**Synonym search tool**

**Running**

I deployed app it is on the following link https://synonyms-search-tool.herokuapp.com/ but I have a problem with heroku-postbuild and the frontend is not shown.

Unfortunately, you have to run locally this app to view the app.

1 Go to https://github.com/sejlap/SynonymsSearchTool and clone or download a repository.

2 Open in VS Code the repo and check if you are in SynonymsSearchTool folder

3 Enter npm install and after npm run watch to start the backend of the application

4 Split your terminal or add a new terminal and click cd search\_tool

5 Enter npm install and after npm start the react app will start

6 The frontend is running on http://localhost:3000/ and the backend and stored data on the http://localhost:5000/sinonimi

**Frontend explanation**

For the frontend, I realized that have little components just components for searching and adding words and synonym. I prefer to use a white or simple design for frontend if there are complex apps with a lot of components but in this case, I decided to make a little different UI and I worked with complement colours dark blue and orange. So, if we look at the psychology in colours blue is the colour that shows trust and strength and orange is a colour that shows enthusiasm and confidence and for me, it was the right choice.

First page Home page You can enter a word and search a synonym If you didn't enter words the message is „First you have to add form“. If you click a right button then page redirect you to the add synonym page where you can add word and synonym. If you have more than one synonym you have to enter it whit whitespaces without a comma.

If you enter a duplicate word or empty inputs the message is „Duplicates and empty inputs are not allowed“. If everything is OK the message is „Successfully added“.

On the frontend, I create Component, Cards using React bootstrap. The word and synonym I put in a JSON object and using Axios I send that data to backend. On the backend side, I store it at a different place. The sent JSON objects are on the route http://localhost:5000/sinonimi and at the same route, I save the data and after getting the specific data for searching. Unit tests are available.

**Backend explanation**

The question is where to store data to have the best complexity and big O notation.

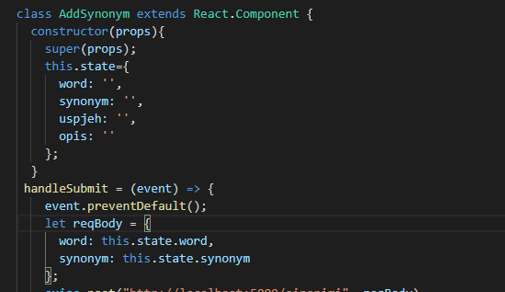
Firstly I think about hashmap and about a Javascript map which has under the hood hash because of the inserting and searching the data have complexity O(log n). When I finally put that object into the map I realized that I need to put that in the array or other structure because I lose the previously inserted map and after I realized that Objects are similar to map and putting the map in the array that means that inserting and searching objects would be complexity O(n) and finally it was a bad decision.

When I realized that previous decision is bad I decided to put JSON objects into the linked list. The class Linked list that I implement with Node and 2 attributes head pointer and size variable.In this way, the adding function have now complexity O(1) and it is so good for time running the operation. But searching objects is still linear so I try to implement binary search because the binary search in the linked list has O(log n) complexity. The problem here is that elements of the list are string objects and we need to sort that objects in lexicographical order and after to call function mid to find a middle element and another function which will compare strings in a lexicographical way. After all of these circumstances, the complexity would be O(nˇ2). So I decided to implement a classic search. Also, tests are available on npm test in the terminal for the backend.

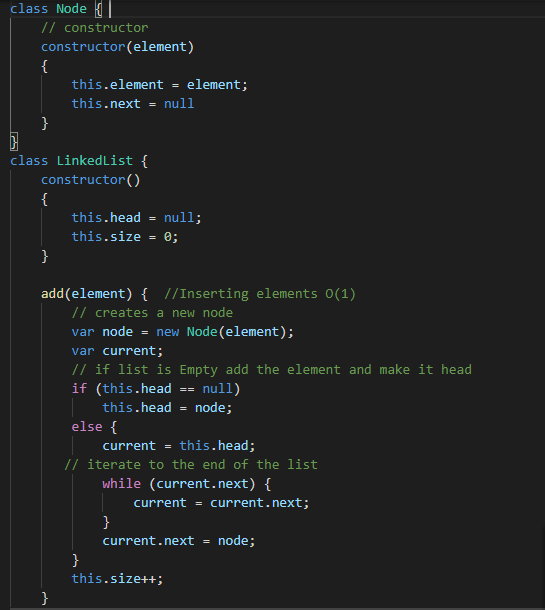
**SOLID principles and design patterns in the app**

**Single responsibility principle**- A class should have one and only one reason to change, meaning that a class should only have one job.

In this case, I have separate classes for each component where every class have one job.

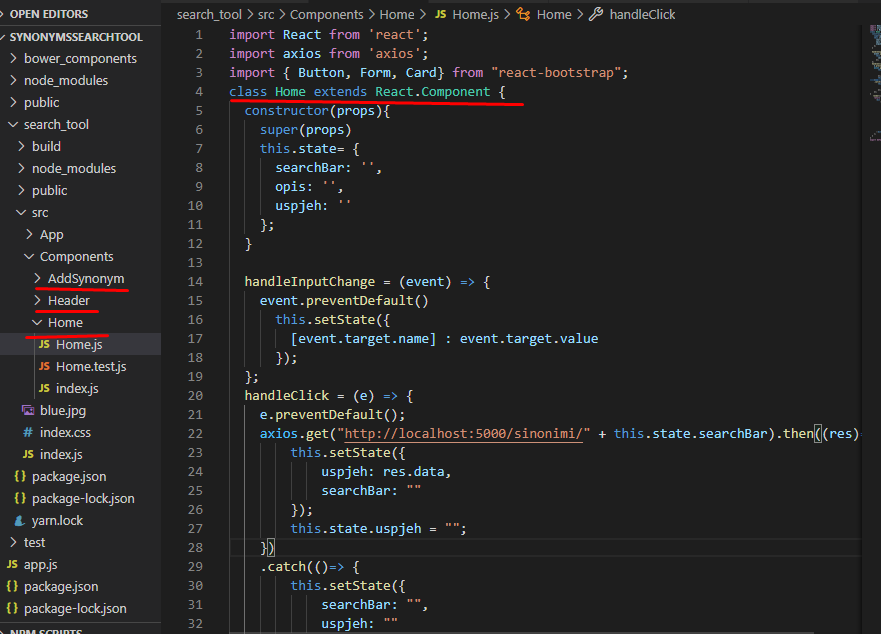


**Open-closed Principle**- Objects or entities should be open for extension, but closed for modification. This principle is visible in the backend. Where I implemented a class Linked List and this class is open for extension but not for class modification.

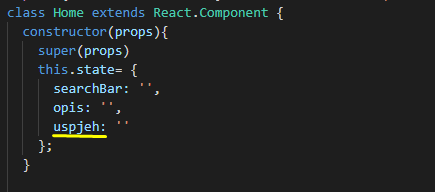


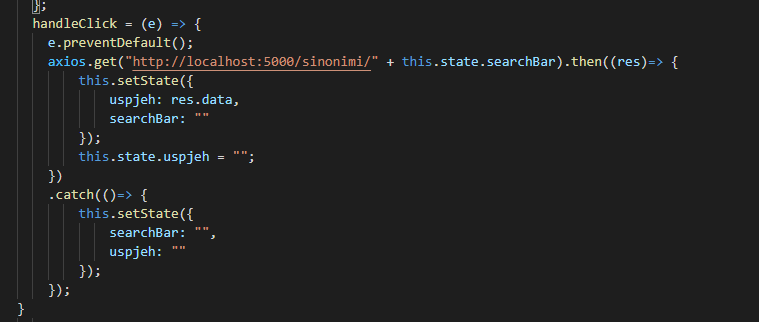
**Interface segregation principle**  
  
A client should never be forced to implement an interface that it doesn’t use or clients shouldn’t be forced to depend on methods they do not use.  
I implemented this principle in the way of Components on frontend where every component have own file and those components won't force the methods that clients won't use.

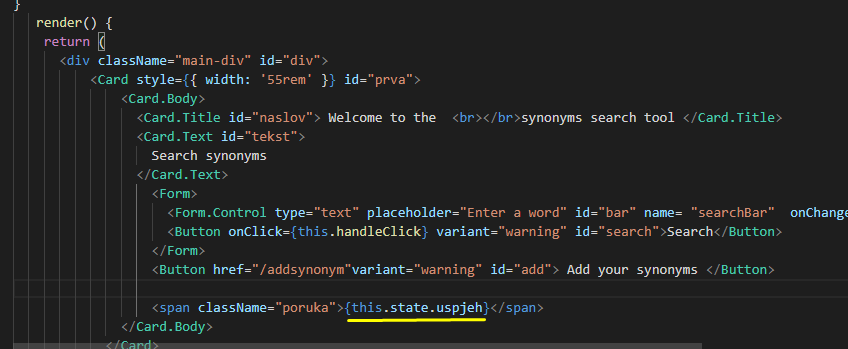
**Dependency inversion principle**  
Entities must depend on abstractions, not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions.  
  
The class Linked list is the high-level module and not depend on the low-level modules like other variables and functions. Exactly other variables and functions depend on Linked List.

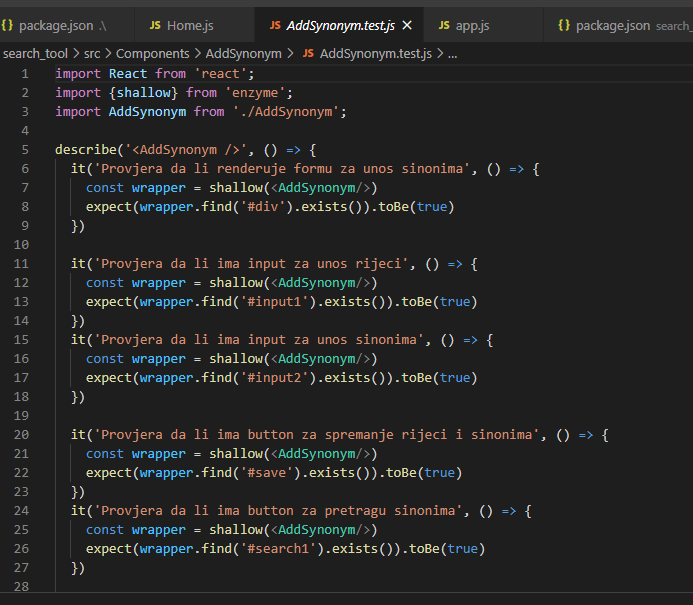
**Design patterns**  
**Module Design Pattern**This pattern is the most common in React.js. It helps us to make decomposition of our project.   
Classes and folders are in Components folder where are module design and it helps us to make a distinct view of our project too.

**Singleton pattern**  
Maybe my favourite pattern because Singleton allows many instances of the same object.  
I didn't use it directly but in the code, in the frontend and backend, you can see the instances of the objects.







**Prototype Design Pattern**  
  
In AddSynonym.test.js I created the shallow objects and Prototype Design Pattern have the objects which are clones (shallow clones) of the original object that are passed around.

**Testing**

Add synonyms

Word: test

Synonyms: analysis approval assessment experiment

Word: wash

Synonyms: bathe shower clean shampoo

Word: clean

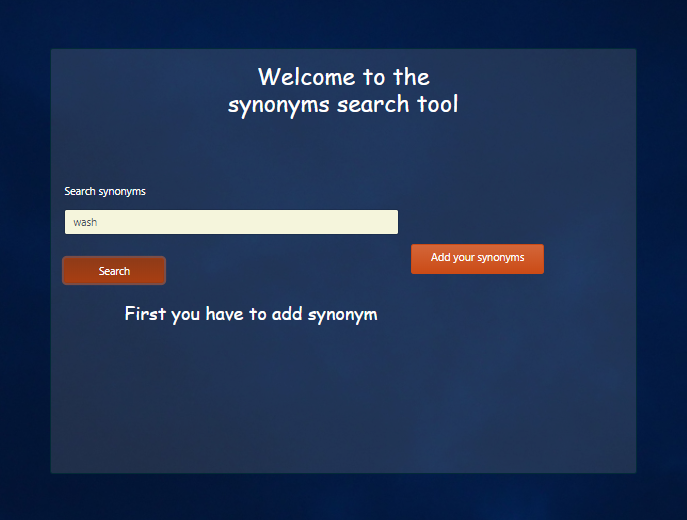
Synonyms: blank bright clear fresh

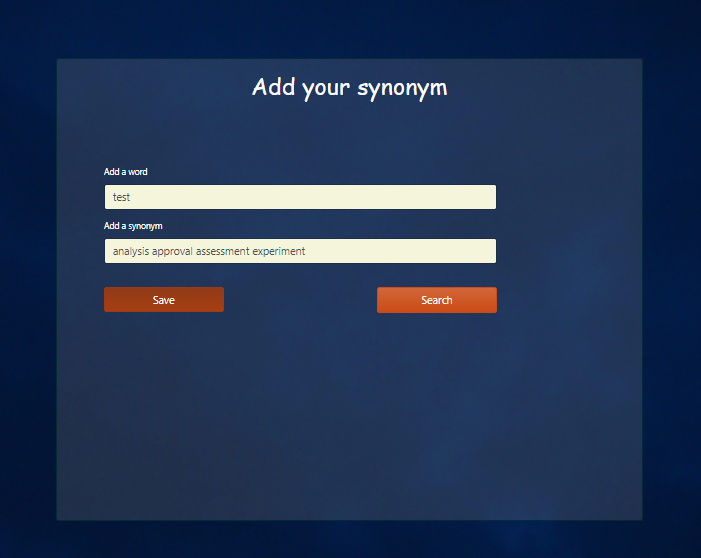
Word: white

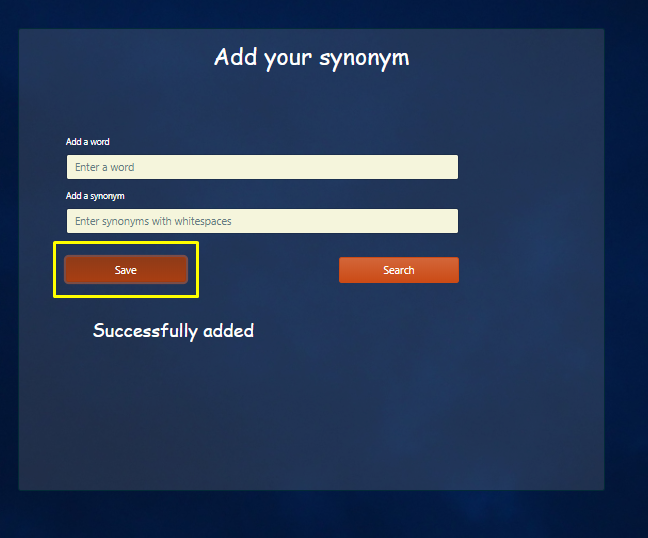
Synonyms: fair ivory light neutral

Word: light

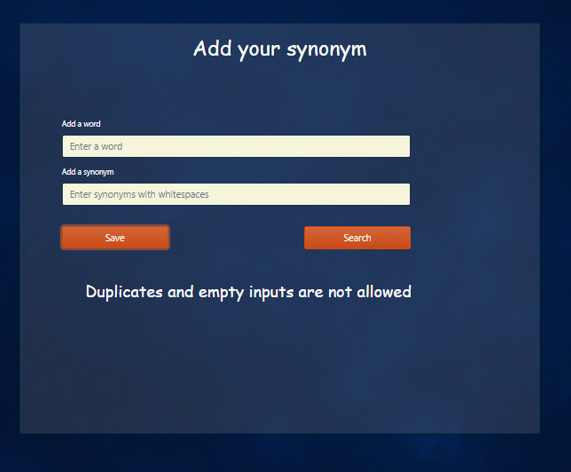
Synonyms: bright luminous rich shiny



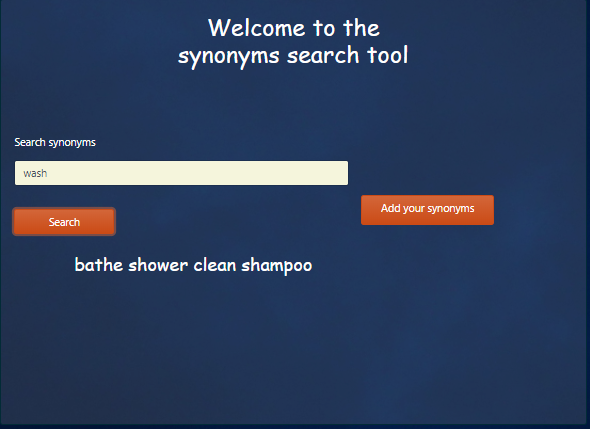




If I try to enter duplicates or empty inputs the message I get is:



When I add all the elements above then I can search now.



Lookups enabled

