**Design of Programming Assignment 3**

**Tech stack used:**

flask, mongodb compass and html are used to create this web application.

Using flask\_pymongo I was able to connect python and mongodb. At the time of establishing connection I have created a new database called '*webAPIs*'

*(app.config["MONGO\_URI"] = "mongodb://localhost:27017/webAPIs")*

*index.html* is design of webpage where the user can insert filters. User can choose to run 1 of 4 queries.

*home.html* bears the results (all the fields of documents returned except ‘*\_id*’) of the search provided

**Parsing the data:**

Given 2 text files, *api.txt* and *mashup.txt*, I have parsed the data with given delimiters as dictionaries and appended them to a list (implemented in *parse.py*). Created field names in the back end according to the instructions provided in the manual for both the collections, *apiData* and *mashupData* collections

These lists have been imported to *main.py* where they are being loaded into the database into respective collections

**Running Application:**

I have created 4 different functions for each query

*For queries 1 and 2:*

I have used pipeline where I am checking for the given inputs

a. *year*: Since we need to check for the year alone, I left the field ‘updated’, which is in string format as is, and performed string match using regex for ‘year’

b. *protocol/ category*: Matching the given inputs with the aforementioned fields in the database

c. *tag*: all the tags are saved as list and the given input is checked if the tag is present or not in every document of that collection (tags are case sensitive)

d. *rating*: Higher precedence is given to 'rating equal to' over greater than and less than, if the former is inputted along with the later ones, app only checks for rating equal to field. This was enforced on back end.

If the rating equal to is not provided only then, the other 2 fields are considered. All the queries are added to a pipeline which are then run through the database

Query 2 also was executed similarly against *mashupData* collection

While returning all the fields, I have omitted ‘\_id’ and rest of the fields that are part of original document are returned on *home.html* page

*For queries 3 and 4:*

Using regex, all the keywords (keywords are *case insensitive*) are checked one after another in a for loop if they are present in one of the 3 fields at least (using ‘*$or*’), and then included in the pipeline.

Once all the keywords are checked, the pipeline is run through the database and the results(all the fields of the document except ‘*\_id*’) are fetched and returned on *home.html* page.

**Screenshots:**

*Database*

Graphical user interface, text, application, email

Description automatically generated

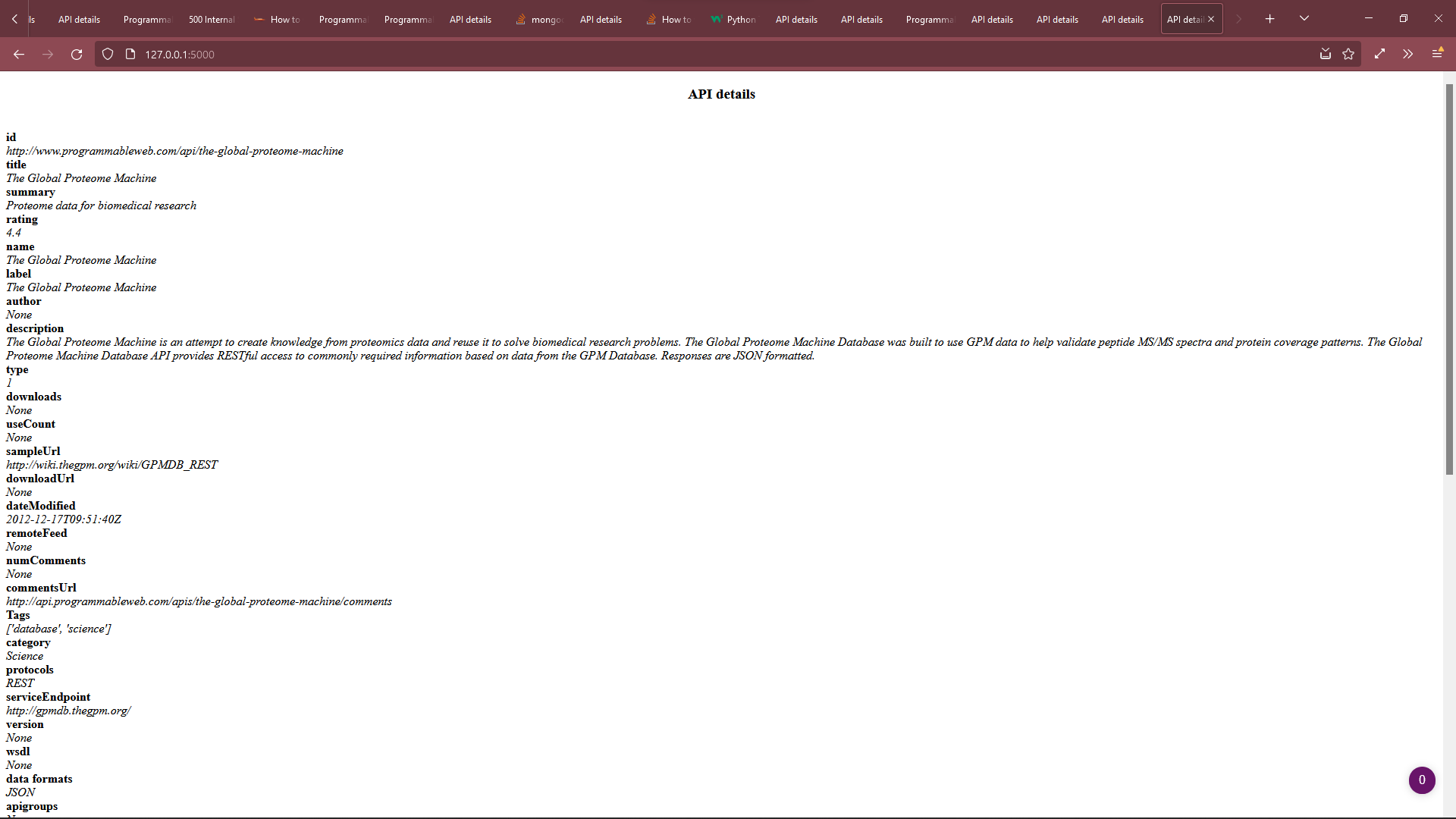
Graphical user interface, text, application, email

Description automatically generated

Query 1:

Graphical user interface, application, Word

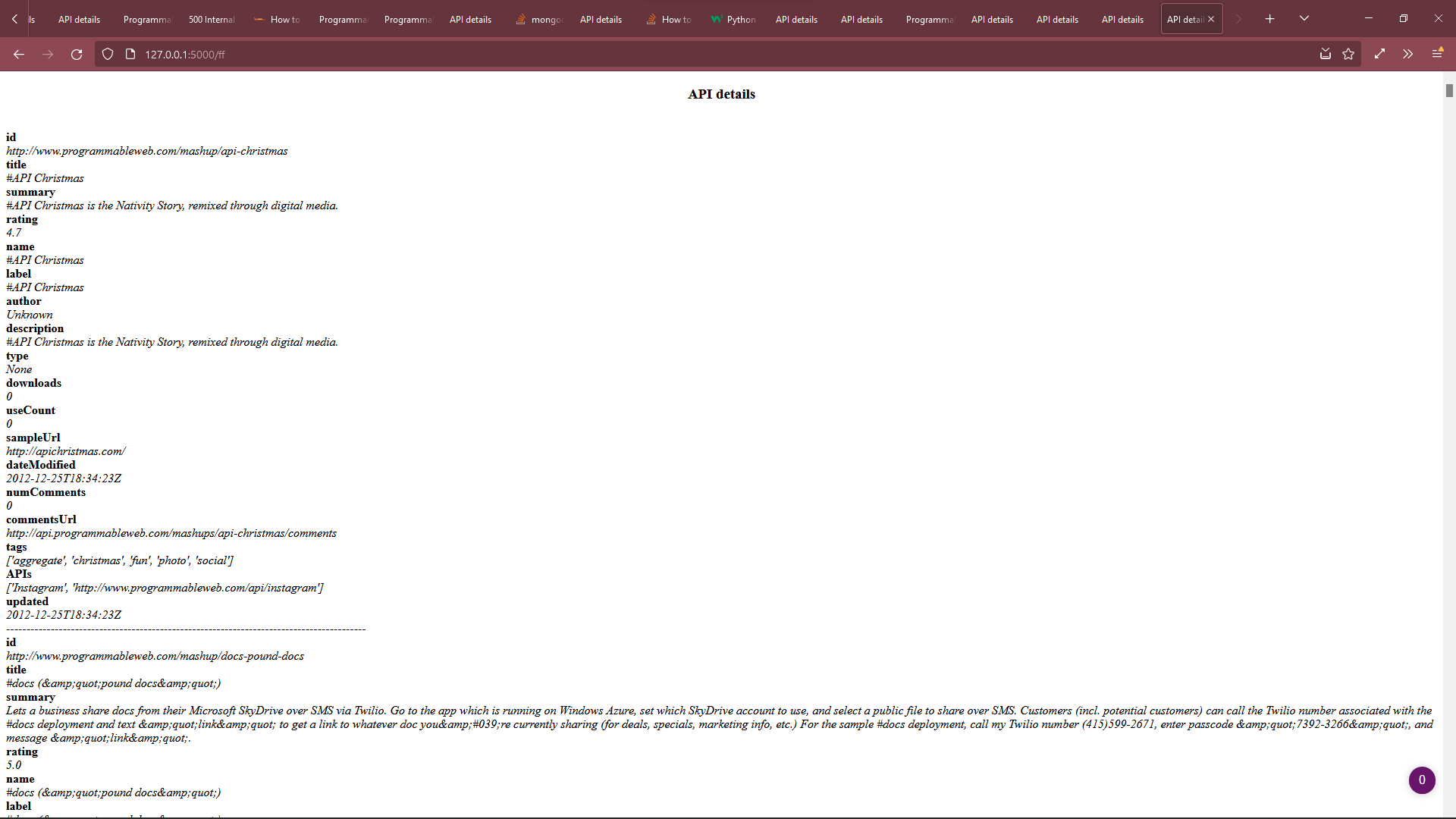
Description automatically generated



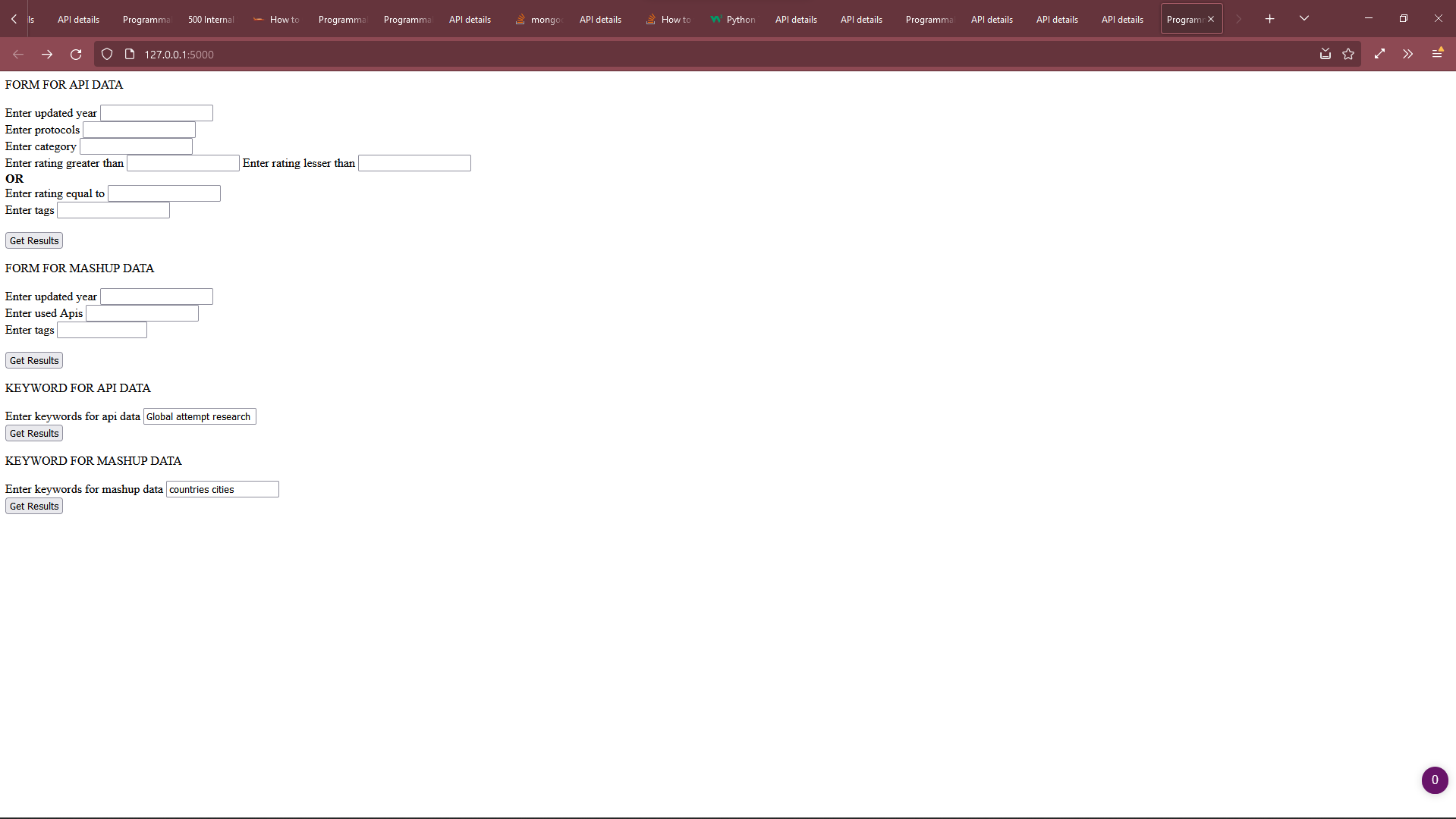
Query 2:

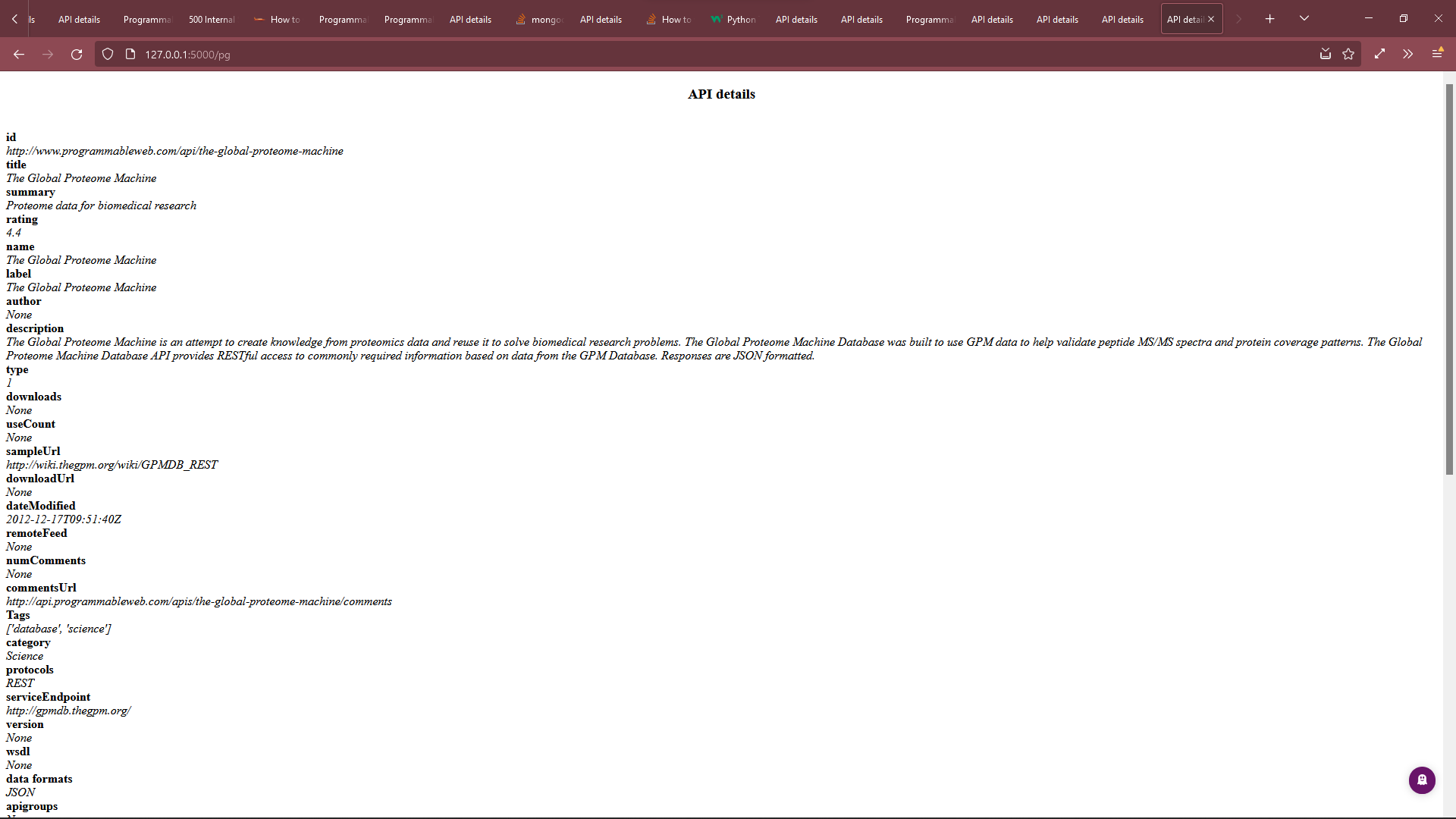
Graphical user interface, application

Description automatically generated



Query 3:





Query 4:

