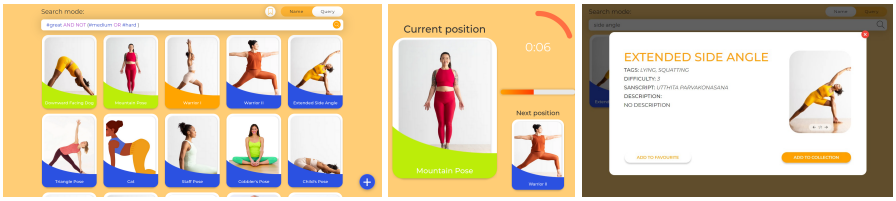
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

Web Tool for Creation, Management, and Use of a Database of Sports Poses

Tomáš Křivánek*



Abstract

This paper is focused on two searching methods and how they can be used in any searching and in any web application. Two most important qualities of any search are speed and accuracy. One of the best methods in terms of speed is so-called livesearch. This method provides results immediately when the user starts typing their search expression. That is why in many cases, the user does not even have to finish their expression, because when seeing the result, the user can immediately access it. Search accuracy is often achieved by using keywords or some part of text. Another common way to organize and fetch documents are tags: a short string which identifies a post or a photo. Multiple tags are then used similarly to plain keywords. This work proposes to use tags as a new searching logical language, where presence and/or absence of tags can be required. For this purpose I created a simple logical language “Queries”, because they can be used very similarly to database queries. In my case, the language and the user interface is made very simple so that a normal user can specify the query himself. This search tool is showcased in a web application focused around yoga poses. Both methods are used there – the user can search in the yoga poses database by name or by the queries language.

Keywords: livesearch — queries — sport — poses — database — quick — search

Supplementary Material: Demonstration Video

*xkriva29@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

This paper is focused on working with yoga poses, but the implemented methods can be used on any sport poses. In terms of yoga, there is also important the term yoga routines/sequences. Yoga poses are almost never exercised independently but they are organised into sequences, where each position is held for certain time interval and followed by another position. It would be great to have a tool which can organise these sequences and makes them easy to train.

There are plenty of datasets with different sport-sposes. Best approach to browsing through yoga poses is on the Yoga journal website [1], where the poses are organised into groups by type. It is by far the most clearly designed website on Internet but still there is no easy way to search through these poses by name or by combination of different properties. Some user interface elements on this websites are just too big and it looks like a waste of space. On the other hand, there are great descriptions and images of each yoga pose so the sportsmen can easily train these poses accurately. There is also a table of all poses included on this website, which I have used as part of my yoga poses dataset. However Yoga Journal website does not

provide a functionality to create yoga sequences.

For composing yoga sequences, there are many separate applications [2], but they share a very big disadvantage, they all require monthly payments. They also provide many great features like mobile/desktop applications, they often have their own graphics, their poses are drawn, which is better, because they are more illustrative than real photographs. For example, Sequence Wiz uses very simple illustrations, and it looks like it is very easy to use. It also provides a feature where users can write notes to each position in sequence.

My application must very clearly visualize database of yoga poses. It is also a demonstration how easy is to implement the livesearch method and how it can be done with any dataset. Users must be able to add new poses to database and add properties to the created poses. Users will be able to add these properties by adding tags. Searching should be possible by name or by logical combination of tags. These logical expressions should be explained very clearly to everyday user, because for non-programmers, logical expression might be confusing. Users will also be able to create their own sequences and train them with the application.

2. Used technologies and languages

Application is based on Model-View-Controller architecture (Fig. 1). Backend (Controller) is programmed in Flask framework, which means it uses Python 3 as programming language. Flask offers easy implementation and manipulation with HTTP requests from application's frontend. Whole application, except of basic configuration, is then basically a specification of answers on HTTP requests which are addressed by a pathname in the URL.

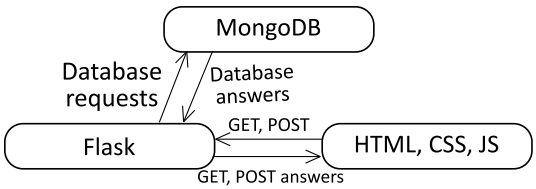


Figure 1. Architecture of the application

NoSQL database MongoDB (Model) was chosen for the application. The PyMongo library is used to communicate with the database, which makes it very easy to connect to the MongoDB atlas server where the database is running. Since it is a NoSQL database, the data are organized into collections. Individual records are stored in BSON format, which stands for binary JSON (for the difference between JSON and BSON see Tab 1). This format makes it much easier to store more complex data structures like arrays or image. For these, there is also an extra library called gridfs that takes care of the distribution of the binary data in the database, since a single record in BSON format cannot exceed 16 MB. These large images are separated into many records. The ability to store images and arrays easily was key to choosing this type of database.

The frontend uses three languages – HTML, CSS and Javascript. In HTML there is used a template engine Jinja2 that allows to add a simple logic to HTML itself. For example, render data from a database using for loops or use conditional statements if-elif-else.The CSS language is used for the graphical styling of the entire website. There are no other CSS frameworks or libraries used. Most of the application code comes from Javascript, mainly because of the abundant use of asynchronous requests and processing the answers. Here jQuery is used. This library allows for a simple signal capture such as button presses, keystrokes or mouse events. Animations on the page are programmed with the help of GSAP library, mainly by the use of a timeline, which allows to chain the animations of elements to be played consecutively.

Table 1. Comparison of JSON and BSON

	JSON	BSON
Encoding	UTF-8	Binary
Data Support	String, Boolean, Number Array	String, Boolean, Number (Integer, Float, Long, Decimal128...), Array, Date, Raw Binary
Readability	Human and machine	Only machine

3. Livesearch

As it was mentioned previously, livesearch is a method when results of a search are displayed dynamically as user writes in the search input box.

Livesearch has many benefits compared to traditional searching [3]:

- Results are shown as you type,
- results narrow as you continue typing,
- if results become too narrow, remove characters to see a broader result.

Many algorithms, including the one that is explained by w3schools [3], use asynchronous requests on the server which then evaluates the results of the search. Basically, when the user changes the contents of the search field, an asynchronous request is sent to the server, which evaluates the search result and sends

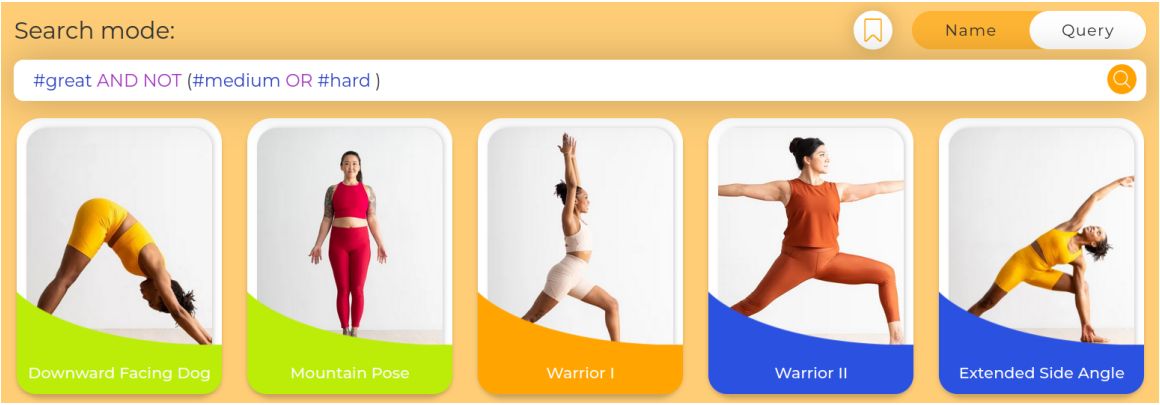


Figure 2. Example of search using the query language with the syntax highlighter.

the search result as a response. Yoga poses does not use this approach, instead it uses regular expressions to match the yoga titles. All titles are displayed at the beginning, when user loads the website.

Algorithm begins as the website loads by applying a jQuery event listener on the search box where users can type. This event listener triggers a function every time a keyboard key is released as user types in the search box. The function that is called on this event proceeds as follows:

1. It stores a string which is currently in the search box to the variable **text**.
2. It uses this variable to create a JavaScript RegExp class with '.*' characters at the beginning and at the end, which results in regular expression that matches each string that contains **text** variable string as a substring. For example when user enters a text: "pose", the regular expression will be initialized in this way :
   `new RegExp(".*pose.*")`
3. It checks if the text is longer or shorter than before a keyUp event.
   (a) If the text got longer it loops through poses which are currently displayed and hides those which does not match the regular expression.
   (b) If the text got shorter it loops through poses which are currently not displayed and show those which does match the regular expression.
   (c) If the text got the same length (but the text changed) it loops through all poses and hides those which does not match the regular expression and display those that does.

This approach not only makes it much faster, because there are no requests on the backend, but it is also optimized because it does not loop through every position but only through a displayed or not displayed subset most of the time.

4. Queries

If the user wants to search by tags, he can switch to search mode by using the button on the top right of the search. This search method is more complex, so it is not evaluated every time the search expression is changed, but the user must confirm the search every time by pressing the return key or clicking the search button on the right side of the input.

A very simple compiler has been developed for the Queries language that consists of four phases:

- Lexical analysis (Lexer) - The job of the tokenizer is to read tokens one at a time from the input stream and pass the tokens to the parser [4].
- Syntax analysis - Syntax refers to the rules that define the structure of a language. Syntax in computer programming means the rules that control the structure of the symbols, punctuation, and words of a programming language [5].
- Expression is transformed from infix format to postfix format.
- Interpretation of the postfix format.

Lexical analysis simply breaks the statement into parts and marks them by their type. There are three types: tag, keyword and brackets. The output of the lexer is an array of objects that contain the types and values of each token. Because the language is so simple and statements are not long, lexer always evaluates whole statement before it proceeds to the syntax analysis.

Syntax analysis works with a table of rules. Each token type has its own rules, which consists of token types that can follow after specific token type. For example keyword AND can be followed by a tag or opening bracket.

Transformation of infix notation to postfix notation uses the sequential algorithm from [6]. For this algorithm to work there needs to be defined a precedence table (see Table 2) that tells the algorithm which operator has higher priority.

Table 2. Precedence table (higher precedence value means higher priority)

Operator	Precedence	Associativity
NOT	3	right-to-left
AND	2	left-to-right
OR	1	left-to-right

After all this is done, the postfix notation is sent to the server by jQuery AJAX request.

Evaluating a query statement uses practically the same algorithm that is used for evaluating a logical or mathematical expression [7]. It scans the postfix expression from left to right. Each time it encounters an operand it pushes it onto the stack, and each time it encounters an operation it performs it with the operands on top of the stack. The token tag is represented by an array of poses that have this tag assigned. That means:

- The result of the binary AND operation is an array of poses that have both tags assigned.
- The result of the binary OR operation is an array of poses that have at least one of the tags assigned to them.
- The result of the unary NOT operation is all poses that are not in the specified array.

After a whole statement is processed server responds to the frontend which poses should be displayed.

5. Sequence builder and trainer

The composition of poses into a sequence can be divided into 4 steps. In the first step, the user chooses which poses he wants to insert into his routine. In the second part, the user can rearrange his selected poses. In the third part, for each position, the user has to add how long the athlete has to stay in the given position. And in the last part, the user specifies the name, difficulty, and caption of the routine and can add an identifying photo to the routine. After these parts, the routine is stored in the database and the user can use the yoga sequence trainer to practice the routine.

Yoga sequence trainer consists of 4 parts (see Figure 3):

- Current position,
- the remaining time, which says how long the athlete has to stay in this position,

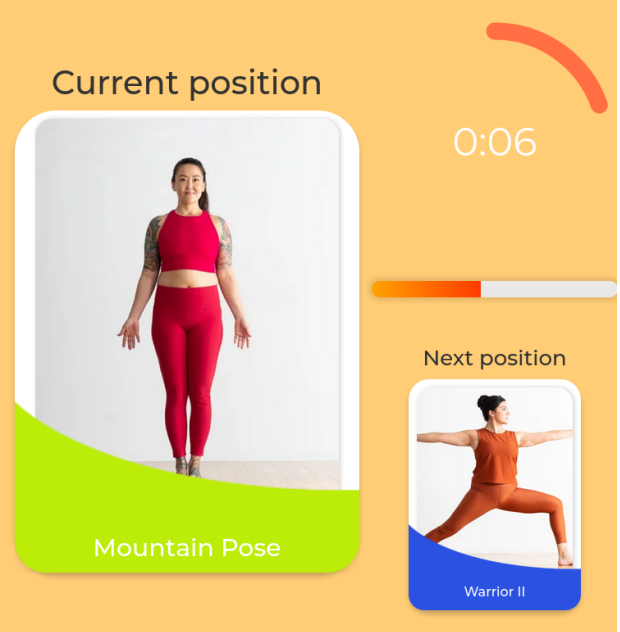


Figure 3. Yoga sequence trainer.

- loading bar that shows what part of the routine the athlete is in
- and the next position.

6. Other application features

There are also other functionalities to improve the user experience:

- Users can freely add tags to poses.
- Users can save their queries.
- Users can share their routines and queries with others.
- Every yoga pose has its own detailed view, which is displayed after clicking on the pose card.
- There are also custom administration pages for the website.

7. Conclusion and Upcoming work

The result of this paper is a web application that offers two different search methods than those commonly used. Livesearch has proven to be a very convenient search method, where users often do not even have to complete a search term and still find a result. Searching with Queries can be confusing for the average user at first, but once the user gets used to this method of searching it can provide more accurate results.

As far as application development is concerned, the programming is practically done, what remains is deep testing with many different users. So I will make the application available to a few users with a test scenario. On this scenario there will be a list of tasks to be completed by the user. I will try to convince

the tester to describe his thoughts to me during the testing session: why he clicks on a particular button on the website? What does he expect when he presses some button on the web application, etc. This whole session will be recorded on camera and I will take notes at the same time. These notes will then help me focus on the biggest flaws. I will be able to review the recording and analyze the user's interactions with the web application in more detail.

The application was tested only by one user till this day. But this single testing showed that the livesearch method suited the particular tester very well and that the Queries language is not complicated at all, even if he is not a person with programming experience.

In the future, the web application could be complemented by a mobile application that would be more focused on creating and training sequences, recording exercise results, counting burned calories, etc..

Acknowledgements

I would like to thank my supervisor prof. Ing. Adam Herout Ph.D. for his help and providing interesting ideas for new features. Another big thank you goes to my girlfriend who helped me a lot with the logo design and was an impartial judge of the application's design.

References

[1] Yoga poses archives. https://www.yogajournal.com/poses/.

[2] Julien. Top 7 best yoga sequence builder apps + 7 tips for planning yoga classes, Feb 2022. https://www.theyoganomads.com/yoga-sequence-builder/.

[3] PHP example - ajax live search. https://www.w3schools.com/php/php_ajax_livesearch.asp.

[4] James Alan Farrell. Anatomy of a compiler, Aug 1995. http://www.cs.man.ac.uk/~pjj/farrell/comp3.html.

[5] Woz U. What is syntax in computer programming?, Dec 2020. https://woz-u.com/blog/what-is-syntax-in-computer-programming/.

[6] Eliezer Dekel and Sartaj Sahni. Parallel generation of postfix and tree forms. *ACM Trans. Program. Lang. Syst.*, 5(3):300–317, jul 1983.

[7] Bohuslav Křena M. Jan Honzík. IAL - 3. přednáška - lineární abstraktní datové typy. Restricted for organization's students, 10 2021.