Geographical Location Analysis using MongoDB.

Jaineel Fenal Dharia, Krunal Chandrakant Parikh, Dhruv Sureshbhai Jasani and Lypsa Nirav Shah
School of Computer Science
University of Windsor
dhariaj@uwindsor.ca, parikh92@uwindsor.ca, jasanid@uwindsor.ca and shah6u@uwindsor.ca

***Abstract*— In this paper, we outline a method for determining the company's most precise location client needs. Many businesses today look for locations with specific amenities, such as restaurants, coffee shops, airports, and hotels close by the corporate location, to bridge the commute gap and save time for everyone employed by the business. Using the aforementioned statement as a problem statement, our product will make use of contemporary APIs and sophisticated database technologies to provide an effective solution, the workings of which will be covered in the study.**

I. INTRODUCTION AND MOTIVATION

A company's success can be strongly impacted by its location. Numerous unique factors must be taken into account when choosing an organization's optimal location, which can be overwhelming for many business owners. This is why it's so important to perform a thorough analysis of a potential business location. When done correctly, this process removes any uncertainty and enables you to make an informed decision.

Our software performs geographic location analysis using geospatial data and data from real-time APIs, assisting any organisation in finding suitable sites based on their needs. The requirements can be of any kind, such as being close to a school, coffee shop, airport, or three kilometres of profitable businesses. Users only need to enter the range and all the requirements they desire based on their company. The user requirements along with real-time API data will be taken into account when developing our solution, resulting in the most precise location range possible.

II. LITERATURE REVIEW

Individuals used paper maps and inquired local people for directions before Google and Apple maps became well-known since they were less at ease using digital maps at the time. Additionally, they are more accurate than they were before . However, since digital maps are more precise and have more attributes that make finding destinations easy, people today have more confidence in them. But as technology develops, additional functions are also in demand. Having discovered a very modest scenario that could be useful for developing corporate location as there aren't many projects focusing on location analysis for firm based locations with consideration of geospatial data as a factor[9].

Using a broad range of capabilities for viewing, analysis, and customization, a product with the name of maptive builds a map [1]. This aids the business in identifying the local market's demographics of customers. Some functionality, such as data filtering according to user demands and a store finder, are shared with our project. The difference between our product and maptive is that our geographic location is based on a specific use case that will assist the company in finding a location that meets all of its needs, including a coffee shop, an airport, parking, etc[10]. Additionally, we will grow our product as an open-source endeavour even though maptive is not free. The majority of the technologies, tools, and libraries used in this project will also be open-source. As a result, interested developers can adapt the project at no expense to suit their needs[8].

In order to find tools and libraries that would be safe, dependable, and versatile to employ in our project, we focused on open-source software. First, Leafmap is a Python module that makes it simple to conduct interactive mapping and geospatial analysis within the Jupyter environment. It is intended to fill the gap for GEE (Google Earth Engine) users who are not GEE users. Additionally, it is a free and open-source Python application that enables users to analyse and visualise geographic data with little to no scripting in a Jupyter environment, such as Google Colab, Jupyter Notebook, or JupyterLab [2].

For the analysis and processing of 2-dimensional geospatial data, such as satellite imagery and output from numerical models, there is also a Python toolkit called Nansat. Nansat improves the widely used Geospatial Abstraction Data Library (GDAL) by combining common capabilities for data analysis and administration and giving the datasets scientific relevance through metadata (e.g., exporting to various data formats) [3].

Now, the main challenge is to swiftly and efficiently extract results from the geographical dataset (spatial data). This happens as a result of the relational database's storage capacity, query performance, and volume of different sorts of data (RDBMS). Our understanding of the geospatial data and all the unique allocations made to this feature in our mongoDB database is aided by the study [4].

The selection of a database to hold the data for geoinformation applications plays a key role here. Therefore, using a NoSQL database is the modern method of storing data, which has several advantages when it comes to handling large amounts of data. Therefore, MongoDB has a lot of advantages. It will execute queries quicker and utilise fewer computer resources by using this NoSQL database [5]. In addition, GeoJSON searches run really well in MongoDB, which will let us create a 2D sphere index that is available in

the collection of the database used. This is the main reason we chose MongoDB as our database[6,7].

III. PROPOSED MODEL

The purpose of the entire model is to use the geospatial data to run GeoJson Queries on the MongoDB database that we have built and utilised for analysing the user input data. Following analysis, our solution will produce the optimal location for the user based on their demands in an interactive map format, providing the optimum user experience. The flow chart presented beforehand clarifies the complete work process.

The complete model is used to gather API data and user requirements, which will be analysed to produce the most precise geographical range. The working of model is described further in point wise manner :

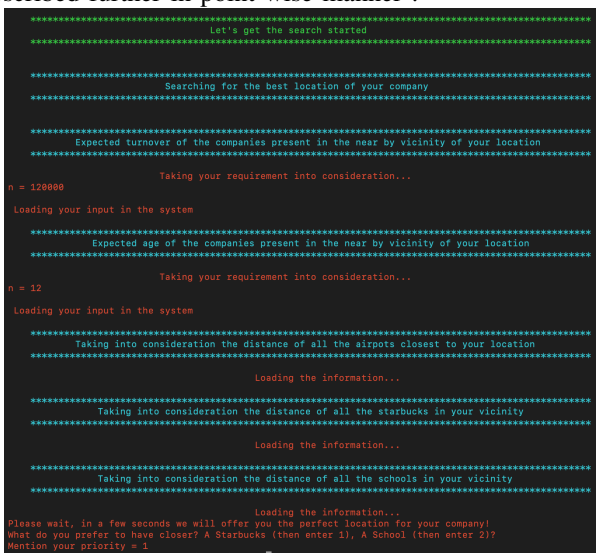

figure 1 : Requirements front end seen by user

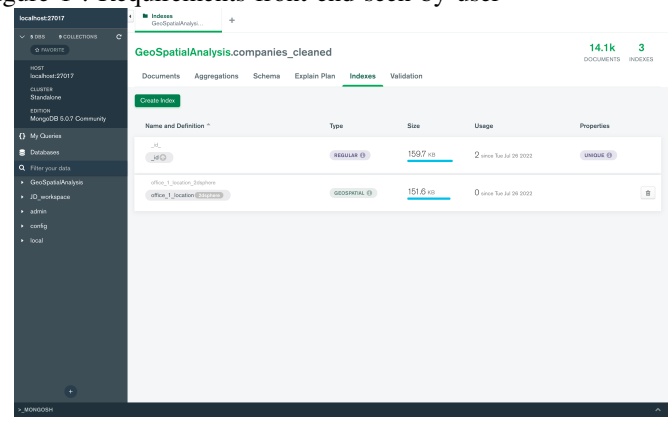

figure 2 : 2D sphere indexing in MongoDB

- Obtaining every user demand is regarded as a priority task in the first step of the process. When deciding on the location coordinates, a product's users typically have the most influence on how the business environment should be. The age and turnover of the companies that the user would want to be located around are both inputable in our model. Depending on the user, each of these user criteria will vary, and they will all be evaluated in light of this information for subsequent processes.
- It is necessary to comprehend the cleaning of the data set and make it compatible with the technological environment for effective outputs before moving forward with the user requirements and combining them with the cleaned data set. The raw data collected in a json file is transformed into a data frame and cleaned using python libraries, after which all the financial data is translated into common currency using our first API, the Exchangerate API, using real-time data obtained from the API used. Additionally, the cleaned data is kept inside a data base mongoDB that will be utilised for indexing later. The cleaned data received in the process's previously described step is then combined with the user needs we established.
- Since the cleaned data has been successfully merged with the user's requirements, there are some additional requirements that the user may be mindful of, such as having coffee shops nearby, airports that are not too far away, and having a school facility close by that will be useful for staff members who have children. All of these points appear to be very crucial, and our model doesn't ignore them either. We will use our Foursquare API to collect all of the relevant data for the aforementioned circumstances, and we will then create a 2D sphere index pointing in the direction of the office, which will aid in locating all of the locations nearby the specified location in real time using the API data [10].This will add up all user requirements and leverage the 2D sphere indexing capability of the mongoDB database for geospatial data to find proximity close to our location.
- The complete set of data will ultimately be reduced to a small number of lines that fulfil the user's requirements before being added to the dataframe. The data frame will be utilised to transfer the data into a csv file for debugging purposes and enter it into a map that we generated using the folium library, which will display the data in real time on a map for improved comprehension and visualisation.

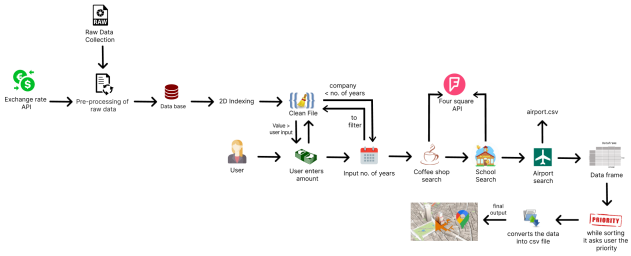

figure 3 : Workflow of the model

IV. RESULTS

The results of the model were stored in 3 different formats each of them having the same data but giving an user a chance for different interpretations of the output. The formats in which the results were stored are:

- The first format is the data frame format which could be used by the developers furthers for understanding any anomaly or will be useful to understand the addition of extra feature to the model.
- The second format id the csv format which could be used by both user and developers as source of secondary data for future work. Its main usage would be allowing a verification step to the graphical visualization of the output.
- The third and the last format which could be used by company executives for their in-depth analysis in a pictorial format consisting of the map with all the important locations accompanied by the main one represented in a pictorial manner.

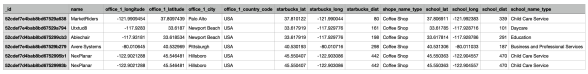

figure 4 : 'CSV' format of final data

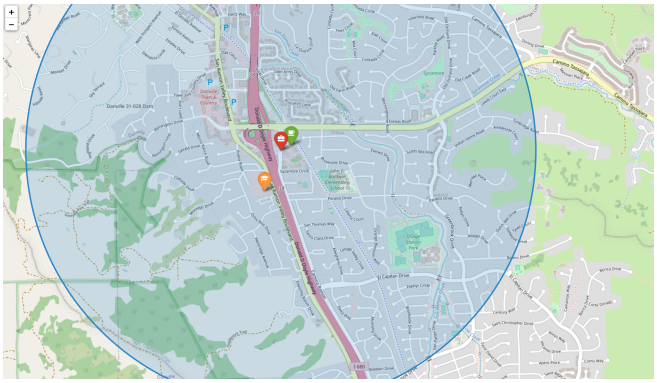

figure 5 : Map format of user case 1

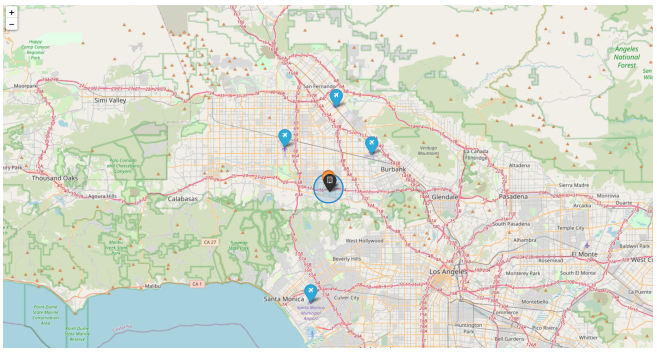

figure 6 : Map 1 format of user case 2

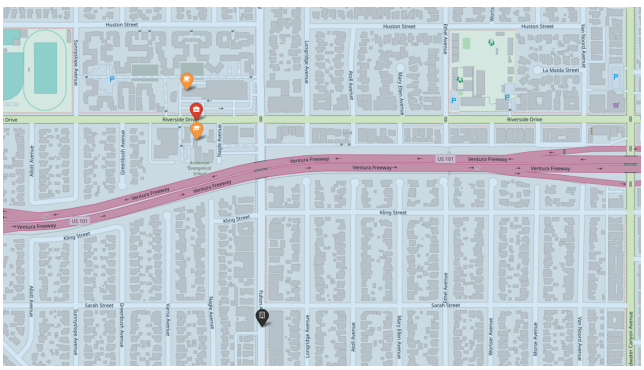

figure 7 : Map 2 format of user case 2

V. LIMITATIONS AND CHALLENGES

It was quite challenging to keep up with the demands of modern technology when building a model heavily dependent on user requirements and real-time data collected from the APIs. Here are a few restrictions we'd want to address:

- One of the issues we had was that we were only able to support a few number of user needs, and when those requirements grew in complexity, their association with the clean data set made it difficult to conduct analysis on.
- The second challenge we encountered was processing such a huge data set with our current technological capabilities. This was time-consuming and made it very challenging to identify a few faults made when developing the model.
- The most significant challenge we encountered was making the most of the free resources offered by the API data generating company. Because we were open source and cost-free, we missed out on many paid functionalities that would have been crucial to developing a more effective model and would have been able to offer flexibility to the user.

VI. CONCLUSION AND FUTURE WORK

A. SUMMARY

In order to put the model into perspective, we were able to produce the set of coordinates for the company's location based on all the user-provided information. The user was able to easily comprehend the functioning and outcome by selecting the best option from the available options and visualising it in a map format. The functionality of 2D sphere indexing offered by the mongoDB database and the real-time data gathered from both of the study's APIs were both significantly dependent on the model. All of the icons for a better understanding of the model were included in the final map output. We experimented with a wide range of user options and ran into issues with many of them, which ultimately demonstrated how solid our model was.

B. FUTURE WORK

It was a great opportunity for all of us to learn new things while working on such a beautiful database design, and using high-end APIs offered us experience handling enormous amounts of data. In the future, we'll work on the model to add more user-requirement options and employ 2D sphere indexing on multiple location attribute variables, which will make it more sophisticated and offer us the chance to work with numerous location attributes of the data along with a variety of parameters.

REFERENCES

[1] Custom Map Creator Map Maker: Maptive mapping software. Maptive. (2021, October 26). Retrieved June 25, 2022, from https://www.maptive.com/

[2] Wu, Q. (2021). Leafmap: A Python package for interactive mapping and geospatial analysis with minimal coding in a Jupyter environment. Journal of Open Source Software, 6(63), 3414.

[3] Korosov, A., Hansen, M., Dagestad, K. F., Yamakawa, A., Vines, A., Riechert, M. (2016). Nansat: A scientist-orientated python package for geospatial data processing. Journal of Open Research Software

[4] da Costa Rainho, F., Bernardino, J. (2018, June). Web GIS: A new system to store spatial data using GeoJSON in MongoDB. In 2018 13th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). IEEE

[5] Pietroń, M. (2019). Analysis of performance of selected geospatial analyses implemented on the basis of relational and NoSQL databases. Polish Cartographical Review, 51.

[6] De Smith, M. J., Goodchild, M. F., Longley, P. (2007). Geospatial analysis: a comprehensive guide to principles, techniques and software tools. Troubador publishing ltd.

[7] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote sensing of Environment, 202, 18-27.

[8] Maantay, J. A., McLafferty, S. (Eds.). (2011). Geospatial analysis of environmental health (Vol. 4). Springer Science Business Media.

[9] Jiang, B., Yao, X. (Eds.). (2010). Geospatial analysis and modelling of urban structure and dynamics (Vol. 99). Springer Science Business Media.

[10] Li, Y., Steiner, M., Wang, L., Zhang, Z. L., Bao, J. (2013, April). Exploring venue popularity in foursquare. In 2013 Proceedings IEEE INFOCOM (pp. 3357-3362). IEEE.