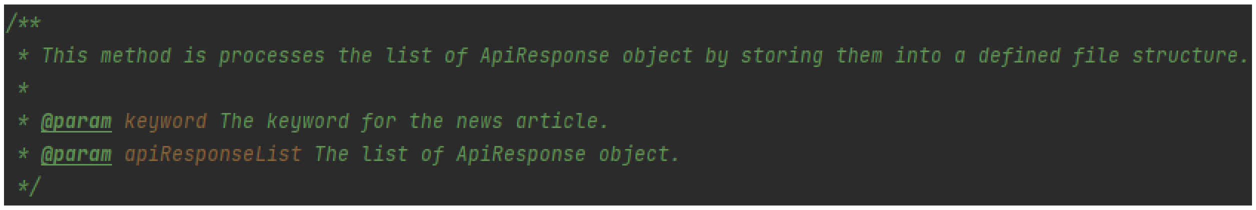
A light-weight data extracting, processing, transformation engine designed using Java programming language (no 3rd party libraries used).

IDE used: IntelliJ IDEA

JDK version: 17

JavaDocs specification for commenting styles, such as @param, @return etc.:



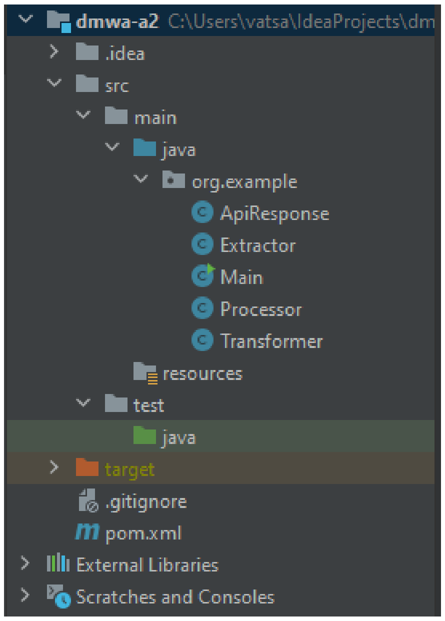
```
/**
 * This method is processes the list of ApiResponse object by storing them into a defined file structure.
 *
 * @param keyword The keyword for the news article.
 * @param apiResponseList The list of ApiResponse object.
 */
```

Fig 1.1: JavaDocs specification example

Design principles used: I have used some of the SOLID design principles in my code. I have created each module independently to reduce code coupling and follow the Single Responsibility principle.

Application type: console-based

Project structure:



```
dmwa-a2  C:\Users\vatsa\IdeaProjects\dm
  .idea
  src
    main
      java
        org.example
          ApiResponse
          Extractor
          Main
          Processor
          Transformer
      resources
    test
      java
  target
  .gitignore
  pom.xml
External Libraries
Scratches and Consoles
```

Fig 2: Project Structure

Extraction Engine Implementation:

Keywords used: “Canada”, “University”, “Dalhousie”, “Halifax”, “Canada Education”, “Moncton”, “hockey”, “Fredericton”, “celebration”

}

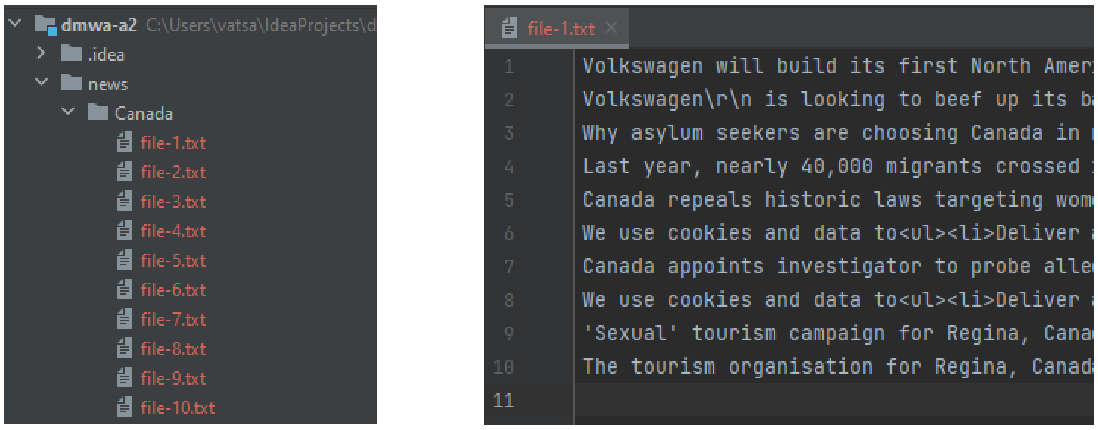
To implement extraction engine, I created an HTTP connection with https://newsapi.org. The response of GET request is in the form of JSON. I have parsed the JSON string using Regex pattern.

Pseudo code of data extraction program:

```
FUNCTION extract(url) {
    TRY
    connection = url.openConnection() // creating an HTTP connection
    connection.setRequestMethod("GET") // sending GET request

    in = new BufferedReader(new InputStreamReader(connection.getInputStream()))
    responseContent = new StringBuffer()

    WHILE (inputLine = in.readLine()) != null
        responseContent.append(inputLine)

    in.close()

    // Matching a Regex pattern to extract the titles and contents from the JSON String
    pattern =
    Pattern.compile("\"title\":\"((?:[^\"\\\\]|\\\\.)*)\"(.*?(?=\"content\":\"([\\s\\S]+?(?=\"}(,\\{|])
    ))))")
    matcher = pattern.matcher(responseContent) apiResponseList = new ArrayList<>();
    WHILE matcher.find()
        apiResponseList.add(new ApiResponse(matcher.group(1), matcher.group(3)))

    RETURN apiResponseList

    CATCH
    (RuntimeException OR IOException e) RETURN NULL
}
```

Data-Processing Engine Implementation:

Once the raw data is captured using the API, this program writes the news contents, and the titles to files. Each file contains only 5 news articles or less.



```
dmwa-a2 C:\Users\vatsa\IdeaProjects\d
  .idea
  news
    Canada
      file-1.txt
      file-2.txt
      file-3.txt
      file-4.txt
      file-5.txt
      file-6.txt
      file-7.txt
      file-8.txt
      file-9.txt
      file-10.txt
```

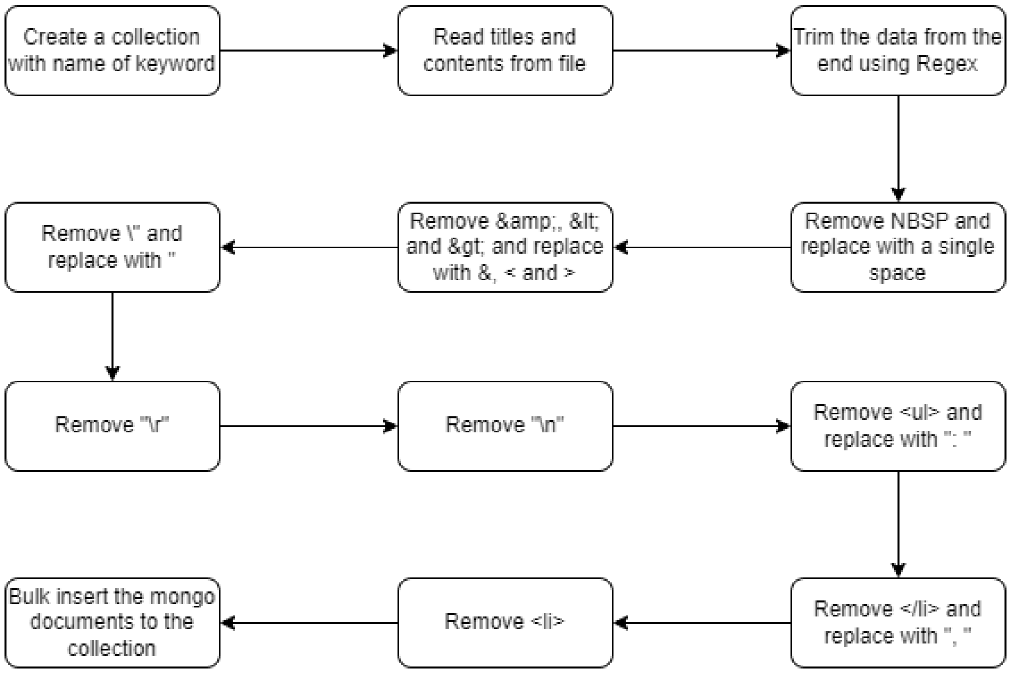
```
file-1.txt
Volkswagen will build its first North Amer
Volkswagen\r\n is looking to beef up its b
Why asylum seekers are choosing Canada in
Last year, nearly 40,000 migrants crossed
Canada repeals historic laws targeting wom
We use cookies and data to<ul><li>Deliver
Canada appoints investigator to probe alle
We use cookies and data to<ul><li>Deliver
'Sexual' tourism campaign for Regina, Cana
The tourism organisation for Regina, Canad
```

Fig 3: Example of file structure after processing the data

Transformation Engine Implementation:

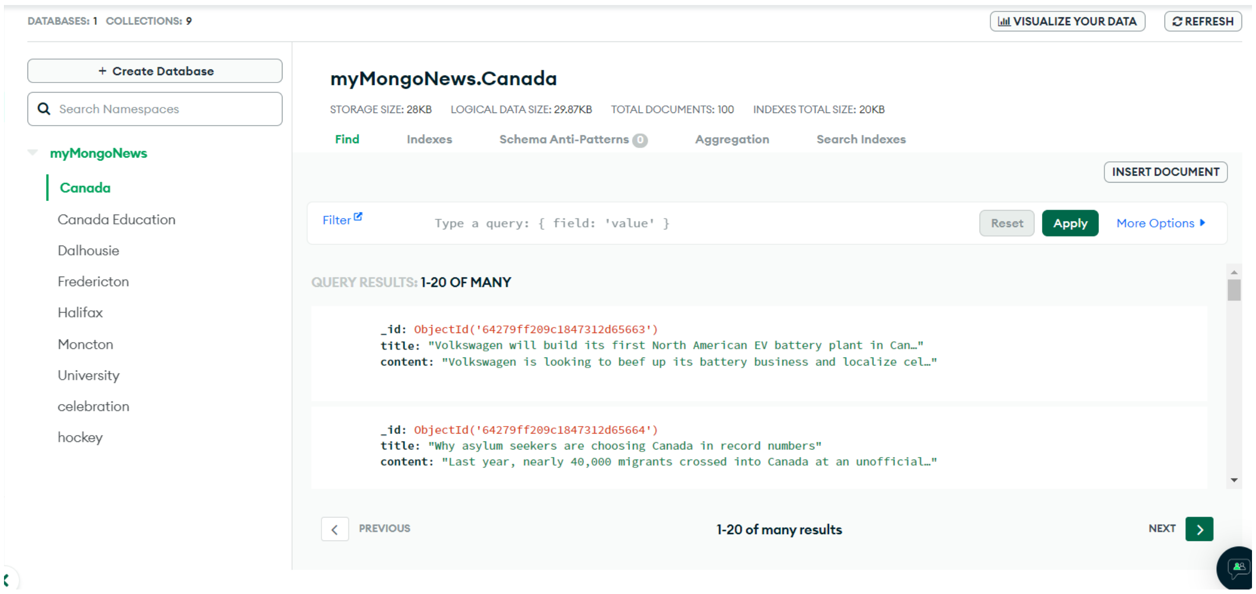
In this step, the raw data stored in files is automatically cleansed and transformed the data, and then upload each record to new MongoDB database "myMongoNews". I have created multiple collections to store the data for each keyword.

Flowchart of the transformation engine:



Create a collection with name of keyword
Read titles and contents from file
Trim the data from the end using Regex
Remove NBSP and replace with a single space
Remove &amp;, &lt; and &gt; and replace with &, < and >
Remove \" and replace with "
Remove "\r"
Remove "\n"
Remove <ul> and replace with ": "
Remove </li> and replace with ", "
Remove <li>
Bulk insert the mongo documents to the collection

Fig 4: Flowchart of the transformation engine



myMongoNews.Canada

STORAGE SIZE: 28KB LOGICAL DATA SIZE: 29.87KB TOTAL DOCUMENTS: 100 INDEXES TOTAL SIZE: 20KB

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('64279ff209c1847312d65663')
title: "Volkswagen will build its first North American EV battery plant in Can…"
content: "Volkswagen is looking to beef up its battery business and localize cel…"

_id: ObjectId('64279ff209c1847312d65664')
title: "Why asylum seekers are choosing Canada in record numbers"
content: "Last year, nearly 40,000 migrants crossed into Canada at an unofficial…"
```

1-20 of many results

Fig 5: Mongo Cloud Database "myMongoNews" after transformation of the data

MongoDB document after the transformation process is completed:

{"_id":{"$oid":"64279ff209c1847312d65663"},"title":"Volkswagen will build its first North American EV battery plant in Canada","content":"Volkswagen is looking to beef up its battery business and localize cell production for its electric vehicles. It has taken another step toward that goal by announcing subsidiary PowerCo's first N"}

References:

- "draw.io", [Online]. Available: https://app.diagrams.net/. [Accessed: 02-Apr-2023].
- "Different ways of Reading a text file in Java", [Online]. Available: https://www.geeksforgeeks.org/different-ways-reading-text-file-java/. [Accessed: 02-Apr-2023].
- "Regex Generator", [Online]. Available: https://regex-generator.olafneumann.org. [Accessed: 02-Apr-2023].
- "Regular Expressions 101, [Online]. Available: https://regex101.com/. [Accessed: 02-Apr-2023].
- "A Solid Guide to SOLID Principles", [Online]. Available: https://www.baeldung.com/solid-principles. [Accessed: 02-Apr-2023].
- "Connection Guide", [Online]. Available: https://www.mongodb.com/docs/drivers/java/sync/v4.3/fundamentals/connection/. [Accessed: 02-Apr-2023].