# University of Windsor

Project Report

COMP 8547 Advanced Computing Concepts – Winter 2025

*Nimbus Navigators*

**(A Cloud Storage Service Analysis)**


Team members:

Aleena Ali Azeem

Simranpreet Kaur

Sai Harinath Reddy Yelluri

Vamsi Krishna Nallani

Priyadharshan Reddy Singareddi Gari

**Introduction:**

Task:

✓ **Web crawler;**
✓ **HTML parser;**
✓ **Spell checking;**
Spell checking can be achieved by constructing a vocabulary based on all existing words in text files.
Alternative word suggestions should be provided if no results are found.
Edit distance algorithm can be used to compare the user's input with existing word from source files.
✓ **Word completion;**
✓ **Frequency count;**
Frequency count shows the user the number of occurrences of a word in a specific url.
✓ **Search frequency;**
Ability to show the word that has been searched before as well as the number of times the word has been
searched.
✓ **Page ranking;**
Page ranking is used to measure importance of a search result based on the number of occurrences.
Search keywords that are repeated more within a web page will be ranked higher than the others.
Ranking web pages can be performed using sorting, heaps or other data structures.
✓ **Inverted indexing**
Inverted indexing allows us to perform quick searches without going through all the files. This can be
represented as an index data structure storing a mapping from content, such as words or numbers, to its
locations in a set of documents.

**Final Project Report**

**COMP-8547: Advanced Computing Concepts – Winter 2025**
**School of Computer Science**

---

**1. Group Information**

**Group Name:** Nimbus Navigator
**Group Number:** 8
**Variant:** Cloud Storage Service Analysis
**Team Lead:** Aleena Ali Azeem

---

**2. Member Contributions**

| **Member Name** | **Feature(s) Developed** | **Java File(s) (Example placeholders)** |
|---|---|---|
| **Simranpreet Kaur** | Spell Checking using Trie and Edit Distance | SpellChecker.java |
| **Aleena Ali Azeem** | Word Completion using Trie | WordCompletion.java |
| **Sai Harinath Reddy Yelluri** | Word Frequency Count using Heap Sort | WordFrequency.java |
| **Vamsi Krishna Nallani** | Search Frequency using KMP Algorithm | SearchFrequency.java |
| **Priyadharshan Reddy Singareddi Gari** | Page Ranking using Frequency Count with Boyer-Moore Algorithm | frequencycount.java |

Note: Each member solely developed their respective features and ensured modular Java implementation with clean integration into the overall system.

---

**3. Additional Features**

Currently, no additional features are implemented outside of the listed core functionalities.

---

**4. Algorithms and Data Structures Used**

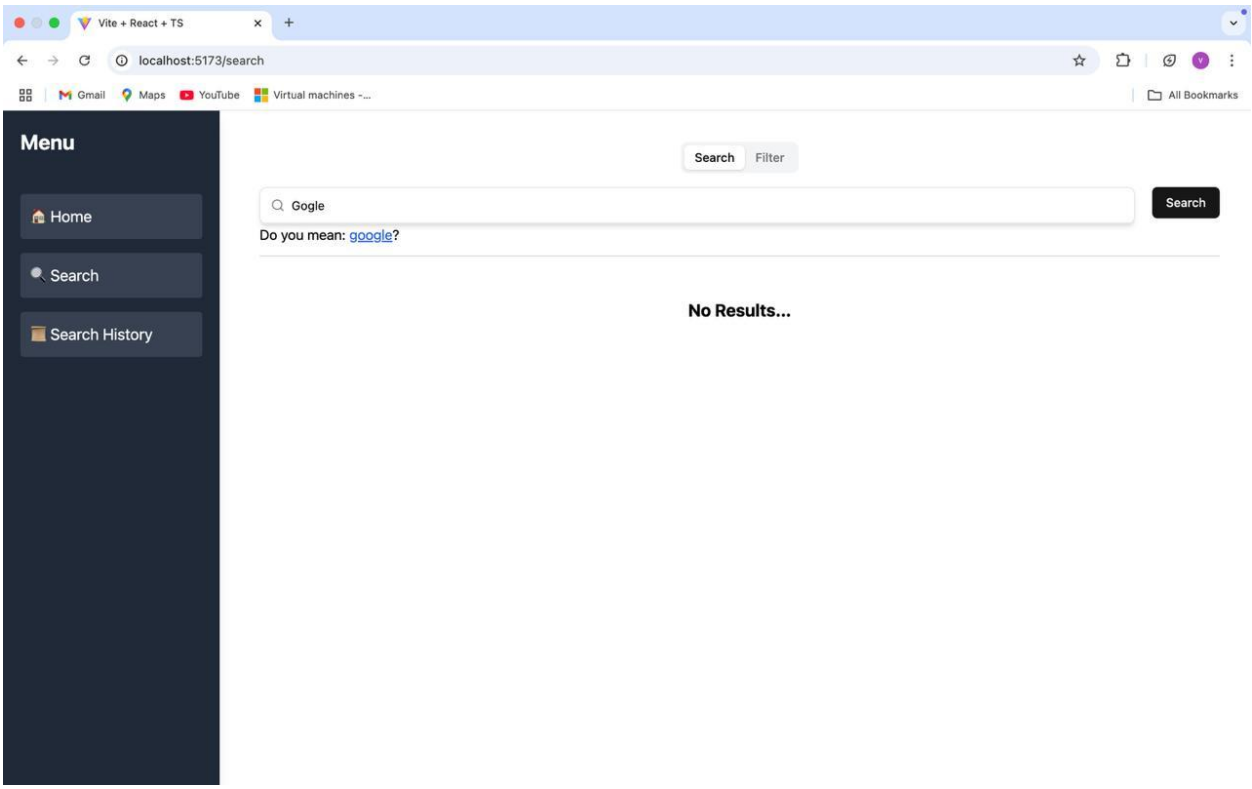
- **Spell Checking (Simranpreet):**
  Implemented using a **Trie data structure** to store the vocabulary of parsed words. The **Edit Distance (Levenshtein)** algorithm helps in finding the closest word suggestions when a search term doesn't match exactly.
- **Word Completion (Aleena):**
  Also uses a **Trie** to allow auto-complete suggestions based on the prefix typed by the user. This feature helps users refine and complete their queries faster.
- **Word Frequency Count (Sai Harinath):**
  Utilizes the **Heap Sort** algorithm to sort the words based on how many times they appear in each document or search scope. The results help quantify keyword density.
- **Search Frequency (Vamsi):**
  Implemented using the **Knuth-Morris-Pratt (KMP)** algorithm to search efficiently for the frequency of repeated queries. It also tracks how often a user has searched for a specific term.
- **Page Ranking (Priyadharshan):**
  Applies the **Boyer-Moore algorithm** to count how often search keywords appear on each page. A ranking score is calculated to display the most relevant results higher.

Each algorithm was chosen specifically for its efficiency and relevance to the task. Trie was excellent for fast lookups; KMP and Boyer-Moore provided optimal search performance; and Heap Sort allowed efficient frequency-based ordering.

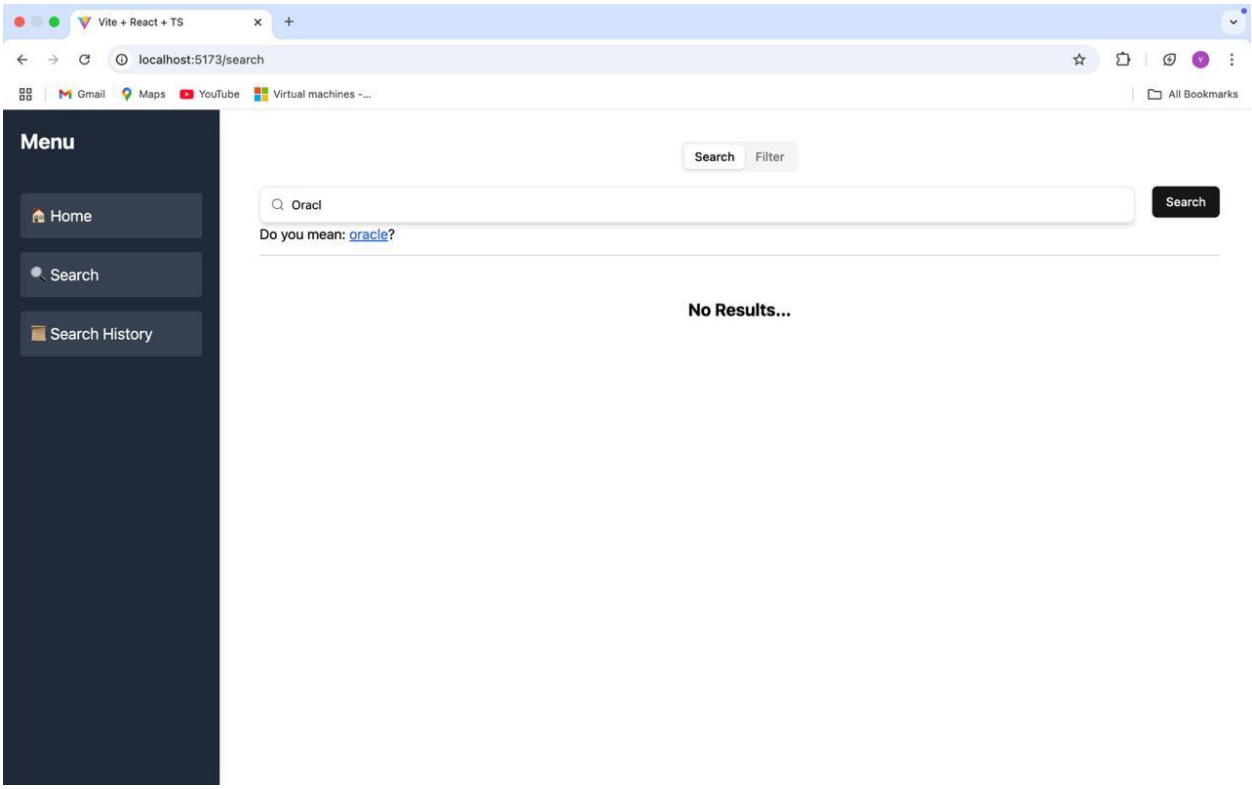
---

**5. Screenshots**

- **Spell Checker Output Sample** – Showing suggestions for mistyped words.




- **Word Completion Demo** – Displaying prefix-based suggestions in real time.



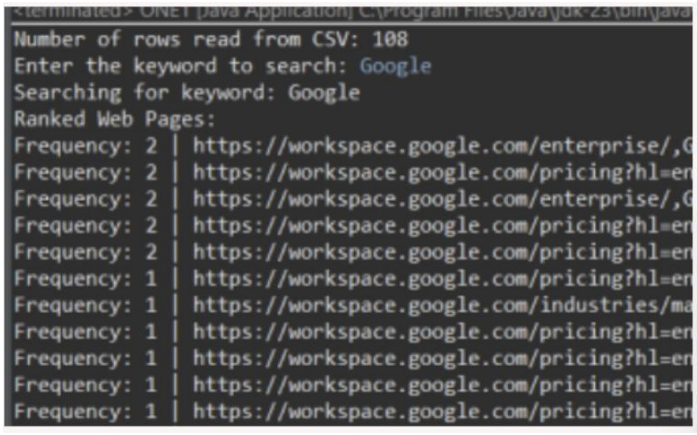

- **Word Frequency Table** – Output of word count sorted using heap.

```
Please enter a word to search for (or type 'done' to exit): com
The word "com" appeared 91 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): goole
The word "goole" appeared 0 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): google
The word "google" appeared 78 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): 55
The word "55" appeared 0 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): 70
The word "70" appeared 55 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): 20
The word "20" appeared 55 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): co
The word "co" appeared 113 times in the CSV files.

Please enter a word to search for (or type 'done' to exit): done
```

- **Search Frequency Tracker** – Logs showing how often each word has been searched.

```
<terminated> KMPSearchTracker [Java Application] /Library/Java/JavaVirtualMachines/jdk
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): gemini
Occurrences: 12
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): dropbox
Occurrences: 0
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): chat
Occurrences: 3
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): hardware
Occurrences: 1
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): applications
Occurrences: 1
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): apps
Occurrences: 1
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): calender
Occurrences: 0
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): docs
Occurrences: 3
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): appsheet
Occurrences: 1
Search Frequency: 1
Enter a word to search (or type 'exit' to quit): exit
```

- **Ranked Page Results** – Display of page ranking based on keyword occurrences.




---

**6. References**


- Java Documentation: https://docs.oracle.com/javase/8/docs/api/
- Data Structure and Algorithms Textbooks and Class Notes
- Online articles on Trie, Heap Sort, KMP, and Boyer-Moore Algorithms
- Open-source Java implementations reviewed for learning purpose