Code - Level Documentation

main.py:

 # Imports:

- `CitationFinder`: This module contains a class responsible for finding citations in text data.
- `requests`: Used for making HTTP requests to fetch data from an API.
- `spacy`: A powerful NLP library used for various text processing tasks.
- `json`: Essential for handling JSON data, including serialization and deserialization.

 # Constants:

- `API_URL`: The endpoint URL from which data will be fetched.

 # Functions:

1. `fetch_data(api_url)`:
   - This function fetches data from a specified API URL.
   - It handles paginated data by continuously fetching until there's no next page.
   - Returns a list of all fetched objects.

2. `get_all_citations()`:
   - Fetches all objects using `fetch_data`.
   - Initializes a `CitationFinder` object.
   - Iterates through all fetched objects, finding citations using the `CitationFinder` class.
   - Returns a list of dictionaries containing object IDs and corresponding citations.

3. `save_dict_to_json(data, filename)`:
   - Saves a dictionary to a JSON file with specified filename.

 # Execution:
- Invokes `get_all_citations` to retrieve citations for all fetched objects.
- Constructs a dictionary containing all citations.
- Saves the dictionary to a JSON file named "all_citations.json".

CitationFinder.py:

 # Imports:

- `torch`: The PyTorch library, which provides functionalities for deep learning tasks.
- `nltk`: A comprehensive toolkit for NLP tasks, used for tokenization and stopwords.

- `AutoModel`, `AutoTokenizer`: Classes from Hugging Face's `transformers` library, enabling easy usage of pre-trained NLP models.

 # Class `CitationFinder`:

1. Constructor `__init__(self, model_name)`:
- Initializes the class instance with a specified model name or uses a default one.
- Loads necessary NLP resources such as tokenizer and model.

2. Method `remove_special_characters(self, text)`:
- Removes special characters from the given text using regular expressions.

3. Method `remove_links(self, text)`:
- Eliminates URLs/links from the provided text using regex patterns.

4. Method `lemmatize_words(self, words)`:
- Lemmatizes a list of words using spaCy, reducing them to their base form.

5. Method `preprocess_text(self, text)`:
- Applies preprocessing steps to the text, including removing special characters, extra spaces, and stop words.

6. Method `embed_text(self, text)`:
- Embeds the provided text into a numerical representation using the loaded model.

7. Method `cosine_similarity(self, embedding1, embedding2)`:
- Computes the cosine similarity between two embeddings, indicating their semantic similarity.

8. Method `get_similarity(self, text1, text2)`:
- Calculates the similarity between two texts based on their embeddings.

9. Method `extract_keywords(self, response)`:
- Identifies keywords (nouns and proper nouns) in the response text.

10. Method `clean_citations(self, citations)`:
- Cleans the extracted citations, removing duplicates and formatting them appropriately.

11. Method `find_citations(self, response, sources, thres=0.8)`:
- Searches for citations in the response text based on provided sources.
- Iterates through each sentence in the response, comparing it with the context of each source.
- If the similarity and keyword presence meet thresholds, adds the citation to the list.
- Returns cleaned citations.

app.py:

This script sets up a Streamlit web application with a wide layout. It displays a title and a table showing all citations with respect to each response, using data loaded from a JSON file. The JSON file is loaded using the load_json_file function, and Streamlit's table function is used to render the data in tabular format.