CodeIt - Search Engine

Points to write in Resume:

- Developed a high-performance search engine using the MERN stack, incorporating modern web technologies to deliver a seamless user experience.

- Scraped and processed a diverse dataset of 3,500 problems from leading coding platforms (LeetCode, InterviewBit, Codeforces) using Selenium, ensuring a comprehensive collection of programming challenges.
  [Just watch this video
  How to Scrape Dynamically Loaded Websites with #Selenium and #BeautifulSoup …
  to know how web scraping works using Selenium and Beautiful Soup 4]

- Implemented robust edge case handling techniques, including converting numbers to words and vice versa, lemmatization, spell-checking, camel case to words conversion, to generate relevant keywords from user query strings.

- Utilized the BM25 algorithm and TF-IDF indexes to rank documents efficiently against query strings, and optimized search speed by leveraging RAM-based indexes and efficient data structures to enhance search performance.

- Incorporated custom scoring metrics, including title matching and source reliability, to further enhance search relevance and user experience, ensuring accurate and trustworthy results appear prominently.

Possible Questions:

1) Why did you use BM25 score criteria over the normal TF-IDF cosine similarity score? [Discussed in Detail in class]
2) What are the different edge cases which you handled while building the search engine? [Discussed in detail in class]

3) Future scope of improvement ?

    - We can add a cache layer before the user query hits the server to reduce the response time.
    - We can look into something known as Sharding (DBMS Concept) which allows us to perform searches over different sets of data concurrently.

- In future we can add a feature to add new questions to our database and then in order to update the indexes (TF-IDF values) we can run a Cron Job which runs every 24 hours to produce the updated index files

4) What do you mean by RAM based indexes ?

   The TF-IDF values for all the documents were stored in two separate .txt files and both the files were read and the corresponding array was generated and loaded onto the RAM at the time of app startup, so that their access became fast.