Data Structure

Team: 10

Team members: 111306001 資管二 林美婷 資管二 111306049徐華均

111306055 資管二 呂家華

Topic: Trip searching website and (Android) APP:

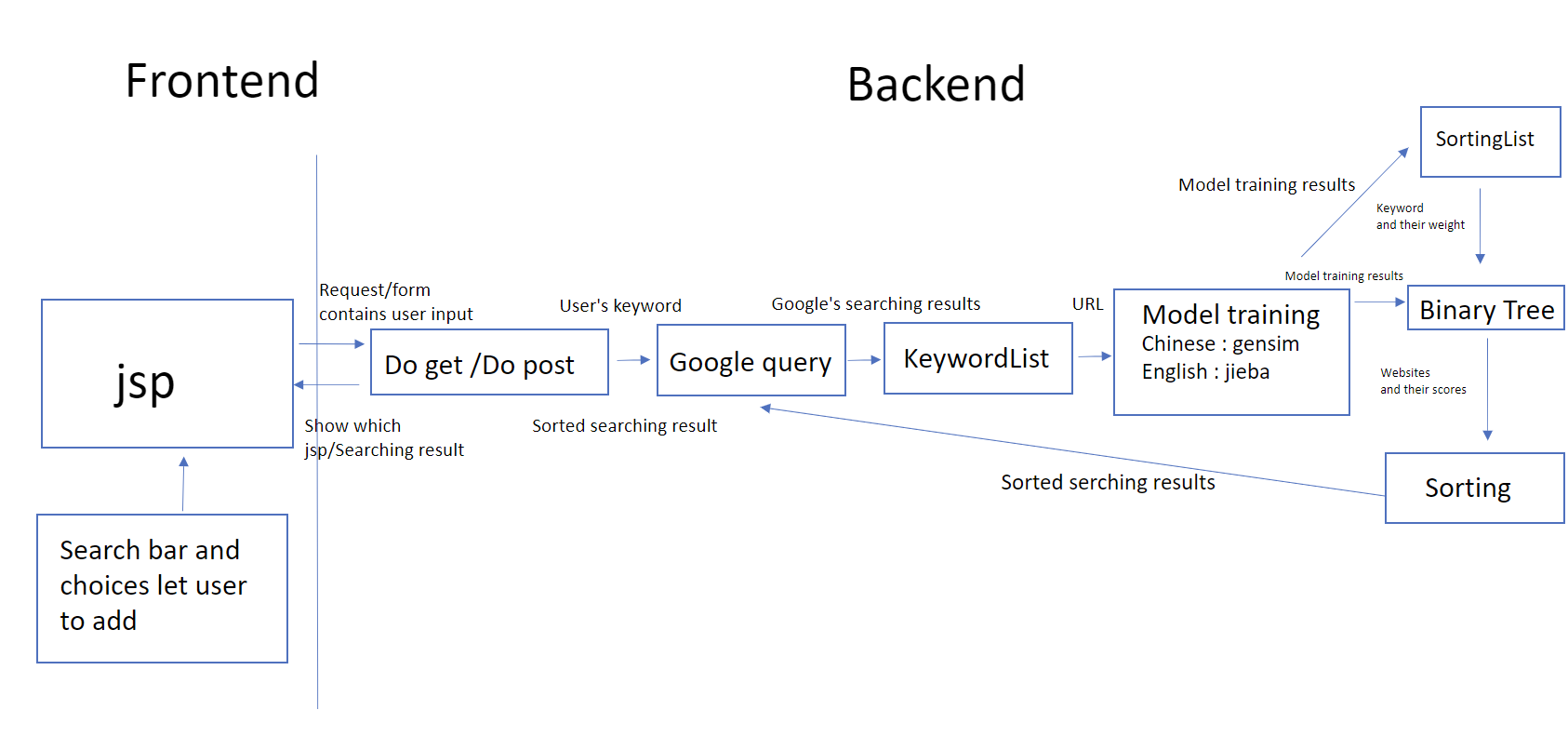
1. Introduction

If we simply enter a specific location into the Google search engine, we may encounter the following issues:

1. Too much ad: The results might be flooded with advertisements.
2. Inaccurate webpage results: Users might receive imprecise information due to various reasons such as incomplete planning or vague search queries.
3. If someone wishes to search for specific information about that location, such as recommended itineraries or visa application procedures, Google might not be able to accurately filter the desired results.

For these reasons, we aim to create a dedicated search engine tailored specifically for international travel purposes.

1. Search tricks



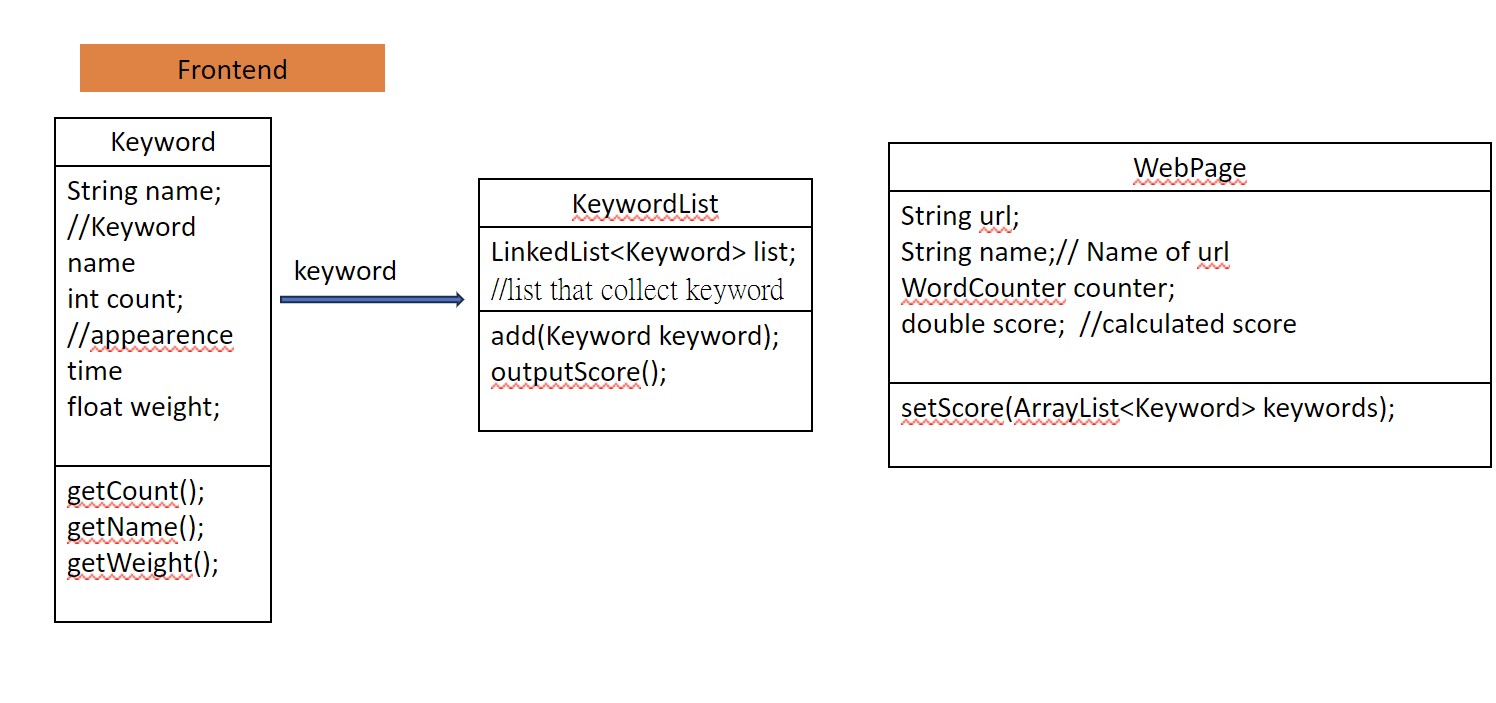
|  |  |
| --- | --- |
| Search tricks | Exmaple |
| User will input journey they want to search on Web page or phone app  \* There will be checkbox for users to select either "Hotel”, “Recommendations”, or "Ticket" to refine their search results. | The user inputs "Europe" on their mobile device and selects the category "Trip." |
| The user's entered keywords and category are input into the Google search engine, generating numerous websites with relevant information. | The user inputs "Europe" on their mobile device and selects the category "Trip plan”. |
| Perform text analysis on the website content to identify additional related keywords (sub-keywords) associated with the user's input. These could aid in more accurately refining the search results. | Analyze the text of the resulting websites to identify similar keywords and their respective similarities. |
| We'll score and filter these sub-keywords to build a scoring tree, using it to assess whether the webpage meets the user's needs. | Further sort the search results based on the most important sub-keywords obtained from the previous step. |
| After constructing the tree, we'll have the final score for each webpage. We'll then sort these scores in descending order and transmit them to the frontend for display to the user. | Once sorted, present the results (websites) to the user in descending order. |

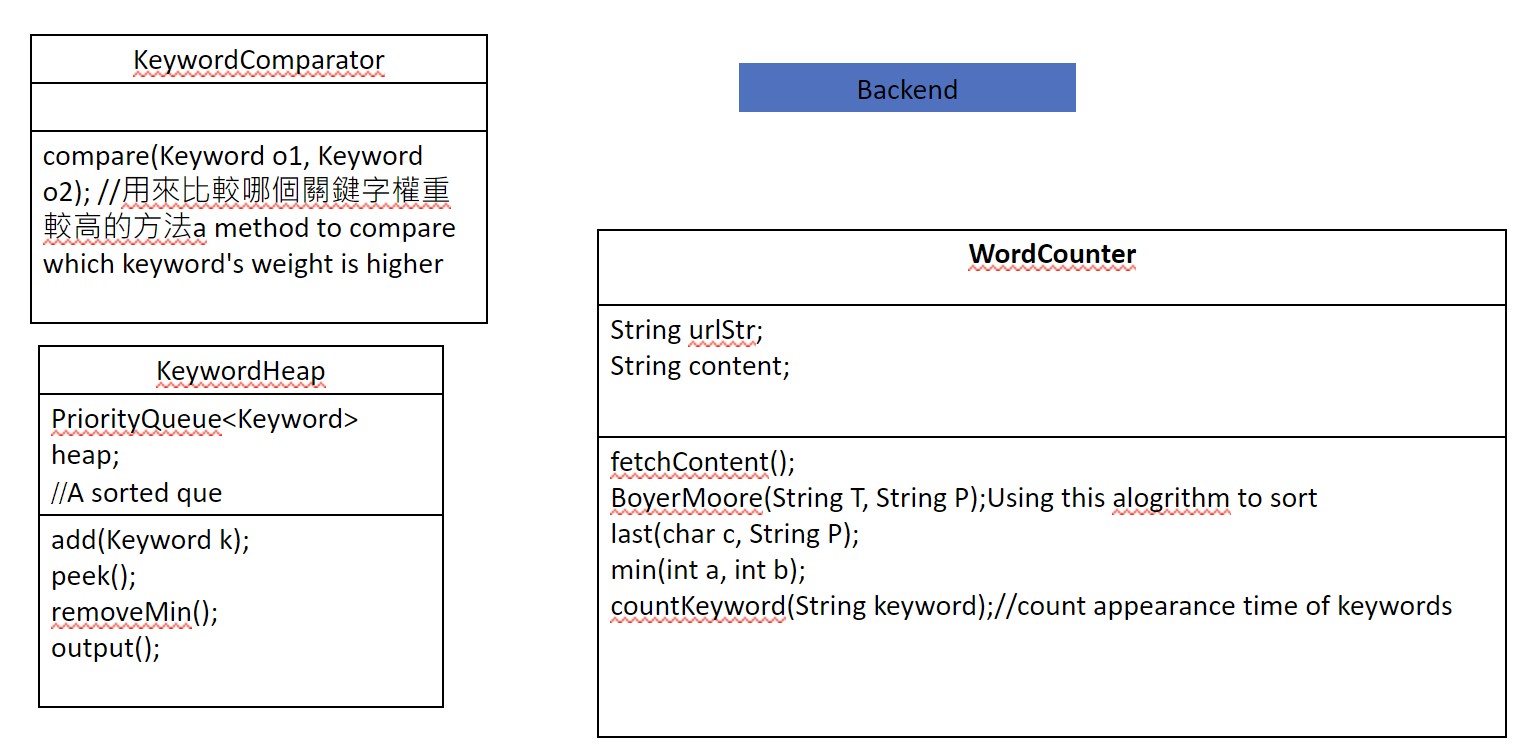
1. Score formulation

Total score = Number of occurrences of a keyword \* Weight of that keyword.

After using a large amount of text to train the word vector of the text, we can use word vector to measure the similarity of two words. The specific similarity calculation method is cosine-similarity (word vector of two words) because similarity can also show how much a certain word resembles the given keyword, all weights are based on the similarity just given. Since the value of similarity is -1~1, if it is 1, and we hope that the weight of the original keyword is still greater than the keyword we recommend, so in the end we will weight = similarity × 0.5. The geographical location is only for convenience of filtering and will not be involved in the weight calculation.

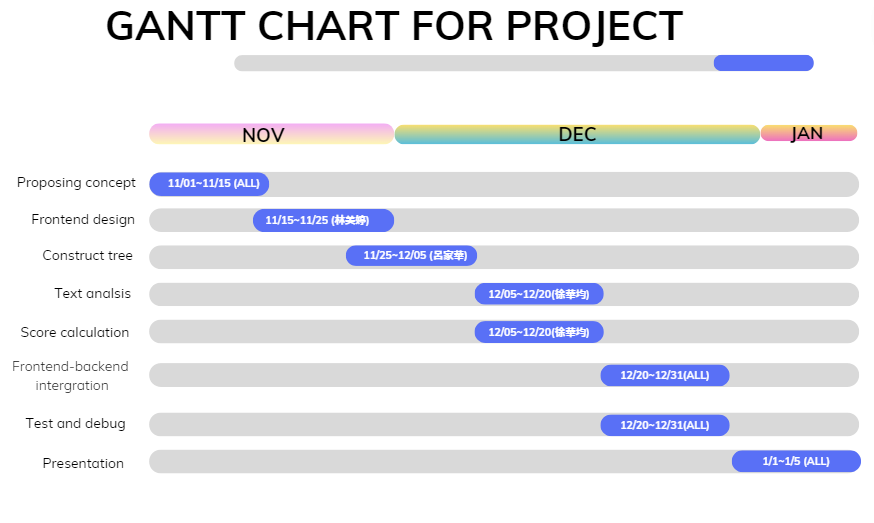
1. System design & Class diagrams



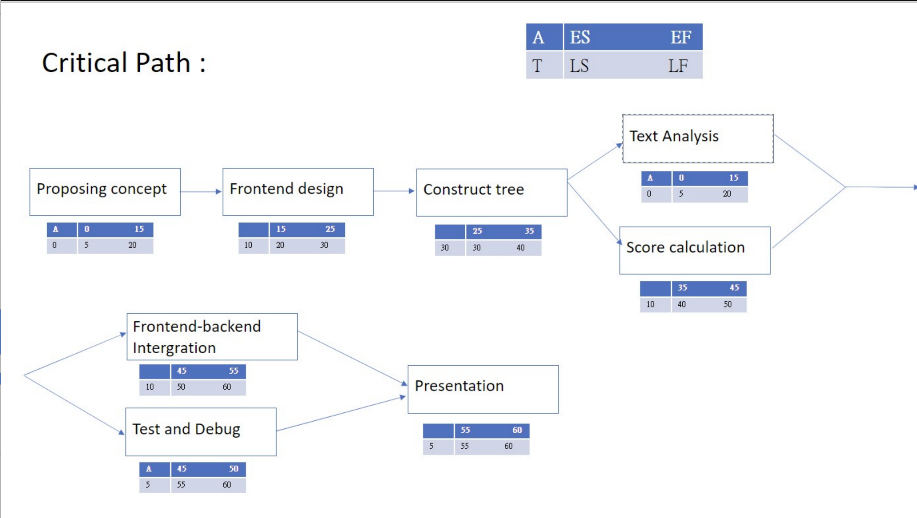


(The class and method would be adjusted before deadline)

1. Frontend: We plan to utilize a cross-platform framework to develop the UI, enabling deployment on both mobile and desktop browsers simultaneously.
2. Submit the keywords to Google search. (Reference: HW4) F
3. Create the scoring tree. (Reference: HW6)
4. For text analysis, we'll employ Word2Vec technology to convert text into vectors and calculate the similarity between two keywords using a pre-trained model in Python. (Reference: HW5)
5. Transmit the computational results back to a Java program for further scoring computations. (Reference: HW3)
6. Once we've calculated the scores for each webpage, we'll sort the webpages based on scores in descending order and then return them to the frontend.
7. Frontend:
   1. Website: (Reference: HW11)
   2. (Android) App: We use Android Studio and google api to develop our (Android) App.
8. Schedule

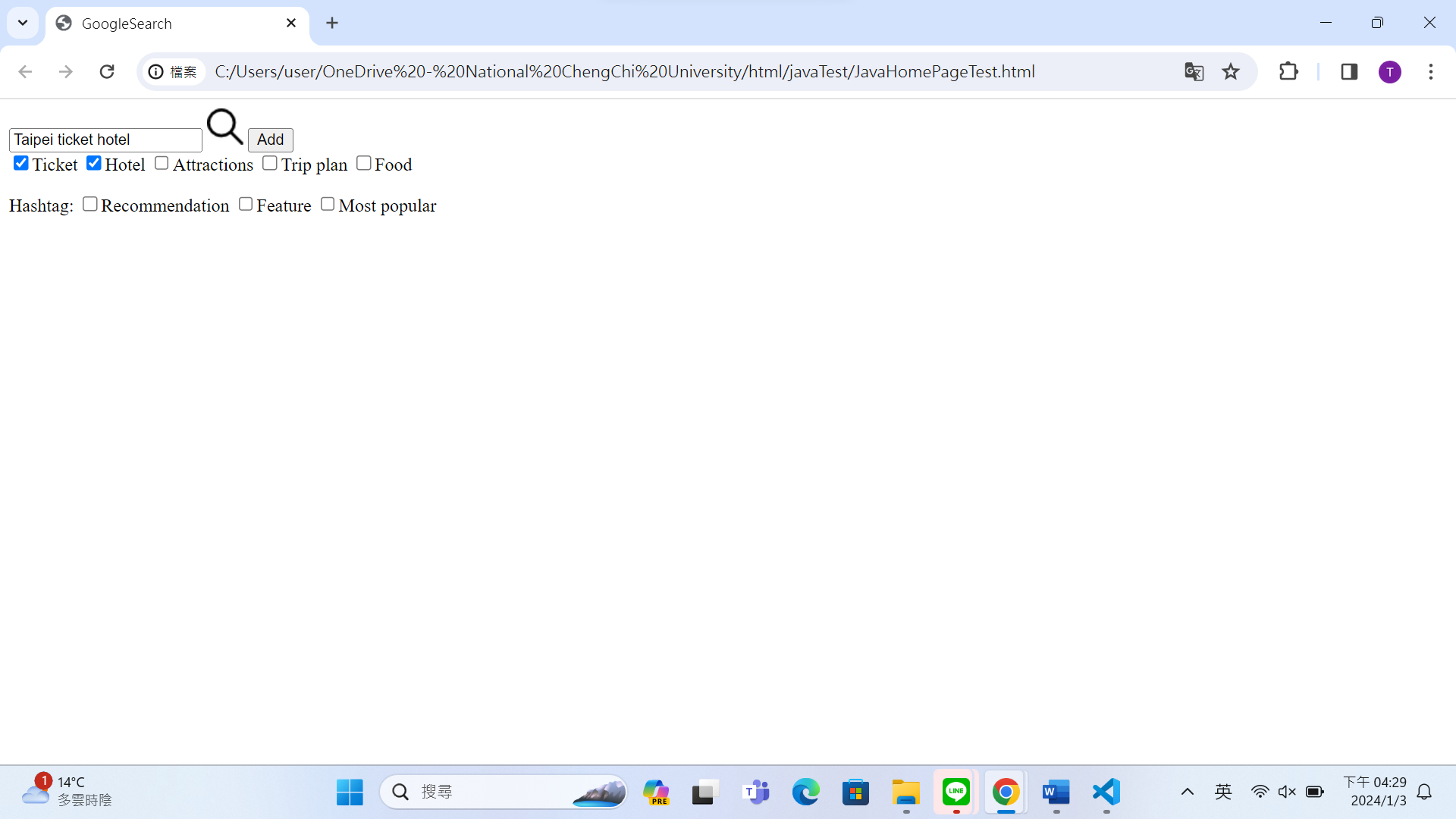


1. Critical path analysis



1. Challenges
   1. Machine learning for finding similar keywords.
   2. Matching the different language. (Including: Java, Python, Html and JavaScript)
   3. Deal with the problems between different
2. Frontend design:
   1. Website:

Based on HW11.



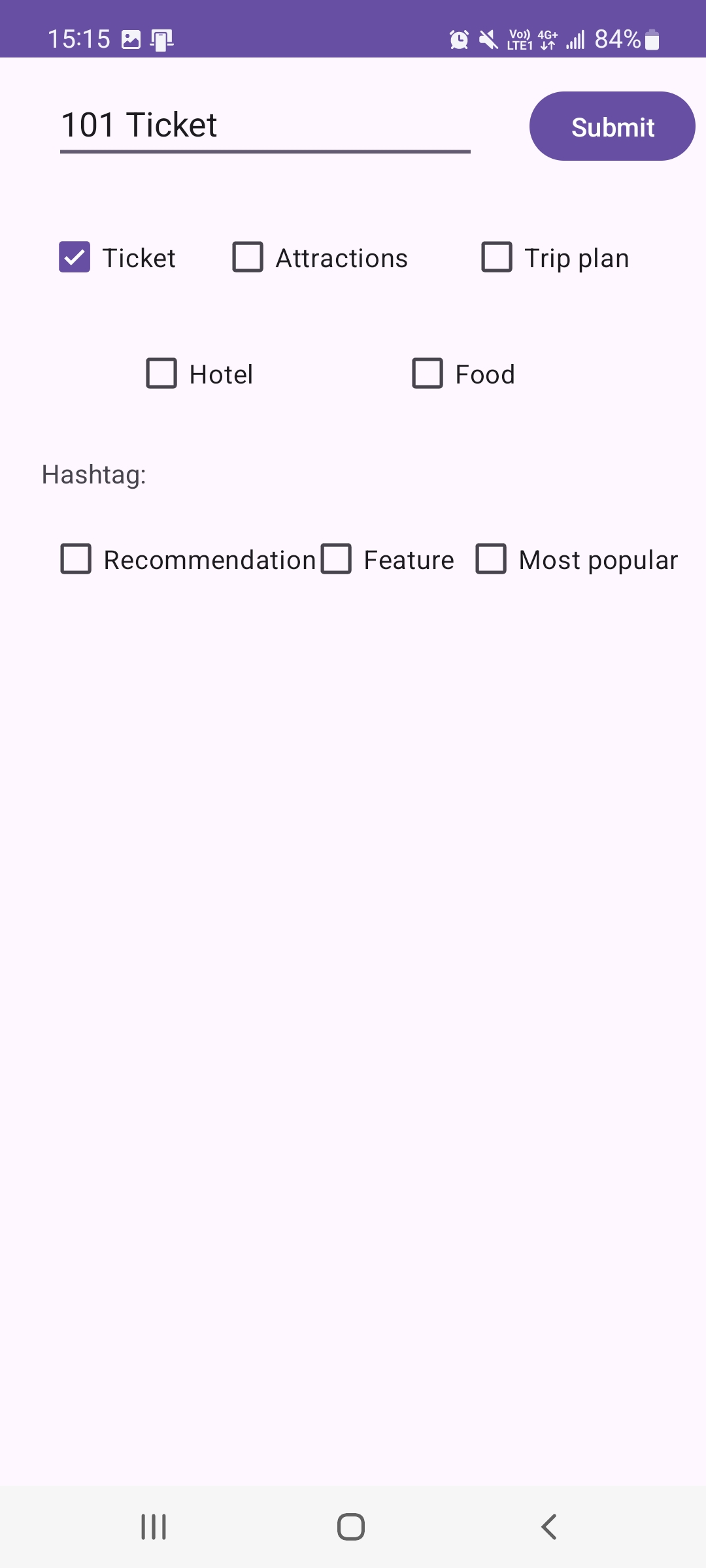
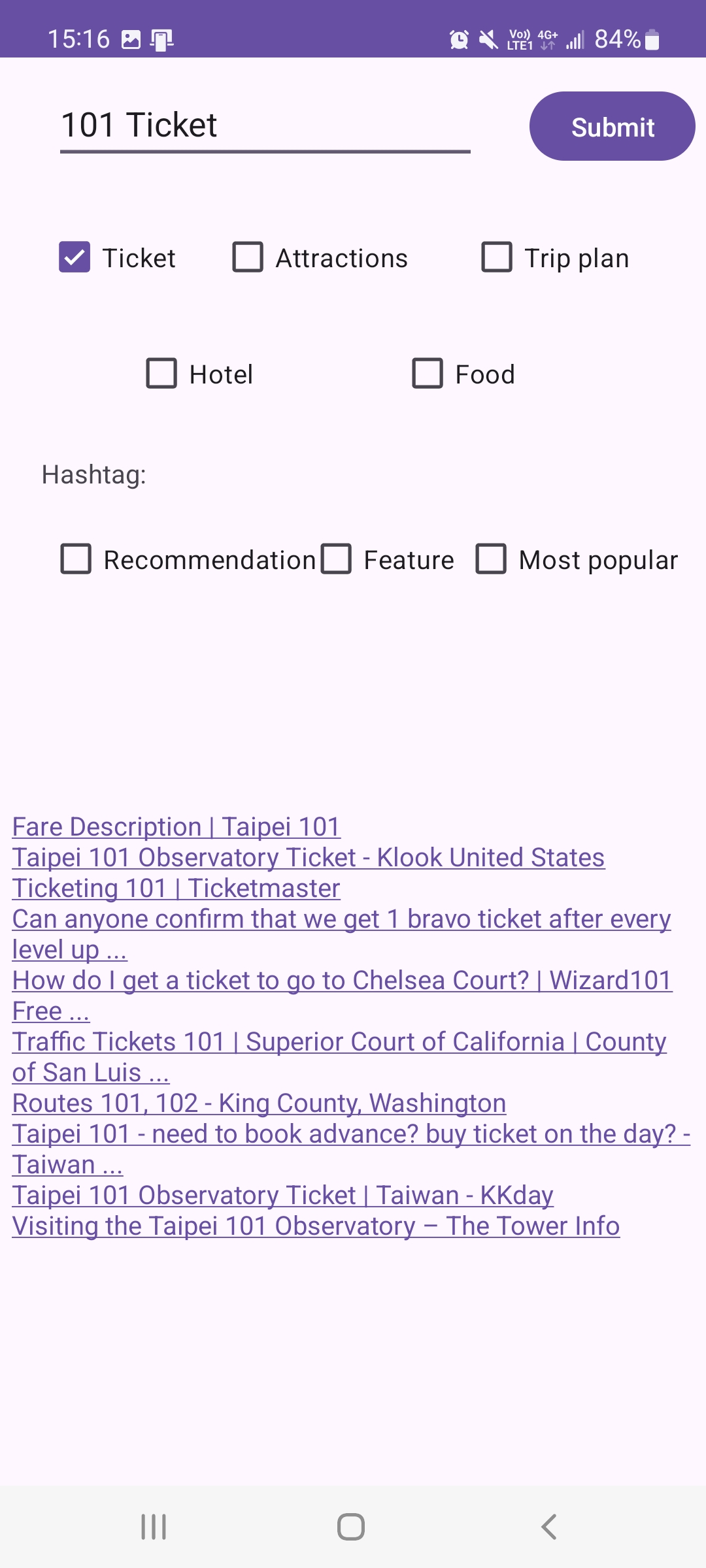
1. Input the place as keyword.
2. Check the checkbox needed.
3. Press “Add” button, it would add the chose keyword.
4. Press the loupe image. The page would show the top 10 results. And the search result shows without ad.
5. There are still some problems we will debug before deadline. (The input can only read the first word in eclipse.)

Note: Since we run the sever on eclipse, and the auto input code is JavaScript. So, the auto input is useless in eclipse.

Demo Video: <https://drive.google.com/drive/folders/1Rz0jtQnwOhPGumtG0DYikmySgAjiFh9z?usp=sharing>

* 1. (Android) APP:

We use Android Studio with java.

1. Input the place as keyword.
2. Check the checkbox needed, it would add the chose keyword.
3. Press the “Submit” button. The page would show the results. And the search result shows without ad.
4. Click link you want it.

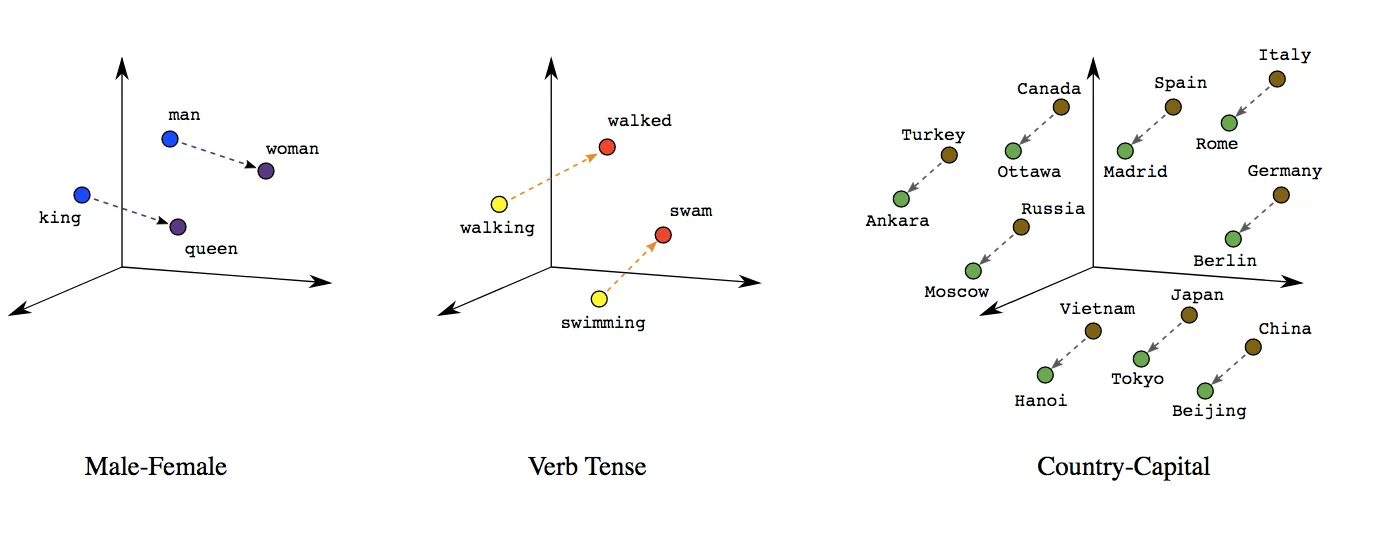
Demo Video: <https://drive.google.com/drive/folders/1Rz0jtQnwOhPGumtG0DYikmySgAjiFh9z?usp=sharing>

1. Search technologis:
   1. NLP :We also utilize word vector model which a way of representing words in a continuous vector space in natural language processing . In various NLP tasks, words are converted from discrete entities into continuous vectors through a certain mapping method. This transformation allows models to learn using techniques like Gradient Descent. By training on substantial amounts of text, word vectors are expected to capture relationships between words using these vector representations. If Word Vectors have been thoroughly learned by the model, we'd expect the vectors for (woman - man) and (queen - king) to be close. Similarly, word vectors representing two semantically similar words should also be close to each other. After get weight figure,we will transmit to our tree and do sorting process.

For English : we use software project called “gensim” which has been trained and testified to do NLP.

For Chinese : we use software project called ”jieba” to cope with word segmentation problem to make sure whether segmentation is precise.

* 1. Word Segmentation:



In addition, when working on English NLP tasks, we typically handle a word as the smallest unit. However, this differs for Chinese. For instance, in the sentence "今天天氣真好" (Today's weather is really nice), the smallest units should be "『今天』『天氣』『真』『好』." Therefore, we need to segment Chinese sentences into individual units before processing them.

Implementation: Word vector: We use the “Gensim” package to handle this, utilizing pre-trained models available online for processing. Word segmentation system: We use the “Jieba” package for this purpose.

* 1. Relative position of cities:

Our goal is to find travel-related keywords, which can be quite challenging. For instance, when searching for the keyword "Brazil" directly through the Google search engine's "related searches," it often provides unrelated keywords like "Brazil time" or "Brazil English." These are not directly related to travel. Hence, we aim to change this situation by considering not only the queried city but also its neighboring cities, recommending nearby cities.

However, parsing city names is not straightforward due to potential variations in their names. For example, "California" and "Cal" may not seem significantly different to many people, but in programming terms, they are entirely different words, just like "California" and "Taiwan." Therefore, we standardize city names to English for consistent evaluation and comparison.

Implementation: Searching for all countries, cities, and their respective latitude and longitude: Utilizing data available online (Data source: <https://simplemaps.com/data/world-cities>).

We will consider a city's latitude, longitude, and population count for recommendations. Translating from Chinese or other languages to English: Utilizing the "google translation" package, which connects to the Google Translate API for translation purposes.

* 1. Python communicates with Java:

The approach we're taking is similar to the front-end HTML and back-end Java setup. We facilitate Java to transmit data to Python over the network using the “flask” package.

* 1. Weight design:

Assuming the keyword represents a region, city, or town, we'll identify the top 10 counties closest to it, all within the same country. Then, using "gensim", we'll measure the word vector distance between these counties and the original keyword, deriving weights. Considering "cosine similarity" as the method for distance measurement in "Gensim", which ranges from [-1, 1], assigning excessively high weights might affect the original keyword computation. Hence, we'll multiply the obtained weights by 0.5 to adjust and use them as weights for the new keyword.

1. Achievement

Overall, for our final project presentation, we attempted to utilize the word vector software package to perform additional computations on Google search results, aiming for higher search accuracy compared to other groups. Moreover, apart from implementing a web-based version, we also ventured into creating a mobile app version on the frontend. This adaptation enables our final project to be compatible with multiple platforms and cater to a broader user base.

1. Future
   1. Diversified Search Functions : Expand the functionalities of the search engine to encompass a wider range of travel information, such as recommendations for attractions, culinary culture, transportation details, etc., to offer a more comprehensive travel planning service.
   2. Technological Innovation and Improvement : Continuously explore the latest technological advancements, such as machine learning, natural language processing, etc., to continually enhance the precision and efficiency of the search engine, ensuring a higher quality user experience.
   3. Integration of AI : Implement AI technologies like ChatGPT to meet users' demand for more customized experiences.
   4. Integrate more software packages in Different Languages : Persist in incorporating word vector software packages in various languages to expand the user base.
2. Work division
   1. 林美婷：Front-end design (Web page and Android APP)：100%、Project writing：40%、Back-end design：Java：95%
   2. 呂家華：Project writing：30%、Back-end design：Java：2%
   3. 徐華均：Project writing：30%、Back-end design：Java：3% and other all code