**Good Design practises and Coding Guidelines**

**Assignment: Search Engine –STAR method**

**Situation**

In modern days, people are always searching for something online. Every app, software that we use has an option called “search” to access desired data. Often this search bar acts as the main interface between the consumer and producer. There is always a need for efficiently performing search engine.

**Task**

I’m going to build a search engine for story books online. Considering the enormous amount of data available it is essential to design a search engine that performs efficiently. A basic search engine would have the 3 components to be implemented.

* A very simple User Interface that has a text box where user can type the request and that is sent to the search backend. The backend should return matched sites sorted by relevance, each with URL, title and a small description. These should be displayed to the user.
* A search backend that should receive and accept the search request and implement relevant algorithm to find website and then sort it based on the relevance.
* An index builder that should be constantly be running to build and redefine the search engine each time.

**Approach**

In order to build the search engine first I would implement it with say 10000 documents. Once this is in the flow and work I can implement it with more number of documents (URLs).

Split the 3 components to many subtasks and implement it in an orderly manner.

* First I would collect a set of URLs the gives the story books to read online and store it in the database.
* Implement the index builder which is the complex part. This starts for some sites in the database and searches in further sites if specified in the sites listed in the database.
* The backend will receive a search query and it first tokenizes it to words. For each word the backend hits the search index.
* Given a word, it will point to the list of URLs with that particular word.
* The searching method would be something like,

**Search request Search Index URL Results**

potterHarry2

Harry1

Potter2

Potter1harry

Harry

Potter

Harry1

Potter2

Potter harry

Harry potter

* **The** UI must have a text box where we can give the search request build using html, css and other frontend languages.
* The HTTP request can be sent to server (backend) and the URLs is fetched based the relevance.

A simple search engine would have class

|  |
| --- |
| #with variables like:  Class search{  String request;  String[] url;  Etc;  # With methods like:  Void split\_request \_word;  Void search\_word;  Void retrieve\_url;  Void sort\_url; |

Thus the search engine can be built by implementing module wise followed by unit testing, by increasing scalability and making it efficient.

**Results**

The final result would be a webpage ,which would act as a user interface that fetches the results and display it. Further the program can be optimized by changing various parameters if necessary.