Search Engine Documentation

Project made by Pablo Sixto García

This project is a small search engine application that utilizes Elasticsearch for search functionality and Streamlit for developing the graphical user interface (GUI). The application allows users to search for Airbnb listings based on various criteria and provides search results with additional information.

Files and Modules

The project consists of the following files:

- utils.py: Contains utility functions related to Elasticsearch, such as creating an index and performing search queries.
    - check_and_create_index(): The function checks if the given index exists in Elasticsearch, if it does not, it creates it with the specified mappings. The mappings define the data model data for the airbnb docs.
    - index_search(): this function performs a search query on the Elasticsearch index and returns the search as a dictionary.
    - calculate_average_price_by_neighborhood(): calculates the average price per neighborhood in the Airbnb data
- load_data.py: Creates a list of Elasticsearch data
    - create_document(): takes a row from the Dataframe as input and constructs a dictionary representing an Elasticsearch document.
- templates.py: Defines HTML templates used for styling and displaying elements in the Streamlit application.
    - load_css(): Returns CSS styles as a string wrapped by a <style> tag
    - number_of_results(): This function takes the total number of search results (‘total_hits’) and the duration of the search
    - search_result(): It generates an HTML string displaying a single search result.
- app.py: The main entry point of the application. It runs the Streamlit application and connects to Elasticsearch.
    - main(): Implements the Streamlit application logic, including handling user inputs, displaying search results, and interacting with Elasticsearch.

Dependencies

The project relies on the following external libraries and frameworks:

- pandas: A data manipulation library used for loading the Airbnb dataset.
- elasticsearch: A Python client for Elasticsearch, used for interacting with the Elasticsearch search engine.
- streamlit: A Python framework for building interactive web applications.

Usage

To run the Search Engine application, follow these steps:

1. Install the required dependencies mentioned above.
2. Ensure that Elasticsearch is running and accessible at `http://localhost:9200`.
3. Execute the `main.py` script using a Python interpreter.
4. The Streamlit application will launch in a web browser.

The easiest way is using the command '$ streamlit run app.py' under directory with all the source code files

Functionality

The Search Engine application provides the following functionality:

Data Loading

The application loads Airbnb listing data from a CSV file hosted on GitHub using the 'pandas' library. The data is read into a pandas DataFrame.

Elasticsearch Integration

The application establishes a connection to Elasticsearch using the Elasticsearch Python client. The Elasticsearch client is used to create an index and load the Airbnb data into Elasticsearch.

Search

- The application allows users to enter search keywords in a text input field.
Keywords:
  - 'host_name'
  - 'neighborhood_group'
  - 'neighborhood'
  - 'room_type'
- Users can also specify a price range using minimum and maximum price inputs.

- Upon clicking the "Search" button, the application performs a search query to Elasticsearch based on the provided criteria.
- The search results are displayed in the application, including the number of results and the duration of the search.
- Each search result includes the listing name, host name, neighborhood group, and neighborhood.
- Users can click on a search result to view more details.

Average Price by Neighborhood

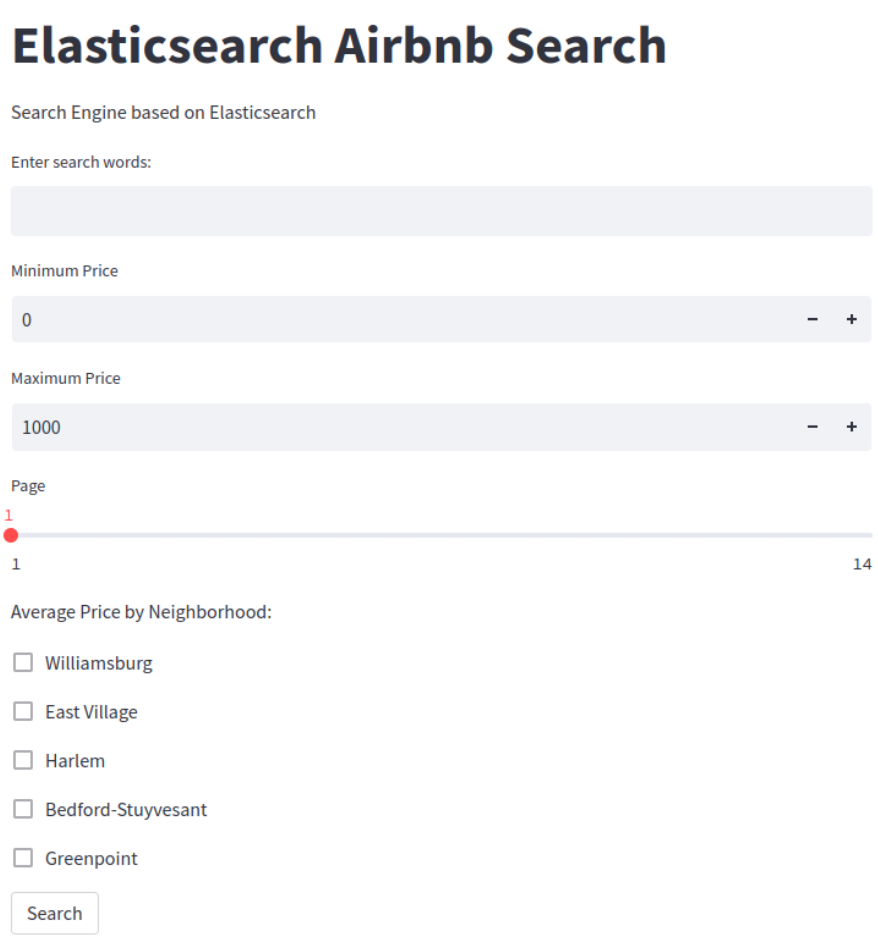
The application calculates the average price per neighborhood using an Elasticsearch aggregation query.The average prices are displayed alongside the corresponding neighborhood names. Users can select and view the average price for specific neighborhoods.

Development and Customization

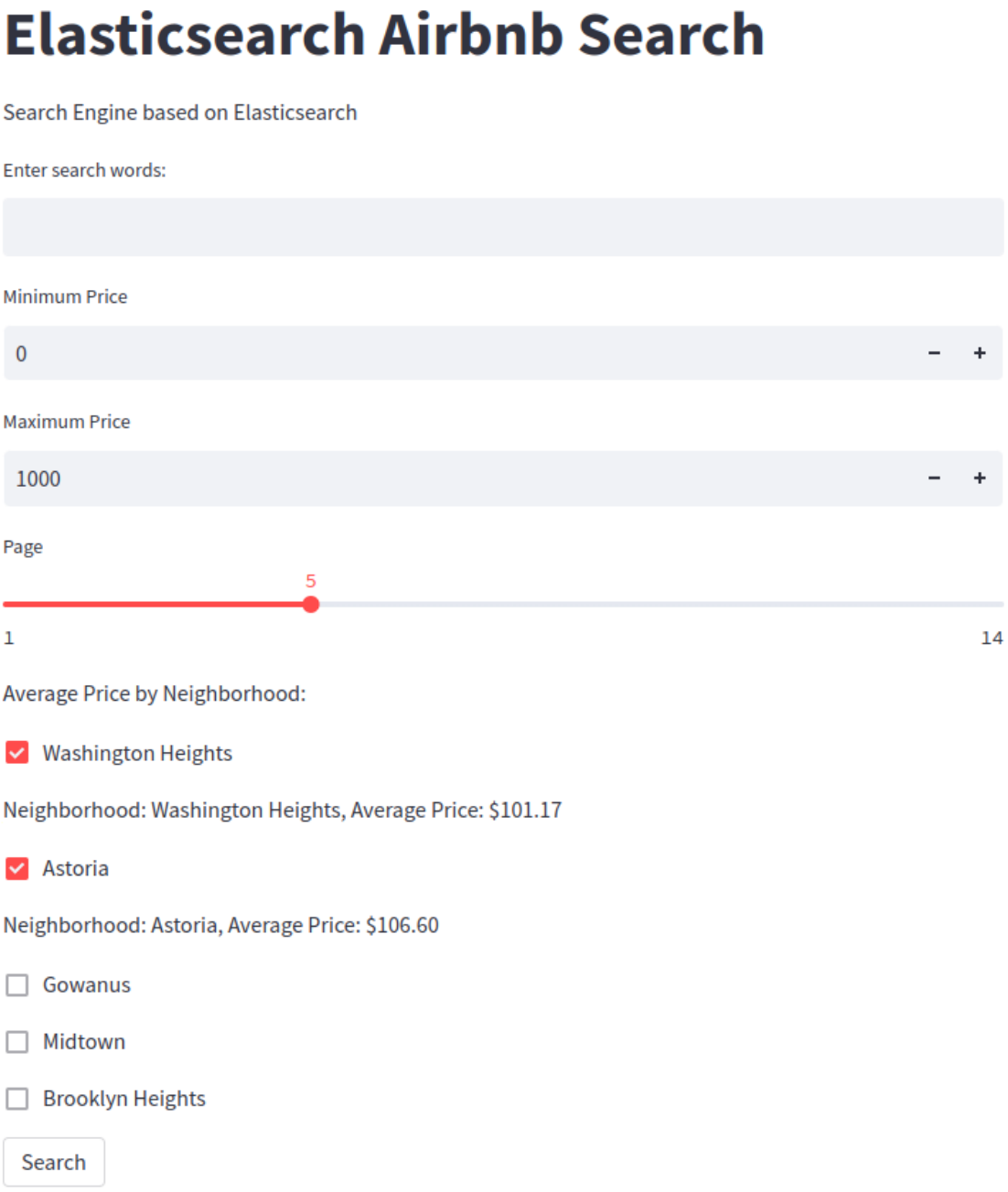
The Search Engine project can be further developed and customized according to specific requirements. Here are some potential areas for improvement:

- Enhancing the search functionality by incorporating additional search fields or filters.
- Improving the user interface by adding more interactive elements, such as dropdowns, checkboxes, or sliders.
- Implementing advanced search features, such as autocomplete suggestions or fuzzy matching.

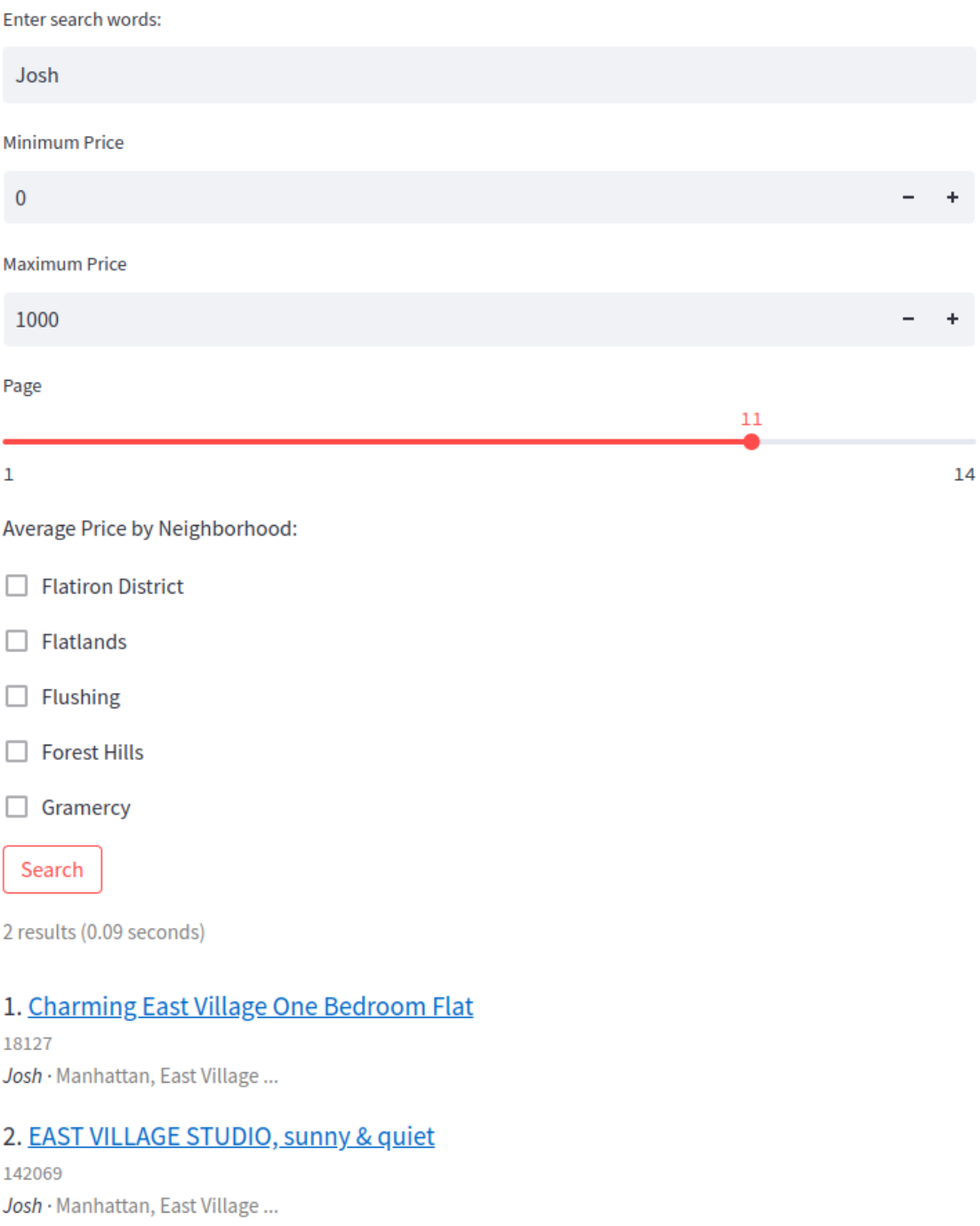
GUI EXAMPLES



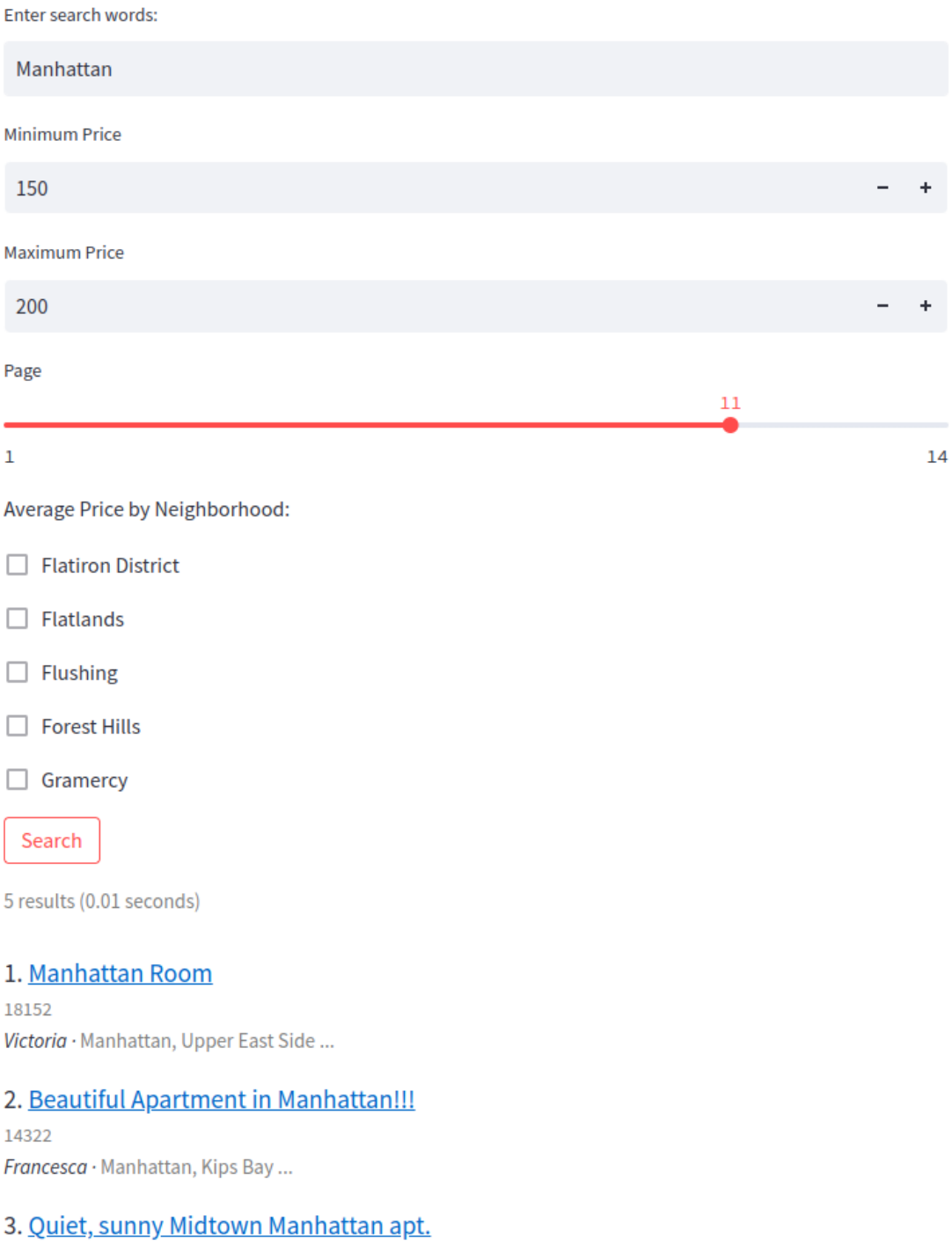
Main view of the app, any search is done yet. With those parameters and filters If you click on ‘search’ without any keyword typed you will get all the documents results. You can iterate the average price by neighborhood list thanks to the pagination made with the scrollbar ‘Page’. You also can add price filters to the searches.



This view shows how the implementation of the calculate_average_price_by_neighborhood(): works , No searches done yet.



Search by the keyword ‘Josh’. You can see the 2 results, no filters applied.



Search by the keyword ‘Manhattan’ filtered by the price range 150-200