`
	`managedBy: ""`
	`nonCommercial: ""`
	`dataLicensing: ""`
	`fees: ""`
	`limits: ""`
	`terms: ""`
	`company: ""`
	`updated: "2012-12-17T09:51:40Z`
	`"`

MASHUP_DATA	Database
This collection stores many attributes like title, name, summary, description, number of comments, tags, and many more. I have used this table for performing 3 queries.	 DATA.MASHUP_DATA STORAGE SIZE: 2.35MB LOGICAL DATA SIZE: 6.59MB TOTAL DOCUMENTS: 7392 INDEXES TOTAL SIZE: 220KB Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes Filter Type a query: { field: 'value' } Reset QUERY RESULTS: 1-20 OF MANY _id: ObjectId('6606236e350f2c4d4288ec48') id: "http://www.programmableweb.com/mashup/.br-domain-name-search" title: ".BR Domain Name Search" summary: "Check on the availability of Brazil (.BR) domain names right from your…" rating: 5 name: ".BR Domain Name Search" label: ".BR Domain Name Search" author: "Unknown" description: "Check on the availability of Brazil (.BR) domain names right from your…" type: "" downloads: "0" useCount: 0 sampleUrl: "http://www.google.com/ig/directory?url=hosting.gmodules.com/ig/gadgets…" dateModified: "2010-10-16T21:35:01Z" numComments: 0 commentsUrl: "http://api.programmableweb.com/mashups/.br-domain-name-search/comments" ▸ Tags: Array (3) ▸ APIs: Array (1) updated: "2010-10-16T21:35:01Z "

Design Diagram

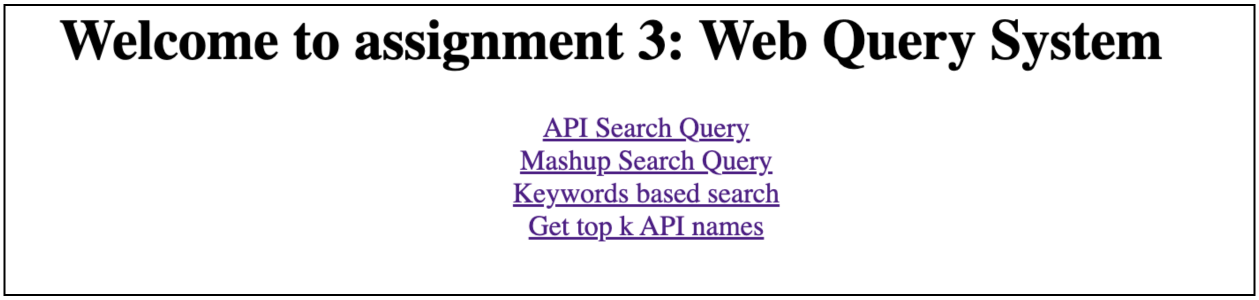

query 1
Takes in user input like updated_year, protocol and returns the API names
query 2
Takes in user input like updated_year, apis, tags and returns the API names
query 3 and 4
returns the names of the APIs from both API, MASHUP with matching keywords
query 5 and 6
returns the top K API names, and Mashup names from Mashup data
Front end
Web based query System
1. API Search Query
2. Mashup Search Query
3.Keywords based search
4. Get top k API names
Actor
Backend Database
API_DATA
MASHUP_DATA

As can be seen from the above diagram, there are three components. The first one is a front-end component that allows the user to query. The other is 6 queries whose working are described as below. And lastly, a Backend database component is used for almost all the queries.

Queries	Explanation
Return the names of APIs based on different criteria, including updated year, protocols, category, rating (such as higher than, equal to, or lower than a given rating), and tags	Used a basic mongodb query that matches the passed parameters like updated_year, category, tags, protocol, ratings against every document in the API_DATA collection
Return the names of Mashups based on different criteria, including updated year, used APIs, and tags	Used a basic mongodb query that matches the passed parameters like updated_year, APIs, tags against every document in the MASHUP_DATA collection
Given a set of keywords,return the names of APIs if all the keywords can be found in the title, summary, or the description of the APIs	Used **regex** keyword in the mongodb query for checking presence of all the user input keywords present in the given fields. And returned only those API names having all the keywords from API table
Given a set of keywords, return the names of Mashups if all the keywords can be found in the title, summary, or the description of the Mashups	Used **regex** keyword in the mongodb query for checking presence of all the user input keywords present in the given fields. And returned only those API names having all the keywords from Mashup table
Return the names of top K APIs that are most frequently used in mashups. K can be any positive integer, such as 10	While loading the data of Mashup in the MASHUP_DATA collection, created a local file that stored the count of each API that was used. Post-loading sorted the contents of the file in reverse order and returned the top k API names.
Return the names of top K mashups that have the greatest number of APIs. K can be any positive integer, such as 10.	While loading the data of Mashup in MASHUP_DATA collection, created a local file that stored the count of API that was used by each mashup API along with its name. Post-loading, sort the contents of the file in reverse order and return the top k mashup names.

Working of the Query system

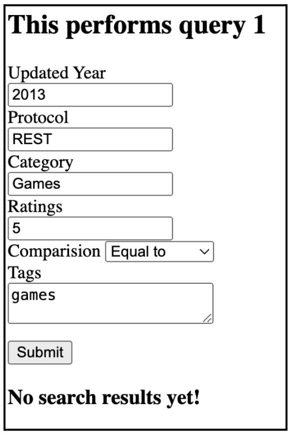
1. On loading the system, the user is displayed with a landing page that consists of 4 links as below.



Welcome to assignment 3: Web Query System

API Search Query
Mashup Search Query
Keywords based search
Get top k API names

2. The first link i.e. **API Search Query** is responsible for retrieving results based on a few criteria including updated year, protocols, category, rating (such as higher than, equal to, or lower than a given rating), and tags From the API_DATA Collection. On clicking that link you will be redirected to a new endpoint **/submitKeywords** as below**.**

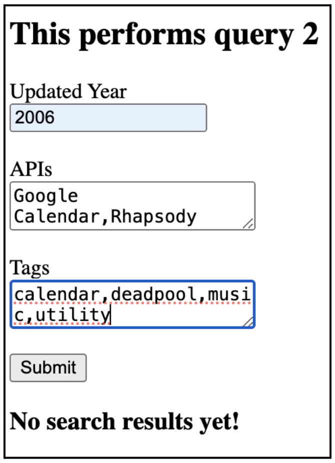


This performs query 1

Updated Year
2013
Protocol
REST
Category
Games
Ratings
5
Comparision Equal to
Tags
games

No search results yet!

3. You can enter values for each user field and get the desired results. It is to be noted that the tags field should be comma-separated. For example **popular,tea.**
4. The second link i.e. **Mashup Search Query** is responsible for retrieving results based on a few criteria including updated year, APIs, and tags From the MASHUP_DATA Collection. On clicking that link you will be redirected to a new endpoint **/resultsmashup** as below**.**



This performs query 2

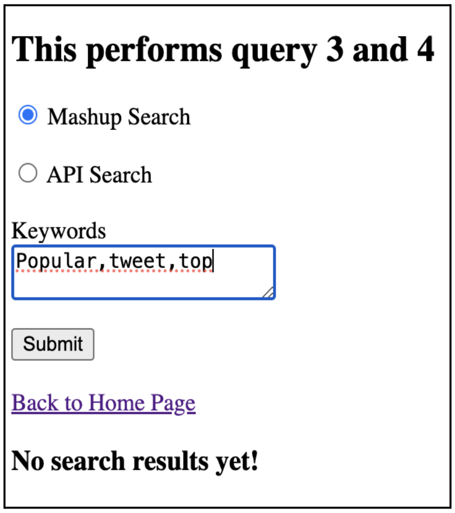
Updated Year
2006

APIs
Google Calendar,Rhapsody

Tags
calendar,deadpool,music,utility

No search results yet!

5. The third link **Keywords based search** is responsible for performing searches based on the user-provided keywords in both API_DATA and MASHUP_DATA and returns the names of the API where those keywords are present in either title, description, or summary. Select the Mashup Search radio button enter the comma-separated keywords and hit submit. Similarly, you can select the API Search radio button enter the comma-separated keywords, and hit submit.



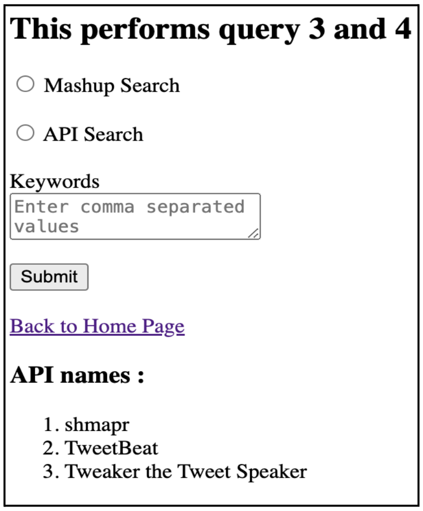
This performs query 3 and 4

● Mashup Search

○ API Search

Keywords: Popular,tweet,top

No search results yet!



This performs query 3 and 4

○ Mashup Search

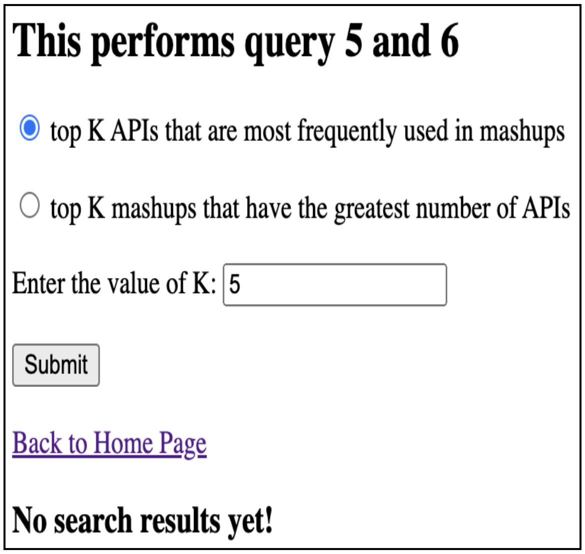
○ API Search

Keywords: Enter comma separated values

API names :

1. shmapr
2. TweetBeat
3. Tweaker the Tweet Speaker

6. The fourth link **Get top K API Names** is responsible for retrieving the top k API names for the Mashup table and top k mashup names for the greatest no. API. Select the 1st radio button enter the value of k and hit submit. Similarly, you can select the 2nd radio button enter the value of k, and hit submit. This will return the top values of API and mashup names using the greatest number of APIs.



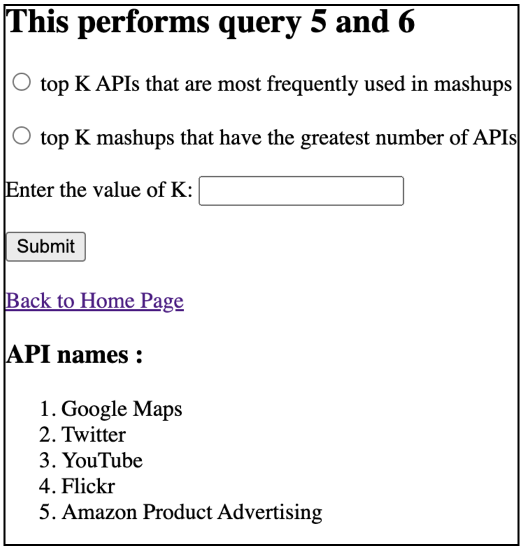
This performs query 5 and 6

● top K APIs that are most frequently used in mashups

○ top K mashups that have the greatest number of APIs

Enter the value of K: 5

No search results yet!



This performs query 5 and 6

○ top K APIs that are most frequently used in mashups

○ top K mashups that have the greatest number of APIs

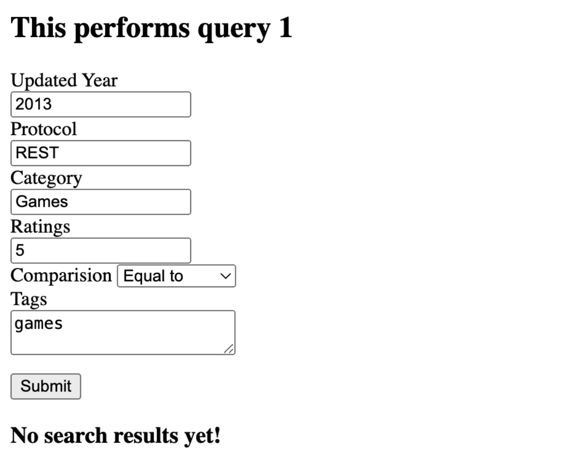
Enter the value of K:

API names :

1. Google Maps
2. Twitter
3. YouTube
4. Flickr
5. Amazon Product Advertising

Testing scenarios of the query system

Clicking on the first line **API Search Query**, will help to perform the 1st query.



This performs query 1

Updated Year: 2013

Protocol: REST

Category: Games

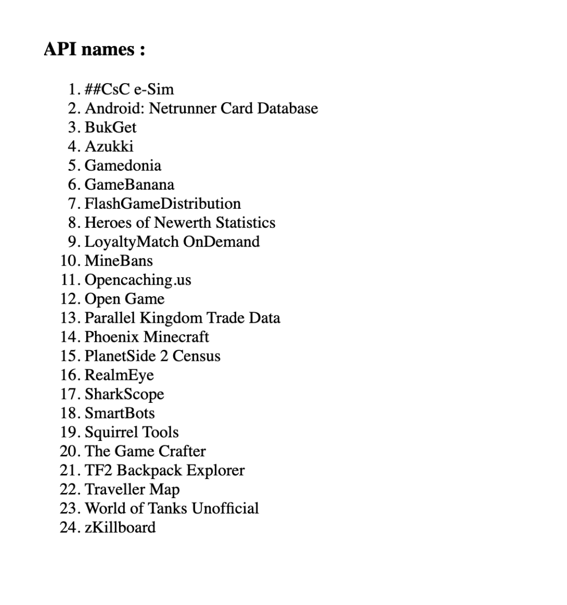
Ratings: 5

Comparision: Equal to

Tags: games

No search results yet!

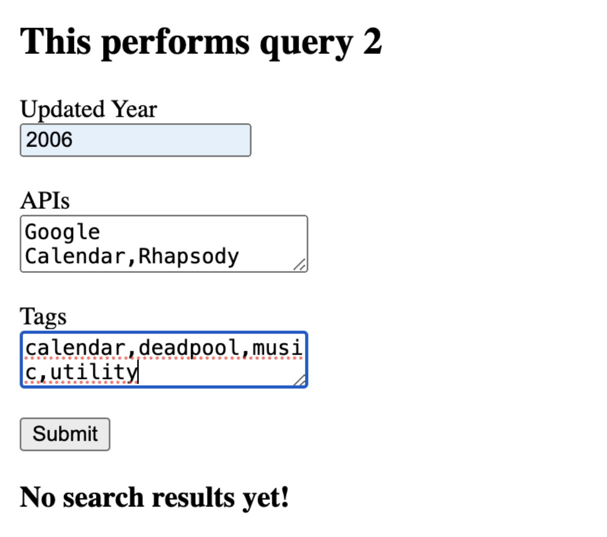
On clicking submit after entering the above values we get the result to the right.



API names :

1. ##CsC e-Sim
2. Android: Netrunner Card Database
3. BukGet
4. Azukki
5. Gamedonia
6. GameBanana
7. FlashGameDistribution
8. Heroes of Newerth Statistics
9. LoyaltyMatch OnDemand
10. MineBans
11. Opencaching.us
12. Open Game
13. Parallel Kingdom Trade Data
14. Phoenix Minecraft
15. PlanetSide 2 Census
16. RealmEye
17. SharkScope
18. SmartBots
19. Squirrel Tools
20. The Game Crafter
21. TF2 Backpack Explorer
22. Traveller Map
23. World of Tanks Unofficial
24. zKillboard

Clicking on the second line **Mashup Search Query**, will help to perform the 2nd query



This performs query 2

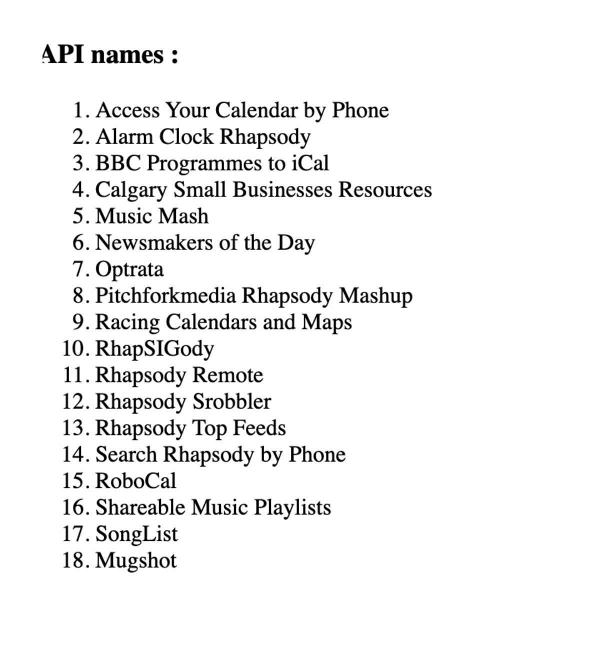
Updated Year: 2006

APIs: Google Calendar,Rhapsody

Tags: calendar,deadpool,music,utility

No search results yet!

On clicking submit we get the result to the right.

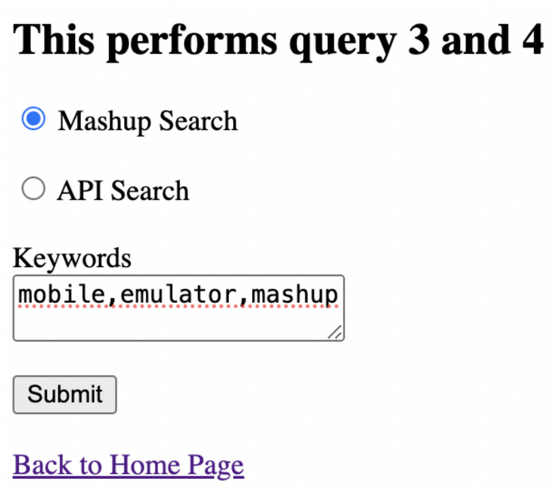


API names :

1. Access Your Calendar by Phone
2. Alarm Clock Rhapsody
3. BBC Programmes to iCal
4. Calgary Small Businesses Resources
5. Music Mash
6. Newsmakers of the Day
7. Optrata
8. Pitchforkmedia Rhapsody Mashup
9. Racing Calendars and Maps
10. RhapSIGody
11. Rhapsody Remote
12. Rhapsody Srobbler
13. Rhapsody Top Feeds
14. Search Rhapsody by Phone
15. RoboCal
16. Shareable Music Playlists
17. SongList
18. Mugshot

Clicking on the third line Keywords based search , will help to perform the 3rd and 4th query.

Query3:



This performs query 3 and 4

● Mashup Search

○ API Search

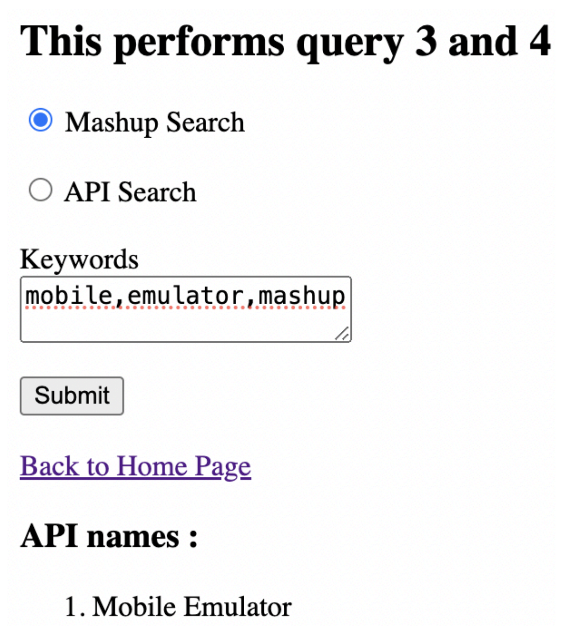
Keywords

mobile,emulator,mashup

Submit

Back to Home Page

On clicking submit we get the result to the right for mashup search



This performs query 3 and 4

● Mashup Search

○ API Search

Keywords

mobile,emulator,mashup

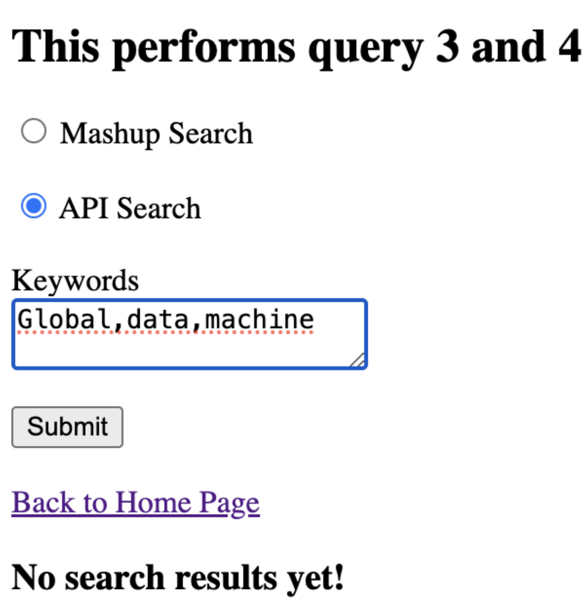
Submit

Back to Home Page

API names :

1. Mobile Emulator

Query4:



This performs query 3 and 4

○ Mashup Search

● API Search

Keywords

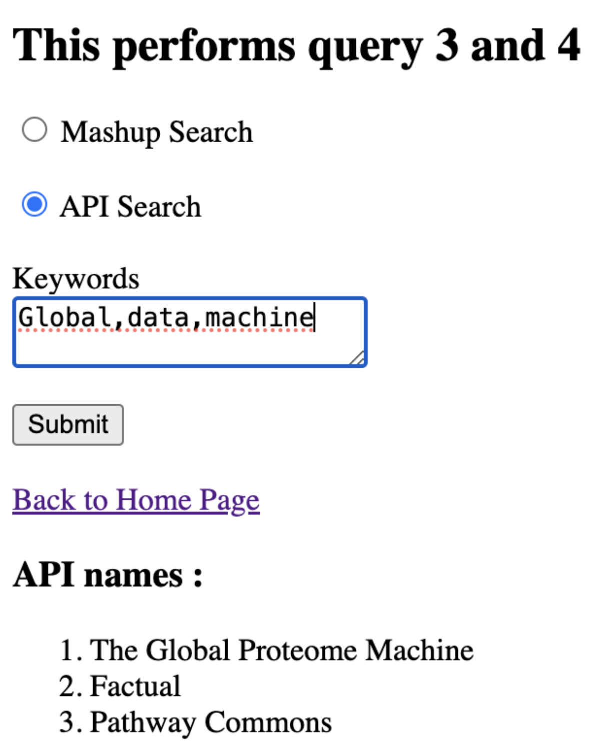
Global,data,machine

Submit

Back to Home Page

No search results yet!

On clicking submit we get the result to the right for the mashup search



This performs query 3 and 4

○ Mashup Search

● API Search

Keywords

Global,data,machine

Submit

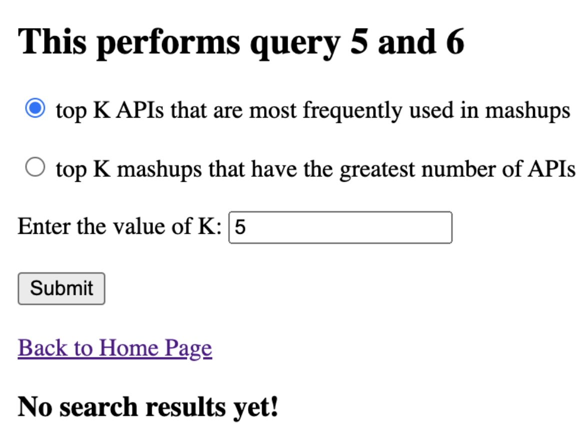
Back to Home Page

API names :

1. The Global Proteome Machine
2. Factual
3. Pathway Commons

Clicking on the fourth line Get top k API names will help to perform the 5th and 6th query.

Query 5



This performs query 5 and 6

● top K APIs that are most frequently used in mashups

○ top K mashups that have the greatest number of APIs

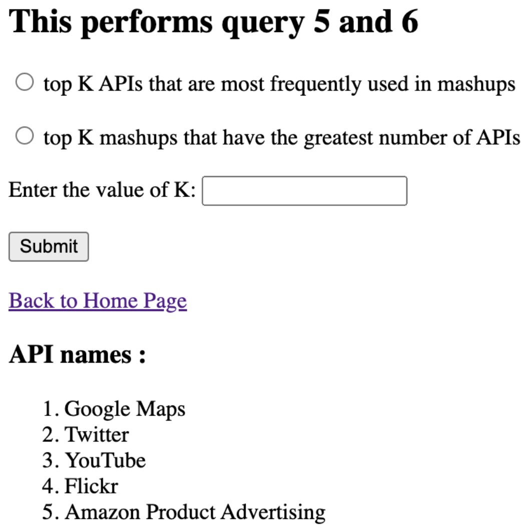
Enter the value of K: 5

Submit

Back to Home Page

No search results yet!

On clicking submit we get the result to the right for the top 5 API



This performs query 5 and 6

○ top K APIs that are most frequently used in mashups

○ top K mashups that have the greatest number of APIs

Enter the value of K:

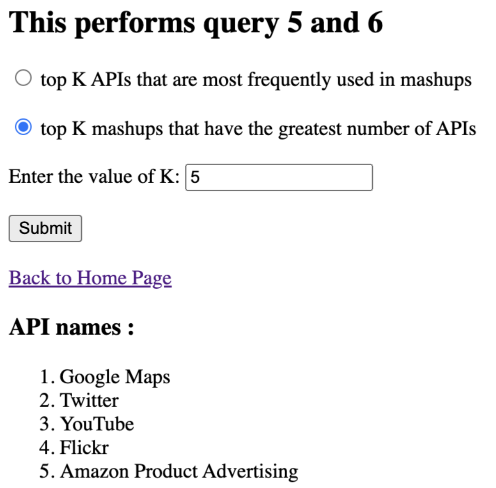
Submit

Back to Home Page

API names :

1. Google Maps
2. Twitter
3. YouTube
4. Flickr
5. Amazon Product Advertising

Query 6



This performs query 5 and 6

○ top K APIs that are most frequently used in mashups

● top K mashups that have the greatest number of APIs

Enter the value of K: 5

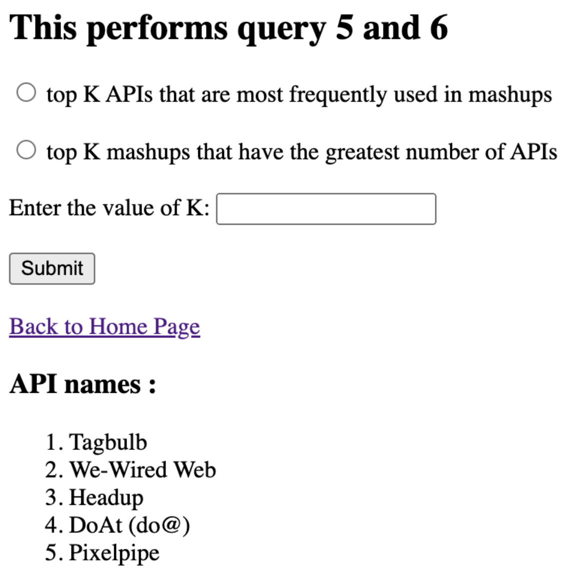
Submit

Back to Home Page

API names :

1. Google Maps
2. Twitter
3. YouTube
4. Flickr
5. Amazon Product Advertising

On clicking submit we get the result to the right for the top 5 Mashup name with greatest APIs



This performs query 5 and 6

○ top K APIs that are most frequently used in mashups

○ top K mashups that have the greatest number of APIs

Enter the value of K:

Submit

Back to Home Page

API names :

1. Tagbulb
2. We-Wired Web
3. Headup
4. DoAt (do@)
5. Pixelpipe

Readme

1. Upon extracting the zip folder, open the folder "**Shreenidhi_Acharya_A3**" in Visual Studio Code or pycharm.
2. Run the command "**python -m venv .**" in a terminal (**CTRL+~)** to create a virtual environment for this application.
   Note: If you get error in the above command, you can use **python3 -m venv .**
3. Run the command "**source bin/activate**" to activate the created environment.
4. Run the command "**pip install -r requirements.txt**" to install all dependencies.
   Note: if you encounter error with pip command use "**pip3 install -r requirements.txt"** instead.

<u>Part 1</u>: Load the database. (This step can be **skipped** as I have already loaded the data in Mongodb atlas)

5. Open a new terminal and run the Python file "parsing_and_loading_data.py". This command will create the databases at the backend and load them with values
   - Command: **python parsing_and_loading_data.py**

   Note: if you encounter error use "**python3 parsing_and_loading_data.py"** instead.
6. After running the above command two new JSON files namely data_api_len.json and data_api.json will be created. These two files will be used for running query 5 and query 6.
   Note: These files are already provided along with the code files

<u>Part 2</u>: Running the application

If you have skipped Part 1, directly run the below command

7. Now it's time to run the entire application. Run the Python file **server.py** to start the application. Once started open your browser and go to http://localhost:8000/index.
   - Command: **python server.py**

   Note: if you encounter error use "**python3 server.py"**