

CS 392 - Capstone Project I



Project 2 – O₂ : A Semantic Search Engine

Group Members -

Aakash Sinha	U101114FCS214
Deep Parikh	U101114FCS185
Karanjit Singh Gill	U101114FCS077
Mohit Raj	U101114FCS394
NVS Abhilash	U101114FCS223
Prateek Agarwal	U101114FCS389

NU Faculty Mentor - Mr. Gaurav Sharma

NIIT Faculty Mentor - Mr. Pradeep Nandi
Mr. Bhavesh Sangwan

Index :

1. Abstract	2
2. Objectives	2
3. Architecture Overview	2
4. General Search Engine and Features of O ₂	3
5. Work Timeline	4
6. Phase 1 : Wireframe and Design	5
a. WireFrame of O ₂	5
b. Schematic Diagram of O ₂	6
c. Progress after Phase 1	6
d. O ₂ after Phase I	7
7. Phase 2 : Implementation of Elasticsearch Capabilities.....	8
a. What is Elasticsearch	8
b. Indexing in Elasticsearch	8
c. Creating Index.....	9
d. Querying	9
e. Elasticseatch RESTful API	10
f. Progress after Phase I	10
g. O ₂ after Phase II	11
8. Phase 3 : Implementation of Semantic Search	12
a. Word Co-occurrence	12
b. Query Expansion	12
c. Word2Vec.....	13
d. Implementing Semantic Search	13
e. Progress after Phase III	13
f. O ₂ after Phase III	14
9. Tools and Technologies Used.....	15
10. Contribution of Team Members.....	16
11. Future Work	16
12. Project Timeline – Gantt Chart.....	17
13. References	17

Abstract (What is O₂):

O₂ leverages Google's search capabilities to implement a semantic search. The core idea in o2 is to allow a user to find documents based on the Intent within the document. For example a user may want to search for an advanced or a basic document on Java programming. The idea of "advanced" or "basic" is represented as an intent graph. A set of terms that if present in the document, reinforces the fact that the document is indeed "basic" or "advanced" in nature. In addition to intents, o2 uses a vocabulary of terms around which these indexes are to be built. For example, o2 can work with the complete set of topics, a student learns in Java and then act as a semantic search engine for these topics.

Objectives :

- To Develop a semantic search engine which uses **ElasticSearch** capabilities for searching documents, stored remotely
- To provide a **minimalist UI** which has upload and search options
- The **Upload** function helps the user to upload all kinds of documents and store them in the local database.
- The **Search** function implements Elasticsearch and query expansion to increase the size of retrieved documents.
- It will be used as a part of an ongoing live project at **NIIT Ltd.**

Architecture Overview :

- Developed a dynamic web application in Eclipse framework.
- Based on Java Server Pages (JSP) and Java Servlets.
- Handled Elastic Search in java for document indexing, searching and retrieval.
- Semantic search is implemented using word2vec and gensim library in Python.

General Search Engines :

Search engines search documents related to the keywords typed into the search engines portals within a very quick time period and submit results from best to worst relating to relevance.

The following takes a look at the statistics and facts relating to Search Engines.

- 84% of submissions are done manually.
- 45% of searches are keyword searches / phrase queries.

Reference : <https://www.optimus01.co.za/search-engine-stats-and-facts/>

Features of O₂ :

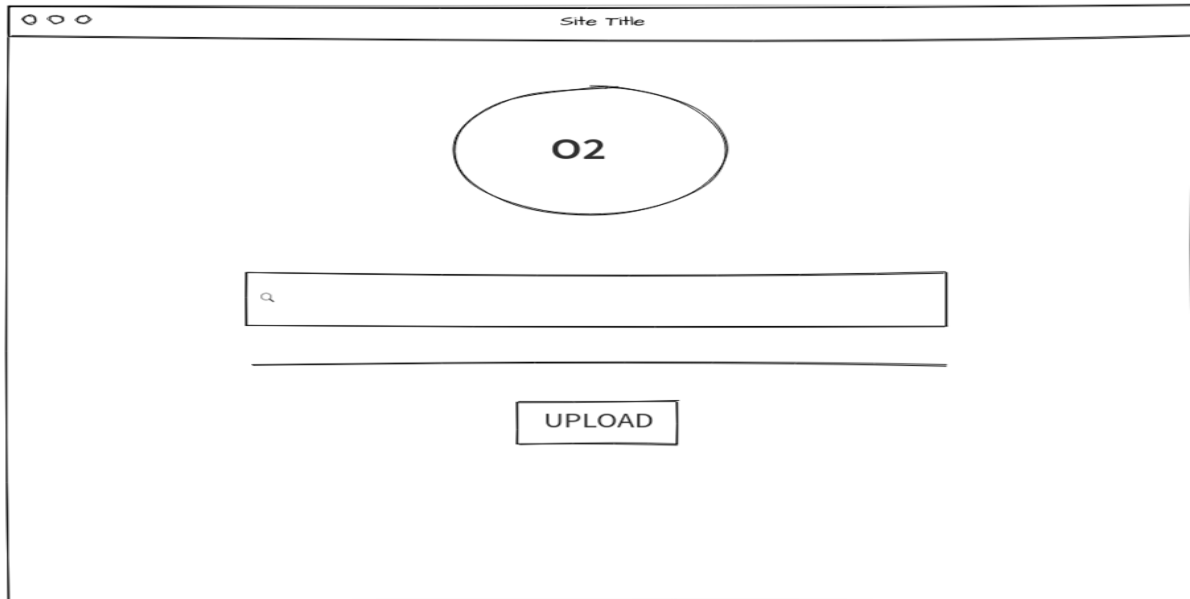
- Document Search Engine.
- Semantic Search Capabilities.
- User can upload documents of supported format.
- Can search .pdf, .doc, .xls, .rtf, .html, .odt, etc .
- Can work on both internet and intranet.

Work Timeline :

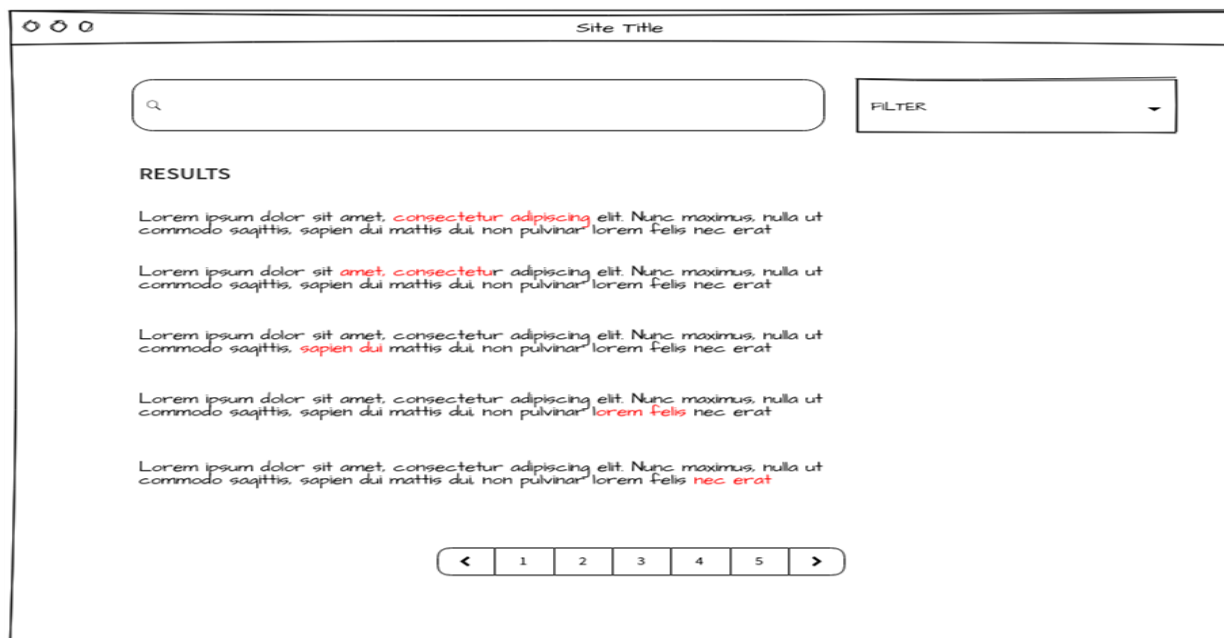
Estimated Deadline	Task
20th August	Wireframe for the Front-End
27th August	Application Architecture (Design Phase)
6th September	Basic JSP application with MVC
13th September	Incorporating suggestions and feedback
20th September	Elastic Search Integration
4th October	Adding advanced features <ul style="list-style-type: none">• Auto-complete• Suggested Search• Search on attachments, etc
26th October	Incorporating suggestion and feedback
8th November	Testing Phase

Phase I : Wireframe and Design

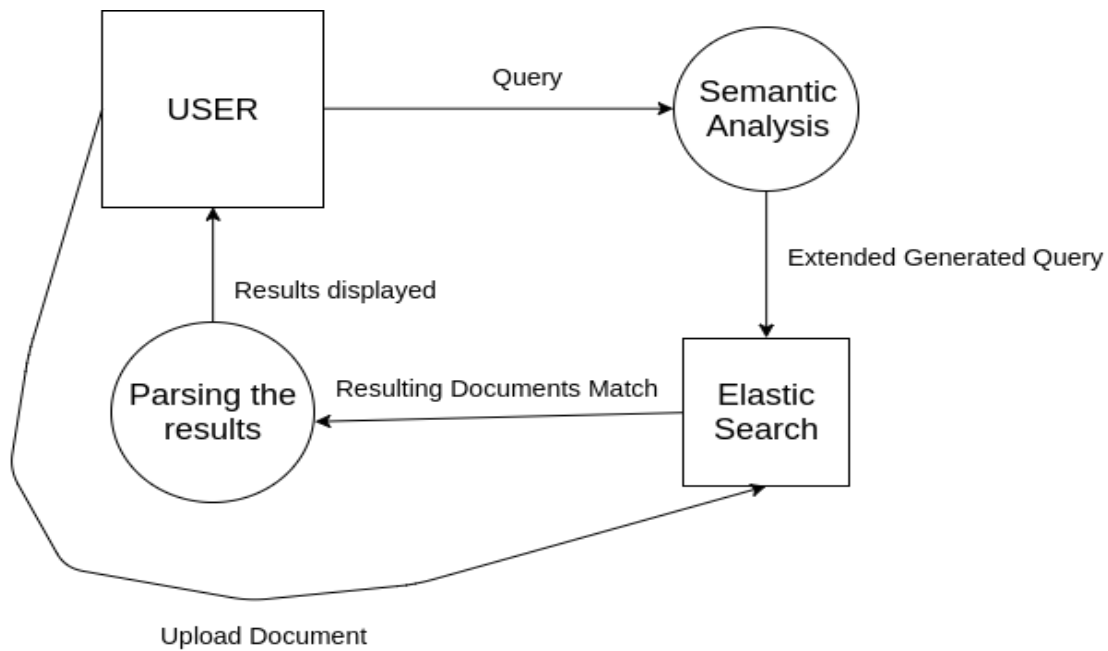
Wireframe of Landing Page for O₂



Wireframe of Search Result Page for O₂



Schematic Diagram of O₂ :

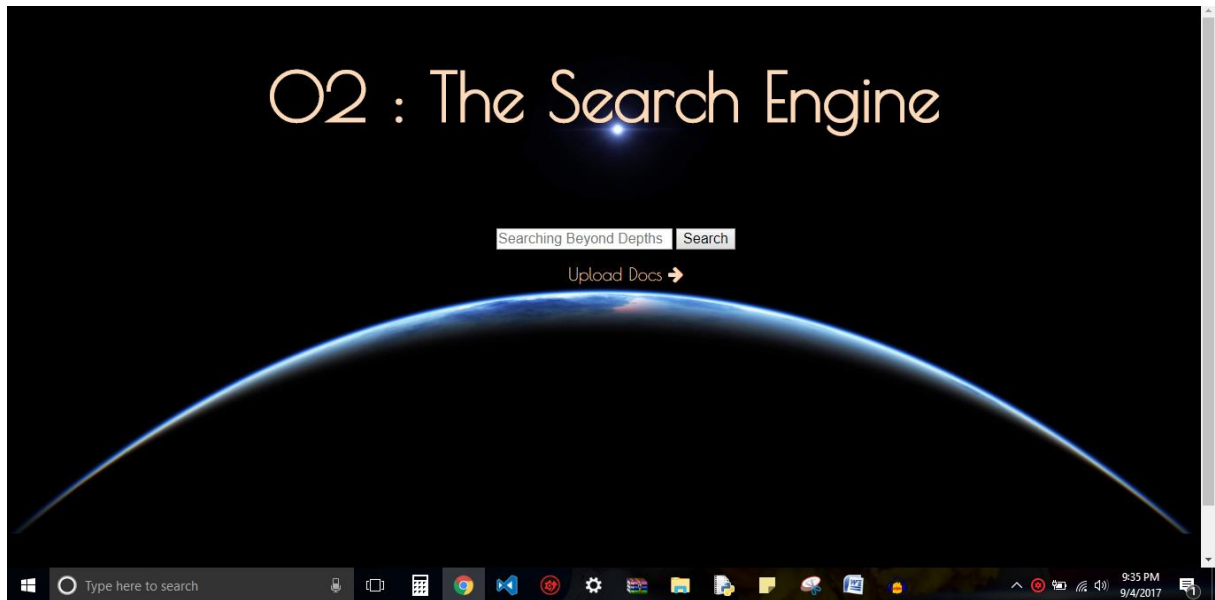


Progress after Phase I :

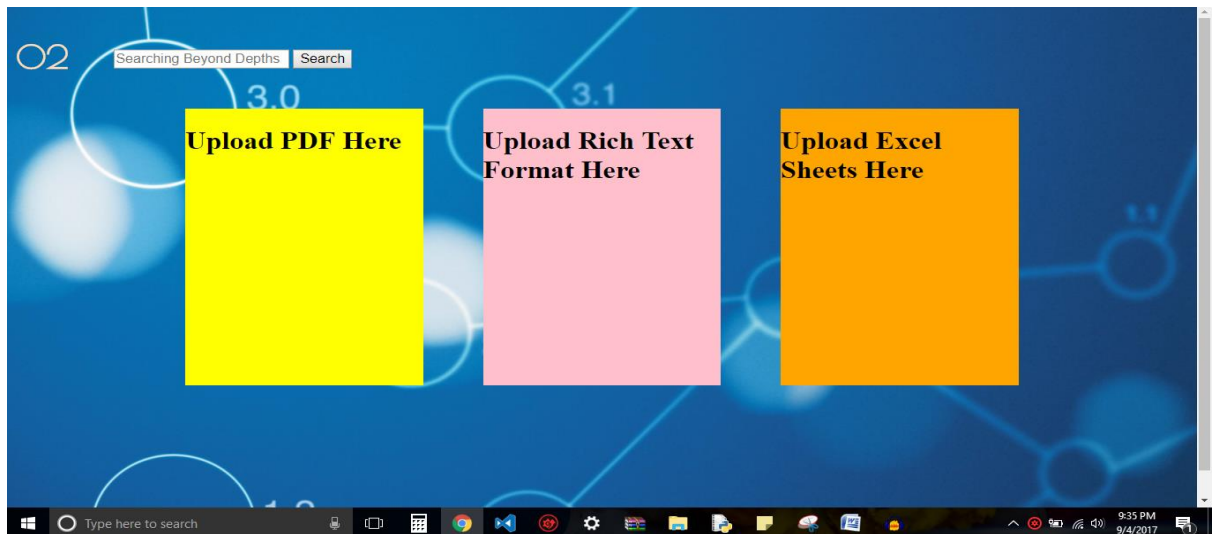
- Wireframe design
- Data Flow Diagram
- JSP - Servlets architecture
 - Front Controller
- Research on Elastic Search.

O₂ after Phase I :

i. Landing Page -



ii. Upload Page –



Phase II : Implementation of ElasticSearch Capabilities

What is ElasticSearch?

Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases.

- It is Fast!
 - Uses inverted index
 - Finite state transducers for full text query
 - BKD-trees for numeric and Geo Data
- It is Scalable
 - Can run on single node.
 - Can run on a large cluster. E.g. 300 Machines
- It is accessible from RESTful web service interface.
- Uses schema less JSON (JavaScript Object Notation) documents to store data.
- Built on Java programming language, hence it can run on different platforms.

Indexing in ElasticSearch :

- **Index** – It is a collection of different type of documents and document properties.
- **Type/Mapping** – It is a collection of documents sharing a set of common fields present in the same index.
- **Document** – It is a collection of fields in a specific manner defined in JSON format.
- **Replicas** – Elasticsearch allows a user to create replicas of their indexes and shards.

Creating Index :

Our system will have 3 Indices:

- Text_index
- Doc_index
- Zip_index

Querying :

API helps to extract type JSON object by performing a get request for a particular document.

```
GET http://localhost:9200/_search?q = name:central
```

```
{
  "took":78, "timed_out":false, "_shards":{"total":10,
"successful":10, "failed":0},
  "hits":{"
    "total":1, "max_score":0.19178301, "hits":[{"
      "_index":"schools", "_type":"school",
      "_id":"1", "_score":0.19178301,
      "_source":{"
        "name":"Central School", "description":"CBSE
Affiliation",
        "street":"Nagan", "city":"paprola",
        "state":"HP", "zip":"176115",
        "location":[31.8955385, 76.8380405],
        "fees":2000,
        "tags":["Senior Secondary", "beautiful
campus"], "rating":"3.5"
      }
    }
  ]
}
```

ElasticSearch RESTful API :

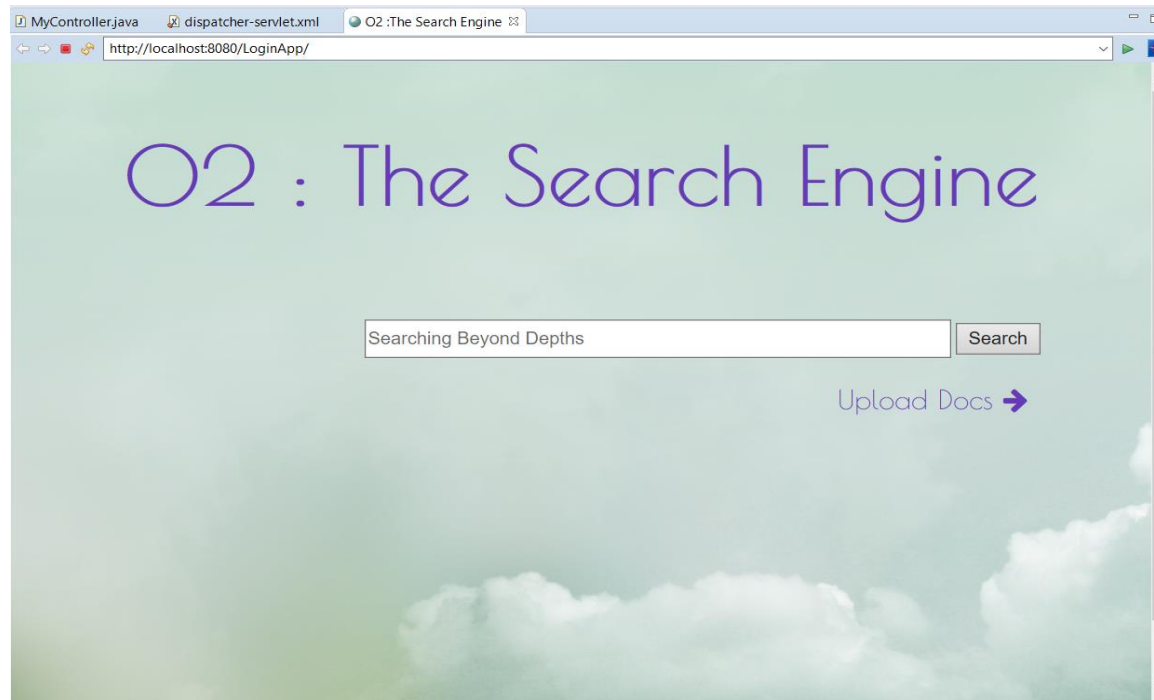
- search() :
 - GET *http:// server:port /index/type/_search?q={query}*
 - 1 index
 - 2 types :
 - text
 - doc
- index() :
 - PUT *http:// server:port /index/type with request JSON body*
 - JSON body
 - {
 - "authorName" : { name }*
 - "content" : { content }*
 - "Timestamp" : { time }*
 - }

Progress after Phase II :

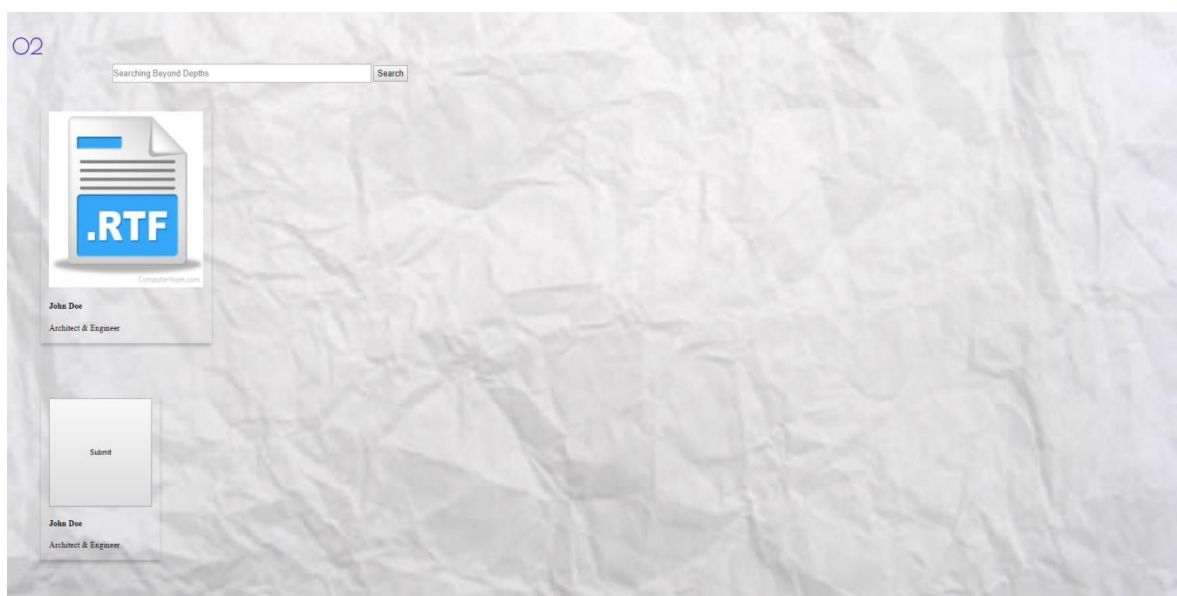
- Back controller to handle API calls to Elastic Search Engine and its capabilities.
- Implementation done in JSP
- Research on Semantic Search and how it works.

O₂ after Phase II :

i. Landing Page -



ii. Search Result Page -



Phase III : Implementation of Semantic Search

Research done on Semantic Search :

1. Word co-occurrence Algorithm –

Steps :


- First, we apply standard text cleaning and processing to queries including stemming, stopwords and punctuation removal as well as spelling correction which are then segmented into bags of words $W = \{w_i \mid i = 1, \dots, N\}$. Each w_i represents a set of variations of a word, e.g., 'lipton', 'liptons', 'lipton's' are all represented by one w_i .
- In order to find topics related to a search query it is useful to look at lift scores. The lift score for word w_i , given action a is defined as - $\text{lift}(w_i ; a) = P(w_i \mid a) / P(w_i)$

Algorithm :

- Create hashmap
- Initialize topics
- Expand Topics
- Form Non-Disjoint Clusters
- Induce Query Similarity
- Form Disjoint Hierarchical Clusters

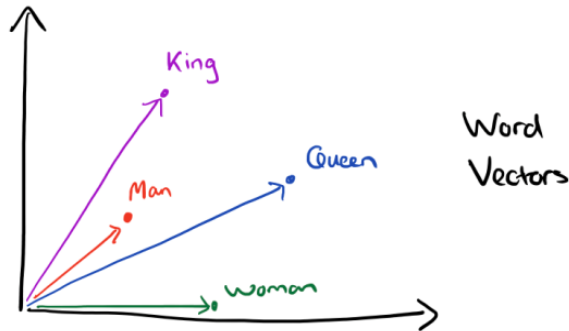
2. Semantic Search using Query Expansion –

We pad the queries with similar meaning words from the corpus to increase the length of search query and subsequently increasing the semantic search size for a user.

“High Level Java”  “Advanced Level Java”

3. Efficient Estimation of Word Representations in Vector Space –

- Word2vec helps in predicting the context of the words
- Why not use synonyms instead?
 - Thresholding is possible



Implementation of Semantic Search :

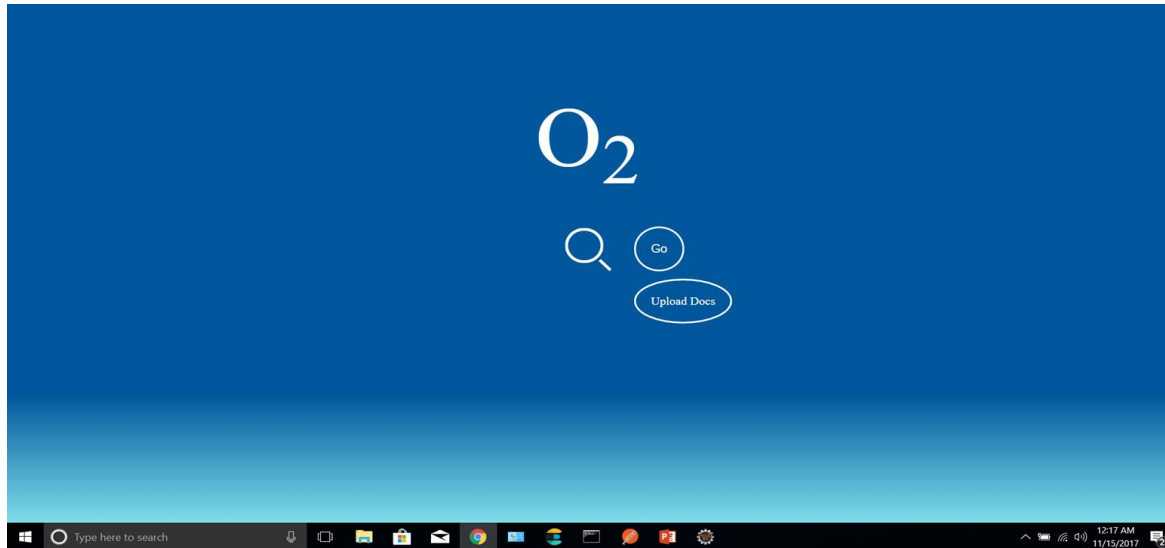
We have implemented query expansion and word2vec. In these algorithms, we are increasing the size of the query by adding similar words from corpus. The similar words are actually gained by converting the words into vectors and then measuring cosine similarity. Then the semantic search engine parses the query by padding extra key words and the elasticsearch capabilities help us to retrieve all the documents matching from the keywords and hence resulting the documents having semantically same meaning as that of the original query search.

Progress after Phase III :

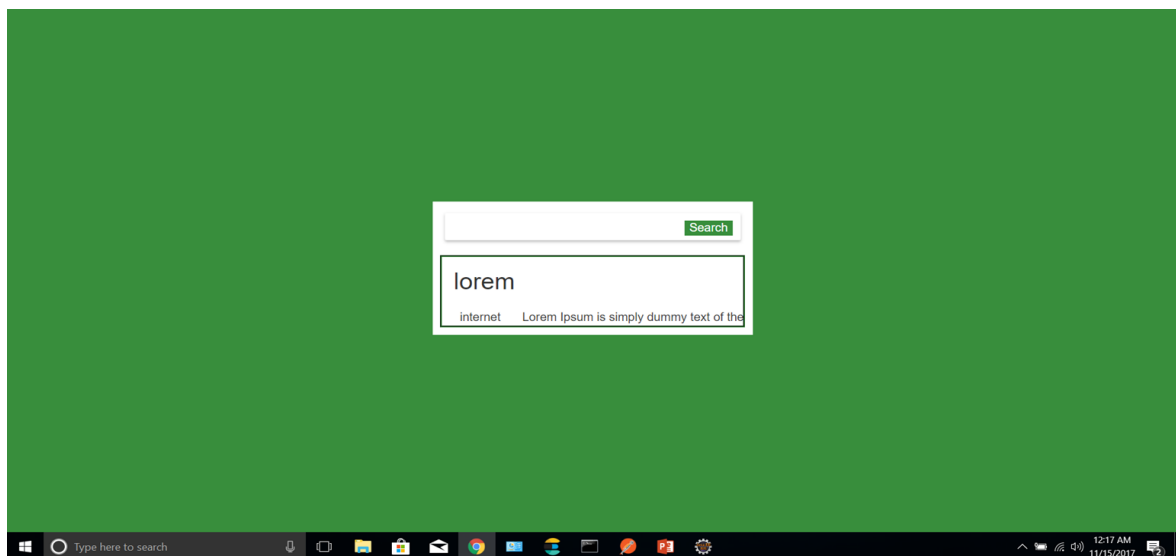
- Implementation of Semantic Search
 - Query Expansion
 - Word2Vec
- User Friendly UI with features like :
 - Search
 - Upload
 - Index

O₂ after Phase III :

i. Landing Page



ii. Search Result Page



Tools and Technologies Used :

- NLTK
- gensim
- Java Server Pages
- Java Servlets
- ElasticSearch
- GitHub for project collaboration
- Trello & Slack for project management



Natural Language
Tool Kit (NLTK)
Basic Text Analytics



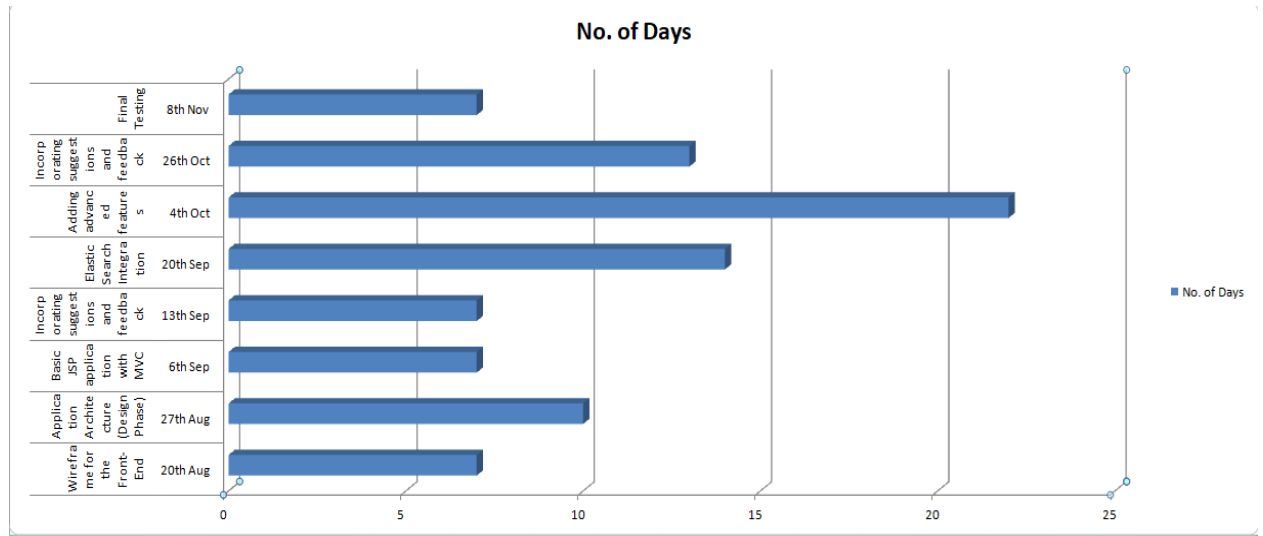
Contribution of Team Members :

Team Members	Roles
Aakash Sinha Karanjit Singh Gill Mohit Raj	UI / UX
Aakash Sinha Karanjit Singh Gill NVS Abhilash	Elastic Search
Deep Parikh NVS Abhilash Prateek Agarwal	Semantic Search
Deep Parikh Mohit Raj Prateek Agarwal	Java Servlet

Future Work :

- Support multiple file formats
- Experiment with other Semantic Search functions like LSA.

Project Timeline : Gantt Chart



References :

- Efficient Estimation of Word Representations in Vector Space. ([Tomas Mikolov](#), [Kai Chen](#), [Greg Corrado](#), [Jeffrey Dean](#), 2013)
- <https://www.optimus01.co.za/search-engine-stats-and-facts>
- <https://www.elastic.co/>
- Improving semantic topic clustering for search queries with word co-occurrence and bigraph co-clustering by Jing Kong_, Alex Scott, and Georg M. Goerg, Google Inc.