A

PROJECT REPORT

ON

**Word-net Visualization**

B.Tech (CE) Sem-VI

In fulfilment of all requirements for

**Bachelor of Technology**

**In**

**Computer Engineering**

**Submitted by**

**Bhimani Sanket R. (CE-007)**

**Dave Meet N. (CE-)**

**Ka. Patel Dhara B. (CE-052)**

Under the Guidance of

**Prof. Brijesh S. Bhatt**



**DEPARTMENT OF COMPUTER ENGINEERING**

**FACULTY OF TECHNOLOGY,**

**DHARMSINH DESAI UNIVERSITY**

COLLEGE ROAD, NADIAD- 387001

**DHARMSINH DESAI UNIVERSITY**

**NADIAD-387001, GUJARAT**



**CERTIFICATE**

**This is to certify that the project carried out in the subject of System Design Practices entitled “Word-net Visualization” and recorded in this report is a work of**

1. **Bhimani Sanket R. ROLL NO: CE-007 ID: 14CEUOG069**
2. **Dave Meet N. ROLL NO: CE-025 ID: 14CEUOS075**
3. **Ka.Patel Dhara B. ROLL NO: CE-052 ID: 14CEUOS056**

**Of Department of Computer Engineering, semester VII. They were involved in Project developing during academic year 2016 -2017.**

**Prof. Brijesh S. Bhatt (Project Guide)**

**Department of computer Engineering**

**Faculty of Technology,**

**Dharmsinh Desai University, Nadiad**

**Dr. C.K.Bhensdadia, (Head of Department)**

**Department of Computer Engineering**

**Faculty of Technology,**

**Dharmsinh Desai University, Nadiad**

**Contents**

1. **Introduction…...............................................................................................................5**
   1. **Technology Used............................................................................................5**
   2. **Project details : Broad specifications……………………………....……………………………6**
   3. **Lexicographical entry file…………………………………………….………………………………..7**
2. **Software Requirement Specification.............................................................................9**
   1. **Introduction ..................................................................................................9**
   2. **Purpose..........................................................................................................9**
   3. **Scope .............................................................................................................9**
   4. **Hardware Requirements ................................................................................9**
   5. **Software Process Model...............................................................................10**
   6. **User Characteristics .....................................................................................10**
   7. **Non-functional Requirements ......................................................................10**
   8. **Installation Requirements ............................................................................10**
   9. **Functional Requirements .............................................................................11**
3. **System Design ...........................................................................................................12**
   1. **Use-case Diagram ........................................................................................12**
   2. **Class Diagram ..............................................................................................13**
   3. **Sequence Diagrams ......................................................................................14**
   4. **Activity Diagrams .........................................................................................15**
4. **Tools and Technology.................................................................................................16**
   1. **To realize Large File Management................................................................16**
5. **Implementation.........................................................................................................19**
   1. **ImplementationEnvironment.......................................................................19**
   2. **Implementation Description.........................................................................19**
   3. **Module Description......................................................................................20**
6. **Testing………………………………………………………………………………………………………….………….21**
   1. **Testing Plan..................................................................................................21**
   2. **Testing Strategy...........................................................................................21**
   3. **Testing Methods…........................................................................................21**
   4. **Test Cases....................................................................................................22**
7. **Screenshots……………………………………………………………………………………………………………...23**
   1. **Search result of a word……………………………………………………………………………....23**
   2. **Clicking on ‘View all words’ button………………………………………………………….….23**
   3. **Viewing word by particular definition…………………………………………………………24**
   4. **Focusing on any word (By left click)………………………………………………………….…24**
8. **Limitations and Future enhancements .......................................................................25**
   1. **Limitations………………………………………………………………………………………………....25**
   2. **Future enhancements………………………………………………………………………….....….25**
9. **Conclusion ................................................................................................................26**
10. **Bibliography……………………………………………………………………………….…………………………….27**

**Abstract**

We often need to use the dictionary. Using that conventional dictionary is more time consuming and uninteresting. Here the Word Search Software helps you in that. The WSS is used to visualize the word and its relations according to different definitions. It realize the concept of Large File Management. It efficiently parse the XML file and find the word in less time also helps user to visualize the relations. WSS provides better and interesting GUI for user to interact.

1. **Introduction**
   1. **Technology Used**

**IDE:** NetBeans

NetBeans IDE 8.1 is the official Integrated Development Environment (IDE) for java applications development. Any application of java can be developed by NetBeans.

**Development Language:** Java

Javais a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

**Mark-up Language:** XML

In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine readable.

**Scripting Language:** Java Script

JavaScript("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites.

**Diagram Tool:** UMLet

UMLet is a UML tool .It is used for creating UML diagrams and other various types of diagram.

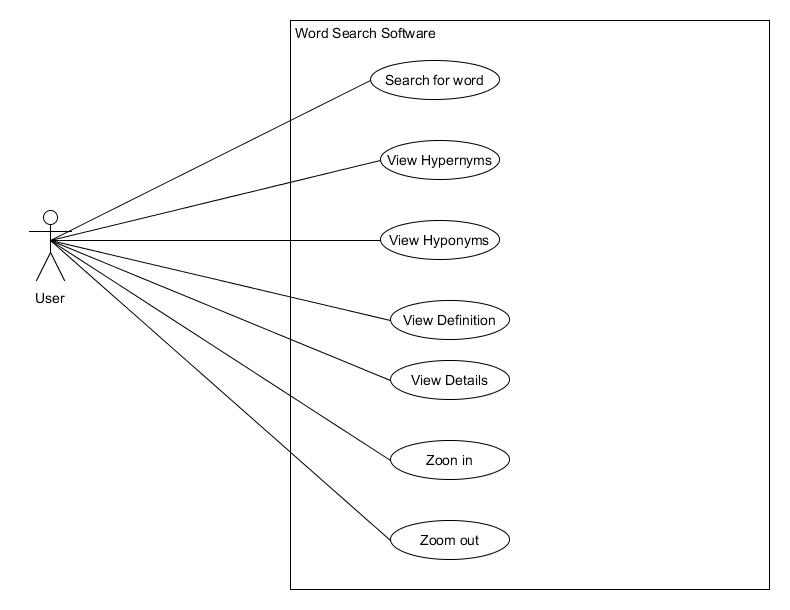
**Intermediate response:** JSON

Java Simple Object Notation is a used to store and retrieve data. It’s a data standard format for data storage gives a specific format to data.

* 1. **Project details : Broad specifications**

Word Search Software is an web application. It is a realization of concept Large File Management System. WSS is used to find word and its meaning. It also helps clients to visualize the word and its relations with other word via graph.

WSS is very simple to use software. User can type in any word, WSS gives all definitions of that word with corresponding hyponyms and hypernyms. WSS has an easy access feature which gives convenience to client to see the related word of particular definition. WSS also give the zoom-in and zoom-out option which makes easy for client to interrelate the words. Right clicking on any word will give search result for that particular word.



* 1. **Lexicographical entry file**

In [mathematics](https://en.wikipedia.org/wiki/Mathematics), the lexicographic or lexicographical order (also known as lexical order, dictionary order, alphabetical order or lexicographic (al) product) is a generalization of the way the [alphabetical order](https://en.wikipedia.org/wiki/Alphabetical_order) of words is based on the alphabetical order of their component letters. This generalization consists primarily in defining a [total order](https://en.wikipedia.org/wiki/Total_order) over the [sequences](https://en.wikipedia.org/wiki/Sequence) (often called [words](https://en.wikipedia.org/wiki/String_(computer_science)) in [computer science](https://en.wikipedia.org/wiki/Computer_science)) of elements of a finite [totally ordered set](https://en.wikipedia.org/wiki/Totally_ordered_set), often called [alphabet](https://en.wikipedia.org/wiki/Alphabet_(formal_languages)).

There are several variants and generalizations of the lexicographical ordering. One variant widely used in [combinatory](https://en.wikipedia.org/wiki/Combinatorics) orders subsets of a given [finite set](https://en.wikipedia.org/wiki/Finite_set) by assigning a total order to the finite set, and converting subsets into increasing sequences, to which the lexicographical order is applied. Another generalization defines an order on a [Cartesian product](https://en.wikipedia.org/wiki/Cartesian_product) of [partially ordered sets](https://en.wikipedia.org/wiki/Partially_ordered_set); this order is a total order if and only if the factors of the Cartesian product are totally ordered.

The snapshot of file used here are:

1. **EWNlexicalEntry.xml** (672809 lines)

It is an XML file which is consisting of word id and actual word.

**Example:**

***<?xml version="1.0" encoding="UTF-8" standalone="no"?>***

***<Lexicon>***

***<LexicalEntry>***

***<Lemma partOfSpeech="r" writtenForm="'tween"/>***

***<Sense synset="eng-3.0-00250898-r"/>***

***</LexicalEntry>***

***</Lexicon>***

**Lexicon**: shows the root element

**LexicalEntry**: represents entry for particular word

**WrittenForm**: represents actual word

**Synset**: represents the id of the word

1. **EWNsynset.xml** (1101043 lines)

It is an XML file which is consist of definition(sense) and relation of word

**Example:**

***<?xml version="1.0" encoding="UTF-8" standalone="no"?>***

***<synset baseConcept="1" id="eng-3.0-00002325-v">***

***<Definition gloss="undergo the biomedical and metabolic processes of respiration by taking up oxygen and producing carbon monoxide ">***

***<Statement example=""/>***

***</Definition>***

***<SynsetRelations>***

***<SynsetRelation relType="has\_verb\_group" target="eng-3.0-00001740-v"/>***

***<SynsetRelation relType="has\_hypernym" target="eng-3.0-02108377-v"/>***

***</SynsetRelations>***

***</synset>***

**Id**: Id of the word

**Gloss**: represents the definition of the word

**relType**: shows the relation type

**target**: shows the id of related word

.

1. **Software Requirement Specification**
   1. **Introduction**

The best way to understand the relation of word and its definitions is via graph. And it is more convenient if the related words are also shown via graph. So our Word Search System does that thing for us. It handles the very much large file and search word and relations efficiently in less time and without overflowing the available RAM memory.

* 1. **Purpose**

The purpose of this is to demonstrate the Large File management. Here in this application works with large XML file .The application is efficiently designed so that it finds the desired data in least time. It is basically an incremental project. Main purpose here is to affectively traverse the XML file and prevent the memory buffer overflow. Another purpose of the system is to visualize the word and its relations.

* 1. **Scope**
* Software like Dictionary by the name “Word Search Software” is to be built.
* The software will allow user to enter a word and then system will provide the related definitions and other related words.
* User can also view graph of just particular definition and corresponding related words.
* User can also see the most recent searched words.
* User can also see the various statistics about the data.
  1. **Hardware Requirements**

Computers having 2 GB RAM will be target audience

* 1. **Software Process Model**
* The application would be developed using Agile method of software development.
* Agile software development (Agile) is a collection of software development methodologies that promote adaptive planning, evolutionary development and delivery, continuous improvement, and a time-boxed period of time to complete a body of work.
* This method is usually adopted when the project size is relatively small, experience of the developers in the domain is less and the requirements aren’t clear as well as they are volatile.
  1. **User Characteristics**

**End User:** User should have basic knowledge of how to interact with browser

* 1. **Non-functional Requirements**
* Reliability:

Users should be prompted with appropriate message when the application isn’t able to process word or isn’t able to load JavaScript file due to change in the path of the file explicitly by the user.

* Performance:

Response time will be kept as low as possible in order to ensure promptness for the user.

* Usability:

User friendly user interface must be designed in order to ensure ease of use.

* 1. **Installation Requirements**

The system to be developed requires any web browser to be present on client’s machine.

* 1. **Functional Requirements**

**R1:** System must provide word’s definitions and related words

**Input:** Word

**Output:** word’s definitions and related words in graph format

**R2:** System must provide facility to focus the word

**Input:** left clicking the word

**Output:** Selected word will be in centre

**R3:** System must provide word list by various definitions

**Input:** clicking on ‘view all words’ button

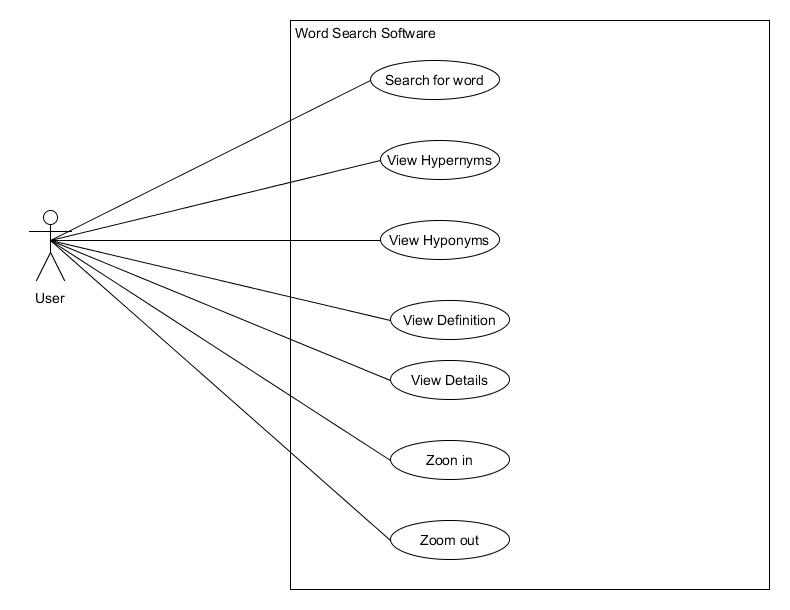
**Output:** list of words will be displayed for various definitions

**R4:** System should search the word whenever right clicked on it

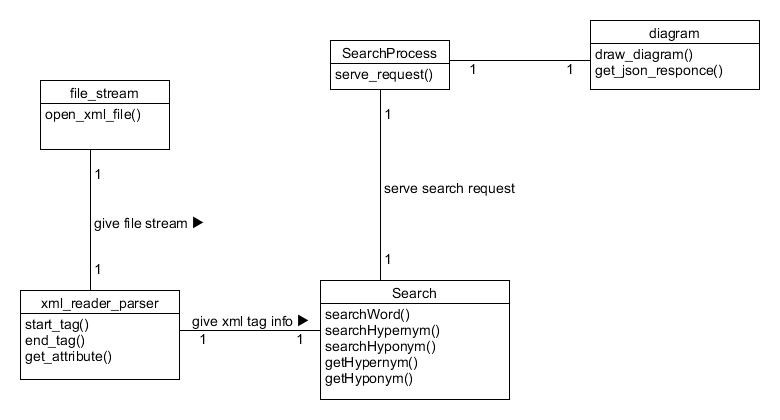
**Input:** right click on word to be searched

**Output:** graph will be displayed with new word’s definitions and related word

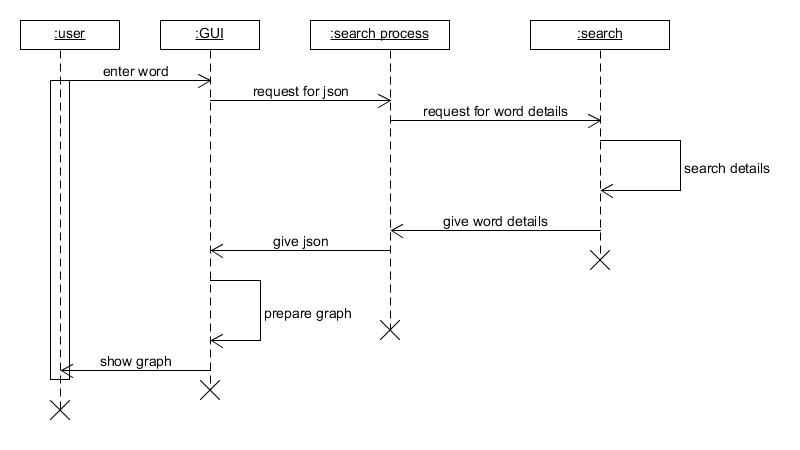
1. **System Design**
   1. **Use-case Diagram**



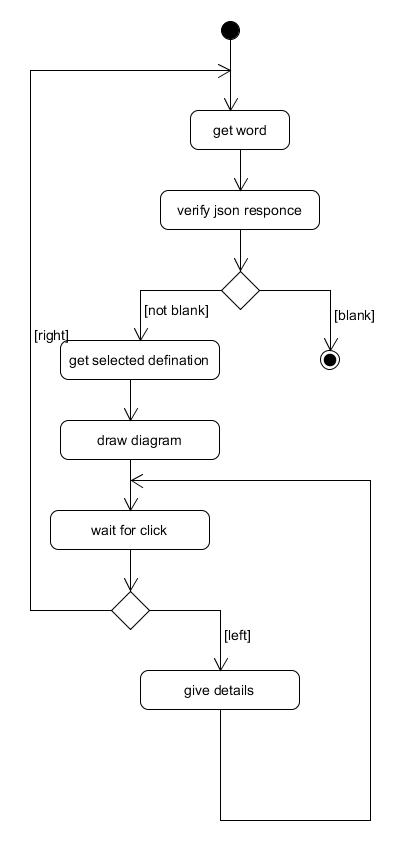
* 1. **Class Diagram**

****

* 1. **Sequence Diagram**

****

* 1. **Activity Diagram**

****

1. **Tools and Technology**
   1. **To realize Large File Management**

**Information about SAX parser:**

SAX (the Simple API for XML) is an event-based parser for xml documents. Unlike a DOM parser, SAX parser creates no parse tree.

**Characteristics of SAX parser:**

* API type: push, streaming
* Easy to use
* CPU and memory efficient
* Forward only
* Read only

**It works in below steps:**

* Reads an XML document from top to bottom, recognizing the tokens that make up a well-formed XML document.
* Tokens are processed in the same order that they appear in the document
* Reports the application program the nature of tokens that the parser has encountered as they occur.
* The application program provides an "event" handler that must be registered with the parser
* As the tokens are identified, callback methods in the handler are invoked with the relevant information

**SAX parser is used when:**

* The file is too big to be loaded in main memory. DOM parser takes much memory to build the tree.
* The problem to be solved involves only part of the XML document
* Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream.

**Disadvantages of using SAX:**

* We have no random access to an XML document since it is processed in a forward-only manner
* If you need to keep track of data the parser has seen or change the order of items, you must write the code and store the data on your own

**Methods used:**

* ***void startElement(String uri, String localName, String qName, Attributes atts)***
  + - Called at the beginning of an element.
* ***void endElement(String uri, String localName,String qName)***
  + - Called at the end of an element.
* ***public void parse(InputSource source)***
  + - Parse an XML document.

These methods notify application about each event like element start or end. So we can design our logic in a way that we can parse and search whole file.

**Example:**

***public class SaxHandler extends DefaultHandler {***

***public void startElement(String uri, String localName,***

***String qName, Attributes attributes) throws SAXException {***

***//code***

***}***

***public void endElement(String uri, String localName,***

***String qName) throws SAXException {***

***//code***

***}***

***private String currentElement() {***

***//code***

***}***

***private String currentElementParent() {***

***//code***

***}***

***}***

This code is a handler we need to link this class with parser for that below code will be useful,

***public static void main (String argv []) {***

***SAXParserFactory factory = SAXParserFactory.newInstance();***

***try {***

***InputStream xmlInput =***

***new FileInputStream("data\\sax-example.xml");***

***SAXParser saxParser = factory.newSAXParser();***

***SaxHandler handler = new SaxHandler();***

***saxParser.parse(xmlInput, handler);***

***for(Driver driver : handler.drivers){***

***System.out.println(driver);***

***}***

***} catch (Throwable err) {***

***err.printStackTrace ();***

***}***

***}***

1. **Implementation**
   1. **Implementation Environment**
   * Java (NetBeans)
   * XML
   * Java script, HTML, CSS
   * JSON
   1. **Implementation Description**

* Implementation is divided into two parts,
  + Client side, where in designing, HTML and CSS used to make visually affective pages. And JavaScript to make graph. Here for designing we have used *“materializedcss”* framework. And for making graph we have used *“jit”* library.
  + Server side, where we are using STAX parser to parse large XML file and then this response is converted to JSON and send to client.
* So, basically communication between client and server is done throw JSON encoding.
* Server side is also divided in to two part,
  + One is searching part
  + Another is to encode the response in to JSON.

XML file

Search from XML file

Convert Hashmap to JSON

Make graph from JSON

Hashmap

JSON resp

When user search any word then from client side AJAX request sent to server and server gives all details for that word in JSON format and this JSON format then converted to graph at client side.

So responsibility of server is to generate only JSON response

* 1. **Module Description**
  + Finding the details from XML files
    - Abstract: It is used to find all details from xml files
    - Input: word
    - Output: Hash map containing all details of provided word
    - Processing logic: It finds the details of that word from large XML files using SAX parser. This generates events like start elements, end elements and like that which is useful to build logic.
  + Generating JSON response
    - Abstract: It generates the JSON response for client.
    - Input: Hash map containing all information related word.
    - Output: JSON response
    - Processing logic: It creates JSON response by parsing Hash map and using some JSON library.
  + Displaying Graph:
    - Abstract: It displays good graph on browser by parsing server response.
    - Input parameter: JSON response
    - Output parameter: Provides contains for canvas tag which causes the animated graphical graph.
    - Processing logic: Here we are using JIT library which convers JSON response to graph.

1. **Testing**
   1. **Testing Plan**

The testing is a technique that is going to be used in the project is black box testing the expected inputs to the system is applied and only the outputs are checked.

* 1. **Testing Strategy**

The development process repeats this testing sub process a number of lines for the following phases

.

* Unit Testing
* Integration Testing

Unit Testing tests a unit of code after coding of that unit is completed. Integration Testing tests whether the previous programs that make up a system, interface with each other as desired. System testing ensures that the system meets its stated design specifications. Acceptance testing is testing by users to ascertain whether the system developed is a correct implementation of the software requirements specification.

Testing is carried out in such a hierarchical manner to that each component is correct and the assembly/combination of component is correct. Merely testing a whole system at end would most likely throw up errors in component that would be very costly to trace and fix. We have performed both Unit Testing and System Testing to detect and fix errors.

* 1. **Testing Methods**

We have performed Black-box testing for the testing purpose. A brief description is given below:

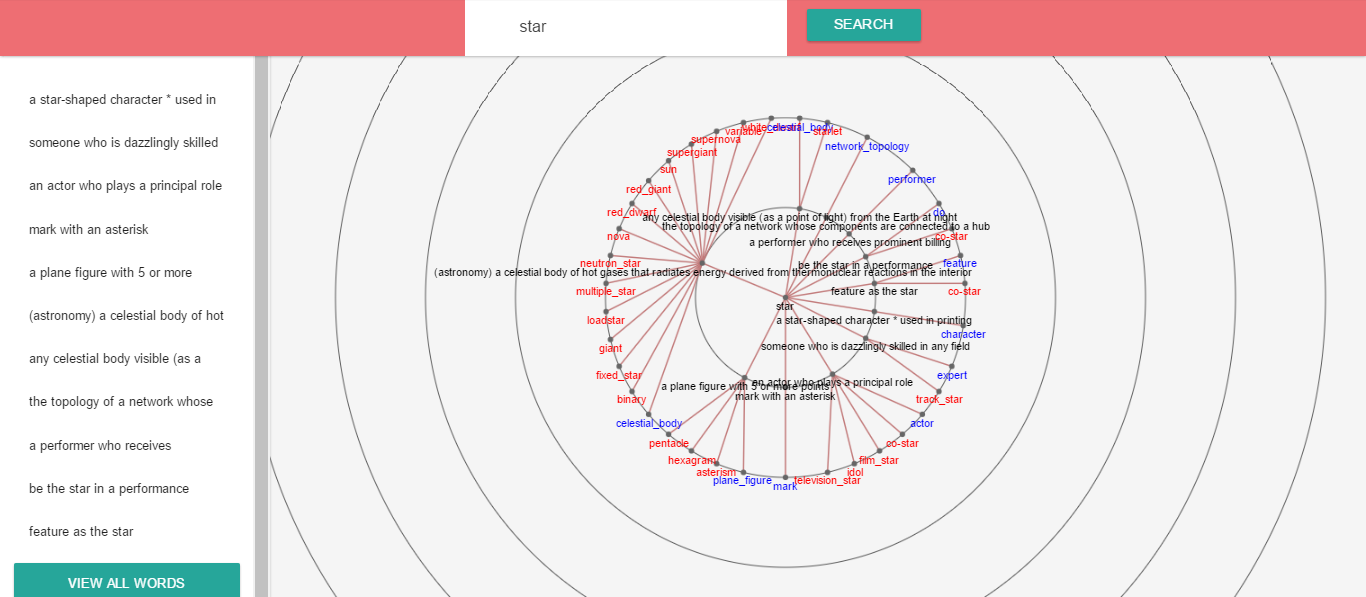
Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

* 1. **Test Cases**

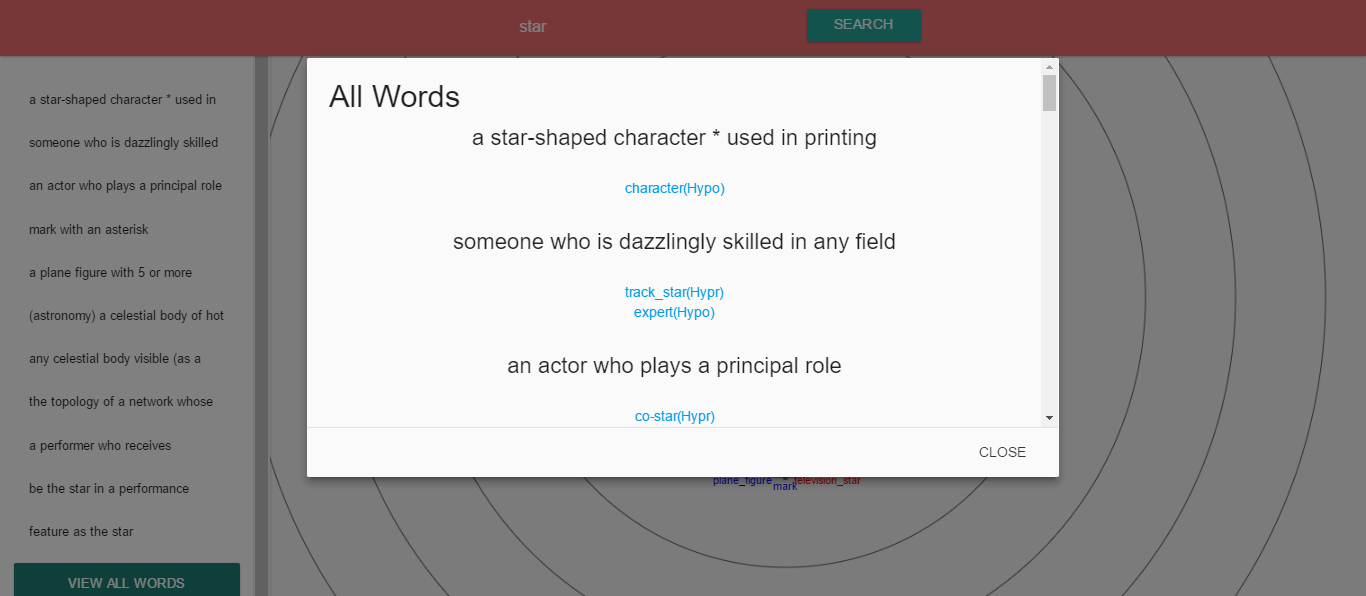
Test cases are described below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test**  **Scenario** | **Test**  **Steps** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** |
| 01 | View search  result | 1.open app  2.Enter word | Word to search | 1.Word’s meaning and related word graph if it is in XML file  2.NULL result will be there if the word is not in source XML file | As per  Expectation |
| 02 | View result  By particular  definition | Click on  Any definition from side panel | Definition | Definition with related words | As per Expectation |
| 03 | View all Word | Click on ‘Show all word’ button | Click on button | List of related words according to meaning | As per Expectation |
| 04 | Focusing on word | Left click on word in graph | Left click  On word | Word in centre of graph | As per Expectation |
| 05 | Exploring word from  graph | Right click  On word in graph | Right click on word | New word with all its definitions and related  words | As per Expectation |

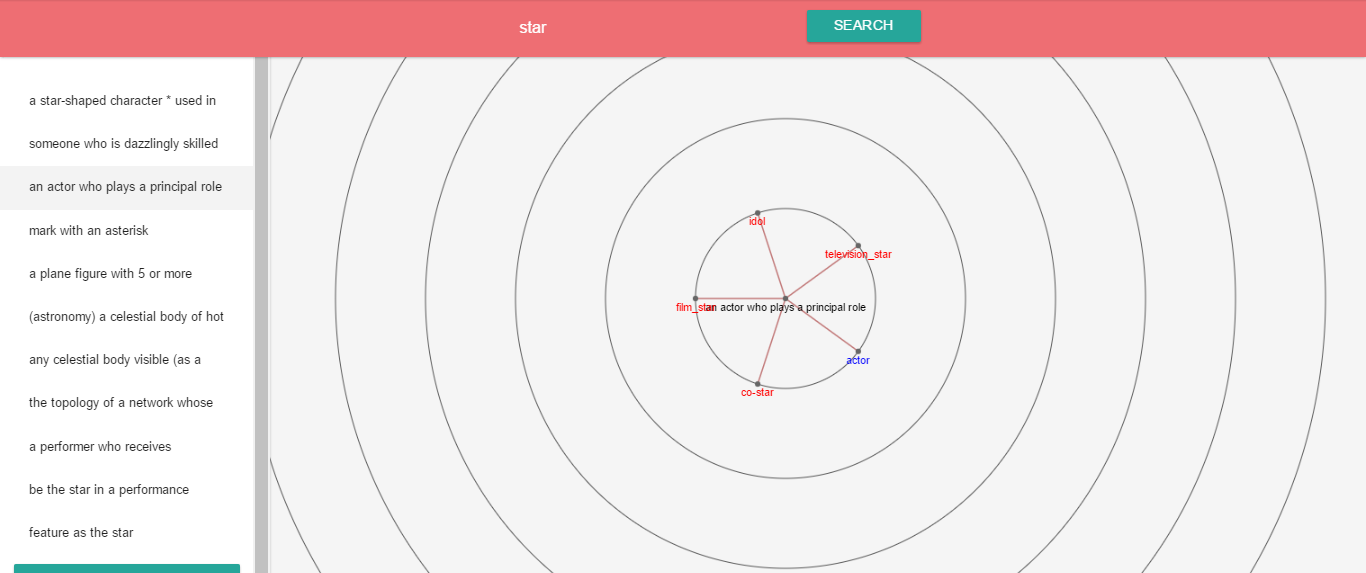
1. **Screenshots**
   1. **Search result of a word**

****

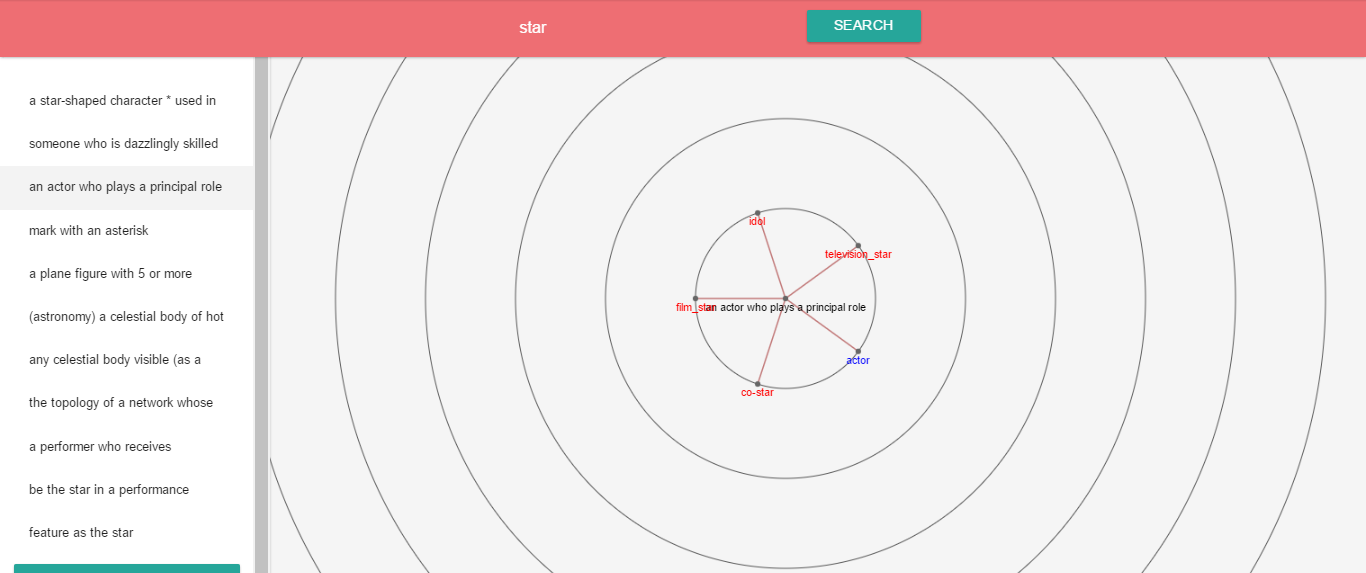
* 1. **Clicking on ‘View all words’ button**

****

* 1. **Viewing word by particular definition**

****

* 1. **Focusing on any word (By left click)**

****

1. **Limitations and future Enhancements**

* 1. **Limitations**
* It takes little more time for displaying results.
* If the files used are more large, memory overflow can occur.
  1. **Future Scope**
* WSS is recently working for English language only; it can be extended in future for more other languages.
* Searching process could be faster.

1. **Conclusion**

The WSS helps user to better visualize the word and related definitions by using the graph and helps user to focus on word by various animation. It uses JIT Rgraph and json for displaying the graph. Basically WSS work on large XML files to realize the concept of Large File Management. It provides a new level to the data visualization.

1. **Bibliography**

* JIT graphs:

<https://philogb.github.io/jit/static/v20/Docs/files/Visualizations/RGraph-js.html>

* XML parser

<https://www.tutorialspoint.com/java_xml/java_stax_parser.htm>