Project 2: Prototype IR System

By

6488072	Pachanitha	Saeheng
6488086	Chutweeraya	Sriwilailak
6488189	Chommakorn	Sontesadisai
6488190	Nattanicha	Sinsawet

Present to

Dr. Suppawong Tuarob

A Report Submitted in Partial Fulfillment of the Requirements for

ITCS 414 Information Retrieval and Storage Faculty of Information and Communication Technology Mahidol University
Semester 1/2023

Table of Contents

Introduction

The specialty of Kanko Hotel Navigator is making it simple for people to locate acceptable lodging in Tokyo. By limiting alternatives, it simplifies the process of making decisions and tackles the difficulty of choosing among a multitude of hotel options. Quick and effective hotel searches are made possible by the website's user-friendly design. The capacity to pair consumers with hotels based on thorough descriptions that cover the hotel's facilities, design, and position in relation to important landmarks is one of its distinctive features. With this method, consumers should be able to select a hotel that not only suits their functional requirements but also their own personal style and preferred atmosphere. Those looking for a hotel experience that goes above and beyond the necessities—a stay that suits their unique preferences and makes the most of their trip to Tokyo—will find this service very beneficial.

Problem we are trying to solve

We're addressing the issue of finding dataset hotel information in Tokyo. Not all the information required is in one place in the current databases. Thus, we've gathered and organized hotel data from several sources. Our objective is to develop a more powerful and efficient search engine specifically for Tokyo hotels. Our goal is to create a platform that makes it easier and easier for consumers to locate what they need by providing detailed and reliable information about every hotel in the city.

Existing relevant systems

We are not aware of any other functional search tools that can look for hotels using their descriptions—which include features and style. The majority of the current platforms don't really target Tokyo and are essentially booking websites for hotel bookings based on geography rather than sophisticated search tools.

- **Agoda** (https://www.agoda.com/en-ca/?ds=yl8NXdlPyOmOqQPs)
  Agoda offers a booking system with filters for price, location, guest ratings, high-resolution photos, map views, local experience information, and over 15 million verified traveler reviews.
- **Booking** (https://www.booking.com/index.en-gb.html)
  Booking.com offers a booking system with options for hotels, flights, customizable filters for dates, location, guest numbers, amenities, ratings, and more.

Implementation

Data Collection

We obtained a CSV dataset, from the website https://www.kaggle.com which contains a wealth of information about hotels in Japan. The dataset includes descriptions of each hotel well as information about their facilities. It also provides details about the hotels location, security measures, staff quality, value for money, latitude and longitude coordinates and an overall summary score. Additionally you can find information such as the hotels name, the city it is situated in the starting price, the distance from the city center and ratings, for its brand and atmosphere.
Here is csv file of our dataset:

hotel	hostel.name	City	price.from	Distance	summary.score	rating.band	atmosphere	cleanliness	facilities	location.y	security	staff	valueformoney	lon	lat	hotel description
1	&And Hostel Akihabara	Tokyo	3600	7.8km from city centre	8.7	Fabulous	8	7	9	8	10	10	9	139.777472	35.697447	&AND HOSTEL AKIHABARA is a conveniently located hotel in Tokyo, just a 5-minute walk from Akihabara Station. The hotel offers
2	&And Hostel Ueno	Tokyo	2600	8.7km from city centre	7.4	Very Good	8	7.5	7.5	7.5	7	8	6.5	139.783667	35.712716	&And Hostel Ueno is a small contemporary hostel with private rooms and dormitory beds, conveniently located close to JR Ueno Station and
3	&And Hostel-Asakusa North-	Tokyo	1500	10.5km from city centre	9.4	Superb	9.5	9.5	9	9	9.5	10	9.5	139.798371	35.727898	Embark on an unforgettable journey in Tokyo by choosing this exceptional property as your starting point. With complimentary Wi-Fi available
4	1night1980hostel Tokyo	Tokyo	2100	9.4km from city centre	7	Very Good	5.5	8	6	6	8.5	8.5	6.5	139.78695	35.724384	1 Night 1980 Hostel Tokyo is a 5-minute walk from Iriya Subway Station Exit 4 and a 15-minute walk from JR Uguisudani Train Station. Each
5	328 Hostel & Lounge	Tokyo	3300	16.5km from city centre	9.3	Superb	8.7	9.7	9.3	9.1	9.3	9.7	8.9	139.745467	35.548044	328 Hostel & Lounge is conveniently located just a 5-minute train ride from Haneda Airport, offering comfortable accommodation with
6	3Q House - Asakusa Smile	Tokyo	2500	10.2km from city centre	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Smile Hotel Asakusa offers comfortable accommodations with a range of amenities for travelers in Tokyo. The hotel is situated just 900 meters
7	Ace Inn Shinjuku	Tokyo	2200	3km from city centre	7.7	Very Good	6.7	7.2	6.8	8.5	7.8	8.5	8.1	139.724304	35.692512	Nishitetsu Inn Shinjuku offers modern and convenient accommodations in the heart of Tokyo. Located just a short 2-minute walk from

We have a dataset that is stored in a CSV file. To make it easier to work with we transformed the CSV file into a JSON format. This initial processing step enables us to extract details as strings, from the data, such as the hotels ID, name, city review ratings for categories and a brief description of each hotel. By doing this we are preparing the data to be presented in a user organized manner, on our hotel search engine.

This Python script is created to convert data from a CSV file into a JSON file. However, it faces problems when trying to convert the CSV file because there are commas after each hotel's information. This issue arises because these extra commas disrupt the structure of the CSV file making it difficult to convert through a website interface.

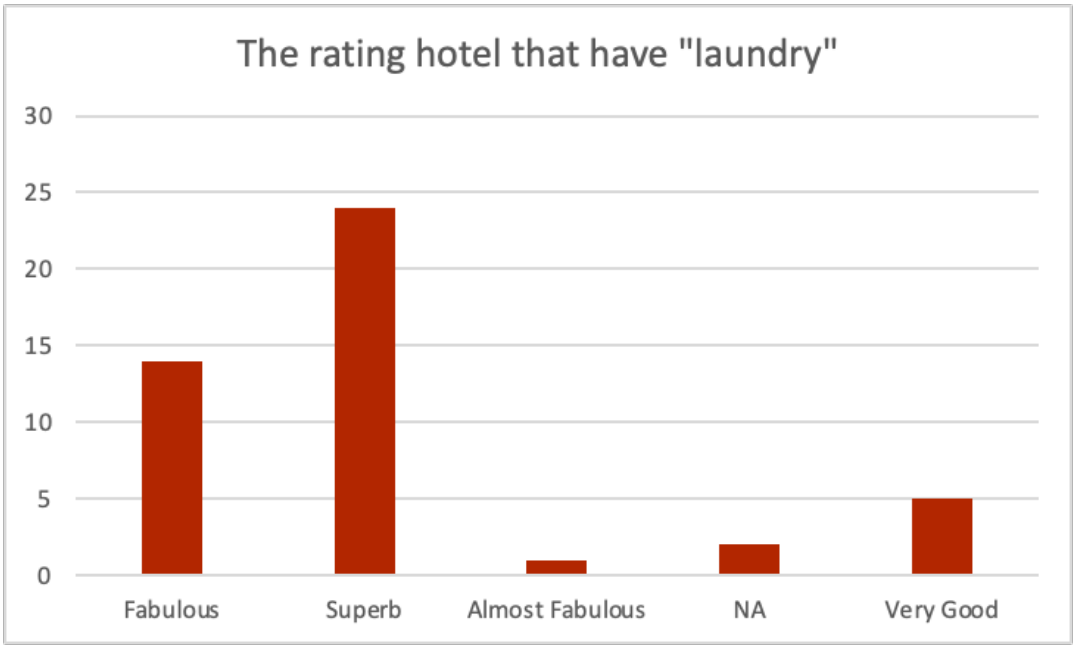
```python
import csv
import json

# Replace 'hostel.csv' with CSV file name and 'hosteldataset.json' with the desired JSON output file name
csv_file = 'hostel.csv'
json_file = 'hosteldataset.json'

data = []

# Specify the encoding when opening the CSV file
with open(csv_file, 'r', encoding='utf-8') as csvfile:
    csv_reader = csv.DictReader(csvfile)
    for row in csv_reader:
        # Create a custom JSON structure
        json_data = {
            "hostel.name": row["hostel.name"],
            "City": row["City"],
            "price.from": row["price.from"],
            "Distance": row["Distance"],
            "summary.score": row["summary.score"],
            "rating.band": row["rating.band"],
            "atmosphere": row["atmosphere"],
            "cleanliness": row["cleanliness"],
            "facilities": row["facilities"],
            "location.y": row["location.y"],
            "security": row["security"],
            "staff": row["staff"],
            "valueformoney": row["valueformoney"],
            "lon": row["lon"],
            "lat": row["lat"],
            "hotel description": row["hotel description"],
            "Link discord": row["Link discord"]
        }
        data.append(json_data)

with open(json_file, 'w', encoding='utf-8') as jsonfile:
    for item in data:
        json.dump(item, jsonfile, ensure_ascii=False, separators=(',', ':'))
        jsonfile.write("\n")
```

Example of document

After using the Python code to transform the CSV file we obtain a JSON file that accurately represents all the information, about the hotels mentioned in the CSV. This JSON version maintains all the information, guaranteeing that the specifics of each hotel are preserved in a well-organized.


```
{
    "hostel.name":"Ikidane House Asakusa Hatago",
    "City":"Tokyo","price.from":"2000",
    "Distance":"10km from city centre",
    "summary.score":"8.9",
    "rating.band":"Fabulous",
    "atmosphere":"10",
    "cleanliness":"10",
    "facilities":"10",
    "location.y":"4",
    "security":"10",
    "staff":"8",
    "valueformoney":"10",
    "lon":"139.795049",
    "lat":"35.722089",
    "hotel description":"This cosy Ikidane House Asakusa Hatago Hotel Tokyo lies in 20 minutes'
    stroll from Tokyo Skytree. The 2-star guest house also offers guests Wi Fi in public areas. Situated in Taito district,
    the accommodation is set 2.5 km away from Edo-Tokyo Museum. Kaminarimon Gate Senso-ji is set within 1.4 km to the property.
    Bathrooms, equipped with a separate toilet, a bath and a shower, also feature a hair dryer and bath sheets.
    The lounge bar is ideal for a relaxing drink. Suzune serves Japanese dishes and lies 200 metres of Ikidane House Asakusa Hatago Hotel.
    Minowa underground station can be reached in 10 minutes by foot. It takes around 41 minutes' drive to get to
    Tokyo International airport, which is 35 km away.",
    "Link discord":"https://cdn.discordapp.com/attachments/1173184141655801927/1173234514529620098/Ikidane_House_Asakusa_Hatago.jpg?
    ex=656336c9&is=6550c1c9&hm=9457dc350bd826ab7afbf0a44f15d290e9831bf2c364be23a75f0b597325a7fd&"
}
```

Data Statistic



This chart appears to depict the number of hotels offering laundry services from 100 documents, categorized by different rating levels.

Here's a summary of the data shown in the bar chart:

- Fabulous: There are around 10 hotels with this rating that offer laundry services.
- Superb: This category has the highest number of hotels, with over 25 hotels rated as superb and providing laundry services.
- Almost Fabulous: There are very few, possibly just 1 or 2, hotels in this rating category with laundry services.
- NA: NA likely stands for not available or not applicable, indicating no rating. There are a small number of such hotels, similar to the "Almost Fabulous" category.
- Very Good: There are around 5 hotels classified as "Very Good" that have laundry services available.

This chart provides a visual representation of the distribution of hotel ratings for those hotels that have laundry facilities. From the chart, it is clear that the majority of hotels with laundry services fall under the "Superb" rating category.

Tool and Software

Here is the list of tools, software, and dependencies that we used in this project, and also what they are used for:

1. Elasticsearch

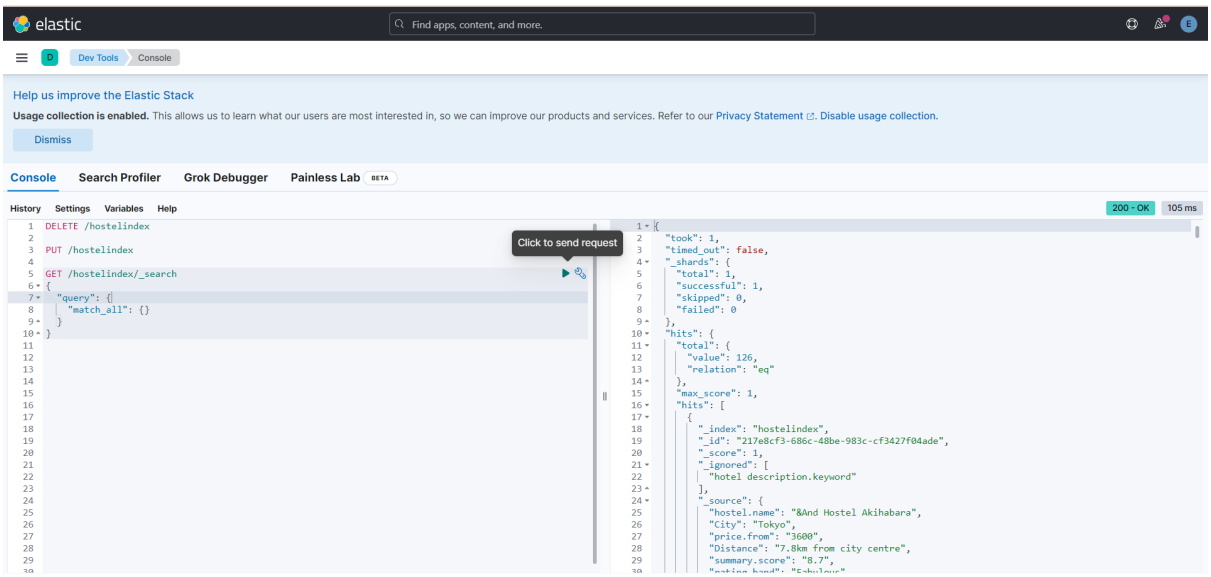
```
PUT /hostelindex

GET /hostelindex/_search
{
  "query": {
    "match_all": {}
  }
}
```

Elasticsearch, which is a distributed, RESTful search and analytics engine. It provides instructions for using the PUT method to create an index without specifying a mapping query. Additionally, Python file named elastic_loader.py, implying this file is used to upload a JSON file to Elasticsearch. There are also example commands shown for interacting with Elasticsearch:

1. A PUT command to create or update an index named hostelindex.
2. A GET command to perform a search query on hostelindex using the _search endpoint, with a query that matches all documents in the index.

The format of the commands suggests that these are HTTP requests, which are typically used in conjunction with tools like curl in a terminal or within a Python script using libraries like requests to interact with the Elasticsearch API.

2. Kibana (Elasticsearch Kibana)



Kibana is a data visualization and management tool included in the Elasticsearch suite. The console part of Kibana, a tool for interacting directly with Elasticsearch data via a web interface, is visible in the screenshot.

- Here are some commands that you can use in the input space on the left;
    - To delete an index called hostelindex use the command DELETE /hostelindex.
    - If you want to create a hostelindex or update an existing one you can use the command PUT /hostelindex.
    - To retrieve data from the hostelindex based on a query you can use the command GET /hostelindex/_search. For example if you want to get all the documents, in the index without any filters you can use the query {"query"; {"match_all"; {}}}.
- To the right you'll find the output section where you can see the outcome of the command executed (the GET request). The response confirms a search (200 OK). Reveals a total of 126 results. This section of the response provides information, about the retrieved data, including hostel names, locations, prices distances, from the city center and a summary score.
- There is a button, in color which displays the text "Click to send request." You can click on it to execute the commands you have entered in the input section.
- On the part of the console there is a notification, from Elastic that mentions about enabling usage collection. This feature is commonly included for purposes.

The Kibana console is a tool that allows developers and administrators to interact directly with their Elasticsearch indices. It enables them to execute queries make modifications and gain insights, from their indexed data.

3. Python

Python is used for several activities, such as converting CSV data to JSON format, storing and retrieving data with Elasticsearch, and maybe creating a web interface with Flask to communicate with Elasticsearch. These are typical duties for online data management and presentation.

- **Flask**

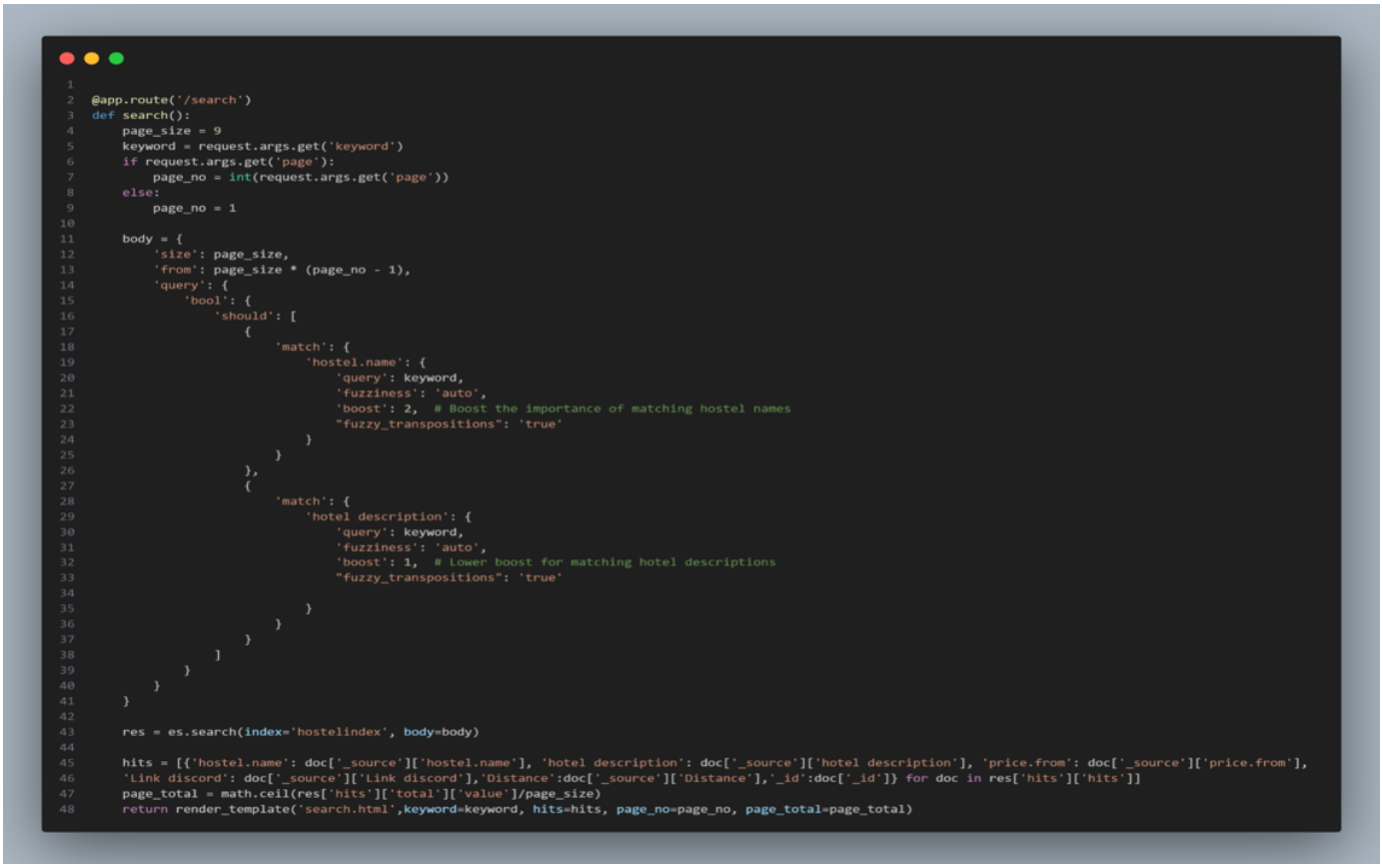
  Flask, a Python library used for creating web applications is capable of displaying web pages. Can be configured to allow users to search for information, on the website. Additionally the code establishes a connection to Elasticsearch, a tool designed for efficient data searching. In order to run the Flask application smoothly it is necessary to set the environment variables "FLASK_APP". FLASK_ENV" as 'app' and 'development' respectively. This particular configuration is vital, for executing the Flask application using the command flask run.

```python
from flask import Flask, request
from markupsafe import escape
from flask import render_template
from elasticsearch import Elasticsearch
import math

ELASTIC_PASSWORD = "Harukyu0004"

es = Elasticsearch("https://localhost:9200", http_auth=("elastic", ELASTIC_PASSWORD), verify_certs=False)
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')
```

These specific details would be incorporated into sections of the code known as route functions. In this case a function, like "index()" would be used. However, it is important

to note that the provided snippet does not demonstrate an instance of Flask requesting data, from Elasticsearch yet.

- **app.route(‘/search’)**
  In a Python web application we often use the decorator @app.route('/search') to associate an URL (in this case '/search') with a Python function. This is a practice when working with web frameworks, like Flask.



```python

@app.route('/search')
def search():
    page_size = 9
    keyword = request.args.get('keyword')
    if request.args.get('page'):
        page_no = int(request.args.get('page'))
    else:
        page_no = 1

    body = {
        'size': page_size,
        'from': page_size * (page_no - 1),
        'query': {
            'bool': {
                'should': [
                    {
                        'match': {
                            'hostel.name': {
                                'query': keyword,
                                'fuzziness': 'auto',
                                'boost': 2,  # Boost the importance of matching hostel names
                                "fuzzy_transpositions": 'true'
                            }
                        }
                    },
                    {
                        'match': {
                            'hotel description': {
                                'query': keyword,
                                'fuzziness': 'auto',
                                'boost': 1,  # Lower boost for matching hotel descriptions
                                "fuzzy_transpositions": 'true'

                            }
                        }
                    }
                ]
            }
        }
    }

    res = es.search(index='hostelindex', body=body)

    hits = [{'hostel.name': doc['_source']['hostel.name'], 'hotel description': doc['_source']['hotel description'], 'price.from': doc['_source']['price.from'],
    'Link discord': doc['_source']['Link discord'],'Distance':doc['_source']['Distance'],'_id':doc['_id']} for doc in res['hits']['hits']]
    page_total = math.ceil(res['hits']['total']['value']/page_size)
    return render_template('search.html',keyword=keyword, hits=hits, page_no=page_no, page_total=page_total)
```

In the provided code snippet we can observe that @app.route('/search') is utilized to link the '/search' URL path to the search() function. Whenever a user visits that URL the search() function is. Seems to handle search queries based on user input. It then returns search results for display, on a web page. This approach is widely used in web development to establish routes that cater to URLs by implementing custom logic.

- **app.route(‘/detail’)**
  This code snippet helps you create a pathway in a web application that can accommodate URLs with varying 'hostel_id' values. It enables you to construct web pages or APIs that respond to input values provided in the URL making your web application more adaptable and interactive.

```python
@app.route('/detail/<string:hostel_id>')
def detail(hostel_id):
    try:
        # Fetch details from Elasticsearch based on the _id
        res = es.get(index='hostelindex', id=hostel_id)

        # Extract relevant information from the Elasticsearch response
        hostel_details = {
            'hostel_name': res['_source']['hostel.name'],
            'price_from': res['_source']['price.from'],
            'description': res['_source']['hotel description'],
            'link_discord': res['_source']['Link discord'],
            'distance': res['_source']['Distance'],
            'rating': res['_source']['summary.score'],  # Add rating field
        }
        return render_template('detail.html', hostel_details=hostel_details)

    except Exception as e:
        # Log the exception for debugging
        print(f"An error occurred: {e}")
        # Return an error page or handle the error appropriately
        return render_template('error.html', error_message='An error occurred while fetching hostel details.')
```

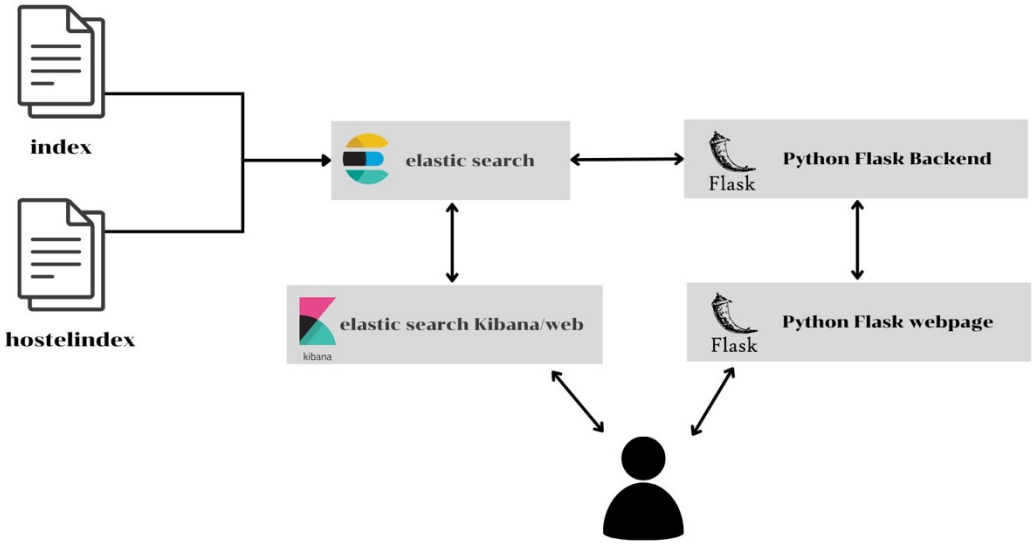
The provided code sets up a route, for displaying information about hostels based on an identifier passed through the URL. It retrieves the details, from an Elasticsearch index. Handles potential exceptions gracefully ensuring that users receive either the hostel details or an error message if any issues arise. This is a common pattern in web development for displaying dynamic content based on user input.

- **Elasticsearch**
  The Elasticsearch class functions, as a toolbox within the Elasticsearch Python client library. Its purpose is to assist in creating tools (instances) that can communicate with Elasticsearch, which's a data storage system. These tools have capabilities, such as data searching, data insertion and data organization within Elasticsearch.

  The provided code acts as a tool for inserting JSON data into an Elasticsearch index. It establishes a connection, to Elasticsearch reads JSON data from a file and utilizes indexing to optimize the process of storing the data in the designated index. This script proves valuable when working with scenarios that involve populating or updating an Elasticsearch database with amounts of JSON information.

```python
from elasticsearch import Elasticsearch, helpers
import ndjson
import argparse
import uuid
import chardet
import certifi


es = Elasticsearch("https://localhost:9200",
    http_auth = ("elastic", "Harukyu0004"),
    scheme = "https", port = 443, maxsize = 5,
          use_ssl=True, ca_certs=certifi.where(), verify_certs=False)

parser = argparse.ArgumentParser()
parser.add_argument('--file')
parser.add_argument('--index')
# parser.add_argument('--type')

args = parser.parse_args()

index = args.index
file = args.file
# doc_type = args.type


with open('hosteldataset.json', 'r', encoding='utf-8') as json_file:
    json_docs = ndjson.load(json_file)

with open('hosteldataset.json', 'rb') as rawdata:
    result = chardet.detect(rawdata.read())
    encoding = result['encoding']

with open('hosteldataset.json', 'r', encoding=encoding) as json_file:
    json_docs = ndjson.load(json_file)

def bulk_json_data(json_list, _index):
    for doc in json_list:
        yield {
            "_index": _index,
            "_id": str(uuid.uuid4()),  # Convert UUID to string
            "_source": doc
        }

try:
    response = helpers.bulk(es, bulk_json_data(json_docs, index))
    print ("\nRESPONSE:", response)
except Exception as e:
    print("\nERROR:", e)
```

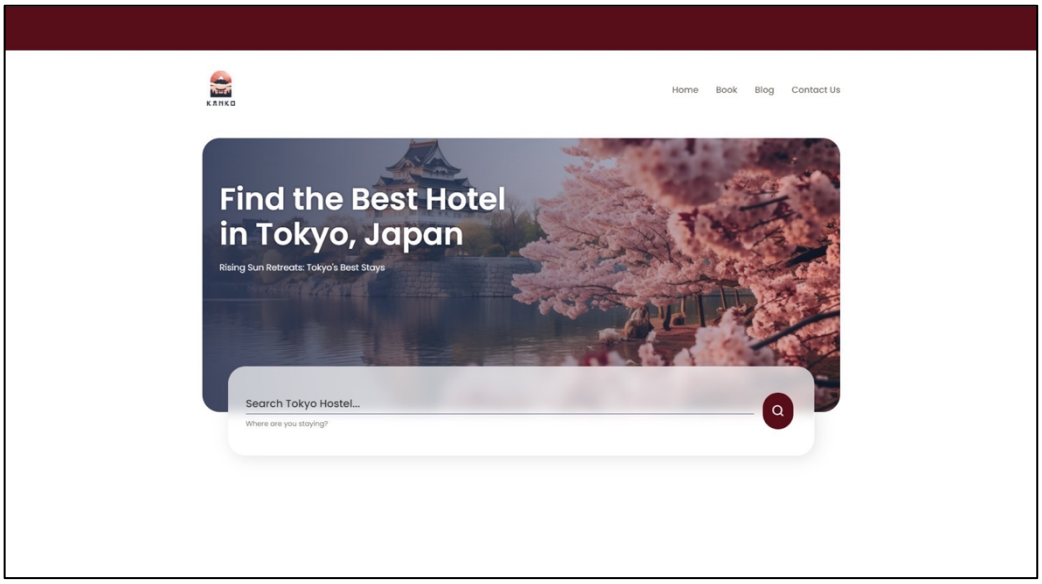
System Diagram



The diagram shows a simple system setup. The main part is the "hostelIndex," used for searches. Another part is used for adding initial data and testing. Elasticsearch organizes this data and sends it to Kibana, which helps users fix any problems in the system.

On the right side of the diagram, Python Flask is used. It connects to Elasticsearch and helps make a webpage using the code explained in the previous page.

System Snapshot



Our web search engine offers a user interface that is inspired by the essence of Japan ensuring an experience, for users looking to find hotels, within the country. With a search bar and a dedicated search button it empowers users to effortlessly explore and discover hotels that perfectly suit their requirements.

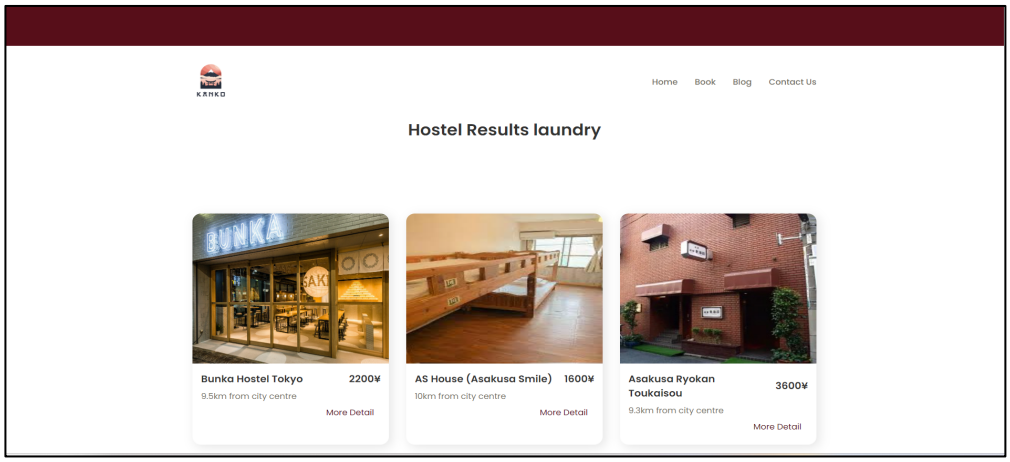
CASE 1: Searching for one word.

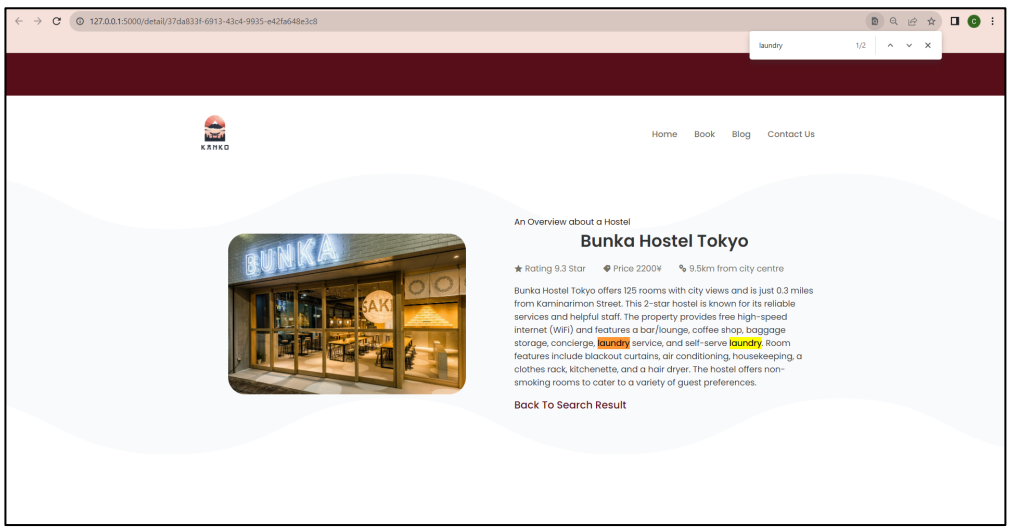
The image is a snapshot from the KANKO search engine, showing results for places to stay in Tokyo with laundry facilities, after searching for the term "laundry".

Here's what the search found:

1. "Bunka Hostel Tokyo", about 9.5km from downtown, with a nightly price of 2200¥. You can see the hostel's name lit up on its entrance in the photo.
2. "AS House (Asakusa Smile)", a bit farther at 10km from the center, costs 1600¥ per night. The photo shows a room with wooden bunk beds.
3. "Asakusa Ryokan Toukaisou", which is 9.3km from the center and a bit pricier at 3600¥. Its photo features the building's brick facade.

Clicking "More Detail" under each listing presumably gives more info about each place. This image tells us that KANKO's search engine can quickly give you a list of hostels with specific features like laundry services, just by searching for one keyword.





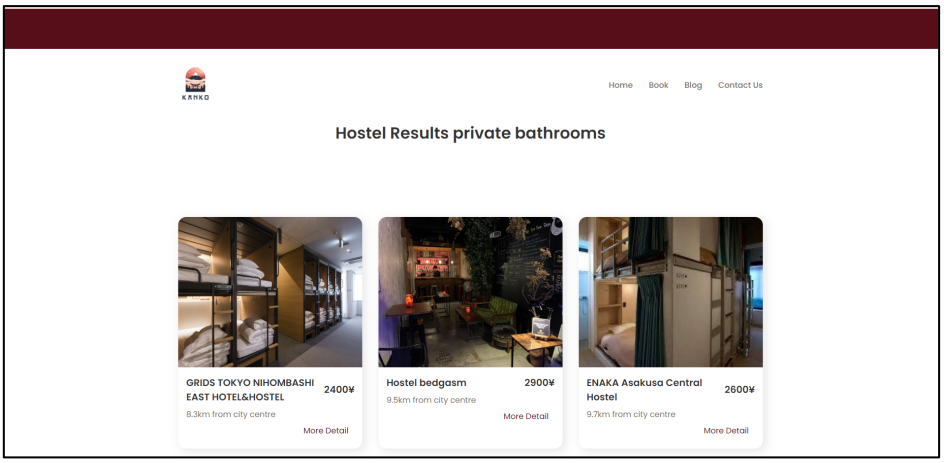
CASE 2: Searching for Multi word.

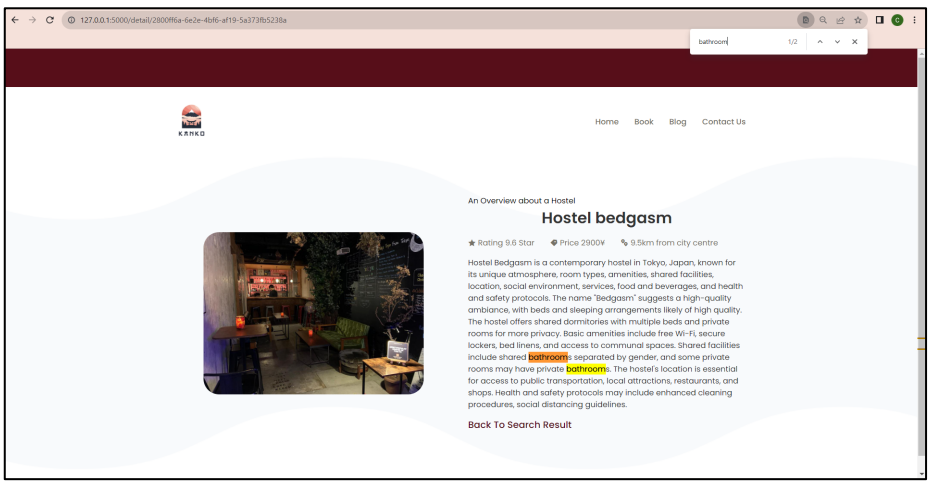
The image is a screenshot from the KANKO website, displaying results for hostels in Tokyo that come with bathroom. The search seems to have been done using the phrase "bathrooms".

Here's what's listed:

1. "GRIDS TOKYO NIHOMBASHI EAST HOTEL&HOSTEL", which has rooms for 2400¥ per night and is 8.3km from downtown. It shows a photo of a tidy room with bunk beds arranged neatly.
2. "Hostel bedgasm", a bit closer to the city center at 5.8km, charges 2900¥ for a night. The accompanying image gives off a warm vibe with its dimly lit social space.
3. "ENAKA Asakusa Central Hostel" is a little farther at 9.7km away, with a price of 2600¥ a night. The photo shows a basic room with bunk beds.

Furthermore, there's more information about "Hostel bedgasm", noting its cozy atmosphere and the bonus of rooms with private bathrooms. The write-up mentions a mix of old and new decor styles and underscores the privacy and comfort provided by en-suite bathrooms in some rooms. There's also a link provided to go back to the search results, making it convenient to keep looking at other options.



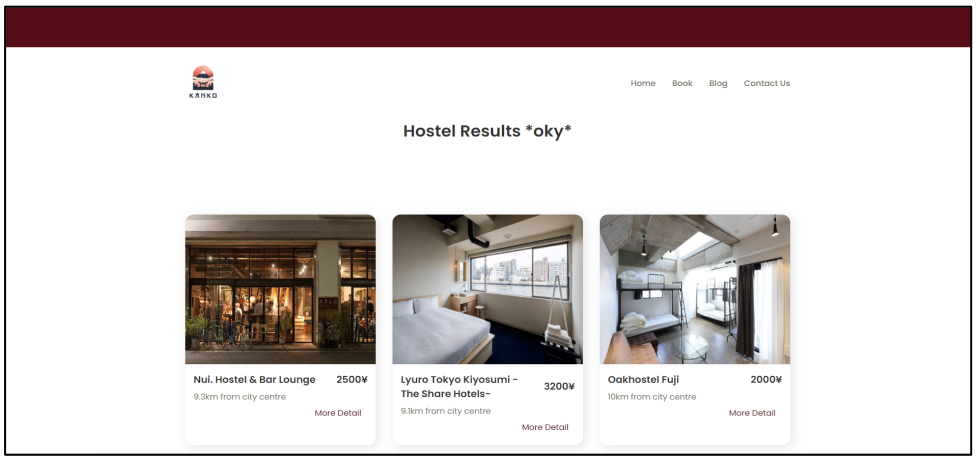


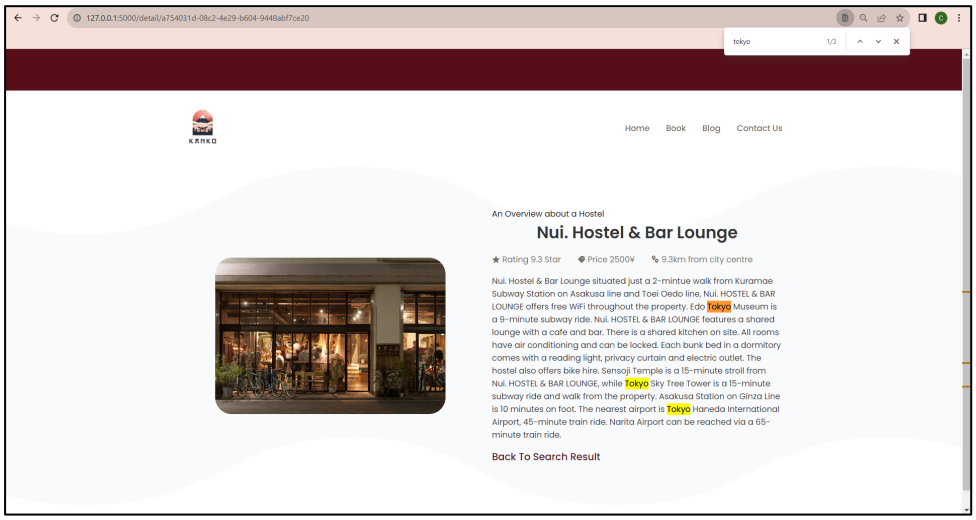
CASE 3: Searching for partial match.

The image shows a KANKO search engine page that's returned some hostels in Tokyo after someone looked up "*oky*". This might be part of a longer word or just a snippet of a hostel's name.
Here's what came up:

1. "Nui. Hostel & Bar Lounge" costs 2500¥ for a stay and it's 3.5km from downtown. The photo shows a cozy bar and lounge spot.
2. "Ivyro Tokyo Kiyosumi - The Share Hotels" has a room for 3100¥, and it's 5km out from the center. Its photo shows a neat, shared room with bunk beds.
3. "Oakhostel Fuji", which is a bit further at 10km from the center, is the cheapest at 2000¥. The picture looks like it's of a neat bathroom or laundry room.

These results indicate that KANKO's search is smart enough to find the right places even if you don't type the whole word. You get pictures, prices, how far they are from the city center, and links to more info, making it easier to find what you need with just a few letters.





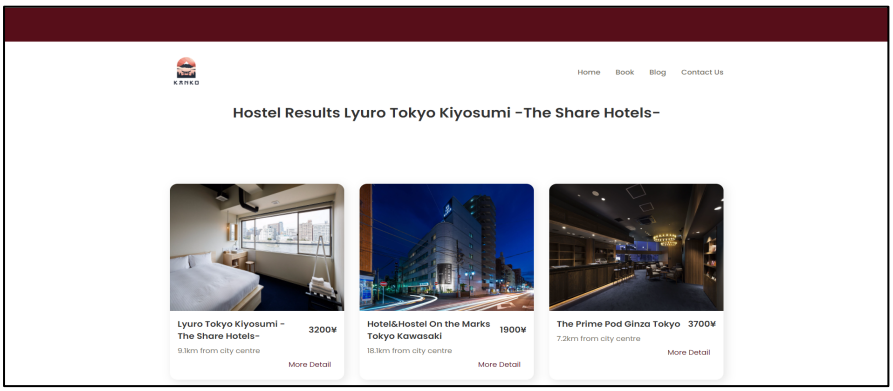
CASE 4: Ranking the best match, searching for the hostel name first (first priority)

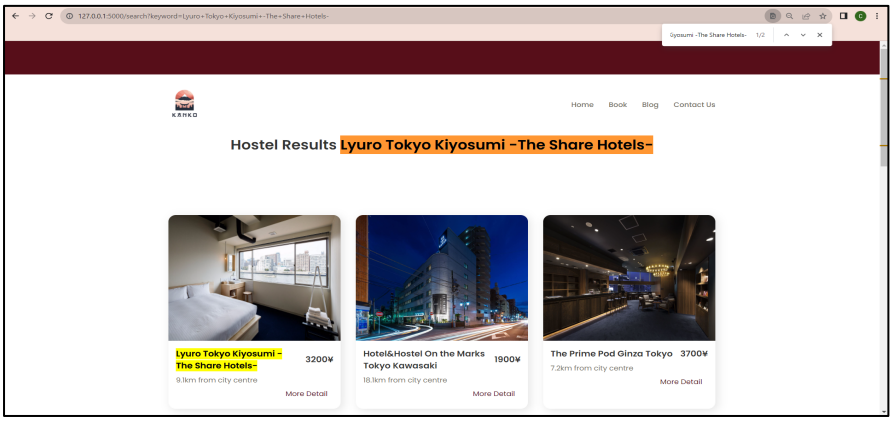
The image is a screenshot from the KANKO website showing results for a hotel named "Lyuro Tokyo Kiyosumi - The Share Hotels". The site seems to sort the results by how closely they match the search term.

Here's what it found:

1. "Lyuro Tokyo Kiyosumi - The Share Hotels", the exact hotel searched for, costs 3200¥ per night and is about 8.1km from the center of Tokyo. The picture shows a well-lit, modern room.
2. "Hotel&Hostel On The Marks Tokyo Kawasaki" comes up next at 1900¥ a night, but it's further out at 18.1km from the center. Its photo shows the building's exterior in the evening.
3. "The Prime Pod Ginza Tokyo" appears third, at 3700¥ for a night and 7.2km from the city center, the nearest of the three. The picture shows a chic bar area inside the hostel.

The search listings seem well-organized, with the exact match appearing first, and the rest follow based on keyword similarity. Each option lists its price, how far it is from the center, and has a photo preview, plus a link to learn more. This neat arrangement implies that KANKO is good at finding exactly what you're looking for by the name of the hotel.





Discussion

Limitations of your system

The search engine is designed to search exclusively within the Tokyo area. This means that users looking for hotels, hostels, or other accommodations with specific amenities, such as laundry services, can only receive results located in Tokyo. While this focuses the search experience for users interested in Tokyo specifically, it also means that those looking for options in other cities or regions will need to use a different search engine or platform that covers a broader geographic area.

Technical difficulties

This report highlights our Technical difficulties with Elasticsearch connectivity and image display issues, detailing the challenges and solutions we found during the project

- **Elasticsearch Connection Error**
  During development, we faced several technical issues with Elasticsearch, including connection problems, certificate errors, SSL issues, and failed connection verifications. Resolving this required extensive research from StackOverflow and other online resources. We experimented with various methods and continuously tested until we found an effective solution.

- **Issues with Image Display**
  After using the converted JSON data on our website, we encountered issues displaying hotel images, which did not appear as expected. This led us to analyze and modify the image links to a different format. This challenge involved finding new image sources and adapting our code to the new format.

Challenges

This is details about the Challenges we encountered in our project, including the issues we found with our website and the methods we implemented to overcome them.

- **Navigating CSS Complexity with Figma**
  At the beginning of our project, we chose Figma to design our website, expecting it to provide CSS suitable for HTML development. However, we encountered challenges as the CSS from Figma was more complex than we had anticipated. This complexity made HTML coding difficult, especially in managing classes and adapting them to the CSS. As a result, we decided to use a ready-made template and modify it to fit our design.

- **Data Conversion and Integration Challenges with Elasticsearch**
  The next step was to integrate with Elasticsearch. This phase tested our skills in working with databases and searching for information efficiently. One of the significant challenges was converting hotel data from CSV to JSON format for our website. The main issue was handling the comma after the first item of each hotel's data. It required meticulous work and extensive testing to format the data correctly and make it effectively usable.

Lessons learned

This reflects the key lessons from our project, including the importance of using the Terminal correctly and the value of teamwork and continuous learning in software development.

- **Using the Terminal**
  We learned an important lesson: not to close the Terminal while the application is running. Closing it can lead to issues that require time-consuming repairs or restarts.

- **Learning and Improving**
  - This project significantly improved our understanding of using different software, focusing on problem-solving and enhancing various aspects of the website.

  - Teamwork was essential, as it enabled us to collectively learn from and address the challenges we faced together.

Opportunities for future improvements

Recognizing the current limitation of the search engine being restricted to Tokyo, there are several opportunities for improvement:

1. **Expanding Geographic Scope**: Extending the search capabilities to other popular regions like Osaka, Kyoto, or Hokkaido would provide users with a wider range of options and make the platform more versatile.
2. **Detailed Filter Options:** Adding filters for more specific search criteria, such as room type, pricing options, proximity to landmarks, or specific services like free Wi-Fi, pet-friendly accommodations, or family rooms, can significantly enhance the user experience.
3. **Multi-Language Support:** Catering to international tourists through multi-language support can broaden the user base and make the platform more accessible.
4. **User Reviews and Ratings Integration:** Incorporating user reviews and a robust rating system can help others make informed decisions based on the experiences of previous guests.
5. **Partnership with Local Businesses:** Collaborating with local tourism businesses could offer users special deals or packages, encouraging bookings through the search engine.
6. **Mobile Optimization:** Ensuring the platform is optimized for mobile devices would cater to on-the-go travelers who rely on their smartphones for bookings.
7. **Real-Time Availability and Pricing:** Implementing a system that provides real-time data on room availability and dynamic pricing would be highly beneficial for users seeking immediate bookings.
8. **Personalization:** Introducing a personalized recommendation system based on user behavior and preferences can improve the search results and satisfaction.
9. **Enhanced Visual Content:** Providing high-quality images and virtual tours of accommodations can greatly influence user decisions and improve the overall appeal of listings.
10. **Sustainable and Eco-Friendly Options:** Highlighting eco-friendly and sustainable lodging options can attract environmentally conscious travelers.

Conclusion

Beyond its role in web development, this project served as a profound learning journey. We honed vital skills in technical problem-solving, navigating data intricacies, fostering teamwork, and embracing adaptability in the face of change. Each obstacle encountered transformed into a precious lesson, enriching our toolset for future endeavors. This experience taught us that growth emerges not only from success but from the challenges that push us to evolve. Armed with these competencies, we are better equipped to tackle the dynamic landscape of future learning opportunities with confidence and resilience.

Reference

1. **HTML/CSS Project – Gtravel**
   WebDesignMastery, "GTravel Project," [Online]. Available: https://github.com/WebDesignMastery/GTravel_18-06-23.git. Accessed on: 2023-11-16.

2. **HTML/CSS Project - WDM-Co**
   WebDesignMastery, "WDM-Co Project," [Online]. Available: https://github.com/WebDesignMastery/WDM-Co_07-07-23.git. Accessed on: 2023-11-16.

3. **Elasticsearch Python Library - Issue 275**
   Elastic, "Issue 275 on Elasticsearch-py," [Online]. Available: https://github.com/elastic/elasticsearch-py/issues/275. Accessed on: 2023-11-18.

4. **Kaggle Dataset - Hostel World**
   K. Ando, "Hostel World Dataset," [Online]. Available: https://www.kaggle.com/datasets/koki25ando/hostel-world-dataset. Accessed on: 2023-11-18.