This is an episode of our SNS project,in these lines of code,I used python + Django + Mongoengine for background processing including analyzing request, database querying and feeding back http response. The following code is used to search tags that a specific user has posted .

```
from datetime import datetime
import string
import random
from taguage.auth import session
from taguage.auth.models import Invitation_code
from taguage.utils import ErrorCode, simple_response, distance
from taguage.users.models import User, convert_ids_to_names, convert_names_to_ids
from taguage.tags.models import Tag, Author, get_latest_tags
from taguage.pushes.models import Push, match_one_push
from taguage.messages.models import new_message
from django.http import HttpResponse
from mongoengine.queryset import Q
import lxml.etree as ET

def search_my_tags(request):
    '''
    Request No. 105.
    Edited by zhangyue.

    Parameter: uid(required), tag(requeired);
    Return: Search specific tags in user's tag list and return the information of the tags.
    '''
    uid = request.GET.get('uid')
    tags = request.GET.get('tag')
    lastId = request.GET.get('lastId')
    if None in (uid, tags):
        return HttpResponse(simple_response(ErrorCode.PARSE_CMD_ERROR))

    try:
        #login?
        if not session.login_verified(request, uid):
            return HttpResponse(simple_response(ErrorCode.LOGIN_REQUIRED))
        #Split parameters into words list.
        words = tags.split(',')
        #Search the tags
        tags_rich = set()
        for word in words:
            tag = Tag.objects(author__id=uid, words_list=word, active=True).first()
            if tag is not None:
                tags_rich.add(tag)
            else:
                continue
        tags_rich = list(tags_rich)
        COUNT = 25
```

```
        if lastId == '-1':
            start = -1
            end = start - COUNT
            tags = tags_rich[start:end:-1]
        else:
            last_tag = Tag.objects(id=lastId).first()
            start = tags_rich.index(last_tag) - len(tags_rich) - 1
            end = start - COUNT
            tags = tags_rich[start:end:-1]
        #Update session timestamp
        session.update_session(request, 'last_action_time', datetime.now())
        #Create response text
        root = ET.Element('Response')
        ET.SubElement(root, 'ErrorCode').text = str(ErrorCode.SUCCESS )
        result = ET.SubElement(root, 'Result')
        for tag in tags:
            item = ET.SubElement(result, 'item')
            ET.SubElement(item, 'tid').text = str(tag.id)
            ET.SubElement(item, 'tag').text = tag.tag and tag.tag or ''
            ET.SubElement(item, 'reason').text = tag.reason and tag.reason or ''
            date = tag.time
            ET.SubElement(item, 'time').text = '%04d%02d%02d%02d%02d' %
(date.year,date.month,date.day,date.hour,date.minute)
            ET.SubElement(item, 'imgOrigin').text = tag.imageOrigin and tag.imageOrigin or ''
            ET.SubElement(item, 'imgBig').text = tag.image128 and tag.image128 or ''
            ET.SubElement(item, 'imgSmall').text = tag.image64 and tag.image64 or ''
            ET.SubElement(item, 'imgTiny').text = tag.image32 and tag.image32 or ''
            at_ids = []
            for at_id in tag.at_list:
                at_ids.append(str(at_id))
            at_names = convert_ids_to_names(at_ids)
            ET.SubElement(item, 'at').text = ','.join(at_names)
            ET.SubElement(item, 'atUid').text = ','.join(at_ids)
            ET.SubElement(item, 'reply').text = str(tag.CommentCount())
            ET.SubElement(item, 'same').text = str(tag.similar_num)
        return HttpResponse(ET.tostring(root, encoding='utf-8', xml_declaration = True,
pretty_print = True))
    except:
        return HttpResponse(simple_response(ErrorCode.REQUEST_FAIL))
    '''
```