Knowledge Graph Construction and Recommender System Development of Tourism in Singapore

Submitted by

Xiong Ying

A final year project report
presented to
Nanyang Technological University
in partial fulfilment of the
requirements for the
Bachelor of Engineering (Computer Science)
Nanyang Technological University

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

Year 2023/2024

Abstract

This research and development project presents a comprehensive investigation into the development and implementation of a knowledge-graph-based recommender system tailored for urban tourism. The recommender system is powered by a recommender engine developed in this project, which utilizes a combination of data mining algorithms, such as heuristic methods, content-based filtering, collaborative filtering, and ensemble learning techniques to generate personalized recommendations for tourist points of interest (POI) in Singapore. A key focus of the study is the evaluation of individual data mining algorithms and ensemble learning strategies to provide insights into their performance across various metrics such as precision, recall, and coverage score. The research identifies the strengths and limitations of each approach, highlighting the importance of a user-centric design and the challenges posed by data and resource constraints. Future work is outlined, including advancements in ensemble learning, database scaling, and user feedback analysis. Overall, the project contributes to the field of knowledge graph and recommender systems by offering a practical framework for developing knowledge-graph-based recommender systems application in urban tourism.

Acknowledgement

I would like to express my sincere gratitude to Assistant Professor Long Cheng for his invaluable guidance and support throughout this project. His mentorship and encouragement have been instrumental in its successful completion.

Table of Contents

List of Figures

1. Introduction

1.1 Background and Motivation

In today's modern world, where information can sometimes be overwhelming, finding useful insights within large datasets can be a challenge. Traditional methods of managing data, like relational databases, may struggle to effectively handle the complex relationships between different pieces of information. However, emerging technologies, such as knowledge graphs, offer a promising solution by organizing data in a structured graph format.

A knowledge graph is a structured representation of data in a graph format, depicting a network of real-world entities. It comprises nodes and edges, where nodes represent entities and edges denote relationships between pairs of entities [1].

The concept of knowledge graphs was popularized by Google's introduction of its knowledge graph in 2012 [2]. Knowledge graphs are constructed from datasets, and they establish connections between entities. Unlike traditional relational databases that utilize foreign keys for managing relationships, knowledge graphs provide a streamlined approach, minimizing computation efforts and facilitating efficient querying [3].

This project is motivated by the increasing importance of knowledge graphs across various fields, especially in enhancing user experiences within tourism applications by providing accurate recommendations for potential points of interest (POI). The significance of this project lies in its potential to improve the experience of tourists searching for information about Singapore's tourist attractions.

1.2 Problem Statement

The project aims to curate data and construct a knowledge graph tailored to Singapore's tourism sector to enhance information retrieval and develop a recommendation system powered by data mining algorithms to offer personalized recommendations to users.

1.3 Objectives

The objective of this project is to build a knowledge graph dedicated to Singapore's tourism sector, emphasizing tourist attractions such as museums, galleries, amusement parks, shopping malls and historical sites. In addition to constructing the knowledge graph, this project also involves the development of a recommender engine. By leveraging data mining algorithms, the recommender engine aims to enhance the accuracy of recommendations provided to users, thereby enhancing the querying experience for tourists and potentially leading to novel business opportunities within the tourism industry [4]. In the end, a user-friendly web-based recommender system powered by the recommender engine will be created to serve as a user-friendly platform for users to explore Singapore's attractions, access relevant information, and view personalized recommendations.

1.4 Scope and Limitations

The scope of this project encompasses the development of a knowledge graph tailored to Singapore's tourism sector, focusing on organizing data related to tourist points of interest (POI). Additionally, the project includes the implementation of a recommender engine to improve the accuracy of recommendations provided to users based on their past interactions.

The integrated recommender system, coupled with a knowledge graph, strives to furnish tourists with an accessible platform to acquire information regarding POI. Its primary objective is to provide users with personalized recommendations swiftly and efficiently.

However, it's essential to acknowledge certain limitations within the scope of this project. Firstly, the knowledge graph will initially exclude data related to hotels and restaurants, focusing solely on tourist attractions. While this streamlines the project's focus, it may limit the comprehensiveness of the recommendations provided, particularly for tourists seeking accommodation and dining options. Additionally, the project's reliance on data from specific sources, such as TripAdvisor, may introduce gaps in the information available within the knowledge graph. Finally, the recommender engine's effectiveness may be impacted by factors such as data sparsity, the quality of user reviews, and the lack of user's personal information due to privacy protection, which could affect the accuracy of recommendations generated.

Despite these limitations, this project aims to leverage knowledge graphs and recommender system to provide recommendations to tourists visiting Singapore, to enhance user experiences in the tourism sector, with potential implications for both tourists and industry stakeholders.

1.5 Outline of the Report

To provide an overview of the report's organization, here is a summary of the content covered in each chapter:

1. **Introduction**: Introduces the project's background, problem statement, objectives, scopes and limitations.
2. **Literature Review**: Surveys existing research on knowledge graphs, recommender systems in tourism.
3. **Data Acquisition and Preprocessing**: Describes the data acquisition, cleaning process and exploratory data analysis (EDA).
4. **Knowledge Graph Construction and Querying**: Presents the creation and analysis of the knowledge graph.
5. **Methodology for Recommender Engine**: Explains the approach for building the recommender engine, the experimental setup and performance evaluation.
6. **System Implementation**: Discusses the implementation of the system, including requirements analysis, web architecture and system validation.
7. **Discussion and Future Work**: Analyzes results and proposes future directions.
8. **Project Management**: Outlines the planned project phases, timeline and schedule.
9. **Conclusion**: Summarizes key result.

1.6 Code Repository

Readers interested in exploring the code implementation can find the GitHub repository at https://github.com/xiong-ying/KG-Rec-Sys-Tourism-SG.

2 Literature Review

2.1 Overview of Knowledge Graphs and their Applications

Knowledge graphs are structured representations of data in a graph format, facilitating the depiction of relationships between entities [5]. Knowledge graphs have emerged as powerful tools across various domains, offering insights and enabling efficient querying of complex datasets. Knowledge graphs are increasingly utilized in various industries for their ability to depict relationships between entities and facilitate efficient querying [6].

For instance, Google's Knowledge Graph, introduced in 2012, revolutionized search engine capabilities by providing instant, relevant information to users based on a semantic understanding of entities and their relationships [7].

Similarly, LinkedIn leverages knowledge graphs to enhance its professional networking platform, enabling features such as personalized job recommendations and skill endorsements by mapping relationships between users, companies, and skills [8].

These real-world applications demonstrate the versatility and value of knowledge graphs in delivering insights and optimizing user experiences.

2.2 Previous Work on Knowledge Graphs for Tourism

Several research projects have explored the construction of knowledge graphs tailored to tourism, particularly in regions like Hainan Province, China, and London/Sardinia.

A case study involving a tourism knowledge graph in China's Hainan Province introduced a pipeline for constructing an event-centric travel knowledge graph (ETKG). The authors utilized structured event data from travel websites and supplemented it with unstructured data from travel notes. Named Entity Recognition (NER) models were employed for entity tagging and attribute extraction, ultimately leading to the creation of the ETKG [9].

Another Hainan Tourism case study demonstrated the construction of a knowledge graph using semi-structured and unstructured data from various travel websites. NER models were tested for labelling entities, and relation extraction models were evaluated based on context semantics. The resulting knowledge graph was successfully constructed, although real-time updates proved challenging due to the dynamic nature of tourism websites [10].

An additional research project presented a framework for a Tourism Knowledge Graph (TKG) related to London and Sardinia. This project utilized data from global travel websites to construct the TKG, leveraging an ontology developed by the team which is called the Tourism Analytics Ontology (TAO). This ontology-based approach facilitated the integration of data and emphasized accommodations rather than points of interest (POI) [11].

These projects collected data from travel websites, incorporating both structured and unstructured data to construct their knowledge graphs. While similar in approach, these projects focused on specific cities, leaving a gap for a comprehensive Tourism Knowledge Graph for Singapore.

2.3 Review of Recommender Systems in Tourism

Recommender systems play a crucial role in enhancing user experiences in tourism applications by providing personalized recommendations for tourist points of interest (POI) [12].

For instance, Expedia, a leading travel booking platform, employs recommendation algorithms to suggest tailored travel packages and activities based on user preferences and booking history [13].

Similarly, Booking.com utilizes machine learning techniques to provide personalized hotel recommendations, taking into account factors such as location, budget, and traveller reviews [14].

Techniques such as content-based filtering, collaborative filtering, and ensemble learning are commonly used in recommender systems to analyze user preferences and make accurate suggestions [15].

These industry examples underscore the important role of recommender systems in optimizing user satisfaction and facilitating personalized travel experiences.

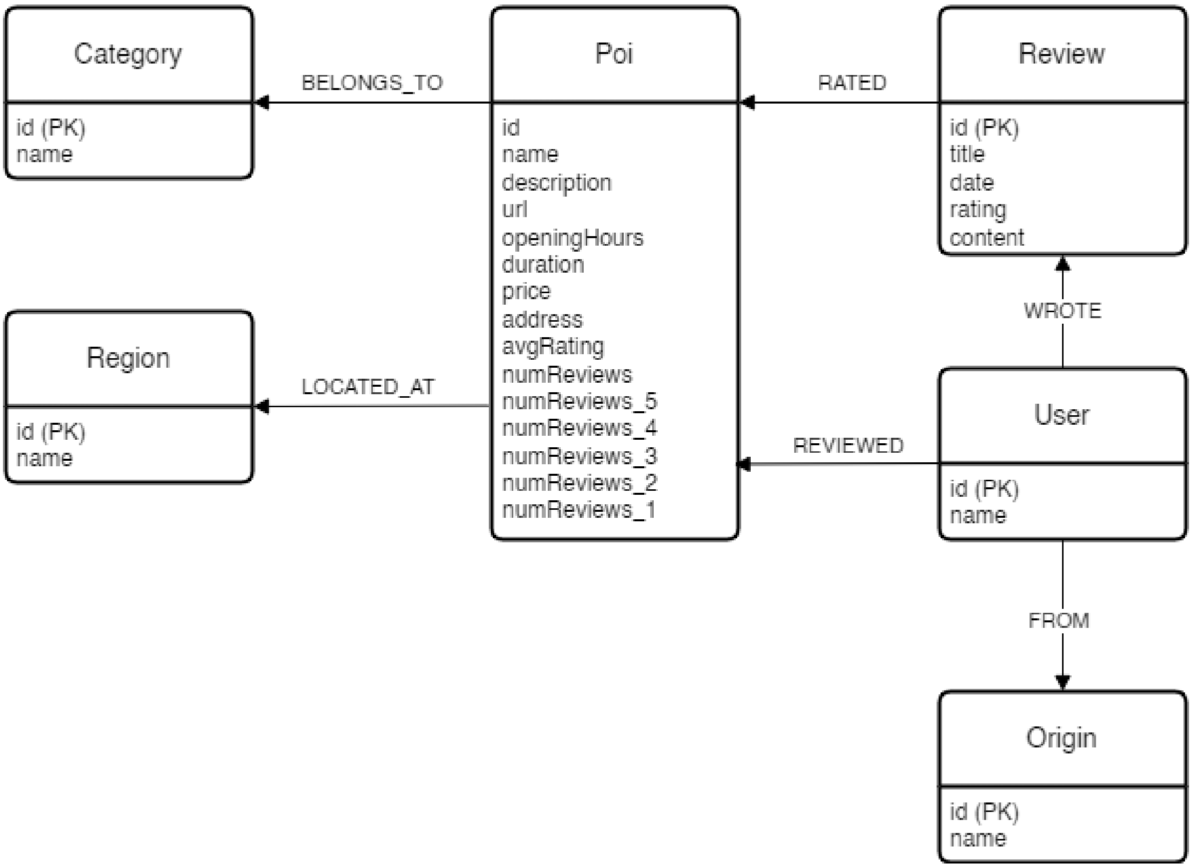
3 Data Acquisition and Preprocessing

3.1 Schema of Knowledge Graph

Before data acquisition, the schema of the graph database is first designed to accommodate different types of data related to tourism points of interest (POI), it captures the essential entities and relationships for a tourism-based recommender system in Singapore [16].

Figure 1 below is an Entity-Relationship Diagram (ERD) defining entities (nodes) and relationships (edges).

Figure 1: Entity-Relationship Diagram (ERD) of Knowledge Graph Schema



It includes nodes representing tourist attractions which are referred to as “Poi” in this schema, other nodes include “Category”, “Region”, “Review”, “User” and “Origin”, along with edges denoting relationships including "BELONGS_TO", “LOCATED_AT”, “RATED”, “WROTE”, "REVIEWED” and “FROM”.

Detail descriptions of the Entities (Nodes) are listed below:

1. **Poi**
   - **Description:** Comprises information about various points of interest (POI) in Singapore.
   - **Properties:** id (unique identifier), name, description, url, openingHours, duration, price, address, avgRating, numReviews, numReviews_5, numReviews_4, numReviews_3, numReviews_2, numReviews_1
2. **Category**
   - **Description**: Represents different categories or types of POIs.
   - **Properties**: id (unique identifier), name
3. **Region**
   - **Description**: Represents different regions or areas within Singapore.
   - **Properties**: id (unique identifier), name
4. **Review**
   - **Description**: User-generated reviews and ratings are collected to provide insights into the popularity and quality of POI.
   - **Properties**: id (unique identifier), title, date, rating, content
5. **User**
   - **Description**: Represents users who have interacted with POI.
   - **Properties**: id (unique identifier), name
6. **Origin**
   - **Description**: Represents the origin or nationality of users.
   - **Properties**: id (unique identifier), name

Detail descriptions of the Relationships (Edges) are listed below:

1. **BELONGS_TO:**
   - **Description**: Connects Poi and Category nodes, indicating the category to which a point of interest (POI) belongs.
   - **Example Relationship**: `(Gardens by the Bay)-[:BELONGS_TO]->(Nature Park)`
2. **LOCATED_AT:**
   - **Description**: Connects Poi and Region nodes, indicating the geographical location of a POI.
   - **Example Relationship**: `(Gardens by the Bay)-[:LOCATED_AT]->(Marina Bay)`
3. **RATED:**
   - **Description**: Connects Review and Poi nodes, indicating that a review has been given for a specific POI.

  - **Example Relationship**: `(Review)-[:RATED]->(Gardens by the Bay)`

4. **WROTE:**
   - **Description**: Connects User and Review nodes, indicating that a user has written a review.
   - **Example Relationship**: `(John Doe)-[:WROTE]->(Review)`

5. **REVIEWED:**
   - **Description**: Connects User and Poi nodes, indicating that a user has interacted with a POI by writing a review.
   - **Example Relationship**: `(John Doe)-[:REVIEWED]->(Gardens by the Bay)`

6. **FROM:**
   - **Description**: Connects User and Origin nodes, indicating the nationality or origin of a user.
   - **Example Relationship**: `(John Doe)-[:FROM]->(Singapore)`

3.2 Data Collection - Web Scraping

Efficient data collection techniques are fundamental for acquiring tourism-related data from online sources. Web scraping, a common method employed in data collection from websites, is employed in this project. Web scraping leverages tools such as Selenium and BeautifulSoup (bs4) to automate the extraction of data from web pages [17].

Selenium is a powerful automation tool that allows for the emulation of user interaction with web browsers, enabling the programmatic navigation of websites and the retrieval of desired HTML content [18].

BeautifulSoup facilitates the parsing of HTML documents, enabling the extraction of specific data elements by identifying the tab or class from the HTML content of web pages [19].

Employing techniques like sending requests with specific User-Agent headers enhances the effectiveness of web scraping efforts. By mimicking the behaviour of legitimate web browsers, these headers help bypass restrictions and prevent websites from blocking the scraping process, ensuring reliable and uninterrupted data retrieval [20].

The primary data source for this project is the travel websites which provide information about points of interest (POI) in Singapore. Web scraping techniques were employed to extract data from the travel website. The process involves utilizing libraries such as Selenium in Python to navigate through web pages, use BeautifulSoup to extract relevant information, and store it in a structured format for further processing [21].

These web scraping techniques enable the project to accumulate a comprehensive dataset for constructing the knowledge graph and implementing the recommender engine.

Here's a breakdown of the technical details involved in each step:

1. Extracting URLs for Each POI in Singapore

1) Navigating to Singapore POI Listing Pages:

Utilized Selenium to browse through the Singapore points of interest (POI) listing pages, each containing a brief overview of POIs. Each page displays 30 POIs, and all listing pages were iteratively flipped through by adjusting the URL to access comprehensive listings. This process allowed us to gather HTML content from all listing pages.

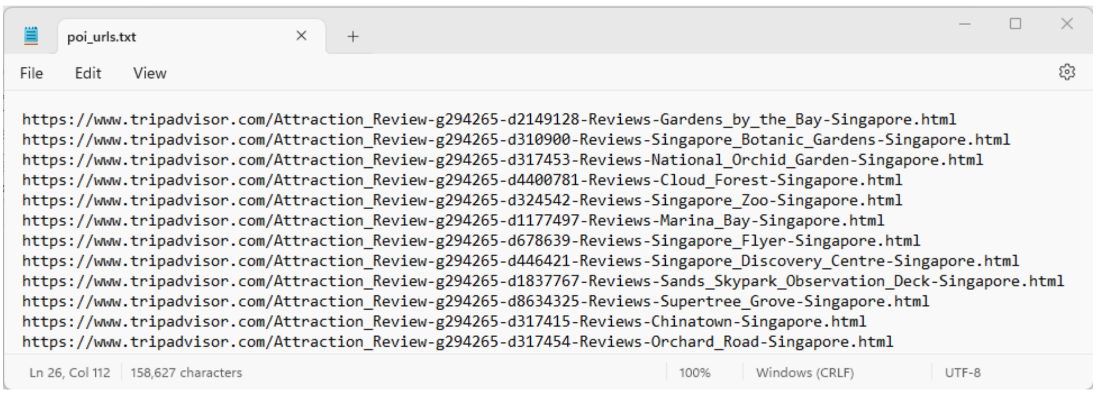
2) Extracting POI URLs:

Employed BeautifulSoup to parse the HTML content and identify the element containing the URL of each POI. By identifying the specific class label, the URLs for all POIs listed on the pages were extracted .

3) Saving Extracted POI URLs:

Saved all extracted POI URLs in a file named **`poi_urls.txt`** for future reference and use.

Figure 2: Screenshot of poi_urls.txt


poi_urls.txt
File Edit View
https://www.tripadvisor.com/Attraction_Review-g294265-d2149128-Reviews-Gardens_by_the_Bay-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d310900-Reviews-Singapore_Botanic_Gardens-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d317453-Reviews-National_Orchid_Garden-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d4400781-Reviews-Cloud_Forest-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d324542-Reviews-Singapore_Zoo-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d1177497-Reviews-Marina_Bay-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d678639-Reviews-Singapore_Flyer-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d446421-Reviews-Singapore_Discovery_Centre-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d1837767-Reviews-Sands_Skypark_Observation_Deck-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d8634325-Reviews-Supertree_Grove-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d317415-Reviews-Chinatown-Singapore.html
https://www.tripadvisor.com/Attraction_Review-g294265-d317454-Reviews-Orchard_Road-Singapore.html
Ln 26, Col 112 158,627 characters 100% Windows (CRLF) UTF-8

2. Extracting Attributes of Each POI

1) Reading POI URLs:

Read the POI URLs from the `poi_urls.txt` file, processing them line by line.

2) Accessing POI Pages:

For each POI URL, use Selenium to navigate to the respective POI page and fetch the HTML content.

3) Extracting POI Attributes:

Utilized BeautifulSoup to extract all attributes of POI nodes, including various details such as name, type, opening hours, description, duration, price, address, region, average rating, and review statistics. Additionally, the region and category were extracted, with some POIs belonging to multiple categories, which were saved as lists.

4) Saving Extracted POI Attributes:

After traversing all POI URLs, saved the extracted attributes in file `poi_info.csv`.

Figure 3: Screenshot of poi_info.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	id	url	name	type	openingHours	description	duration	price	address	region	avgRating	numReviews	numReviews_5	numReviews_4	numReviews_3	numReviews_2	numReviews_1
2	2149128	https://www.t	Gardens by the Bay	Points of Interest &	5:00 AM - 2:00 AM	An integral part of Singap	More than 3 hours	8.01	18 Marina Gard	Central Area/City Area	4.5	60393	43439	13817	2541	406	199
3	310900	https://www.t	Singapore Botanic Gardens	Parks, Gardens	5:00 AM - 12:00 AM	This national park is ope	1-2 hours		1 Cluny Road, S	Tanglin	4.5	20016	14192	4899	822	66	37
4	4400781	https://www.t	Cloud Forest	Points of Interest &	9:00 AM - 9:00 PM		2-3 hours	11.42	18 Marina Gard	Central Area/City Area	4.5	15161	11177	3078	715	138	53
5	324542	https://www.t	Singapore Zoo	Zoos	8:30 AM - 6:00 PM	Set in a rainforest enviror	More than 3 hours	33.49	80 Mandai Lake	Central Water Catchment	4.5	22544	14405	6127	1503	319	190
6	678639	https://www.t	Singapore Flyer	Points of Interest &	10:00 AM - 10:00 PM	At 165 metres tall, Singap	1-2 hours		30 Raffles Avenu	Marina Centre	4.5	17410	9284	5753	1960	277	137
7	1837767	https://www.t	Sands Skypark Observation Deck	Points of Interest &	9:30 AM - 11:00 PM	Located on the roof of Ma	1-2 hours	24.41	10 Bayfront Aver	Marina Bay	4.5	17000	9572	4860	1853	463	253
8	8634325	https://www.t	Supertree Grove	Points of Interest & Landmarks, Observatories & Planetariums			1-2 hours		18 Marina Gard	Central Area/City Area	4.5	6112	4435	1370	261	36	10
9	317415	https://www.t	Chinatown	Neighborhoods	8:00 AM - 12:00 AM	For a fascinating peek int	1-2 hours		Crot Trengganu	Outram	4	14539	5591	6234	2286	306	122
10	644919	https://www.t	Merlion Park	Monuments & Statues		Standing at 8.6 metres hi	1-2 hours		1 Fullerton Road	Marina Bay	4	9643	3697	4039	1658	185	64

3. Extracting User-Generated Reviews of Each POI

1) Reading POI URLs:

Read the POI URLs from the `poi_urls.txt` file, processing them line by line.

2) Accessing Review Pages:

For each POI URL, adjust the URL to navigate to the user review section. As each review page displays 10 reviews, navigate through all pages of reviews by adjusting the URL accordingly. Selenium is used to browse through the pages and fetch the HTML content.

3) Extracting Review Attributes:

Utilized BeautifulSoup to extract attributes of each review, including the ID, title, content, rating and date. Additionally, extracted basic user information, and URL of the review page.

4) Saving Extracted Reviews:

After traversing all review pages, save the extracted review attributes in a `reviews.csv` file.

Figure 4: Screenshot of review.csv

	A	B	C	D	E	F	G	H	I	J
1	poiID	username	location	review_id	title	date	rating	user_group	content	review_url
2	2149128	kra63	Sydney, Australia	735976516	Great potential but experience marred by o	January 1, 2020	5		We visited during the evening to catch th	https://www
3	2149128	KLPLeeds	Leeds	752547885	Wow wow wow - MUST DO!!!	April 12, 2020	5		This is free and the most amazing thing	https://www
4	2149128	mhsiong	Wellington, New Zealand	742358229	Breathtaking views of the city and great lan	February 1, 2020	5		I am not a plant person and I love Gard	https://www
5	2149128	Philthetrav	Avoca Beach, Australia	753097566	Amazing gardens and landscaping.	May 2, 2020	5		This fantastic area is located by the har	https://www
6	2149128	G8nzgirl	Auckland Central, New Zealand	737918097	GARDEN OASIS IN THE CITY	January 8, 2020	5		There is no cost to walking around the (	https://www
7	2149128	Exeter2010	Exeter, UK	742836580	Wonderful views from above	February 3, 2020	5	Couples	This is a truly wonderful area. We visite	https://www
8	2149128	RebeccaPi	London, UK	749991652	Gardens Rhapsody highlight of my trip	March 9, 2020	5	Couples	We visited the gardens both in the day a	https://www
9	2149128	Ozzy-Kunn	Melbourne, Australia	737255634	A Befitting Beauty Spot on Singapore.	January 6, 2020	5	Couples	Brilliant Brilliant Brilliant ! This man-m	https://www
10	2149128	garfield19	Kuala Lumpur, Malaysia	739577396	MRT station to Garden By the Bay	January 17, 2020	5	Family	Alight from Station Bayfront MRT and w	https://www

3.3 Data Preprocessing

Data cleaning and preprocessing are crucial steps to ensure the quality and consistency of the collected data. This involves tasks such as removing duplicate entries, handling missing values and standardizing data formats [22].

Challenges encountered during data collection and preprocessing, such as ensuring data quality, handling dynamic web content, and managing large volumes of data, were addressed through iterative refinement of scraping scripts.

Additionally, strategies such as introducing random time intervals between requests and implementing error-handling mechanisms were employed to mitigate issues related to website access and data retrieval [23].

The scraped data represents a representative portion of the data of the available data from the travel website due to time and resource constraints. Sampling this subset took approximately 21 hours of run time. As far as recommendation systems are concerned, this is a relatively modestly sized graph, with around 69 points of interest (POI), 58,656 users, and 90,454 total reviews [24].

While all the data are now extracted into `poi_info.csv` and `reviews.csv` files, further processing is required to split necessary parts into nodes and relationships for loading into the

Neo4j graph database. Dataframes containing only essential information was created for each node and relationship, preparing them for loading into Neo4j, and saved each dataframe into separate .csv files.

The .csv files are subsequently uploaded to GitHub for future access by the Neo4j graph database during the data loading process.

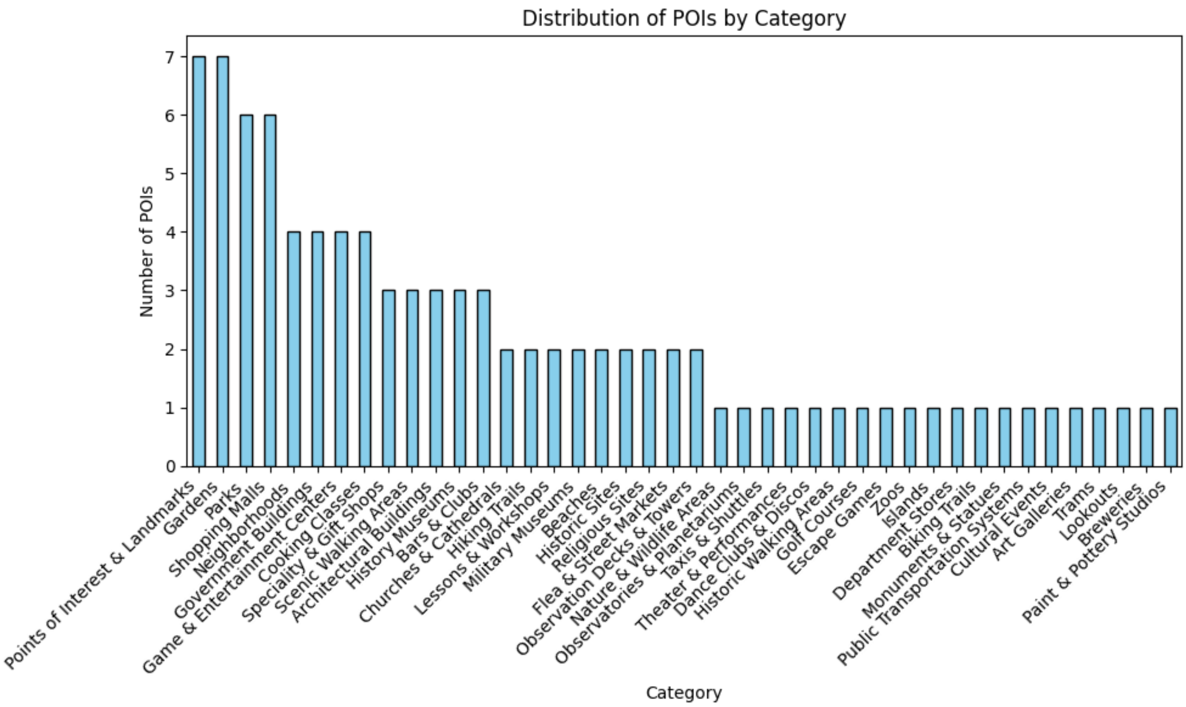
3.4 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is performed on the extracted data to gain insights into its characteristics and distributions.

3.4.1 Exploration of POIs

1. Distribution of POIs by Category

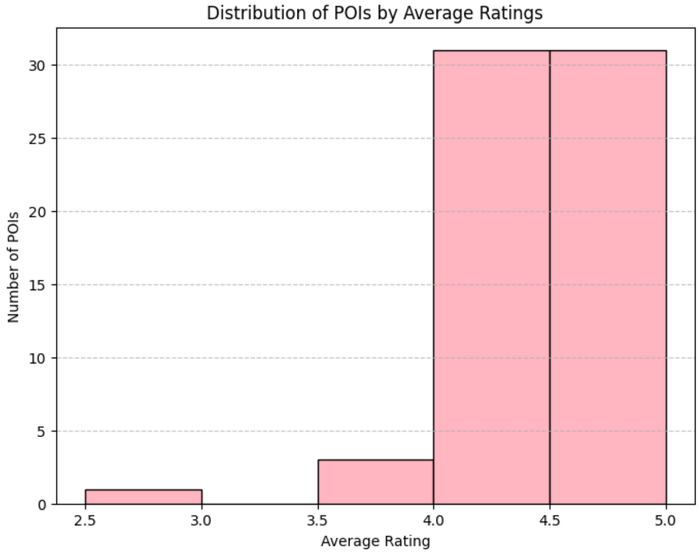
Figure 5: Distribution of POIs by Category



The distribution of Points of Interest (POIs) across different categories was analyzed. Among the 42 categories identified, "Points of Interest & Landmarks" emerged as the category with the highest number of POIs, followed by "Gardens." This distribution shows the diversity of POI available, with certain categories being a little more dominate than others.

2. Distribution of POIs by Average Ratings

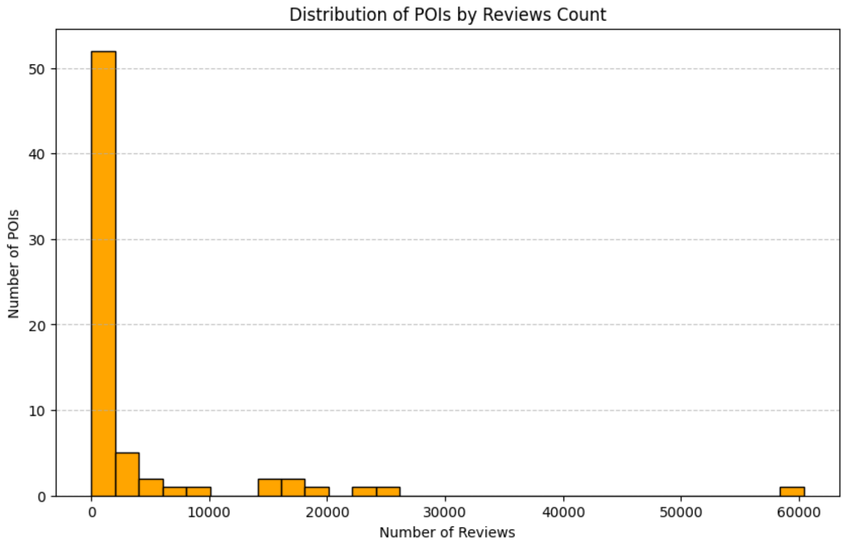
Figure 6: Distribution of POIs by Average Ratings



The average ratings of POIs were examined, showing that most POIs received ratings of 4.0 and 4.5. This suggests a generally high level of satisfaction among visitors, with only a minority of POIs receiving lower ratings.

3. Distribution of POIs by Reviews Count

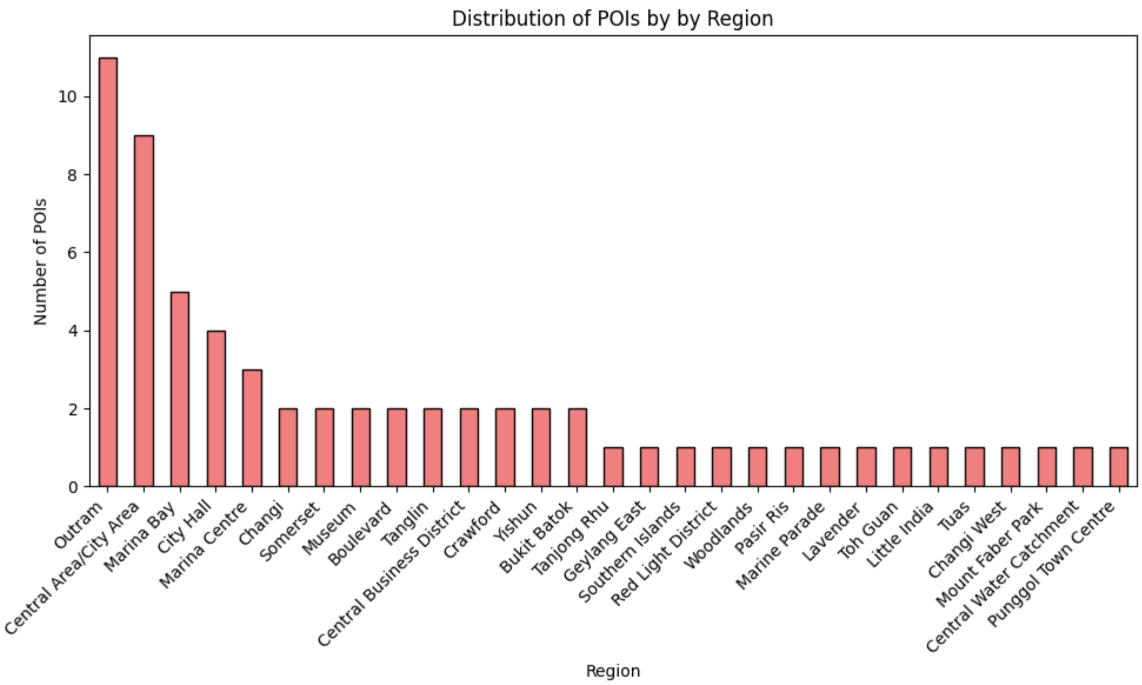
Figure 7: Distribution of POIs by Reviews Count



The popularity of POIs varied significantly, with the majority receiving fewer than 2,000 reviews. While some POIs attracted a large number of visitors and gathered extensive reviews, others remained relatively undiscovered. This disparity underscores the uneven distribution of tourist interest across different POIs.

4. Distribution of POIs by Region

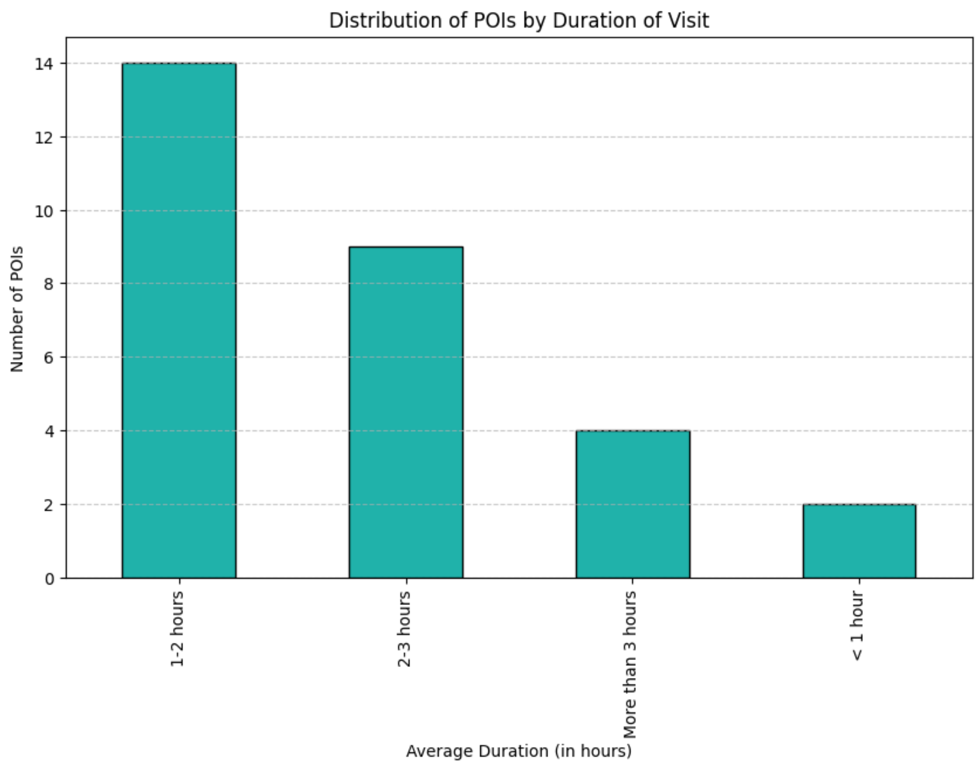
Figure 8: Distribution of POIs by Region



The distribution of POIs by region highlighted "Outram" as the region with the highest number of POIs, followed by "Central Area/City Area." However, several regions were found to have only a single POI, indicating disparities in tourism infrastructure.

5. Distribution of POIs by Duration of Visit

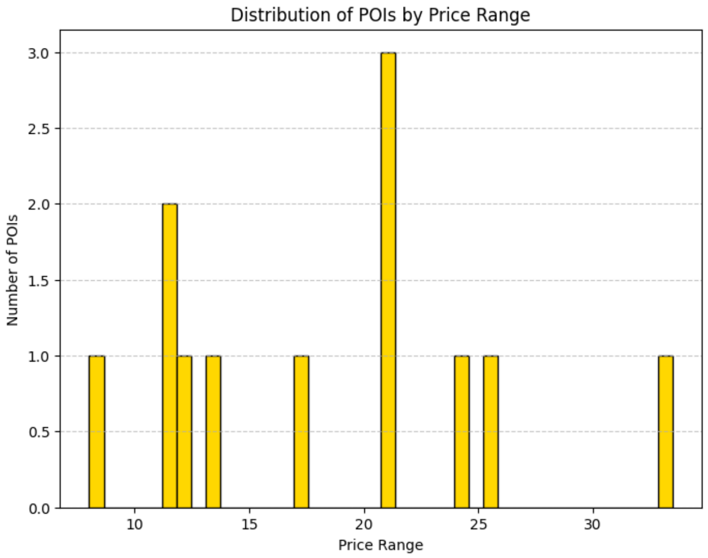
Figure 9: Distribution of POIs by Duration of Visit



The average duration of visits to POIs was examined, revealing that most POIs required 1-2 hours for exploration. This suggests that visitors typically allocate a few hours to explore individual POIs, with only a small number of POIs requiring less than an hour for a visit.

6. Distribution of POIs by Price Range

Figure 10: Distribution of POIs by Price Range

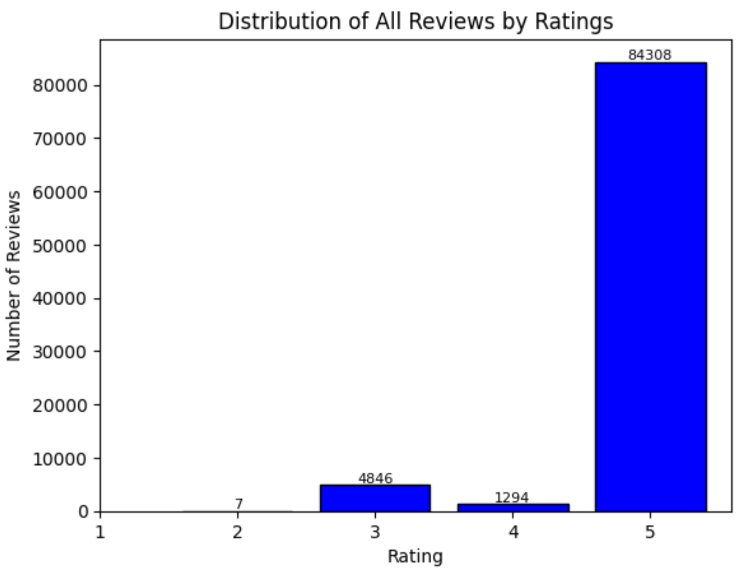


The price range of POIs was examined, revealing a diverse distribution ranging from S$0 to S$35. This indicates that POI caters to a range of budgets, with offerings available across different price points.

3.4.2 Exploration of Reviews

1. Distribution of All Reviews by Ratings

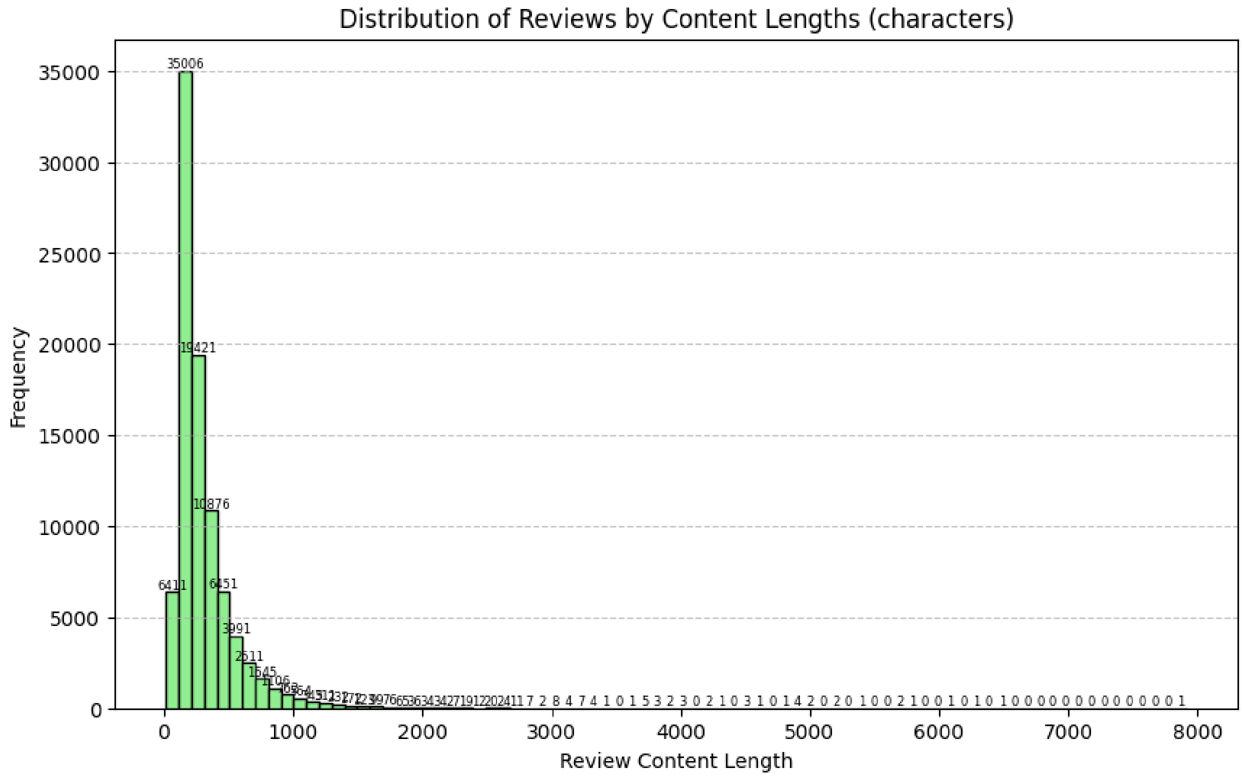
Figure 11: Distribution of All Reviews by Ratings



A highly skewed distribution of ratings was observed in the review data. The overwhelming majority of reviews are rated with 5 stars, indicating an excellent experience. Conversely, there are only 7 reviews rated with 2 stars, indicating a poor experience. This trend suggests a general bias towards positive experiences among reviewers, potentially influenced by factors such as user expectations and the tendency to share positive experiences more frequently.

2. Distribution of Reviews by Content Lengths (characters)

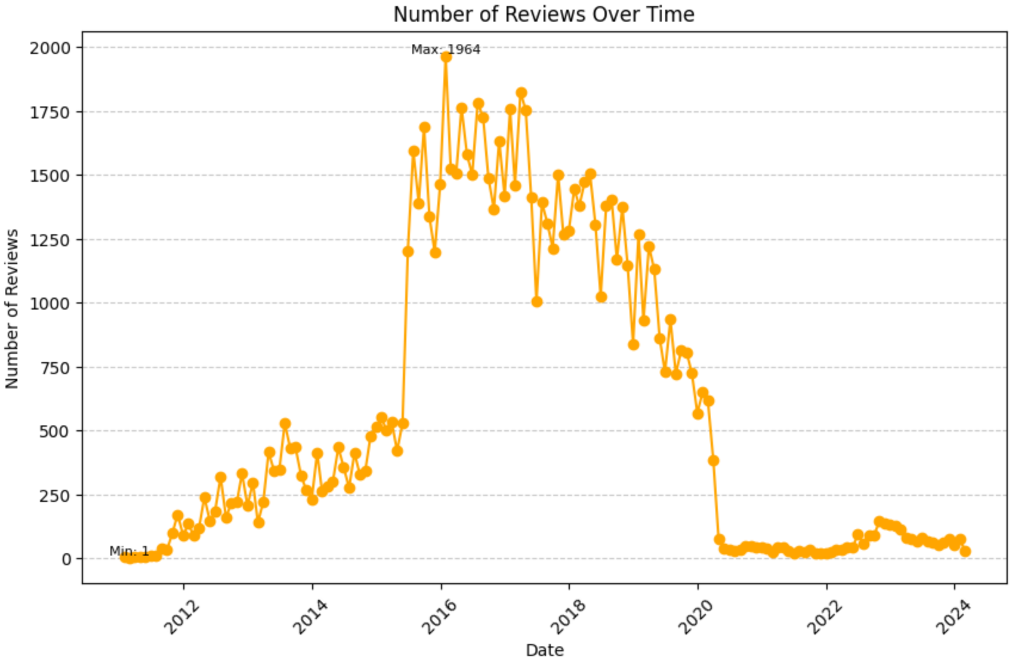
<u>Figure 12: Distribution of Reviews by Content Lengths (characters)</u>



An analysis of review lengths indicated that the majority of reviews were concise, with lengths typically ranging from 100 to 200 characters. While some longer reviews were observed, they remained relatively rare. This suggests that users tend to provide brief content for review.

3. Number of Reviews Over Time

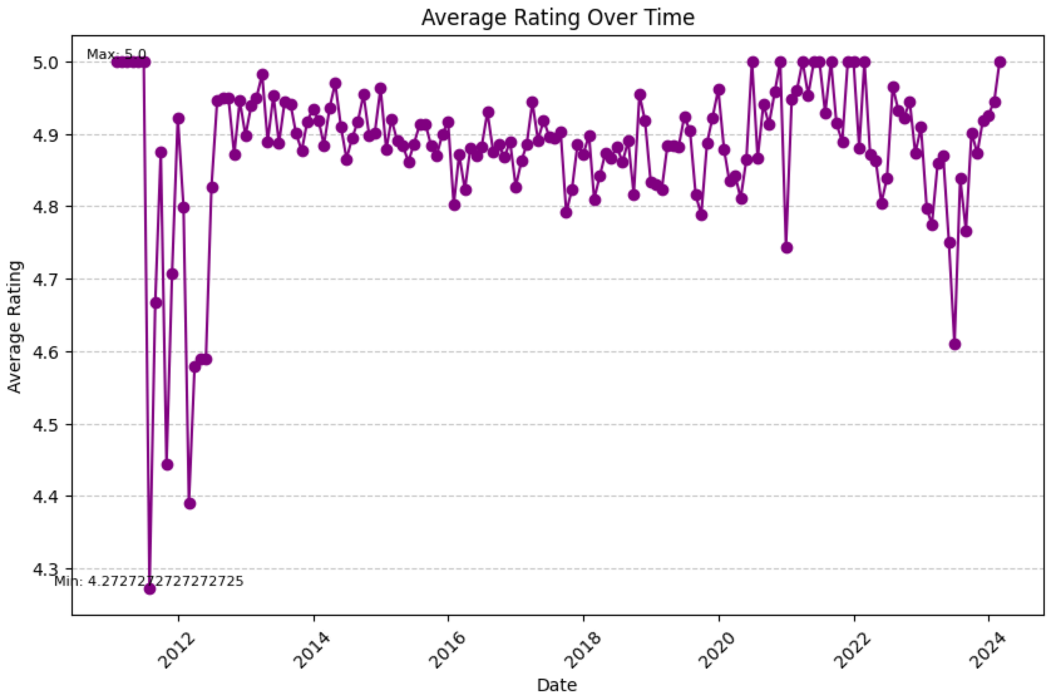
<u>Figure 13: Number of Reviews Over Time</u>



The temporal trend of review counts exhibited a significant increase from late 2015 to early 2020, followed by a notable decline thereafter. This pattern can be attributed to the impact of the COVID-19 pandemic on travel and tourism, resulting in reduced user activity and engagement with travel websites.

4. Average Rating Over Time

<u>Figure 14: Average Rating Over Time</u>

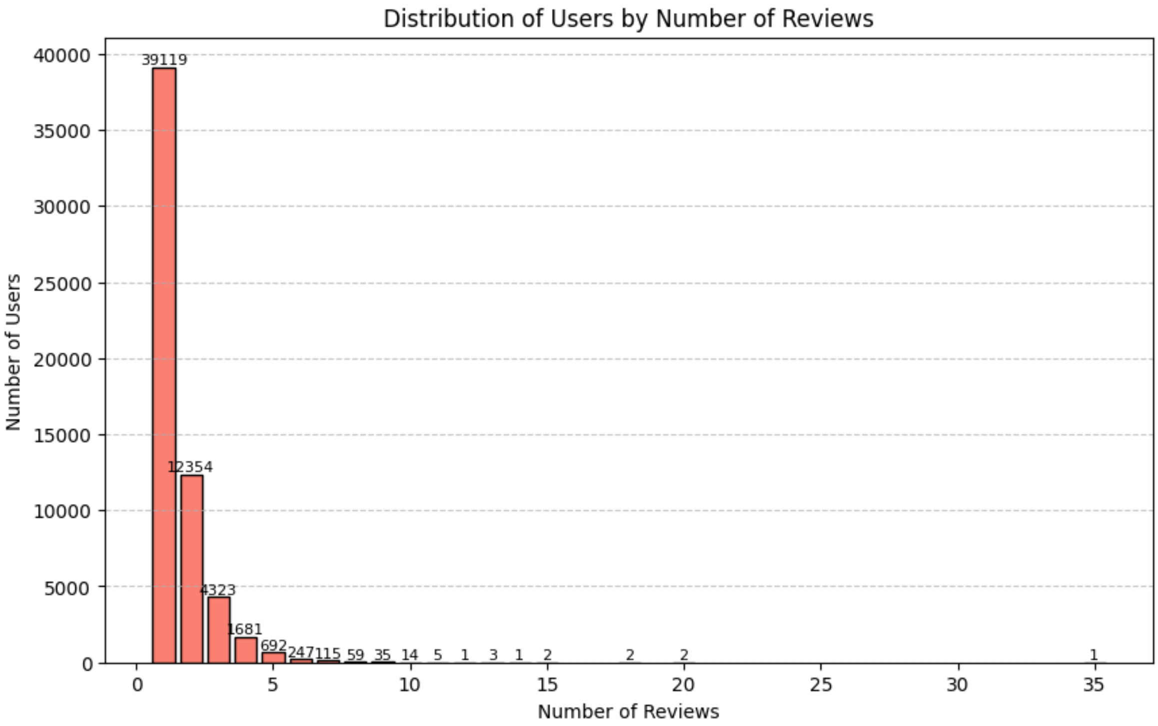


The average rating trend over time revealed consistently high ratings, with values ranging from 4.27 to 5.0. This indicates a sustained level of satisfaction among visitors, reflecting positively on the quality and appeal of points of interest (POI).

3.4.3 Exploration of Users

1. Distribution of Users by Number of Reviews

Figure 15: Distribution of Users by Number of Reviews



An analysis of user engagement patterns and activity levels within the review dataset was conducted by examining the frequency of users who had written a specific number of reviews. The majority of users were found to have written only a single review, indicating sparse user interaction data and presenting challenges for recommendation systems. Content-based recommendation strategies may be necessary to address the cold start problem posed by the majority of users [25].

4 Knowledge Graph Construction and Querying

4.1 Knowledge Graph Technologies

Neo4j, a leading graph database management system, is a popular choice for building knowledge graphs due to its user-friendly graphical interface and Cypher query language [26]. Cypher query language enables users to interact with the graph database efficiently, allowing for seamless querying and manipulation of data [27].

Neo4j's Python driver facilitates integration with Python-based applications, offering flexibility in developing custom solutions tailored to specific project requirements [28]. Additionally, the Neo4j Graph Data Science library is an advanced analytical tool for exploring and deriving insights from the graph data [29].

These technologies collectively contribute to the project's objective of creating an accessible and efficient platform for managing and querying tourism-related data, enabling the construction of a comprehensive knowledge graph tailored to Singapore's tourism sector.

4.2 Loading Data into Neo4j Graph Database

After completing the data cleaning and preprocessing steps, the next phase involves loading the prepared data into the Neo4j graph database. This operation is conducted using Cypher queries executed with the Neo4j Python driver. The process includes mapping the structured data from tables onto the graph schema, where nodes and relationships are created. Special consideration is given to ensuring data integrity during the loading process, which involves avoiding the creation of duplicate entities or relationships by performing existence checks and optimizing query performance for large volumes of data by loading data in batches.

The detailed process involves:

1. Setting up a Neo4j graph database instance, either in a sandbox environment or a local machine.

2. Creating a Python driver to interact with the Neo4j graph database by utilizing the Neo4j GraphDatabase library.

3. Defining constraints for the graph database to enforce the presence of a unique identifier property ('id') for each node, ensuring data integrity.

4. Importing the preprocessed .csv files generated in the previous data preprocessing step. These files are hosted on GitHub, and access is facilitated by specifying the file URLs.

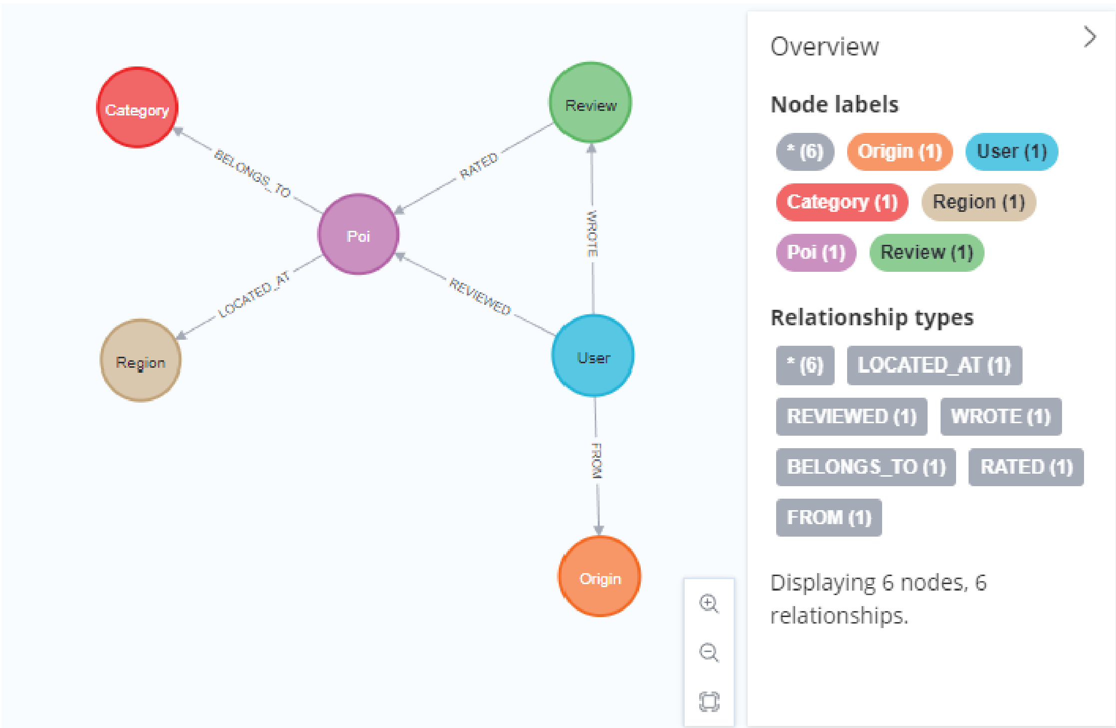
5. Processing each .csv file to import nodes and relationships into the Neo4j knowledge graph using Cypher import queries executed through the Neo4j Python driver.

4.3 Overview of the Constructed Knowledge Graph

The constructed knowledge graph serves as a comprehensive representation of Singapore's tourism landscape, it is structured to reflect the hierarchical and interconnected nature of tourism data, enabling seamless navigation and exploration.

For a comprehensive understanding of the graph's structure and components, please refer to section 3.1, which provides detailed insights into the schema of the Knowledge Graph. Figure 16 below illustrates the schema of the constructed Neo4j knowledge graph.

Figure 16: Schema of Constructed Neo4j Knowledge Graph



Basic information about the Knowledge Graph:

1. **Number of Nodes**

- Number of "Poi" Nodes: 69
- Number of "Category" Nodes: 42
- Number of "Region" Nodes: 29
- Number of "User" Nodes: 58656
- Number of "Origin" Nodes: 9403
- Number of "Review" Nodes: 90454

2. **Number of Relationships**

- Number of "BELONGS_TO" Relationships: 95
- Number of "LOCATED_AT" Relationships: 65
- Number of "RATED" Relationships: 90454
- Number of "WROTE" Relationships: 90454
- Number of "REVIEWED" Relationships: 90454
- Number of "FROM" Relationships: 51278

4.4 Graph Visualization and Analysis

Neo4j provides visualization tools to assist users in understanding the structure and dynamics of the knowledge graph [30]. The graph database is loaded into neo4j sandbox, which can be opened by a browser and is used to render the graph in a visually interpretable format, allowing exploring relationships and patterns intuitively [31]. Tools such as Neo4j Bloom provide interactive visualizations, enabling users to traverse the graph, zoom in on specific nodes, and uncover hidden connections [32].

An analysis of the knowledge graph was conducted, which entailed exploring different queries to extract insights. These queries covered a range of aspects, such as obtaining relevant information about points of interest (POI), understanding user preferences, exploring category-based offerings, gaining insights based on regions, analyzing price ranges, examining duration of visits, investigating ratings, assessing user engagement, studying review distribution, and identifying top-rated POIs.

The Cypher queries and visualization results for the above analysis are appended in **Appendix A: Knowledge Graph Queries**.

These queries provide insights into visitor preferences, POI popularity, and overall tourism dynamics, facilitating data-driven decision-making and enhancing the overall tourism experience in Singapore, and how graph database facilitates efficient querying to extract valuable insights.

5 Methodology for Recommender Engine

In this section, the methodology employed to develop an effective recommender engine for the tourism knowledge graph will be presented, leveraging various data mining algorithms including content-based filtering and collaborative filtering, as well as ensemble learning techniques, including evaluating the performance of the recommender engine.

5.1 Overview of Recommender Algorithms

In the realm of recommendation engines, various data mining algorithms are used to analyze user preferences and historical data to generate personalized recommendations. Three primary techniques commonly employed in recommender systems include content-based filtering, collaborative filtering, and ensemble learning [33].

1. Content-Based Filtering

This approach recommends items solely based on their features and characteristics, aiming to match user preferences with item attributes [33]. For instance, in the context of tourism, content-based filtering might suggest similar points of interest (POI) based on the POI that the user is currently viewing. This method relies on the extraction of relevant features from only items, such as average ratings, descriptions, or categories. Recommendations are then generated by identifying items with similar attributes.

In content-based filtering, various similarity measures are employed to quantify the resemblance between items based on different categories of attributes.

For numerical attributes, similarity is often computed using metrics like Euclidean distance [34]. Euclidean distance measures the straight-line distance between two points in a multi-dimensional space, providing a measure of similarity based on the difference between numerical values [35].

Formula to Calculate Euclidean Distance:

$$d(p_1, p_2) = \sqrt{\sum_{i \in \text{item}} (s_{p_1} - s_{p_2})^2}$$

Formula to Calculate Similarity based on Euclidean Distance:

$$\frac{1}{1 + d(p_1, p_2)}$$

For categorical attributes, Jaccard similarity is commonly utilized. Jaccard similarity measures the similarity between two sets by comparing their intersection over union, effectively quantifying the proportion of common categorical values between items [36]. This metric is particularly useful for categorical attributes where items are represented as sets of categories.

Formula to Calculate Jaccard Similarity:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

In the case of textual attributes such as item descriptions, textual similarity measures like cosine similarity or Jaccard similarity applied to term frequency-inverse document frequency (TF-IDF) vectors are often employed [37]. Cosine similarity measures the cosine of the angle between two TF-IDF vectors, capturing the similarity in terms of the distribution of important terms across documents. Jaccard similarity, when applied to TF-IDF vectors, assesses the similarity between texts based on the presence or absence of terms [38].

Formula to Calculate Cosine Similarity:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

By leveraging these similarity measures tailored to different categories of attributes, content-based filtering systems can effectively match user preferences with item attributes, enabling the generation of accurate recommendations.

2. Collaborative Filtering

Unlike content-based filtering, collaborative filtering recommends items based on the behaviours of users. By analyzing user-item interaction data, collaborative filtering identifies patterns and similarities among users' preferences, allowing for the prediction of items that a user might like

[39]. This technique can be further divided into user-based collaborative filtering and item-based collaborative filtering, each focusing on different aspects of user-item interactions [40].

In collaborative filtering, the K-Nearest Neighbours (KNN) algorithm is a fundamental component used to identify similarities between users or items based on their interaction patterns [41]. In the context of user-based collaborative filtering, KNN identifies users with similar preferences by computing the distances or similarities between their interaction. Similarly, in item-based collaborative filtering, KNN identifies items that are similar to each other based on users' interactions with them [42]. By leveraging the KNN algorithm, collaborative filtering systems can efficiently identify nearest neighbours or similar items, thereby enhancing the accuracy of recommendation predictions.

Fast Random Projection (FastRP) embeddings play a pivotal role in collaborative filtering by transforming the high-dimensional user-item interaction graph into a lower-dimensional space while preserving significant relationships [43]. Employing FastRP enables the efficient generation of embeddings for users and items, simplifying the computation of similarities between them. By utilizing FastRP-generated embeddings, collaborative filtering algorithms can identify users with similar preferences or items that are often interacted with by the same users, facilitating the provision of personalized recommendations tailored to individual user preferences [44].

3. Ensemble Learning

Ensemble learning combines multiple recommendation algorithms to improve recommendation accuracy and robustness [45]. By leveraging the strengths of individual algorithms and mitigating their weaknesses, ensemble methods aim to achieve better performance compared to standalone techniques [45]. Ensemble learning can be implemented through various strategies, such as majority voting, model stacking, bagging, or boosting, each offering unique advantages in terms of recommendation quality and diversity [46].

5.2 Overview of Experimental Setup and Evaluation Metric

1. Experimental Setup

The experimental setup involves partitioning the dataset into training and testing sets to evaluate the performance of the recommender engine [47]. Sparse data are not included in the test set, for example, if the user only reviewed less than 5 POIs, it will be challenging to use this kind of sparse data to validate the efficiency of the recommender engine by seeing whether the prediction appears in the true interaction. Among these potential data to be used for validating the recommender system, train-test split was adopted, with 90% of the data used for training and 10% for testing about 500 instances in the test dataset.

2. Evaluation Metrics

To assess the effectiveness of recommendation engines, several evaluation metrics are commonly used, including precision, recall, and coverage [48], to gain valuable insights into the performance and efficacy.

- **Precision**: Precision measures the proportion of recommended POIs that are relevant to the user's preferences [49]. A high precision indicates that a large proportion of the recommended items are indeed of interest to the user, reflecting the system's ability to provide accurate suggestions.

$$Precision\ Score = \frac{Relevant\ Retrieved\ Items}{All\ Retrieved\ Items}$$

- **Recall**: Recall quantifies the proportion of relevant items that are successfully recommended to the user [49]. It evaluates the system's ability to retrieve all relevant items from the dataset, ensuring that less relevant items are overlooked in the recommendation process.

$$Recall\ Score = \frac{Relevant\ Retrieved\ Items}{All\ Relevant\ Items}$$

- **Coverage**: Coverage measures the percentage of all items in the dataset that are successfully recommended to users [50]. It assesses the system's ability to provide diverse and comprehensive recommendations, ensuring that users are exposed to a wide range of items.

$$Coverage\ Score = \frac{Relevant\ Retrieved\ Items}{All\ Items}$$

- **F1 Score**: The F1 score combines precision and recall into a single metric using their harmonic mean, thereby offering a balanced representation of both aspects. An ideal F-score reaches 1.0, denoting flawless precision and recall, while the minimum value is zero [51].

$$F1\ Score = \frac{2\ \times Precision\ \times Recall}{Precision + Recall}$$

The precision, recall, and coverage scores were used as evaluation metrics to measure the effectiveness of the recommender system. Precision measures the proportion of recommended items that are relevant to the user's preferences, while recall measures the proportion of relevant items that are successfully recommended. Coverage assesses the breadth of the recommendation space covered by the system. F1 Score evaluates the balance of precision and recall measures.

5.3 Content-Based Filtering (CBF) Recommendation Algorithm

Content-based filtering (CBF) is a recommendation approach that suggests items to users based on the attributes and features of those items [52]. It focuses on analyzing the characteristics of items to match them with user preferences. CBF encompasses various methods, including heuristic approaches and node similarity methods, each tailored to leverage specific item attributes for generating recommendations.

5.3.1 Algorithm 1: Heuristic Algorithm

The heuristic method serves as a simple yet effective approach to recommending points of interest (POI) to users. It involves ranking POIs based on predefined criteria such as category, region and popularity. While straightforward and intuitive, this method provides a baseline for comparison with more advanced algorithms.

This section provides a detailed technical overview of the heuristic method's implementation:

1. Input Parameters:

- `poi_id`: Identifier of the POI that is currently being viewed by user as a reference.

2. Algorithm Workflow:

1) Retrieve POIs in the Same Region:

The algorithm first identifies other POIs located in the same region as the input POI. This step aims to recommend POIs near to the reference POI’s location or user’s area of interest.

2) Retrieve POIs in the Same Category:

Next, the algorithm retrieves additional POIs belonging to the same category as the reference POI. This step focuses on recommending POIs that align with the user's interests based on the category of the reference POI.

3) Merge and Aggregate Recommendations:

The algorithm aggregates recommendations obtained from both the region-based and category-based queries above. It computes the total weight of each recommended POI by summing the occurrences of POIs across both criteria.

4) Rank and Filter Recommendations:

Recommendations are ranked based on their total weight, prioritizing POIs with higher occurrences. Duplicate recommendations are removed to ensure a distinct set of suggestions.

5) Output Recommendations:

The algorithm generates a list of recommended POIs, consisting of the input POI ID, and recommended POI IDs. These recommendations serve as personalized suggestions tailored to the user's recent interactions.

3. Output Format:

The output is a dataframe containing input POI IDs and recommended POI IDs.

4. Evaluation Result:

- **Precision Score:** 0.167
- **Recall Score:** 0.244
- **Coverage Score:** 0.652
- **F1 Score:** 0.198

The evaluation result demonstrates lower precision, recall and F1 scores. However, its coverage score indicates that it covers a substantial portion of the item space, making it suitable for recommending a diverse range of items. Despite its lower precision, recall and F1 score, the algorithm's ability to cover a large portion of the item space enhances its utility in providing varied recommendations.

While the Heuristic Algorithm (Algorithm 1) may not be optimal as a standalone recommendation algorithm due to its lower performance scores, it serves as a benchmark for comparison. While simple, it offers insights into the performance of more complex algorithms.

5.3.2 Algorithm 2: Node Similarity Algorithm

The similarity between POIs was computed based on attributes such as category, location, description, price, etc. POIs with high similarity scores are recommended to users who have shown interest in reference POI.

The features of POIs can be categorized into three main types—numerical, categorical, and textual—computing the similarity between different types of features is not straightforward [53].

Preprocessing Techniques

Different preprocessing techniques are applied to various types of features to prepare the data for similarity computation:

- **Numerical: Min-Max Normalization**

  Numerical features, such as price and average rating, undergo min-max normalization [54]. This technique scales the numerical values to a fixed range (between 0 and 1), preserving the relationships between data points while ensuring that they are on a comparable scale [55]. Min-max normalization is suitable for numerical features as it prevents attributes with larger scales from dominating the similarity calculation.

- **Categorical: One-Hot Encoding**

  For categorical features like category and region, one-hot encoding is employed. This technique converts categorical variables into binary vectors, where each category becomes a separate binary attribute [56]. One-hot encoding ensures that categorical variables are represented in a format suitable for computation while maintaining their distinct categories [57]. It allows the algorithm to capture the relationships between different categories without imposing any ordinality or hierarchy among them [58].

- **Textual: Token Count Vectorization**

  Textual features, such as descriptions, are processed using token count vectorization [59]. This method converts text documents into numerical vectors by counting the frequency of each word (token) in the document [60]. Token count vectorization captures the frequency of words by representing it as a vector in a high-dimensional space [61]. It allows the algorithm to compute similarity between text based on the frequency and distribution of words.

Similarity Metrics

Different similarity metrics are applied to each type of feature to compute the similarity:

- **Numerical: Euclidean Similarity**

  For numerical features, Euclidean similarity is used, which measures the geometric distance between data points in a multidimensional space [62]. Euclidean similarity is appropriate for numerical attributes as it captures both the magnitude and direction of differences between values [35].

- **Categorical: Jaccard Similarity**

  Categorical features employ Jaccard similarity, which calculates the intersection over the union of binary vectors representing categories [63]. Jaccard similarity is well-suited for categorical attributes as it evaluates the similarity based on the presence or absence of categories, disregarding the magnitude or order of values [64].

- **Textual: Cosine Similarity**

  Textual features utilize cosine similarity, which measures the cosine of the angle between two vectors representing text [65]. Cosine similarity is ideal for textual attributes as it assesses similarity based on the orientation of vectors in the vector space, effectively capturing the semantic similarity between documents regardless of their length or magnitude [66].

1. Input Parameters:

- **`poi_id`**: Identifier of the reference POI for which recommendations are generated.

2. Algorithm Workflow:

1) Extract Raw Data of POIs and Attributes:

Raw data of POIs and their attributes, including numerical, categorical, and textual features, are extracted from the neo4j graph database.

2) Data Preprocessing:

- **Numerical Features:** Min-Max normalization is applied to numerical attributes such as price, average rating, and number of reviews to scale them between 0 and 1.
- **Categorical Features:** One-hot encoding is performed on categorical attributes like category, and region to convert them into binary vectors.

- **Textual Features:** Token count is computed for textual attributes like description using the CountVectorizer Python library to extract and quantify textual information [67].

3) Compute Pair-wise Similarity:

Pair-wise similarity between POIs is calculated based on their numerical, categorical, and textual attributes. Jaccard similarity is used for categorical attributes of POIs, Euclidean distance similarity for numerical attributes of POIs, and cosine similarity for textual attributes of POIs. Weighted overall similarity is computed considering the contributions of each attribute type.

4) Write Similarity Relationships to Graph Database:

Similarity relationships "CBF_SIMILAR" between POIs with property "score" are created in the Neo4j graph database based on the computed similarity scores.

5) Output Recommendations:

The algorithm generates a list of recommended POIs based on their similarity to the input POI. These recommendations are ranked by their similarity scores in descending order, ensuring that the most similar POIs are prioritized for recommendation.

3. Output Format:

The output of the node similarity method is a dataframe containing the input POI ID (`poi_id`) and recommended POI IDs (`rec_poi_id`).

4. Evaluation Result:

- **Precision Score:** 0.347
- **Recall Score:** 0.287
- **Coverage Score:** 0.884
- **F1 Score:** 0.314

Node Similarity algorithm (Algorithm 2) exhibits improved precision, recall and F1 scores compared to Heuristic Algorithm (Algorithm 1). With a higher precision score, this algorithm provides recommendations that are more relevant to users' preferences. Algorithm 2 has a slightly higher recall score compared to Algorithm 1, indicating its ability to retrieve a substantial portion of relevant items.

Additionally, Algorithm 2 has the highest coverage score among all algorithms, indicating its effectiveness in recommending items across a broad spectrum. Given its highest coverage score, it is well-suited for introducing random new items from the entire item space to users.

5.4 Collaborative Filtering (CF) Recommendation Algorithm

Collaborative filtering analyzes user-item interactions to recommend items to users with similar preferences, encompassing two primary approaches: User K-nearest Neighbours (UserKNN) and Item K-nearest Neighbours (ItemKNN) [68].

K-nearest Neighbours (KNN) is a classic machine learning algorithm used for classification and regression tasks [69]. For user-based collaborative filtering, UserKNN identifies users with similar preferences by considering their interactions with common items [70]. Similarly, ItemKNN identifies items that are often interacted with by the same users, implying similarity in user preferences [71]. UserKNN and ItemKNN leverage FastRP-generated embeddings to compute similarities.

Fast Random Projection (FastRP) is a dimensionality reduction technique employed to transform the high-dimensional user-item interaction graph into a lower-dimensional space while preserving important relationships [72]. This technique efficiently generates embeddings for users and items, facilitating the computation of similarities between them, based on their proximity in the embedding space [73].

By combining FastRP for dimensionality reduction and KNN for similarity computation, the collaborative filtering recommendation algorithm can effectively analyze user-item interactions and provide personalized recommendations [74].

During the KNN process, hyperparameter tuning is a common step, involving the optimization of the hyperparameter *k* (the number of top neighbours) in KNN algorithms. This optimization aims to enhance performance by identifying the most effective configuration for each algorithm [75].

5.4.1 Algorithm 3: User K-Nearest Neighbours (UserKNN) Algorithm

User-based collaborative filtering, specifically User K-nearest neighbours (UserKNN), identifies similar users based on their past interactions and recommends points of interest (POI) reviewed by those users.

1. Input Parameters:

- `user_id`: Identifier of the user for whom recommendations are generated.

2. Algorithm Workflow:

1) Projection Graph:

The algorithm first projects the user-item interaction sub-graph using the Graph Data Science (GDS) library [76]. The sub-graph contains nodes for users and POIs, along with the relationship "REVIEWED" indicating user interactions with POIs.

2) Create Fast RP Embeddings:

Fast Random Projection (FastRP) is applied to the projected sub-graph to generate embeddings representing users and POIs. This step utilizes the "rating" property of “REVIEWED” relationships to capture the strength of user interactions and the topological structure of the sub-graph. The resulting embeddings are stored as a property labelled "embedding" for each node within the projected sub-graph [77].

3) Similarity with User-based KNN:

User-based K-Nearest Neighbours (UserKNN) is performed on the graph with the optimal topK hyperparameter, identifying similar users based on their embedding vectors. The computed similarities are written back to the neo4j graph database as relationships labelled "CF_SIMILAR_USER" with corresponding similarity scores as a property “score”.

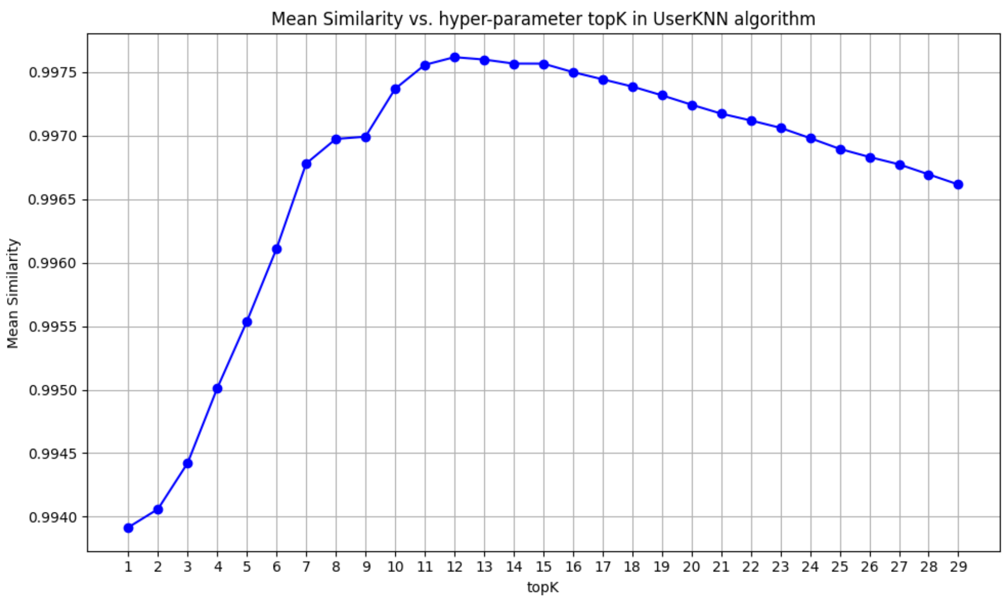
4) Make Recommendations:

Using the computed similarities, the algorithm retrieves the most similar users, prioritizing users with higher similarity scores, and sorts out the POIs reviewed by these similar users as recommendations. Recommendations are then sorted based on both user similarity and POI average ratings.

3. Hyper-Parameter Tuning (topK):

In the hyper-parameter tuning process for the topK parameter in the UserKNN algorithm, the value of *k was* systematically varied from 1 to 29 to evaluate its impact on the algorithm's performance [78]. For each value of *k*, the algorithm computed the similarity between users and their k nearest neighbours based on the generated embeddings.

Figure 17: Mean Similarity vs. Hyper-Parameter topK in UserKNN Algorithm



The results in the above diagram indicated that as the value of *k* increased, the mean similarity also increased, reflecting a broader scope of similar users considered in the recommendation process. However, beyond a certain point, the increase in mean similarity levelled off, suggesting reduced results in performance improvement. The optimal value of *k* was determined to be 12, where the mean similarity reached its highest value of 0.9976.

This value balances the trade-off between capturing sufficient user similarities for accurate recommendations while avoiding excessive computational overhead. Consequently, k=12 was chosen as the optimal hyperparameter for the UserKNN algorithm.

Additionally, the mean similarity of user nodes tends to be very high at around 0.99. This anomaly can be attributed to the characteristics of the dataset. As indicated in section 3.4, the analysis of the Distribution of All Reviews by Ratings revealed that the overwhelming proportion of reviews are 5-star ratings. Moreover, the Distribution of Users by Number of Reviews showed that the majority of users only write one review. Given these factors and the constraints imposed by privacy protection, where personal information about users is unavailable to compute individual characteristics of users, the similarity of ratings in single reviews tends to be highly similar, resulting in a substantial number of user pairs with a similarity of 1. This phenomenon significantly influences the mean similarity, leading to the very high mean similarity values.

4. Output Format:

The output of the UserKNN recommendation algorithm is a dataframe containing user IDs and recommended POI IDs.

5. Evaluation Result:

- **Precision Score:** 0.712
- **Recall Score:** 0.948
- **Coverage Score:** 0.623
- **F1 Score:** 0.814

User KNN (Algorithm 3) stands out with higher precision, its high precision score indicates a strong ability to recommend items that closely match users' preferences. Furthermore, Algorithm 3 achieves an impressive recall score, the highest among all algorithms, suggesting its effectiveness in retrieving a large proportion of relevant items. The algorithm achieving the highest F1 score implies its superior performance in recommending points of interest within this dataset.

Algorithm 3 stands out as a robust recommendation algorithm, showcasing impressive precision, recall, coverage score, and balanced performance as the F1 score is high, making it well-suited for diverse use cases.

5.4.2 Algorithm 4: Item K-Nearest Neighbours (ItemKNN) Algorithm

Item-based collaborative filtering identifies similar POIs based on user interactions and recommends those points of interest (POI) to users who have shown interest in related items.

1. Input Parameters:

- `poi_id`: Identifier of the POI for which recommendations are generated.

2. Algorithm Workflow:

1) Projection Graph:

The algorithm projects the user-item interaction sub-graph using the Graph Data Science (GDS) library. This sub-graph includes nodes representing users and points of interest (POI), connected by the "REVIEWED" relationship, indicating user interactions with POI.

2) Create Fast RP Embeddings:

Fast Random Projection (FastRP) is utilized on the projected sub-graph to generate embeddings representing POI and users. The "rating" property of the "REVIEWED" relationships is leveraged to capture the strength of user interactions and the sub-graph's topological structure. The resulting embeddings are stored as a property labelled "embedding" for each node within the projected sub-graph.

3) Similarity with Item-based KNN:

Item-based K-Nearest Neighbours (ItemKNN) is executed on the graph with the optimal topK hyperparameter, identifying POIs similar to the target POI based on their embedding vectors. The computed similarities are written back to the Neo4j graph database as relationships labelled "CF_SIMILAR_POI" with corresponding similarity scores as a property “score”.

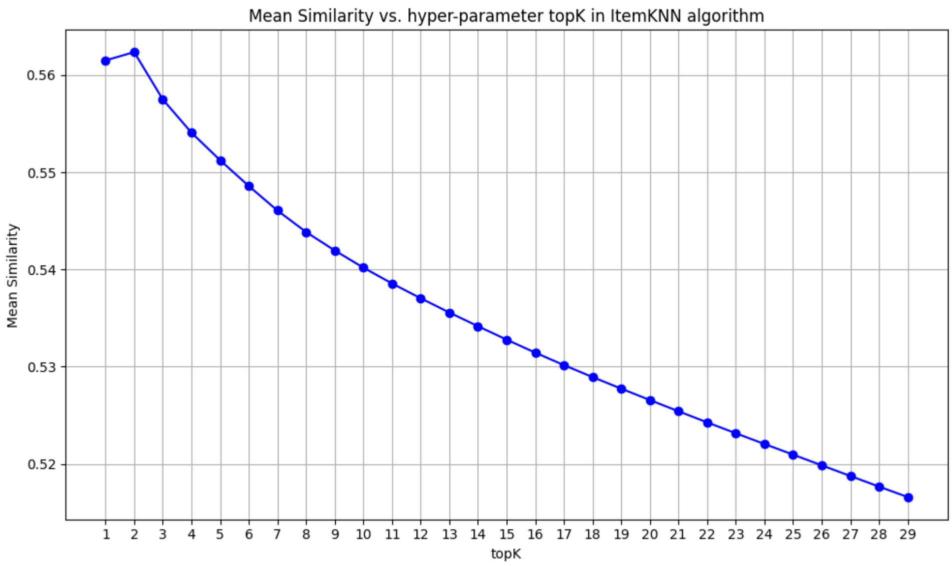
4) Make Recommendations:

Recommendations are generated based on the computed similarities. The algorithm retrieves POIs similar to the target POI, prioritizing those with higher similarity scores. Recommendations are then sorted based on both POI similarity and their average ratings.

3. Hyper-Parameter Tuning (topK):

In the hyper-parameter tuning process for the topK parameter in the ItemKNN algorithm, the value of *k* was systematically varied from 1 to 29 to evaluate its impact on the algorithm's performance. For each value of *k*, the algorithm computed the similarity between POI and their nearest neighbours based on the generated embeddings.

Figure 18: Mean Similarity vs. Hyper-Parameter topK in ItemKNN Algorithm



The results indicated that as the value of *k* increased, beyond a certain point, the mean similarity decreased, suggesting a narrower consideration of similar POIs in the recommendation process. The optimal value of *k* was determined to be 2, where the mean similarity reached its highest value of 0.5624.

4. Output Format:

The output of the ItemKNN recommendation algorithm is a dataframe containing input POI IDs and recommended POI IDs.

5. Evaluation Result:

- **Precision Score:** 0.968
- **Recall Score:** 0.057
- **Coverage Score:** 0.333
- **F1 Score:** 0.107

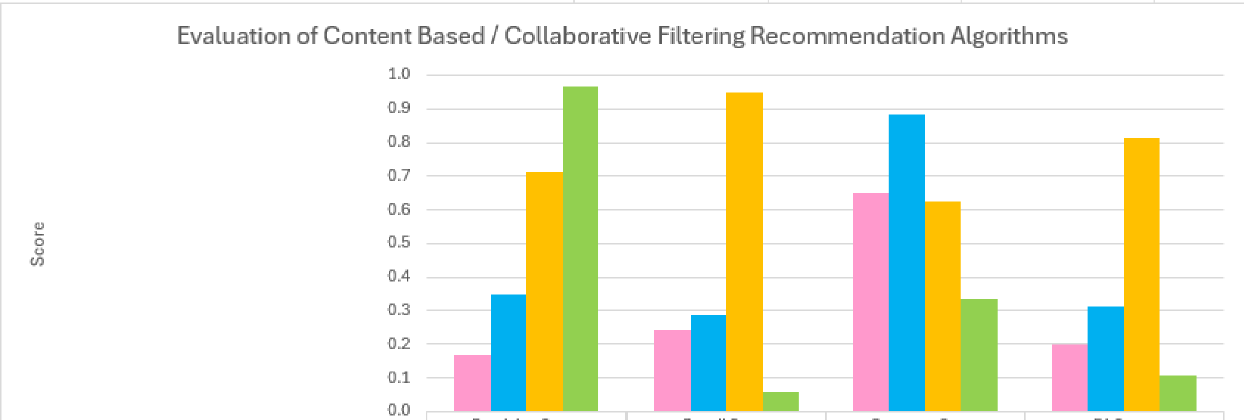
Item KNN (Algorithm 4) delivers an exceptional precision score but exhibits a very low recall score compared to other algorithms. While Algorithm 4 excels in recommending highly relevant items with the highest precision score, its limited ability to retrieve a wide range of relevant items results in the lowest recall score, which also results in a very low F1 score suggesting the imbalanced performance. Additionally, the algorithm's coverage score indicates that it covers a smaller portion of the item space compared to all previous algorithms.

It is best suited for use cases prioritizing high accuracy while disregarding the introduction of random new items for users to discover potential interests.

5.5 Algorithms Performance Comparison and Insights

The evaluation of individual algorithms provides valuable insights into their performance, facilitating informed decisions for algorithm selection tailored to specific use cases and priorities. The performance of different algorithms was summarized below and analyzed to determine the most effective approach for recommending points of interest (POI).

Figure 19: Evaluation of Content-Based / Collaborative Filtering Recommendation Algorithms



	Precision Score	Recall Score	Coverage Score	F1 Score
Algo 1. Content Based Filtering - Heuristic	0.167216062331435	0.244201312910284	0.652173913043478	0.19850587
Algo 2. Content Based Filtering - Node Similarity	0.347126436781609	0.287345385347288	0.884057971014492	0.314419573
Algo 3. Collaborative Filtering - UserKNN with FastRP	0.712360289283366	0.948358862144420	0.623188405797101	0.813591139
Algo 4. Collaborative Filtering - ItemKNN with FastRP	0.967741935483871	0.057088487155090	0.333333333333333	0.107816712

Summary of potential use scenarios for each algorithm:

- **Algorithm 1: Heuristic Algorithm** serves as a baseline method, offers minimal performance, reserves as a backup option for extreme cases where all other algorithms are not applicable.
- **Algorithm 2: Node Similarity Algorithm** is ideal for scenarios prioritizing the introduction of random new items to help users discover potential interests.
- **Algorithm 3: User K-Nearest Neighbours (UserKNN) Algorithm** emerges as a robust choice for recommendation systems across various use cases, demonstrating balanced precision, recall, and coverage scores.
- **Algorithm 4: Item K-Nearest Neighbours (ItemKNN) Algorithm** is recommended when achieving high accuracy is paramount, although with narrower choices.

5.6 Ensemble Learning Framework

Ensemble learning combines the predictions of multiple base algorithms to improve the overall accuracy and robustness of the recommender system [45]. A simplified ensemble techniques with majority voting was employed to aggregate the predictions of heuristic methods, content-based filtering, and collaborative filtering [79]. By leveraging the diversity of individual algorithms, ensemble learning combines the recommendations generated by various algorithms and selects the most frequently recommended items and mitigates the weaknesses of individual algorithms and yields more accurate recommendations [80].

1. Input Parameters:

- **`poi_id`**: Identifier of the reference point of interest (POI).
- **`user_id`**: Identifier of the user for whom recommendations are generated.
- **`algo_combination`**: List of integers representing the chosen combination of algorithms for ensemble learning. Each integer corresponds to a specific algorithm:
    - Algorithm 1: Content-Based Filtering - Heuristic Algorithm
    - Algorithm 2: Content-Based Filtering - Node Similarity Algorithm
    - Algorithm 3: Collaborative Filtering - UserKNN Algorithm
    - Algorithm 4: Collaborative Filtering - ItemKNN Algorithm

2. Algorithm Workflow:

1) Implementation of Individual Algorithm:

Individual algorithm is implemented, with all requisite preprocessing steps conducted as detailed in Sections 5.3 and 5.4.

2) Individual Algorithm Recommendations:

For each algorithm specified in **`algo_combination`**, recommendations are generated using the corresponding method.

3) Ensemble Recommendation:

The ensemble recommendations are generated by aggregating the outputs of all selected algorithms using the majority voting method. Only items that are recommended by multiple algorithms are included in the final ensemble recommendation list. The ranking of these recommendations is determined by their frequency across all algorithm outputs. In the case of ties, items are prioritized based on their average ranking across all selected algorithms.

4) Output Generation:

The final recommendations, along with their corresponding User and POI identifiers, are returned as the output as a dataframe.

3. Output Format:

The output of the ensemble recommendation algorithm is a dataframe containing the recommended POI IDs (`rec_poi_id`), corresponding user IDs (`user_id`), and input POI IDs (`poi_id`) of each recommendation.

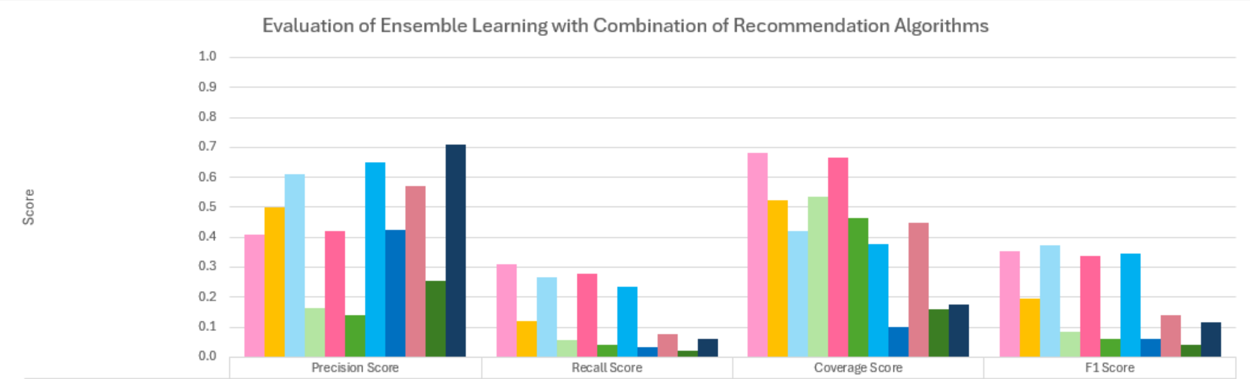
4. Evaluation Result:

To assess the effectiveness of ensemble learning, all eleven different combinations derived from the four individual algorithms was tested, specified by the `algo_combination` parameter, as outlined in the section on input parameters.

The details of performance evaluation are in **Appendix B: Performance Evaluation of Ensemble Learning with All Combination of Algorithms**.

Figure 20 below shows an overview of the evaluation metrics for the ensemble strategy with all different combinations of Algorithms.

Figure 20: Evaluation of Ensemble Learning with Combination of Recommendation Algorithms



	Precision Score	Recall Score	Coverage Score	F1 Score
Ensemble Learning - Algo 1,2,3,4	0.409778812572759	0.308096280087527	0.681159420289855	0.351736198
Ensemble Learning - Algo 2,3,4	0.499089253187613	0.119912472647702	0.521739130434782	0.193366267
Ensemble Learning - Algo 1,3,4	0.610169491525423	0.267833698030634	0.420289855072463	0.372262774
Ensemble Learning - Algo 1,2,4	0.161800486618004	0.058205689277899	0.536231884057971	0.085613132
Ensemble Learning - Algo 1,2,3	0.422060766182298	0.279649890590809	0.666666666666666	0.336404317
Ensemble Learning - Algo 1,2	0.139534883720930	0.039387308533917	0.463768115942029	0.061433447
Ensemble Learning - Algo 1,3	0.651387213510253	0.236323851203501	0.376811594202898	0.346820809
Ensemble Learning - Algo 1,4	0.423529411764705	0.031509846827133	0.101449275362318	0.058655804
Ensemble Learning - Algo 2,3	0.570063694267515	0.078336980306346	0.449275362318840	0.137745287
Ensemble Learning - Algo 2,4	0.253731343283582	0.022319474835886	0.159420289855072	0.041029767
Ensemble Learning - Algo 3,4	0.710000000000000	0.062144420131291	0.173913043478260	0.114285714

In summary, ensemble learning with specific combinations of algorithms demonstrates a well-balanced performance, by the following configurations:

- Ensemble Learning with Algorithms 1,2,3,4
- Ensemble Learning with Algorithms 1,3,4
- Ensemble Learning with Algorithms 1,2,3
- Ensemble Learning with Algorithms 1,3

The four configurations listed exhibit comparable performance scores for precision, recall, coverage, and F1 score. For our tourism recommender system, our objective is to provide precise recommendations while also allowing users to explore new spots. Employing multiple algorithms in ensemble learning ensures the system's robustness and readiness to handle unseen data, thereby yielding unbiased results.

As a result, **"Ensemble Learning with Algorithms 1,2,3,4"** is chosen as the primary algorithm for this recommender engine due to its well-balanced performance and incorporation of the highest number of algorithms in the ensemble learning framework. As a result, it emerges as the top preferred choice for our recommender system.

5.7 Hybrid Algorithm

Based on the analysis above, a hybrid algorithm was adopted in the recommender system based on preferences of order:

1. **Priority of Ensemble Learning**: Ensemble Learning with Algorithms 1,2,3,4 is prioritized to provide comprehensive recommendations as the top priority.

2. **Fallback to Individual Algorithms**: If Ensemble Learning with Algorithms 1,2,3,4 generates an insufficient number of recommendations, individual algorithms were employed in descending order of F1 score. Algorithm 3 UserKNN Algorithm is prioritized for balanced recommendations with highest F1 score. Algorithm 2 Node Similarity Algorithm is next, providing moderate precision and a broader item range, potentially introducing new items. Followed by Algorithm 4 ItemKNN Algorithm for high precision and a narrower range of items. Finally, Algorithm 1 Heuristic Algorithm serves as a backup, offering lower precision recommendations.

3. **Adaptive Recommendation Generation**: Each algorithm is executed whenever the required input (`user_id`, `poi_id`) is available. If the input is unavailable, the algorithm is skipped, and the next one is considered. Recommendation generation stops once a predefined number of recommendations is collected.

This final hybrid algorithm ensures the robustness and effectiveness of the recommender system, catering to varying use cases and data availability.

6 System Implementation

To showcase the integration of the developed recommender engine with the constructed knowledge graph, both components will be integrated into a web application. This section provides an in-depth description of the implementation process for the demo web application, detailing how the knowledge graph and recommender engine functionalities are incorporated to deliver personalized recommendations and interactive user experiences.

6.1 Functional Requirements

As this web application serves as a demonstration of how a knowledge-graph-based recommender engine can be integrated into a web application, it will focus solely on essential functionalities to optimize time and resources.

- **Recommendation generation:** Utilize recommender engine to generate personalized recommendations for users.
- **Integration with Neo4j database:** Retrieve data from Neo4j graph database related to users, POIs, and their interactions.
- **Responsive user interface:** Design a user-friendly interface that adapts to different screen sizes and devices.
- **User authentication:** Allow users to log in and log out.
- **Profile management:** Enable users to view their profiles.

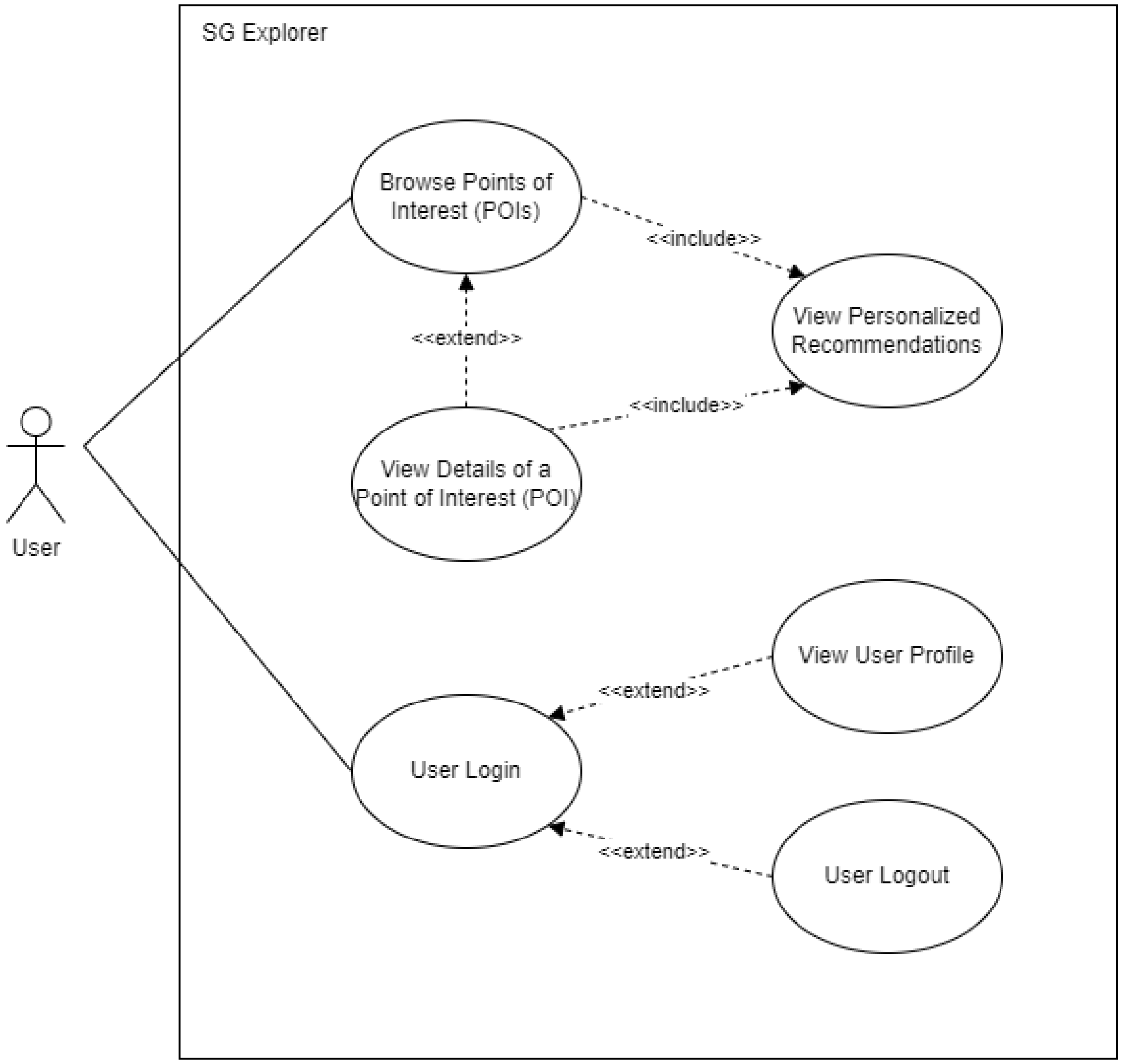
6.1.1 Use Cases

The use case scenarios for the demo web application include users being able to access a list of points of interest (POIs), view detailed information about a specific POI by clicking on it from the list, and receive personalized recommendations for POIs in Singapore based on their past interactions and POI data.

Additionally, users have the capability to log in to view their profiles and use the logout function.

These use case diagram of the demo web application is illustrated in the Figure 21.

Figure 21: Use Case Diagram of Demo Web Application of Recommender System



The detailed use case descriptions are in **Appendix C: Web Application Use Case Descriptions**.

6.1.2 User Interface

The user interface design focuses on simplicity, responsiveness, and ease of navigation, ensuring a consistent and visually appealing design across different devices and screen sizes.

The demo web application comprises four main pages:

1. **Home Page:** Displays the complete list of POIs, display recommendations. For logged-in users, personalized recommendations, titled 'You might like this,' are provided in the right column, generated based on user's past interactions. However, for guest users, without access to their interaction history, baseline recommendations of the most popular places are provided instead.

Figure 22: User Interface of Home Page for Guest User

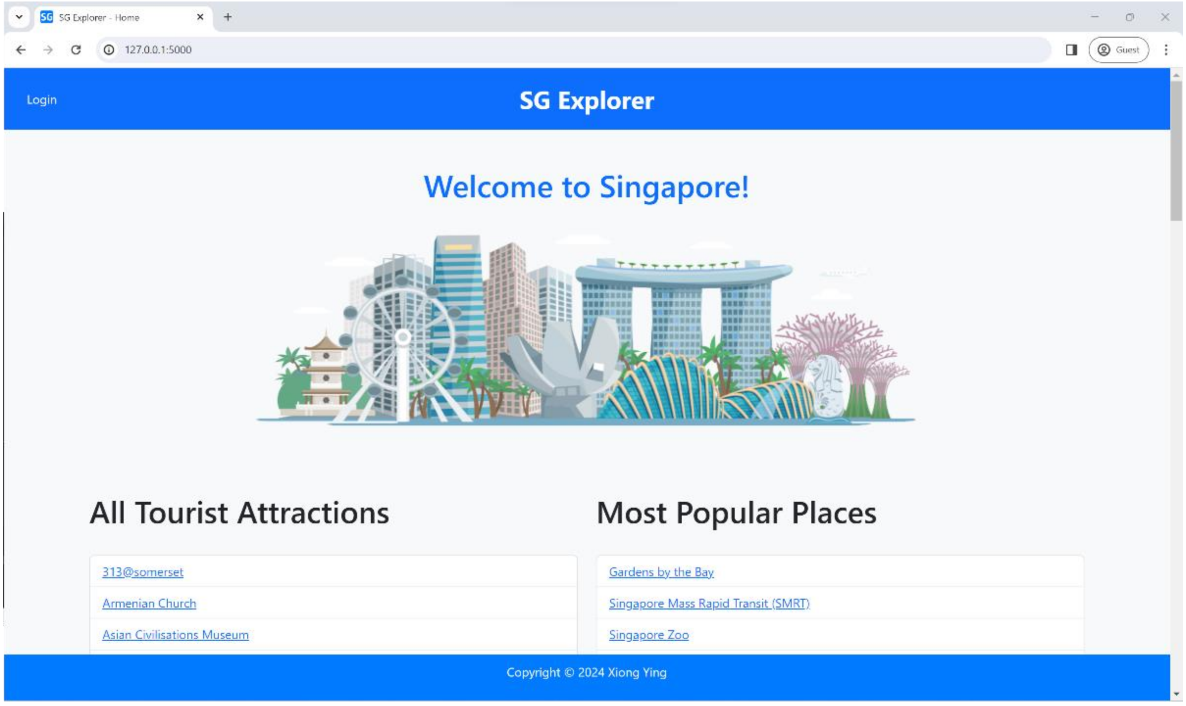
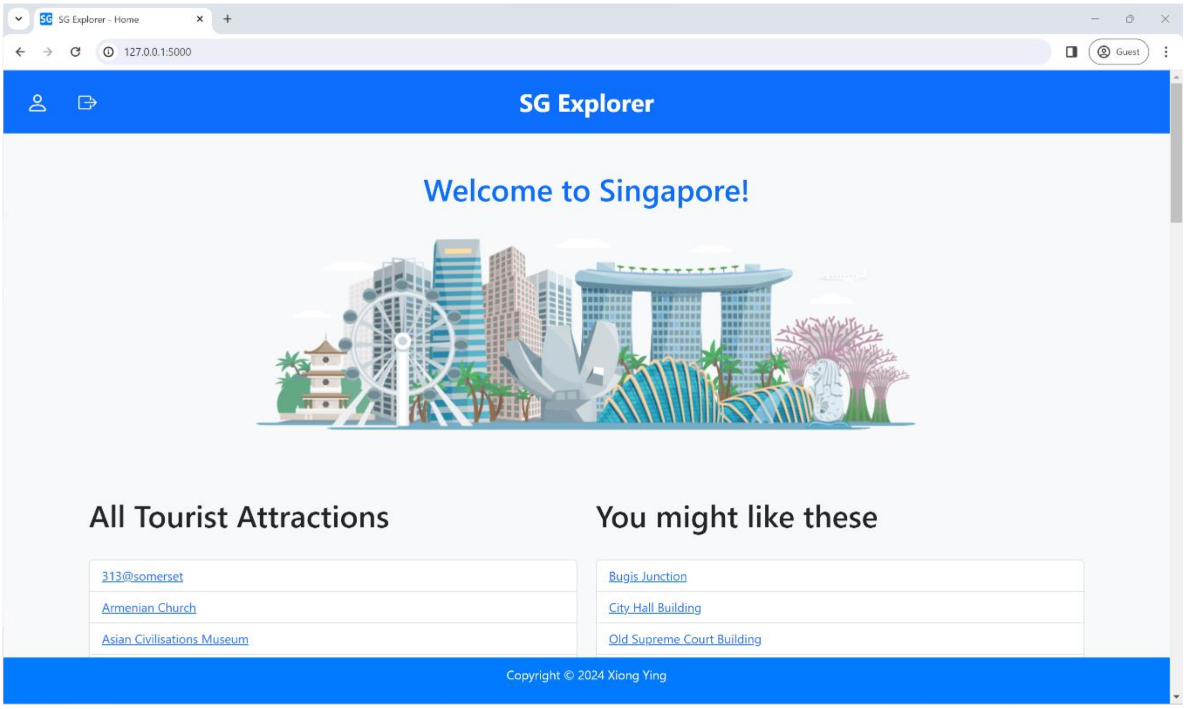


Figure 23: User Interface of Home Page for Logged-in User



2. **POI Page:** Provide comprehensive details about a chosen POI and showcases personalized recommendations. For logged-in users, the recommendations are tailored based on their past interactions, resulting in a distinct set compared to the generic recommendations provided to guest users, which are solely based on the currently viewed POI.

Figure 24: User Interface of POI Page for Guest User

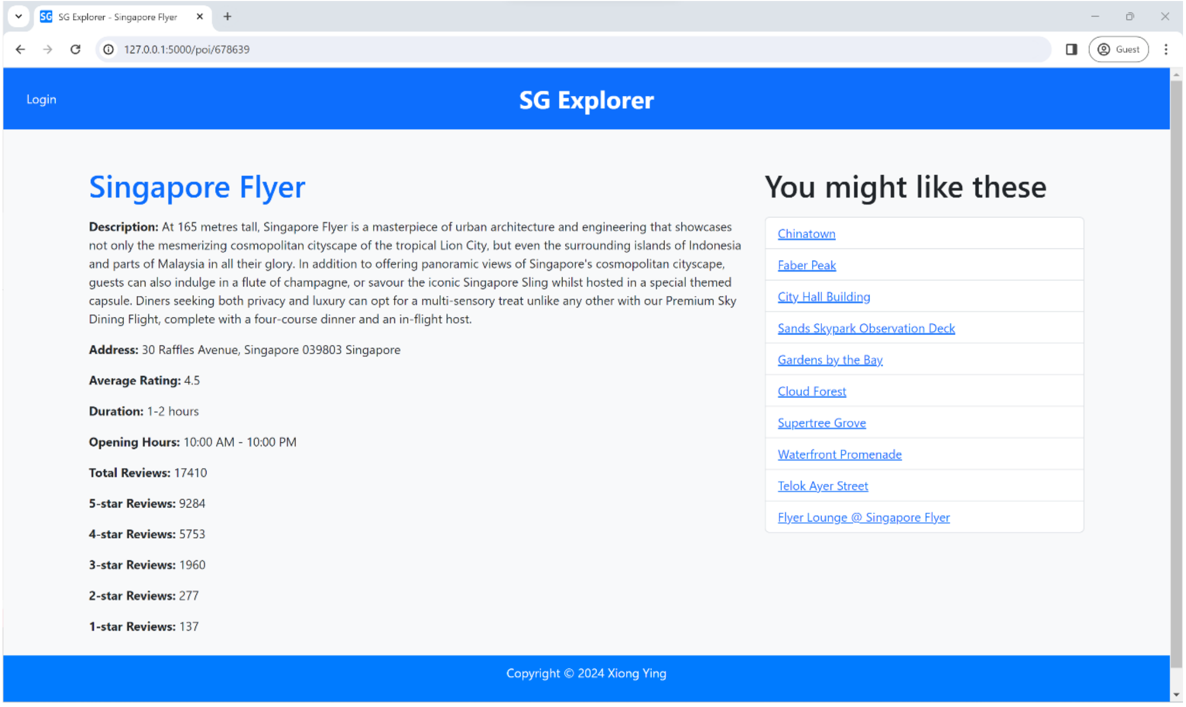
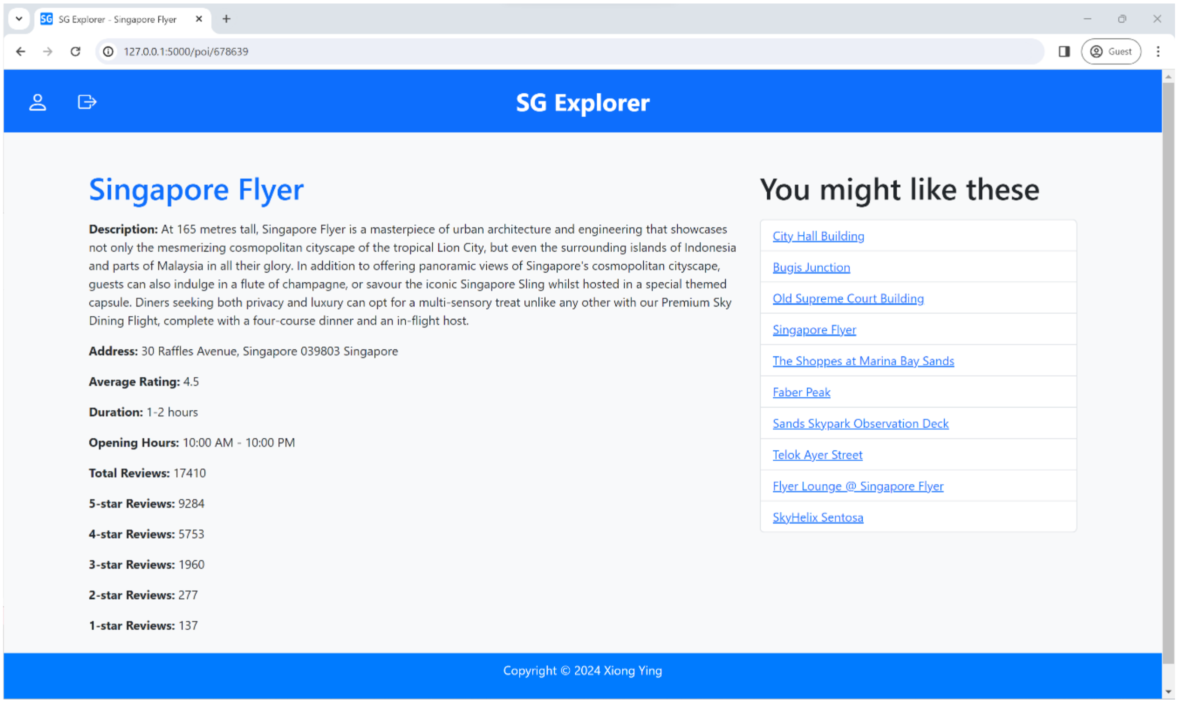


Figure 25: User Interface of POI Page for Logged-in User



3. **Login Page:** Allows users to enter their account credentials for logging in. If incorrect credentials are entered, users will see an error message in red stating 'Invalid username or password,' alerting them to the login issue.

Figure 26: User Interface of Login Page

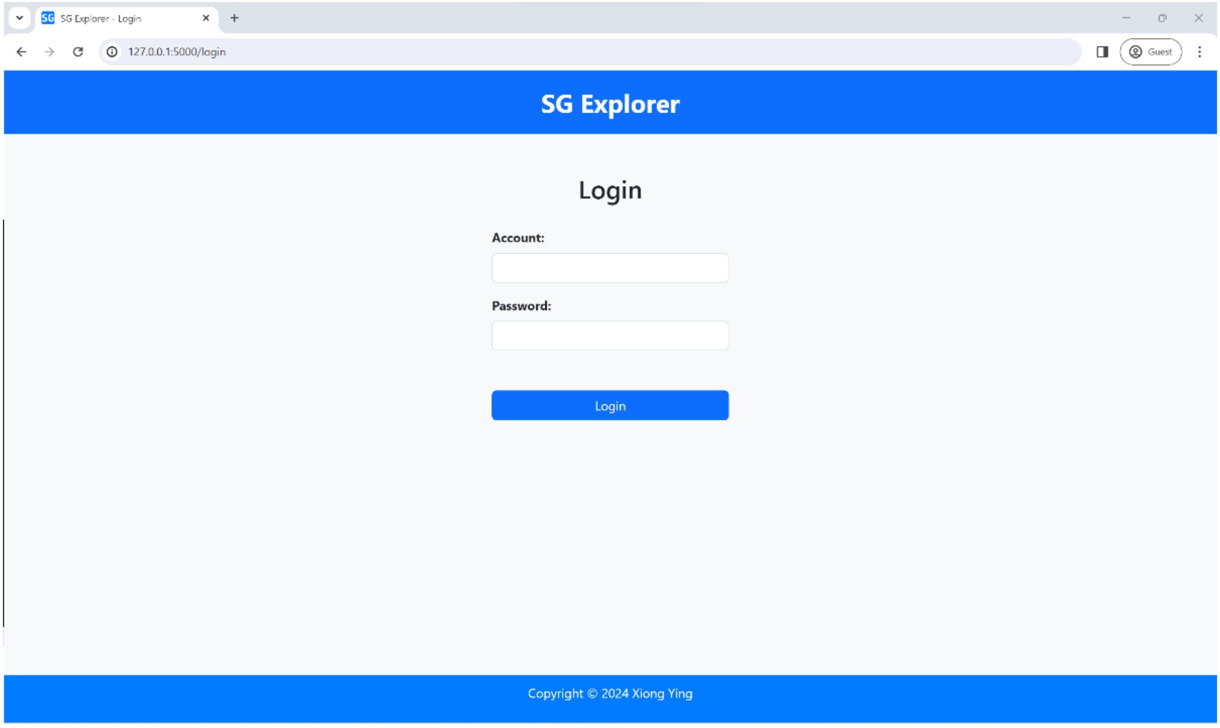
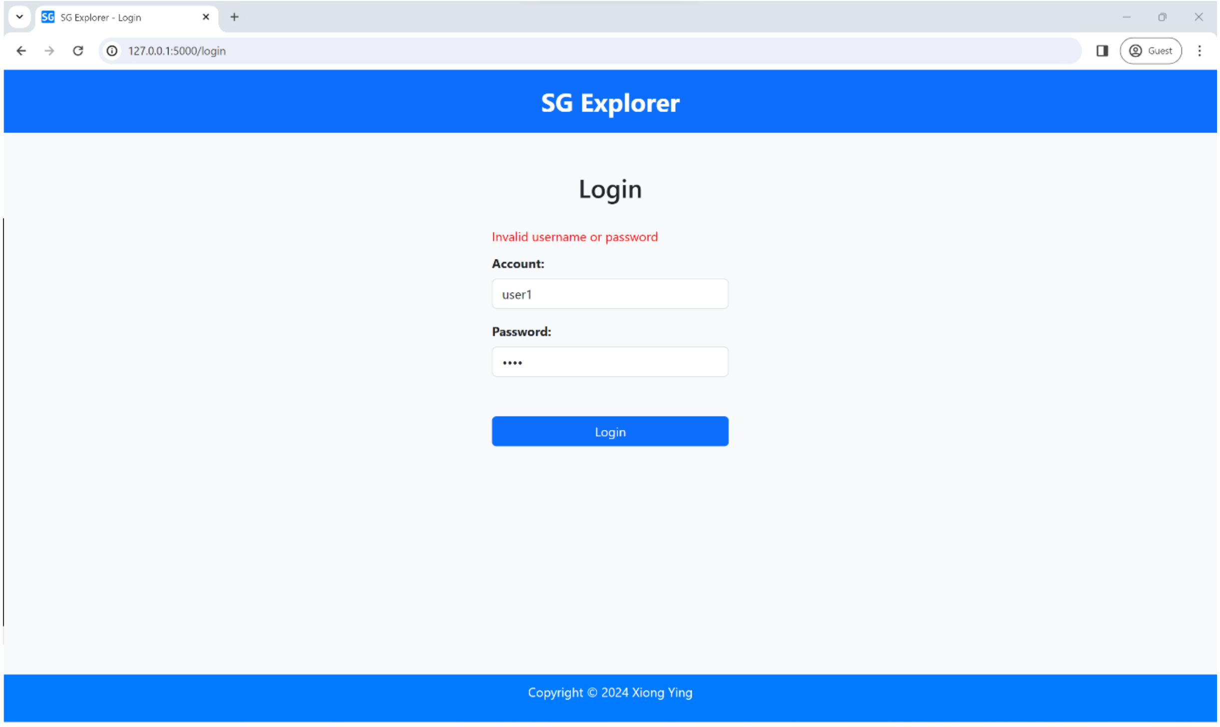
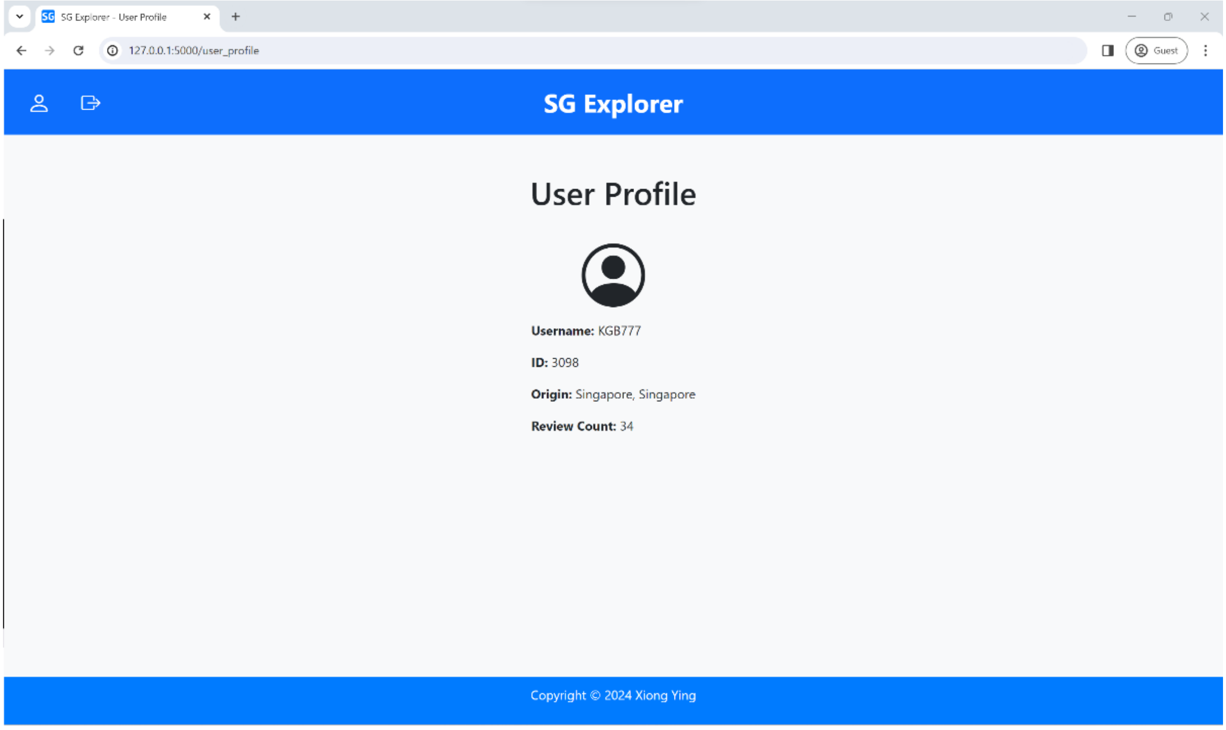


Figure 27: User Interface of Login Page with Incorrect Credentials Entered



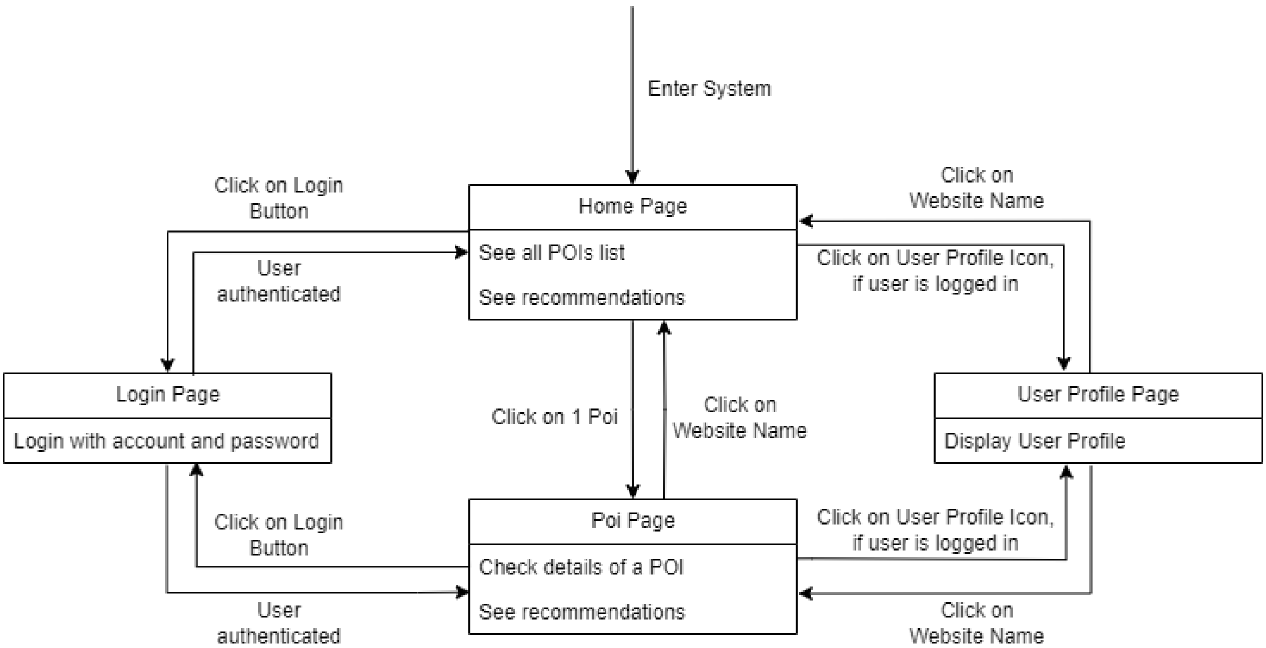
4. **User Profile Page:** After logging in, users can view their profile information.

Figure 28: User Interface of User Profile Page



This dialog map illustrates the user's navigation through the various pages of the web application. When users enter the website, they will land on the home page. Clicking on any point of interest (POI) listing from the home page will direct them to the POI detail page. Upon clicking the 'Login' button, users will be directed to the Login Page. Once logged in, they can access the 'User Profile' button to view their profile page. Additionally, regardless of the page they are on, clicking on the website name in the middle of the navigation bar will return them to the home page.

Figure 29: Dialog Map of the Demo Web Application



6.2 Non-functional Requirements

Below are the non-functional requirements for the demo web application:

1. **Performance**: The application must ensure fast response times for user interactions and recommendation generation. It should be capable of handling multiple concurrent requests without significant latency.

2. **Scalability**: The system should be designed to efficiently handle a growing number of users and data, involving optimizing database queries [81].

3. **Usability**: Creating an intuitive user interface is essential to enhance user experience and engagement. The application should be accessible across different devices and screen sizes, with user-friendly interactions [82].

4. **Maintainability**: The codebase should be well-organized and documented to facilitate easy maintenance and future enhancements. This involves following coding best practices and conducting regular code reviews [83].

6.3 Web Application Architecture

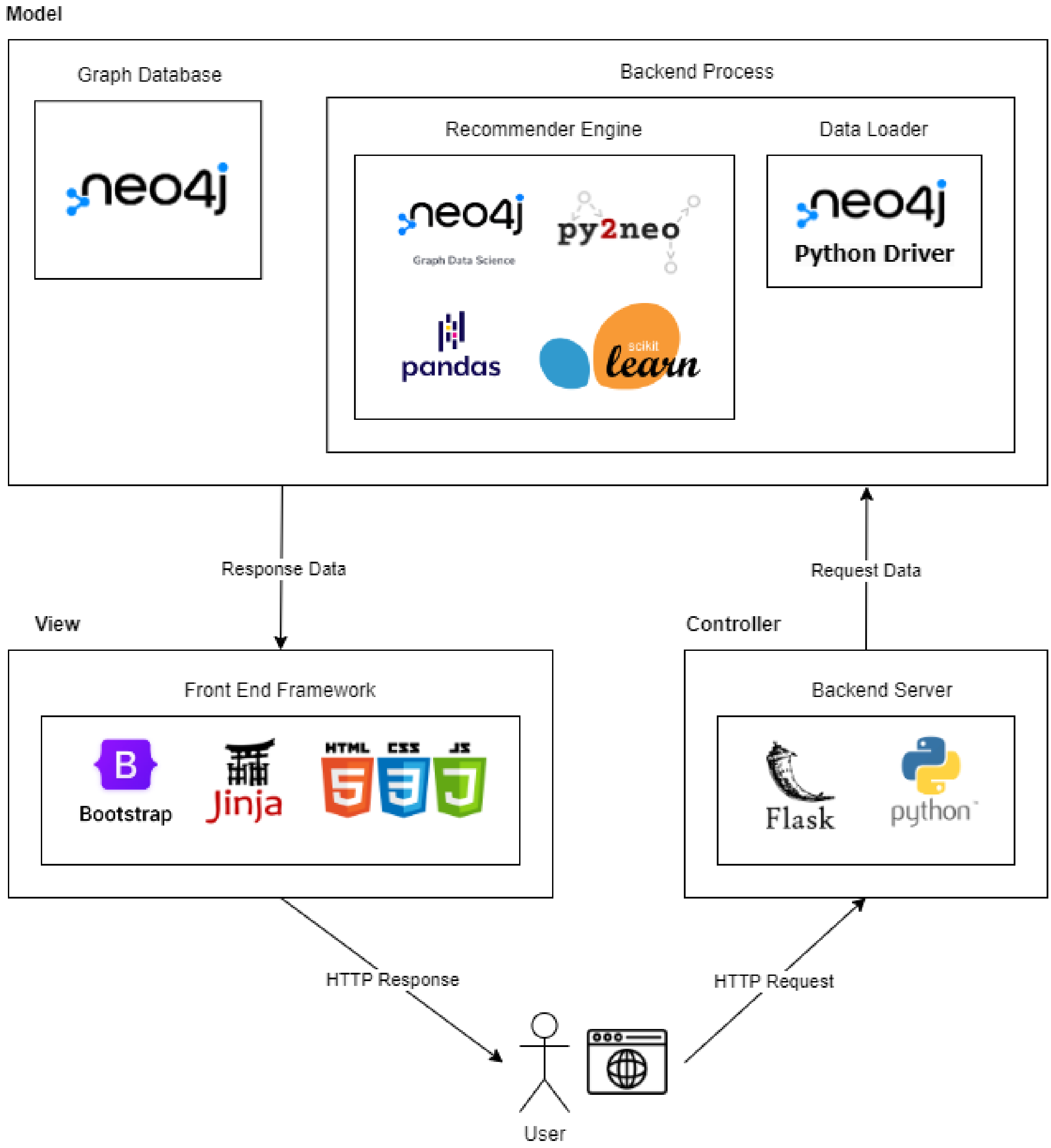
The primary objective of this project is to showcase the integration of the developed recommender engine and Neo4j graph database within web applications. To achieve this, the aim is to utilize lightweight web development tools that facilitate rapid development, by seamlessly integrating Flask's backend functionalities with Bootstrap's frontend components, along with Jinja web templates [84].

The web application employs the Model-View-Controller (MVC) Architecture, which consists of three main components:

- Model: establishes the data structure.
- View: manages the user interface and presents the data.
- Controller: modifies both the model and view in response to user interactions [85].

Below is the diagram illustrating the architecture of the web application:

Figure 30: Software Architecture of the Demo Web Application



Components of the architecture:

1. **View**:

   The frontend utilizes Bootstrap framework for responsive design, ensuring optimal user experience on various devices. Bootstrap serves as a widely adopted open-source framework for constructing responsive web applications [86]. Leveraging its library of pre-designed HTML, CSS, and JavaScript components, combined with the Jinja web template engine, visually appealing and adaptable user interfaces can be efficiently built [87].

2. **Controller**:

   On the backend, Flask emerges as an ideal choice due to its lightweight and adaptable nature as a micro web framework for Python [88]. Additionally, it seamlessly supports the integration of static files like images, CSS and JavaScript, facilitating the effortless incorporation of Bootstrap assets into Flask applications [89]. Flask is also used to handle routing, authentication, and interaction with the Neo4j database [90].

3. **Model**:

   Neo4j graph database is integrated to store and retrieve user data, points of interest (POI) information, and their relationships. This allows for efficient querying and recommendation generation based on graph-based algorithms.

   The backend process comprises two modules:

   - **Data loader:** It verifies whether the database is initially initialized and empty. If so, it proceeds to load the data into the Neo4j graph database.

   - **Recommender:** This module consists of the recommendation generation process, which involves deploying and integrating various algorithms, including content-based filtering, collaborative filtering, and ensemble learning.

In summary, the integration of Flask, Bootstrap and neo4j graph database provides a flexible framework for the development of web applications. This enables the effective demonstration of the capabilities of the knowledge-graph-based recommender engine.

6.4 System Validation

The implemented system undergoes thorough testing and validation procedures to ensure alignment with project objectives regarding functionality and performance [91]. This includes conducting unit tests on individual components, integration tests on system modules, and end-to-end testing of user workflows [92].

6.4.1 Unit Testing

Unit testing conducted is covering two backend modules and the web application:

1. **Recommender Module**

- Test Case 1: Evaluates the response when only the `poi_id` input is available.
- Test Case 2: Evaluates the response when only the `user_id` input is available.
- Test Case 3: Evaluates the response when both input `poi_id` and `user_id` are available.
- Test Case 4: Evaluates the response when no input data is provided.

2. **Data Loader Module**

- Test Case 5: Validates the process when the Neo4j database is initially initialized and empty, without any data.
- Test Case 6: Ensures proper handling when Neo4j database already contains data.

3. **Web Application Module**

- Test Case 7: Tests the functionality of navigating to the home page.
- Test Case 8: Ensures proper user login and authentication.
- Test Case 9: Validates the functionality of viewing user profiles.
- Test Case 10: Assesses the user logout process.

6.4.2 Integration Testing

The following integration test cases cover various scenarios of system functionality:

- Test Case 11: Assesses the system behavior when starting with an empty database.
- Test Case 12: Evaluates the response when the application begins with a pre-loaded database.
- Test Case 13: Verifies the behavior when a user clicks on a specific POI.

6.4.3 User Workflow Testing

The system's end-to-end functionality is tested with three different user accounts, each representing varying usage scenarios:

- Test Case 14: Evaluates the workflow with a user account who has written 34 reviews.

- Test Case 15: Evaluates the workflow with a user account who has written only 5 reviews.
- Test Case 16: Evaluates the workflow with a user account who has written 1 single review.
- Test Case 17: Evaluates the workflow when accessed as a guest user.

In summary, all test cases have successfully passed validation.

Details of all test cases are provided in the **Appendix D: System Validation Test Cases and Results**.

7 Discussion and Future Work

7.1 Strengths and Limitations of the Approach

The approach adopted in this project exhibits several strengths along with certain limitations.

Strengths:

- **Comprehensive Evaluation:** The project undertook a thorough evaluation of individual algorithms and ensemble learning strategies, providing valuable insights into their performance across various metrics.
- **Balanced Performance:** Ensemble learning with specific combinations of algorithms demonstrated well-balanced performance, ensuring robustness and readiness for handling data.
- **User-Centric Design:** The recommender system prioritizes providing precise recommendations while also allowing users to explore new spots, enhancing user experience and engagement.

Limitations:

- **Data Limitations:** Due to restrictions on web scraping from travel sites and hardware resource constraints when dealing with a potentially overwhelming amount of data from the entire Singapore landscape, only a portion of the available data in Singapore is utilized for demonstration purposes.
- **Resource Constraints:** Due to limited time and resources, the web application implemented only essential functionalities, which may restrict the scope of the demonstration.
- **Algorithm Complexity:** The integration of multiple algorithms in ensemble learning adds complexity to the system, potentially leading to increased computational cost with a larger scale database.

7.2 Future Work and Potential Improvements

Moving forward, several areas offer opportunities for future work and potential improvements:

- **Advancing Ensemble Learning**: Delving into additional machine learning models and fine-tuning ensemble learning methods can elevate recommendation precision and breadth.
- **Database Scaling**: Expanding the database capacity to accommodate larger and more diverse datasets can improve recommendation accuracy and system efficiency.
- **User Feedback Analysis**: Exploring methods for explicitly or implicitly gathering user feedback and interaction data can provide insights into user satisfaction and engagement levels. This information can then be leveraged to refine recommendation algorithms and strategies.

By addressing these areas, the recommender system can adapt to evolving user preferences and deliver enhanced personalized experiences.

8 Project Management

An iterative approach was adopted to swiftly respond to new ideas, optimize resource allocation, and maintain flexibility in project execution. This flexibility was essential to adapt to changing requirements, prioritize tasks, and deliver incremental updates. Given the novelty of the project area and its solo nature, an agile project management method and incremental development proved most suitable.

Throughout the project period, one full day per week is dedicated to tasks from ideation and exploration to implementation and documentation.

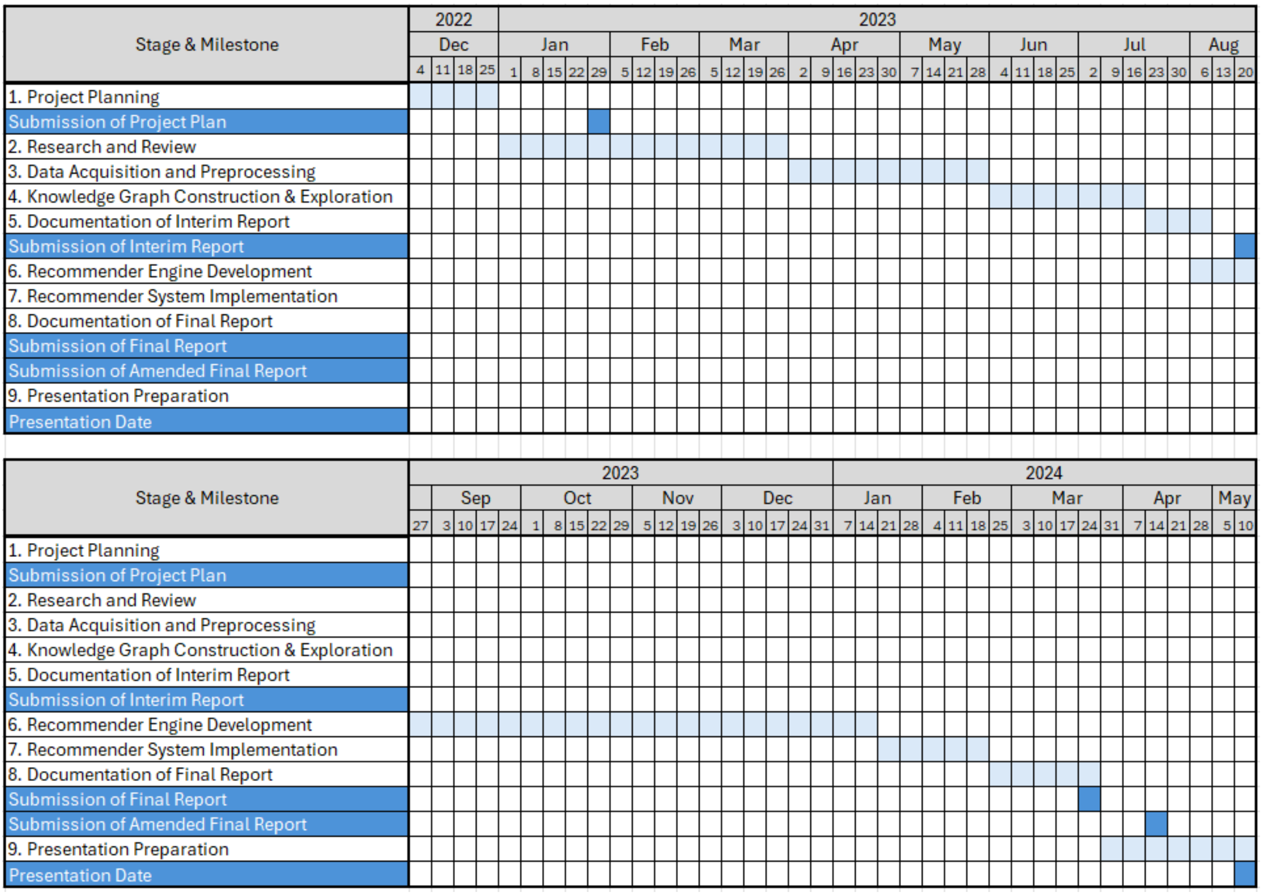
The project comprises the following tasks:

1. Project Planning
2. Research and Review
3. Data Acquisition and Preprocessing
4. Knowledge Graph Construction & Exploration
5. Documentation of Interim Report
6. Recommender Engine Development
7. Recommender System Implementation
8. Documentation of Final Report
9. Presentation Preparation

Despite the need for flexibility, effective project management was essential to meet deadlines. Task scheduling with Gantt charts offers a visual representation of project timelines and milestones [93]. This enabled the efficient allocation of resources and ensured tasks were completed in a timely manner.

Below is the Gantt chart outlining the project schedule with dates, including key milestones:

Figure 31: Gannt Chart for Project Schedule Management



This schedule served as a roadmap for project execution, guiding efforts and ensuring tasks were completed promptly to achieve project objectives.

9 Conclusion

The project offers insights into the development of knowledge-graph-based recommender systems tailored for urban tourism. It outlines the end-to-end process, covering data acquisition and preprocessing, knowledge graph construction and querying, recommender algorithm methodology and evaluation, as well as system implementation and comprehensive testing.

Contributions to the field involve potential advancements in graph-based recommender engine and knowledge graph applications, potentially offering benefits such as enhanced personalized experiences for users in urban tourism settings.

References

[1] "What is a knowledge graph?," IBM, [Online]. Available: https://www.ibm.com/topics/knowledge-graph#:~:text=A%20knowledge%20graph%2C%20also%20known,the%20term%20knowledge%20%E2%80%9Cgraph.%E2%80%9D. [Accessed 29 Jan 2023].

[2] Lisa Ehrlinger, Wolfram Wöß, "Towards a Definition of Knowledge Graphs," in Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics, Leipzig, Germany, 2016.

[3] "Knowledge Graphs - A mirror of (tourist) reality.," OPEN DATA, 10 Nov 2019. [Online]. Available: https://open-data-germany.org/en/knowledge-graphs/. [Accessed 10 Jun 2023].

[4] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas S., "Knowledge Graphs," ACM Computing Surveys, vol. 54, no. 4, p. 71:1, 2021.

[5] A. Das, "Knowledge Graphs: Deep Dive into its Theories and Applications," Analytics Vidhya, 19 May 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/01/knowledge-graphs-deep-dive-into-its-theories-and-applications/#:~:text=Knowledge%20graphs%20are%20used%20to,people%2C%20things%2C%20and%20concepts.. [Accessed 25 Feb 2024].

[6] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa & Francesco Osborne, "Knowledge Graphs: Opportunities and Challenges," Artificial Intelligence Review, vol. 56, p. 13071–13102, 2023.

[7] A. Singhal, "Introducing the Knowledge Graph: things, not strings," Google, 16 May 2012. [Online]. Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/. [Accessed 25 Feb 2024].

[8] Qi He, Bee-Chung Chen, Deepak Agarwal, "Building The LinkedIn Knowledge Graph," LinkedIn, 6 Oct 2016. [Online]. Available: Building The LinkedIn Knowledge Graph. [Accessed 25 Feb 2024].

[9] Jie Wu, Xinning Zhu, Chunhong Zhang, Zheng Hu, "Event-centric Tourism Knowledge Graph—A Case Study of Hainan," in Knowledge Science, Engineering and Management, 2020.

[10] Dinghe Xiao, Nannan Wang, Jiangang Yu, Chunhong Zhang, Jiaqi Wu, "A Practice of Tourism Knowledge Graph Construction based on Heterogeneous Information," in Proceedings of the 19th Chinese National Conference on Computational Linguistics, Haikou, China, 2020.

[11] Alessandro Chessa, Gianni Fenu, Enrico Motta, Francesco Osborne, Diego Reforgiato Recupero, Angelo Salatino, Luca Secchi, "Data-driven Methodology for Knowledge Graph Generation within the Tourism Domain," Semantic Web 0 IOS Press, vol. 0, no. 1, pp. 3314-4528, 2022.

[12] F. Ricci, "Recommender Systems in Tourism," Handbook of e-Tourism, pp. 457-474, 2022.

[13] A. Morvan, "Expedia Group @ RecSys 2021," Expedia Group Technology, 21 Oct 2021. [Online]. Available: https://medium.com/expedia-group-tech/expedia-group-recsys-2021-ff791f42ba07. [Accessed 25 Feb 2024].

[14] "Machine Learning in production: the Booking.com approach," Booking.com Data Science, 29 Oct 2019. [Online]. Available: https://booking.ai/https-booking-ai-machine-learning-production-3ee8fe943c70. [Accessed 25 Feb 2024].

[15] "Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering," Towards Data Science, 29 Jul 2020. [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421. [Accessed 25 Feb 2024].

[16] "Defining a Graph Schema," TigerGraph, [Online]. Available: https://docs.tigergraph.com/gsql-ref/current/ddl-and-loading/defining-a-graph-schema#:~:text=A%20graph%20schema%20is%20a,(properties)%20associated%20with%20it.. [Accessed 3 Mar 2024].

[17] V. Tatan, "In 10 minutes: Web Scraping with Beautiful Soup and Selenium for Data Professionals," Towards Data Science, 17 Jun 2019. [Online]. Available: https://towardsdatascience.com/in-10-minutes-web-scraping-with-beautiful-soup-and-selenium-for-data-professionals-8de169d36319. [Accessed 3 Mar 2024].

[18] "The Selenium Browser Automation Project," Selenium, [Online]. Available: https://www.selenium.dev/documentation/. [Accessed 3 Mar 2024].

[19] M. Breuss, "Beautiful Soup: Build a Web Scraper With Python," RealPython, [Online]. Available: https://realpython.com/beautiful-soup-web-scraper-python/. [Accessed 3 Mar 2024].

[20] "Pretending I'm a human(while web scraping)," 5 Jun 2022. [Online]. Available: https://medium.com/@dungwoong/pretending-im-a-human-while-web-scraping-d5464e36f24. [Accessed 3 Mar 2024].

[21] "A complete guide to Web Scraping with Selenium & Python in 2024," Nanonets, 6 Feb 2024. [Online]. Available: https://nanonets.com/blog/web-scraping-with-selenium/. [Accessed 3 Mar 2024].

[22] Oanottage, "Data Preprocessing and Cleaning," 19 Jul 2023. [Online]. Available: https://medium.com/@oanottage/data-preprocessing-and-cleaning-b64c2098b743. [Accessed 3 Mar 2024].

[23] "Web Scraping：How to bypass anti-scraping techniques," Octoparse, 8 May 2019. [Online]. Available: https://octoparsewebscraping.medium.com/web-scraping-is-a-technique-that-enables-quick-in-depth-data-retrieving-1d65eda4ef40. [Accessed 3 Mar 2024].

[24] Z. Blumenfeld, "Exploring Practical Recommendation Systems In Neo4j," Towards Data Science, 18 Dec 2021. [Online]. Available: https://towardsdatascience.com/exploring-practical-recommendation-engines-in-neo4j-ff09fe767782. [Accessed 3 Mar 2024].

[25] M. Milankovich, "The Cold Start Problem for Recommender Systems," 14 Jul 2015. [Online]. Available: https://medium.com/@markmilankovich/the-cold-start-problem-for-recommender-systems-89a76505a7. [Accessed 3 Mar 2024].

[26] "Neo4j," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Neo4j. [Accessed 10 Jun 2023].

[27] "Query a Neo4j database using Cypher," Neo4j, [Online]. Available: https://neo4j.com/docs/getting-started/cypher-intro/. [Accessed 10 Mar 2024].

[28] "Build applications with Neo4j and Python," Neo4j, [Online]. Available: https://neo4j.com/docs/python-manual/current/. [Accessed 10 Mar 2024].

[29] "Neo4j Graph Data Science," Neo4j, [Online]. Available: https://neo4j.com/product/graph-data-science/. [Accessed 10 Mar 2024].

[30] "Knowledge Graphs," Neo4j, [Online]. Available: https://neo4j.com/use-cases/knowledge-graph/. [Accessed 10 Mar 2024].

[31] "Neo4j Browser," Neo4j, [Online]. Available: https://neo4j.com/docs/browser-manual/current/. [Accessed 10 Mar 2024].

[32] "Neo4j Bloom," Neo4j, [Online]. Available: https://neo4j.com/product/bloom/. [Accessed 10 Mar 2024].

[33] B. Rocca, "Introduction to recommender systems," Towards Data Science, [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada. [Accessed 10 Mar 2024].

[34] A. Roy, "Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering," Towards Data Science, 29 Jul 2020. [Online]. Available:

https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421. [Accessed 10 Mar 2024].

[35] M. W., "Similarity Measures in Data Science: Euclidean distance & Cosine similarity," ASEAN+3 Macroeconomic Research Office, 25 Sep 2022. [Online]. Available: https://www.linkedin.com/pulse/similarity-measures-data-science-euclidean-distance-cosine-wynn/. [Accessed 10 Mar 2024].

[36] C. Zhou, "Understanding Jaccard Similarity: A Powerful Tool for Data Analysis," 7 Sep 2023. [Online]. Available: https://medium.com/@conniezhou678/understanding-jaccard-similarity-a-powerful-tool-for-data-analysis-42abaaafd782. [Accessed 10 Mar 2024].

[37] A. Jain, "TF-IDF Vectorization with Cosine Similarity," 4 Feb 2024. [Online]. Available: https://medium.com/@anurag-jain/tf-idf-vectorization-with-cosine-similarity-eca3386d4423#:~:text=Cosine%20similarity%20ranges%20from%20%2D1,the%20product%20of%20their%20magnitudes. [Accessed 10 Mar 2024].

[38] "What are the pros and cons of using cosine similarity vs. Jaccard similarity for text analysis?," LinkedIn, [Online]. Available: https://www.linkedin.com/advice/3/what-pros-cons-using-cosine-similarity-vs. [Accessed 10 Mar 2024].

[39] "How do you choose between collaborative and content-based filtering for your recommender system?," LinkedIn, [Online]. Available: https://www.linkedin.com/advice/0/how-do-you-choose-between-collaborative-content-based. [Accessed 10 Mar 2024].

[40] C. Pinela, "Recommender Systems — User-Based and Item-Based Collaborative Filtering," 6 Nov 2017. [Online]. Available: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f. [Accessed 3 Mar 2024].

[41] S. Li, "How Did We Build Book Recommender Systems in An Hour Part 2 — k Nearest Neighbors and Matrix Factorization," Towards Data Science, 20 Sep 2017. [Online]. Available: https://towardsdatascience.com/how-did-we-build-book-recommender-systems-in-an-hour-part-2-k-nearest-neighbors-and-matrix-c04b3c2ef55c. [Accessed 10 Mar 2024].

[42] A. O. Memetoglu, "4-) Collaborative Filtering and KNN," 31 May 2022. [Online]. Available: https://medium.com/@aliozan_memetoglu/4-collaborative-filtering-and-knn-f997f8993256. [Accessed 10 Mar 2024].

[43] "Fast Random Projection," Neo4j, [Online]. Available: https://neo4j.com/docs/graph-data-science/current/machine-learning/node-embeddings/fastrp/. [Accessed 10 Mar 2024].

[44] Xiangyu Zhao, Maolin Wang, Xinjian Zhao, Jiansheng Li, Shucheng Zhou, Dawei Yin, Qing Li, Jiliang Tang, Ruocheng Guo, "Embedding in Recommender Systems: A Survey," Xinjian Zhao, 2023.

[45] A. Bonnet, "What is Ensemble Learning?," 24 Nov 2023. [Online]. Available: https://encord.com/blog/what-is-ensemble-learning/#:~:text=Ensemble%20learning%20is%20a%20powerful,and%20tastes%20of%20different%20users.. [Accessed 10 Mar 2024].

[46] Ammar Mohammed, Rania Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," Journal of King Saud University - Computer and Information Sciences, vol. 35, no. 2, pp. 757-774, 2023.

[47] "How can you test the accuracy of a recommendation system?," LinkedIn, [Online]. Available: https://www.linkedin.com/advice/1/how-can-you-test-accuracy-recommendation-nnore. [Accessed 10 Mar 2024].

[48] M. Malaeb, "Recall and Precision at k for Recommender Systems," 14 Aug 2017. [Online]. Available: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54. [Accessed 10 Mar 2024].

[49] P. Huilgol, "Precision and Recall | Essential Metrics for Machine Learning (2024 Update)," Analytics Vidhya, 21 Dec 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/. [Accessed 10 Mar 2024].

[50] "Coverage," UiPath, [Online]. Available: https://support.reinfer.io/support/solutions/articles/48001167109-coverage#:~:text=Coverage%20is%20a%20term%20frequently,Validation%20as%20a%20percentage%20score. [Accessed 10 Mar 2024].

[51] " The F1 score," Joos K, [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6. [Accessed 24 Mar 2024].

[52] P. Nicoletti, "Chapter 42 - Content Filtering," Computer and Information Security Handbook, pp. 723-744, 2009.

[53] I. Mebsout, "Multivariate matching engine on Textual, Categorical and Numerical data," Towards Data Science, 20 Oct 2022. [Online]. Available: https://towardsdatascience.com/multivariable-matching-engine-on-textual-categorical-and-numerical-data-d90b0dca7f4c. [Accessed 10 Mar 2024].

[54] H. Singh, "Data Normalization for Numeric features," Analytics Vidhya, 27 Dec 2019. [Online]. Available: https://medium.com/analytics-vidhya/data-transformation-for-numeric-features-fb16757382c0. [Accessed 10 Mar 2024].

[55] "What are the advantages of feature scaling in ML?," LinkedIn, [Online]. Available: https://www.linkedin.com/advice/3/what-advantages-feature-scaling-ml-skills-machine-learning-

lvfne#:~:text=Min%2DMax%20Scaling%3A%20This%20method,ranging%20from%200%20to%20100%2C000. [Accessed 10 Mar 2024].

[56] S. Saxena, "What are Categorical Data Encoding Methods | Binary Encoding," Analytics Vidhya, 24 Sep 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/. [Accessed 10 Mar 2024].

[57] I. D. Baruah, "All you need to know about encoding techniques!," ANOLYTICS, 30 Sep 2023. [Online]. Available: https://medium.com/anolytics/all-you-need-to-know-about-encoding-techniques-b3a0af68338b#:~:text=One%2Dhot%20encoding%20is%20the,category%20as%20a%20binary%20vector. [Accessed 10 Mar 2024].

[58] S. Gupta, "“One-Hot Encoding: A Comprehensive Guide with Python Code and Examples for Effective Categorical Data Representation”," 2 Jul 2023. [Online]. Available: https://medium.com/@creatorvision03/one-hot-encoding-a-comprehensive-guide-with-python-code-and-examples-for-effective-categorical-2fbbc111c320. [Accessed 10 Mar 2024].

[59] C. GOYAL, "Part 5: Step by Step Guide to Master NLP – Word Embedding and Text Vectorization," Analytics Vidhya, 22 Jun 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/. [Accessed 10 Mar 2024].

[60] P. R. Ganjeer, "Text to Numerical Vector Conversion Techniques," Analytics Vidhya, 28 Jun 2021. [Online]. Available: https://medium.com/analytics-vidhya/text-to-numerical-vector-conversion-techniques-f2b97ab9b895. [Accessed 10 Mar 2024].

[61] S. Eskandar, "Exploring Feature Extraction Techniques for Natural Language Processing," 26 Apr 2023. [Online]. Available: https://medium.com/@eskandar.sahel/exploring-feature-extraction-techniques-for-natural-language-processing-46052ee6514. [Accessed 10 Mar 2024].

[62] M. Harmouch, "17 types of similarity and dissimilarity measures used in data science.," Towards Data Science, 14 Mar 2021. [Online]. Available: https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681. [Accessed 10 Mar 2024].

[63] F. Karabiber, "What is Jaccard Similarity?," LearnDataSci, [Online]. Available: https://www.learndatasci.com/glossary/jaccard-similarity/. [Accessed 10 Mar 2024].

[64] T. Davi, "Understanding Different Distance Measures," 14 Jun 2023. [Online]. Available: https://www.linkedin.com/pulse/understanding-different-distance-measures-tiago-davi-1f/. [Accessed 10 Mar 2023].

[65] Jiawei Han, Micheline Kamber, Jian Pei, "2 - Getting to Know Your Data," in Data Mining (Third Edition), 2012, pp. 39-82.

[66] S. Prabhakaran, "Cosine Similarity – Understanding the math and how it works (with python codes)," Machine Learning Plus, [Online]. Available: https://www.machinelearningplus.com/nlp/cosine-similarity/. [Accessed 10 Mar 2024].

[67] "sklearn.feature_extraction.text.CountVectorizer," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed 10 Mar 2024].

[68] Rakesh4real, "User-Based and Item-Based Collaborative Filtering — Part 5," Fnplus Club, 29 Jul 2019. [Online]. Available: https://medium.com/fnplus/user-based-and-item-based-collaborative-filtering-b73d9b2badba. [Accessed 10 Mar 2024].

[69] T. Srivastava, "A Complete Guide to K-Nearest Neighbors (Updated 2024)," 4 Jan 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=The%20K%2DNearest%20Neighbors%20(KNN)%20algorithm%20is%20a%20popular,training%20dataset%20as%20a%20reference. [Accessed 10 Mar 2024].

[70] Amy, "Recommendation System: User-Based Collaborative Filtering," GrabNGoInfo, 16 Apr 2022. [Online]. Available: https://medium.com/grabngoinfo/recommendation-system-user-based-collaborative-filtering-a2e76e3e15c4. [Accessed 10 Mar 2024].

[71] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering University of Minnesota, 2001.

[72] C. Sullivan, "Behind the scenes on the Fast Random Projection algorithm for generating graph embeddings," Towards Data Science, 19 Aug 2021. [Online]. Available: https://towardsdatascience.com/behind-the-scenes-on-the-fast-random-projection-algorithm-for-generating-graph-embeddings-efb1db0895. [Accessed 10 Mar 2024].

[73] Xiangyu Zhao, Maolin Wang, Xinjian Zhao, Jiansheng Li, Shucheng Zhou, Dawei Yin, Qing Li, Jiliang Tang, and Ruocheng Guo, "Embedding in Recommender Systems: A Survey," 2023.

[74] "Product recommendations with kNN based on FastRP embeddings," Neo4j, [Online]. Available: https://neo4j.com/docs/graph-data-science-client/current/tutorials/fastrp-and-knn/. [Accessed 10 Mar 2024].

[75] S. Mudadla, "Data Science Interview Questions on related to the K-Nearest Neighbors (KNN).," 29 Oct 2023. [Online]. Available: https://medium.com/@sujathamudadla1213/data-science-interview-questions-on-related-to-the-k-nearest-neighbors-knn-939f954e0bd0. [Accessed 10 Mar 2024].

[76] "Getting started," Neo4j, [Online]. Available: https://neo4j.com/docs/graph-data-science-client/current/getting-started/. [Accessed 10 Mar 2024].

[77] "Node embeddings," Neo4j, [Online]. Available: https://neo4j.com/docs/graph-data-science/current/machine-learning/node-embeddings/. [Accessed 10 Mar 2024].

[78] S. Agrawal, "Hyperparameter Tuning of KNN Classifier," 5 Jun 2023. [Online]. Available: https://medium.com/@agrawalsam1997/hyperparameter-tuning-of-knn-classifier-a32f31af25c7. [Accessed 10 Mar 2024].

[79] "Guide to Simple Ensemble Learning Techniques," Data Science Wizards, 6 Jun 2023. [Online]. Available: https://medium.com/@datasciencewizards/guide-to-simple-ensemble-learning-techniques-2ac4e2504912#:~:text=Max%20or%20majority%20voting%20techniques,votes%20as%20the%20final%20prediction. [Accessed 10 Mar 2024].

[80] "Introduction to Ensemble Learning Methods," LinkedIn, 23 Oct 2023. [Online]. Available: https://www.linkedin.com/pulse/introduction-ensemble-learning-methods-data-and-analytics-magazin/. [Accessed 10 Mar 2024].

[81] C. S. Mullins, "Scalability and Performance: Different but Crucial Database Management Capabilities," 18 Dec 2023. [Online]. Available: https://www.dbta.com/Editorial/Think-About-It/Scalability-and-Performance-Different-but-Crucial-Database-Management-Capabilities-161866.aspx#:~:text=Database%20scalability%20refers%20to%20the,handle%20increased%20demand%20over%20time. [Accessed 17 Mar 2024].

[82] "Usability," Interaction Design Foundation, [Online]. Available: https://www.interaction-design.org/literature/topics/usability. [Accessed 17 Mar 2024].

[83] S. Matade, "Code Reviews : Best Practices for Software Development," 11 Jun 2023. [Online]. Available: https://medium.com/@shreyasmatade/code-reviews-best-practices-for-software-development-da58a112faa0. [Accessed 17 Mar 2024].

[84] M. Joshi, "The Art of System Design: A Journey with Jinja, Flask, and More!," 3 Jan 2024. [Online]. Available: https://www.linkedin.com/pulse/art-system-design-journey-jinja-flask-more-manish-joshi-tdkvc/. [Accessed 17 Mar 2024].

[85] Sadika, "The MVC Architecture," 19 Sep 2023. [Online]. Available: https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2. [Accessed 17 Mar 2024].

[86] Ryan, "The Perfect Blend of Bootstrap and UI/UX: Crafting Exceptional Digital Experiences," 27 Dec 2023. [Online]. Available: https://webplanex.medium.com/the-perfect-blend-of-bootstrap-and-ui-ux-crafting-exceptional-digital-experiences-29cd4ce3bee9. [Accessed 17 Mar 2024].

[87] A. Dyouri, "How To Use Templates in a Flask Application," DigitalOcean, 14 Sep 2021. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-use-templates-in-a-flask-application. [Accessed 17 Mar 2024].

[88] Harisudhan.S, "Flask : A Web Framework," 9 Mar 2024. [Online]. Available: https://medium.com/@speaktoharisudhan/flask-a-web-framework-435de255e81b. [Accessed 17 Mar 2024].

[89] S. Mudadla, "What kind of static files we can include in Flask?," 25 Nov 2023. [Online]. Available: https://medium.com/@sujathamudadla1213/what-kind-of-static-files-we-can-include-in-flask-4a80080e6023. [Accessed 17 Mar 2024].

[90] J. T, "Flask User Auth with Neo4j," TheStartup, 30 Jun 2020. [Online]. Available: https://medium.com/swlh/flask-user-auth-with-neo4j-cca79aa90f57. [Accessed 17 Mar 2024].

[91] "How do you align testing with project objectives in SDLC?," LinkedIn, [Online]. Available: https://www.linkedin.com/advice/3/how-do-you-align-testing-project-objectives#:~:text=The%20first%20step%20to%20align,improving%20performance%2C%20or%20enhancing%20security. [Accessed 17 Mar 2024].

[92] S. PITTET, "The different types of software testing," Atlassian, [Online]. Available: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing. [Accessed 17 Mar 2024].

[93] "What Is a Gantt Chart? (Examples & Templates)," ProjectManager, [Online]. Available: https://www.projectmanager.com/guides/gantt-chart#:~:text=The%20Gantt%20chart%20timeline%20is,longer%20the%20task%20will%20take.. [Accessed 17 Mar 2024].

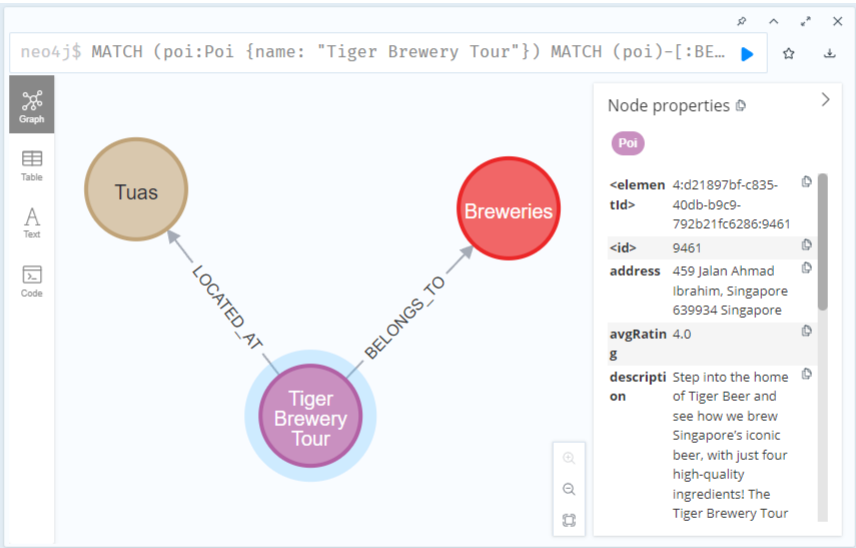
Appendix A: Knowledge Graph Queries

1. Relevant Information about POI

This query provides comprehensive details about a specific point of interest (POI), including its category and location. This enables users to understand the context and attributes associated with the POI.

1) Find relevant information about POI.

Query	`MATCH (poi:Poi {name: "Tiger Brewery Tour"})`
	`MATCH (poi)-[:BELONGS_TO]->(category:Category)`
	`MATCH (poi)-[:LOCATED_AT]->(region:Region)`
	`RETURN poi,category,region`

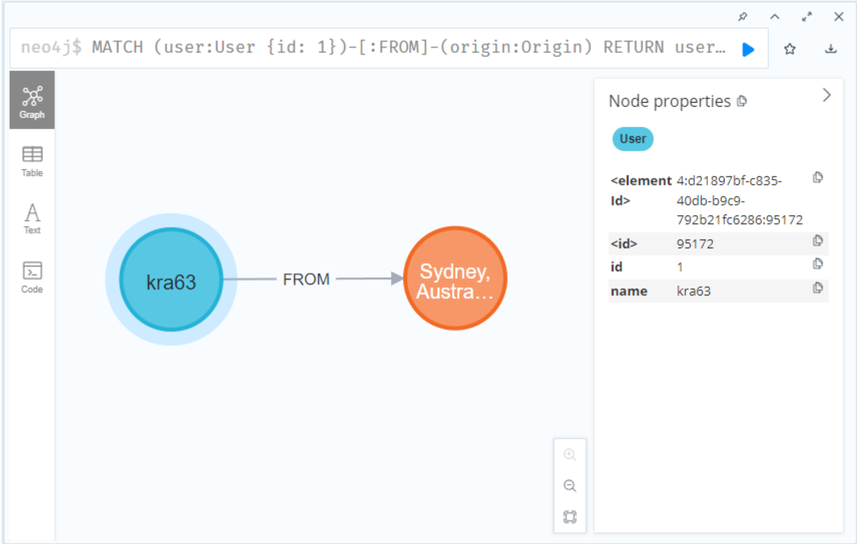
Result	

2. Relevant Information about a User

By retrieving information about a particular user, such as their ID and origin, it offers insights into user demographics, aiding in personalized recommendations and targeted marketing strategies.

1) Find relevant information about a user.

Query	`MATCH (user:User {id: 1})-[:FROM]-(origin:Origin)` `RETURN user,origin`

<table>
<tr><td>Result</td><td>
</td></tr>
</table>

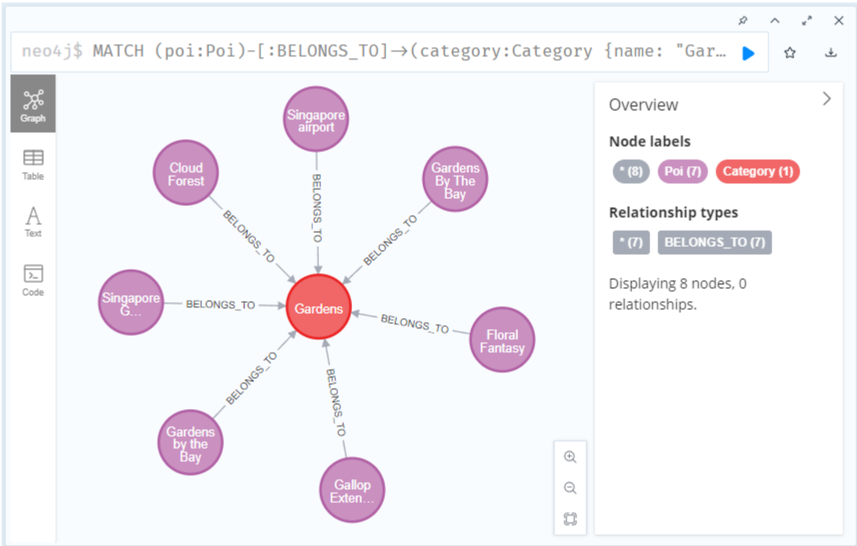
3. Category-based Offerings

Identifying POIs that offer specific experiences, such as gardens, historical museums, or scenic landscapes, facilitates itinerary planning and caters to the diverse interests of visitors.

1) Find points of interest (POI) that offer "Gardens" experiences.

<table>
<tr><td>Query</td><td>

```
MATCH (poi:Poi)-[:BELONGS_TO]->(category:Category {name: "Gardens"})
RETURN poi, category
```

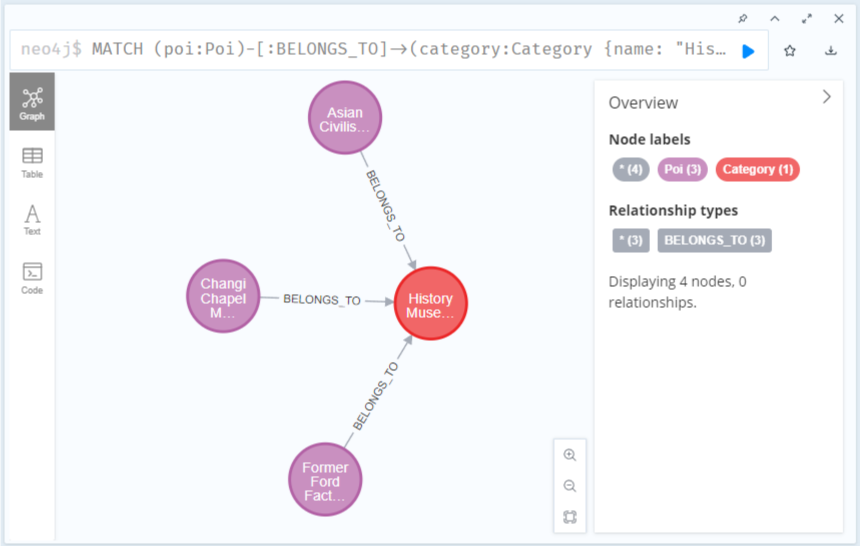
</td></tr>
<tr><td>Result</td><td>
</td></tr>
</table>

2) List historical museums that offer insights into Singapore's history.

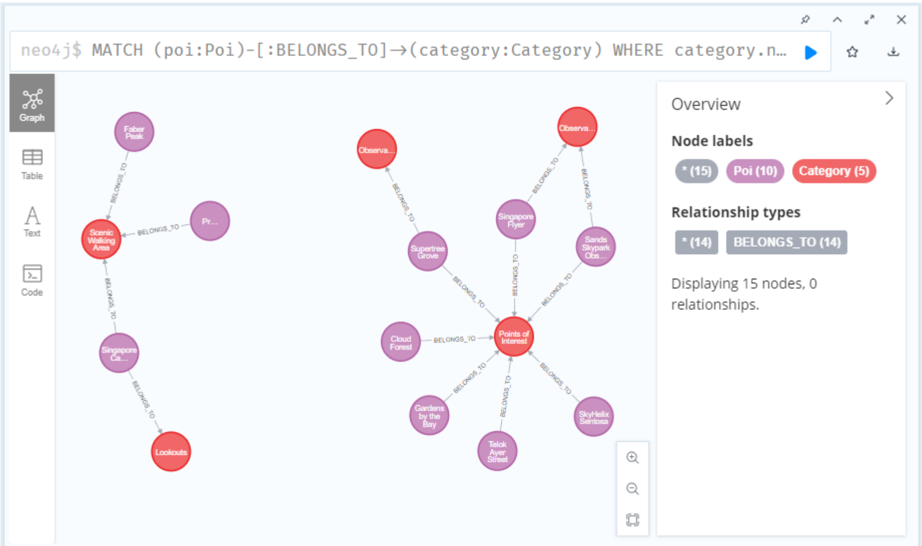
<table>
<tr><td>Query</td><td>

```
MATCH (poi:Poi)-[:BELONGS_TO]->(category:Category {name: "History
Museums"})
```

</td></tr>
</table>

	RETURN poi, category
Result	

3) List points of interest (POI) with scenic landscapes, or iconic landmarks for photography enthusiasts and sightseeing.

Query	MATCH (poi:Poi)-[:BELONGS_TO]->(category:Category) WHERE category.name IN ["Points of Interest & Landmarks", "Lookouts", "Observation Decks & Towers", "Observatories & Planetariums", "Scenic Walking Areas"] RETURN DISTINCT poi, category
Result	

4. Region-based Insights

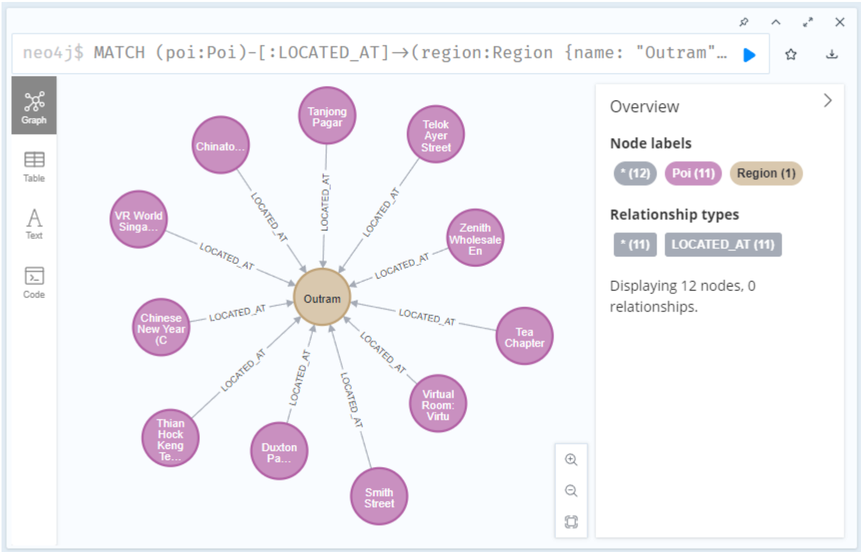
Listing all POIs within a specific region, such as "Outram," allows for region-specific analysis, highlighting popular tourist zones and distribution patterns of POIs across different areas.

1) List all points of interest (POI) in Region "Outram".

<table>
<tr><td>Query</td><td>

```
MATCH (poi:Poi)-[:LOCATED_AT]->(region:Region {name: "Outram"})
RETURN poi, region
```

</td></tr>
<tr><td>Result</td><td>



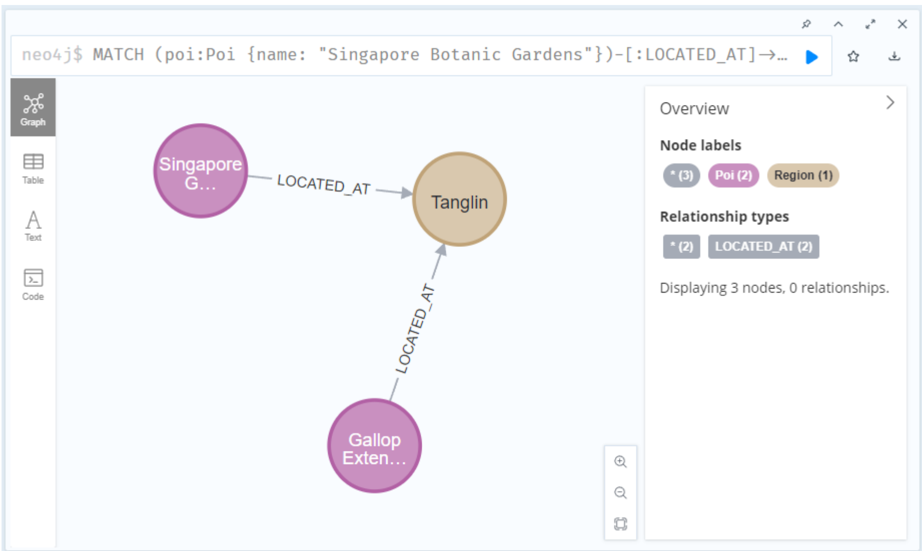
</td></tr>
</table>

2) List all the points of interest (POI) that are in the same region as "Singapore Botanic Gardens".

<table>
<tr><td>Query</td><td>

```
MATCH (poi:Poi {name: "Singapore Botanic Gardens"})-
[:LOCATED_AT]->(region:Region)
WITH region
MATCH (poi:Poi)-[:LOCATED_AT]->(region:Region)
RETURN poi, region
```

</td></tr>
<tr><td>Result</td><td>



</td></tr>
</table>

5. Price Range Analysis

Understanding the price distribution of POI within a specific range, such as SGD 10 to SGD 20, assists in budget planning for travellers and provides insights into the affordability of tourist experiences.

1) List points of interest (POI) with pricing information within a price range from SGD 10 to SGD 20.

<table>
<tr><td>Query</td><td>MATCH (poi:Poi)
WHERE poi.price >= 10 AND poi.price <= 20
RETURN poi.name AS PoiName, poi.price AS PoiPrice</td></tr>
<tr><td>Result</td><td>neo4j$ MATCH (poi:Poi) WHERE poi.price ≥ 10 AND poi.price ≤ 20 RET…
<table>
<tr><th></th><th>PoiName</th><th>PoiPrice</th></tr>
<tr><td>1</td><td>"Cloud Forest"</td><td>11.42</td></tr>
<tr><td>2</td><td>"Asian Civilisations Museum"</td><td>12.01</td></tr>
<tr><td>3</td><td>"Changi Experience Studio"</td><td>17.09</td></tr>
<tr><td>4</td><td>"Floral Fantasy"</td><td>11.42</td></tr>
<tr><td>5</td><td>"SkyHelix Sentosa"</td><td>13.7</td></tr>
</table>
Started streaming 5 records after 10 ms and completed after 11 ms.</td></tr>
</table>

2) List the reference price of "Singapore Zoo".

<table>
<tr><td>Query</td><td>MATCH (poi:Poi {name: "Singapore Zoo"})
RETURN poi.name AS PoiName, poi.price AS PoiPrice</td></tr>
<tr><td>Result</td><td>neo4j$ MATCH (poi:Poi {name: "Singapore Zoo"}) RETURN poi.name AS Po…
<table>
<tr><th></th><th>PoiName</th><th>PoiPrice</th></tr>
<tr><td>1</td><td>"Singapore Zoo"</td><td>33.49</td></tr>
</table>
Started streaming 1 records after 9 ms and completed after 11 ms.</td></tr>
</table>

6. Duration of Visit Analysis

Identifying POIs with varying visit durations, such as 1 to 2 hours, helps in optimizing travel itineraries and allocating time efficiently during sightseeing trips.

1) List points of interest (POI) that only take 1 to 2 hours to visit.

Query

```
MATCH (poi:Poi {duration:"1-2 hours"})
RETURN poi.name AS PoiName, poi.duration AS PoiDuration
```

Result

neo4j$ MATCH (poi:Poi {duration:"1-2 hours"}) RETURN poi.name AS PoiName, …

	PoiName	PoiDuration
1	"Singapore Botanic Gardens"	"1-2 hours"
2	"Singapore Flyer"	"1-2 hours"
3	"Sands Skypark Observation Deck"	"1-2 hours"
4	"Supertree Grove"	"1-2 hours"
5	"Chinatown"	"1-2 hours"
6	"Merlion Park"	"1-2 hours"
7		

Started streaming 14 records after 10 ms and completed after 11 ms.

7. Rating Insights

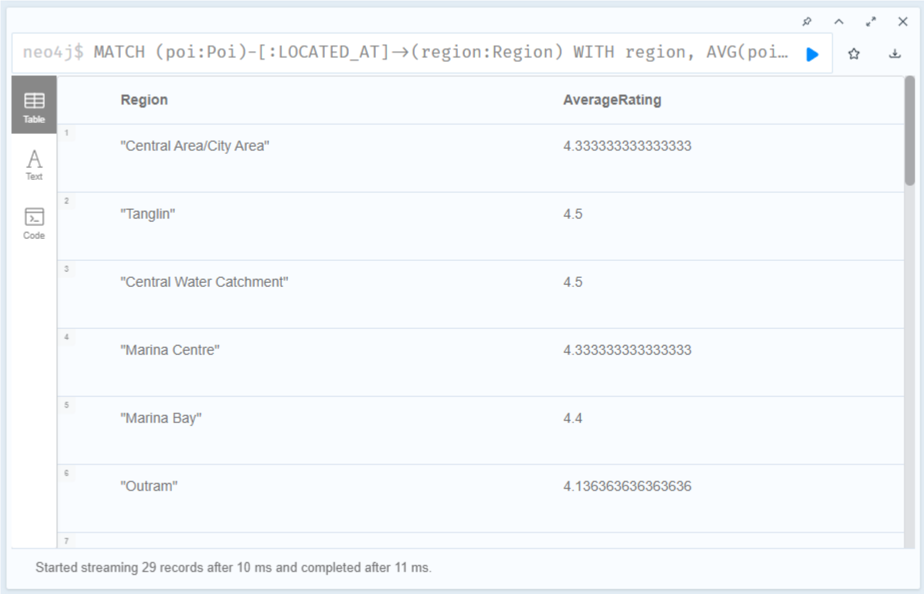
Analyzing the average rating of POIs enables the identification of highly-rated POIs, contributing to visitor satisfaction and guiding future decision-making for tourism development and marketing efforts.

1) Find the Average Rating for POIs in Each Region

Query

```
MATCH (poi:Poi)-[:LOCATED_AT]->(region:Region)
WITH region, AVG(poi.avgRating) AS avgRating
RETURN region.name AS Region, avgRating AS AverageRating
```

Result



neo4j$ MATCH (poi:Poi)-[:LOCATED_AT]→(region:Region) WITH region, AVG(poi…

	Region	AverageRating
1	"Central Area/City Area"	4.333333333333333
2	"Tanglin"	4.5
3	"Central Water Catchment"	4.5
4	"Marina Centre"	4.333333333333333
5	"Marina Bay"	4.4
6	"Outram"	4.136363636363636
7		

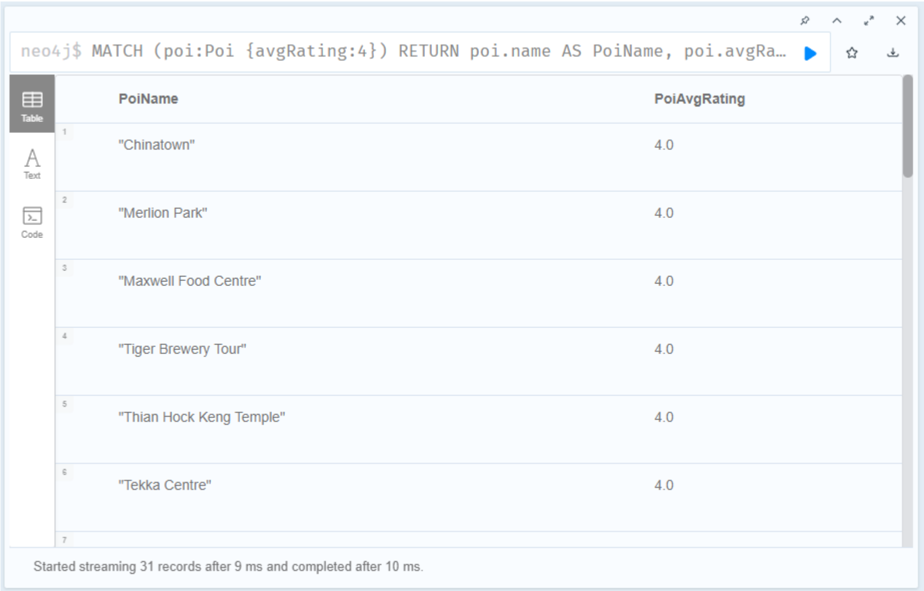
Started streaming 29 records after 10 ms and completed after 11 ms.

2) List points of interest (POI) that have an average rating of 4

Query

```
MATCH (poi:Poi {avgRating:4})
RETURN poi.name AS PoiName, poi.avgRating AS PoiAvgRating
```

Result



neo4j$ MATCH (poi:Poi {avgRating:4}) RETURN poi.name AS PoiName, poi.avgRa…

	PoiName	PoiAvgRating
1	"Chinatown"	4.0
2	"Merlion Park"	4.0
3	"Maxwell Food Centre"	4.0
4	"Tiger Brewery Tour"	4.0
5	"Thian Hock Keng Temple"	4.0
6	"Tekka Centre"	4.0
7		

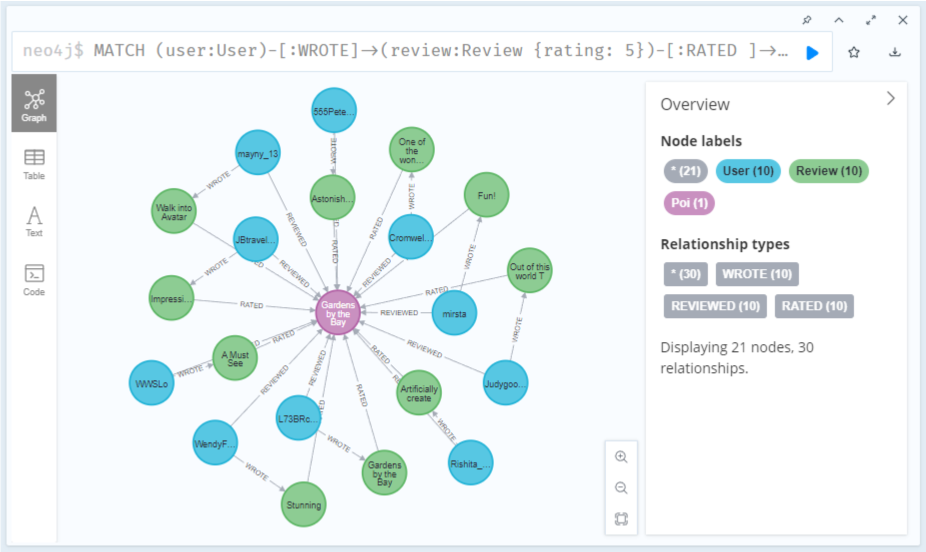
Started streaming 31 records after 9 ms and completed after 10 ms.

8. User Engagement Analysis

Discovering users who have written reviews with a specific rating or visited POIs within a particular category offers insights into user engagement levels and preferences, facilitating targeted engagement strategies.

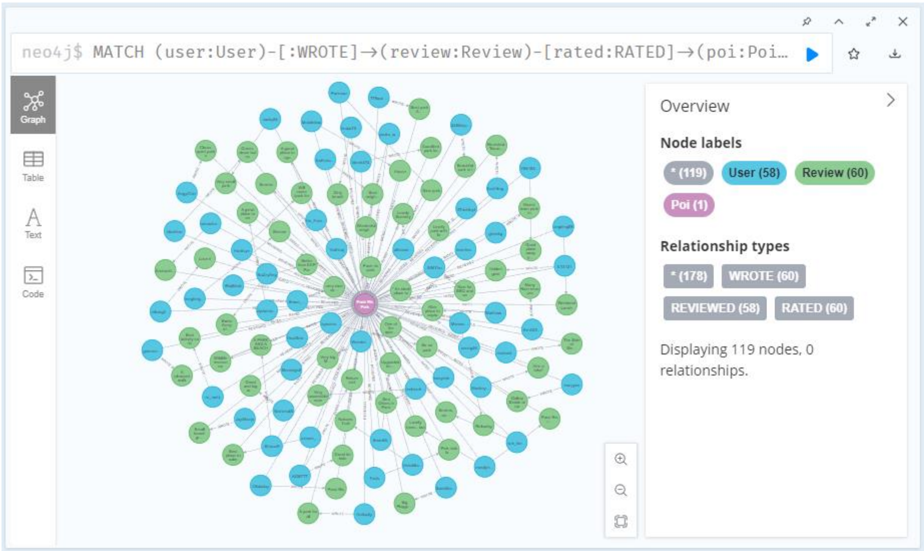
1) Find 10 Users who Wrote Reviews with a Rating of 5

Query	`MATCH (user:User)-[:WROTE]->(review:Review {rating: 5})-`
	`[:RATED ]->(poi:Poi)`
	`RETURN user, review, poi`
	`LIMIT 10`

Result	

2) Find Users who Reviewed a Specific POI

Query	`MATCH (user:User)-[:WROTE]->(review:Review)-[rated:RATED]->(poi:Poi`
	`{name: 'Pasir Ris Park'})`
	`RETURN user, review, poi`

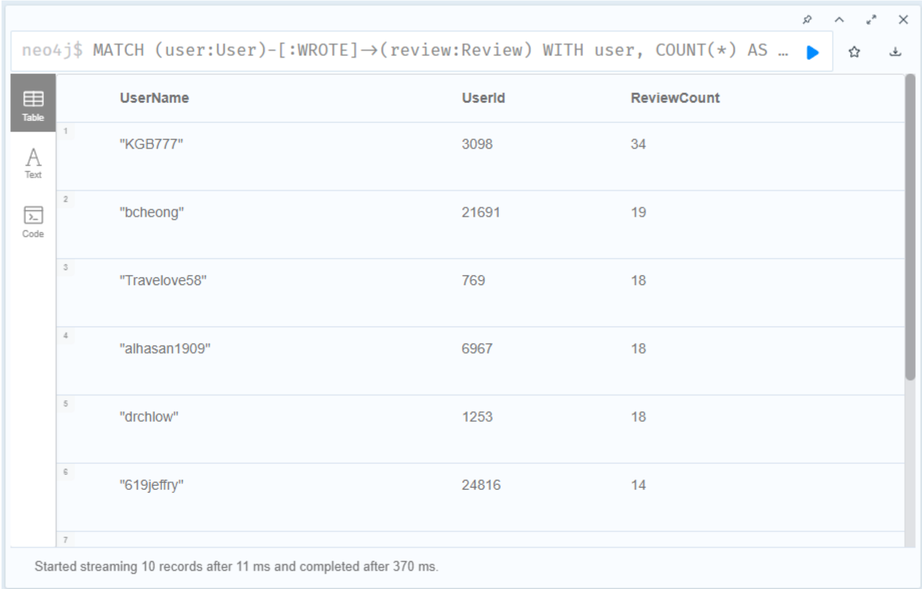
Result	

3) Find the Top 10 Active Users with Most Reviews

Query	`MATCH (user:User)-[:WROTE]->(review:Review)` `WITH user, COUNT(*) AS reviewCount`

```
ORDER BY reviewCount DESC
RETURN user.name AS UserName, user.id AS UserId, reviewCount AS
ReviewCount
LIMIT 10
```

Result



neo4j$ MATCH (user:User)-[:WROTE]→(review:Review) WITH user, COUNT(*) AS …

	UserName	UserId	ReviewCount
1	"KGB777"	3098	34
2	"bcheong"	21691	19
3	"Travelove58"	769	18
4	"alhasan1909"	6967	18
5	"drchlow"	1253	18
6	"619jeffry"	24816	14
7			

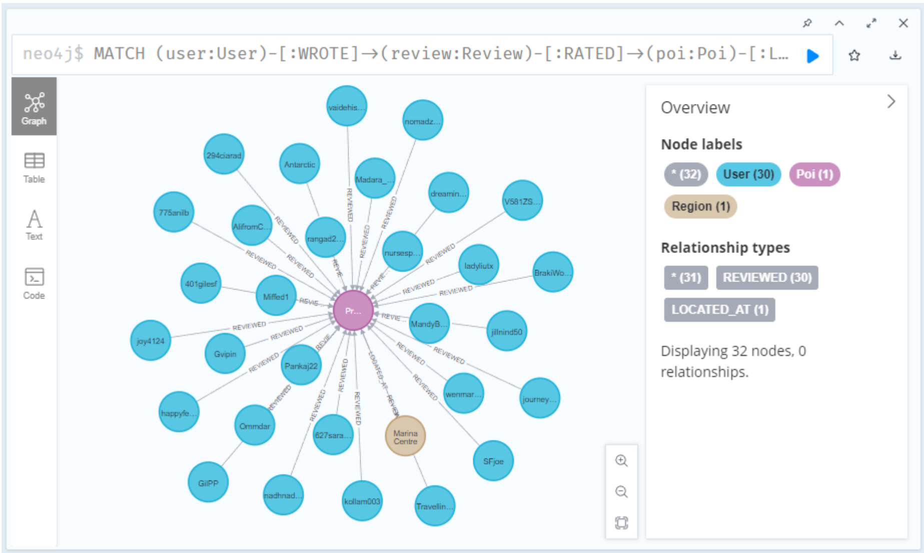
Started streaming 10 records after 11 ms and completed after 370 ms.

4) Find Users who Reviewed POIs in the Region “Marina Centre”

Query

```
MATCH (user:User)-[:WROTE]->(review:Review)-[:RATED]->(poi:Poi)-
[:LOCATED_AT]->(region:Region {name: 'Marina Centre'})
RETURN user, poi, region
LIMIT 30
```

Result



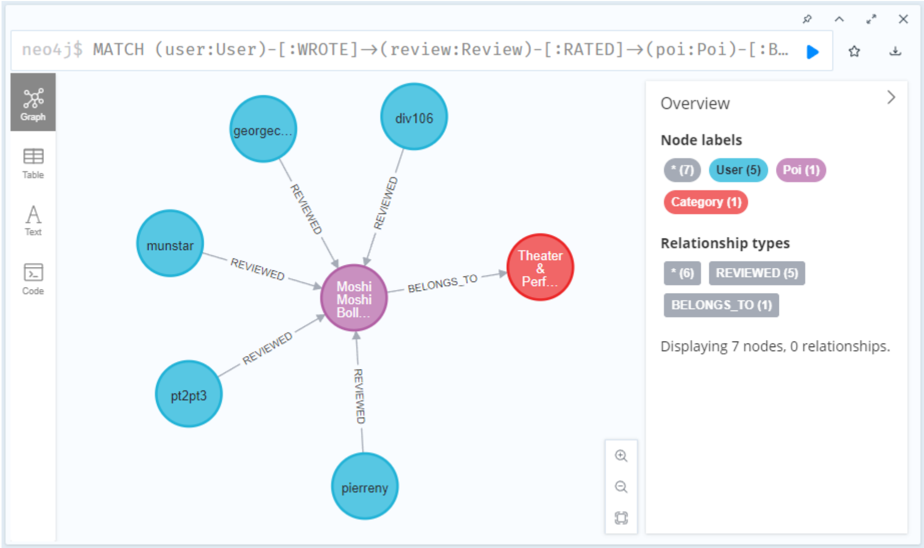
5) Find Users who Visited POIs in a Specific Category "Theater & Performances"

Query

```
MATCH (user:User)-[:WROTE]->(review:Review)-[:RATED]->(poi:Poi)-
[:BELONGS_TO]->(category:Category {name: 'Theater & Performances'})
```

```
RETURN user, poi, category
```

Result



9. Review Distribution

Examining the distribution of reviews per rating category provides a holistic view of visitor feedback and sentiment towards POI, guiding improvements and quality assurance initiatives.

Discovering the top 10 active users with the most reviews offers insights into user behaviour and engagement patterns, aiding in user segmentation and personalized marketing strategies.

1) Find the Number of Reviews per Rating

Query

```
MATCH (review:Review)
RETURN review.rating AS Rating, COUNT(*) AS ReviewCount
ORDER BY Rating
```

Result

neo4j$ MATCH (review:Review) RETURN review.rating AS Rating, COUNT(*) AS R…

	Rating	ReviewCount
1	2.0	7
2	3.0	4676
3	4.0	1225
4	5.0	79721

Started streaming 4 records after 8 ms and completed after 217 ms.

2) Find POIs with the Highest Number of Reviews

Query	MATCH (poi:Poi)<-[rated:RATED]-(:Review) WITH poi, COUNT(rated) AS numReviews ORDER BY numReviews DESC RETURN poi.name AS PoiName, numReviews AS NumReviews
Result	

	PoiName	NumReviews
1	"Gardens by the Bay"	21381
2	"Singapore Mass Rapid Transit (SMRT)"	18064
3	"Singapore Zoo"	7453
4	"Singapore Botanic Gardens"	6887
5	"Singapore Flyer"	5324
6	"Cloud Forest"	5126
7		

Started streaming 56 records after 10 ms and completed after 426 ms.

10. Top-rated POIs

Identifying the top-rated POI based on user reviews highlights the most popular and highly recommended destinations, serving as valuable insights for tourists and tourism stakeholders.

1) Find the Top 10 Top-Rated POIs

Query	MATCH (poi:Poi)<-[r:RATED]-(review:Review) WITH poi, AVG(review.rating) AS avgRating ORDER BY avgRating DESC RETURN poi.name AS PoiName, avgRating AS AverageRating LIMIT 10

Result	

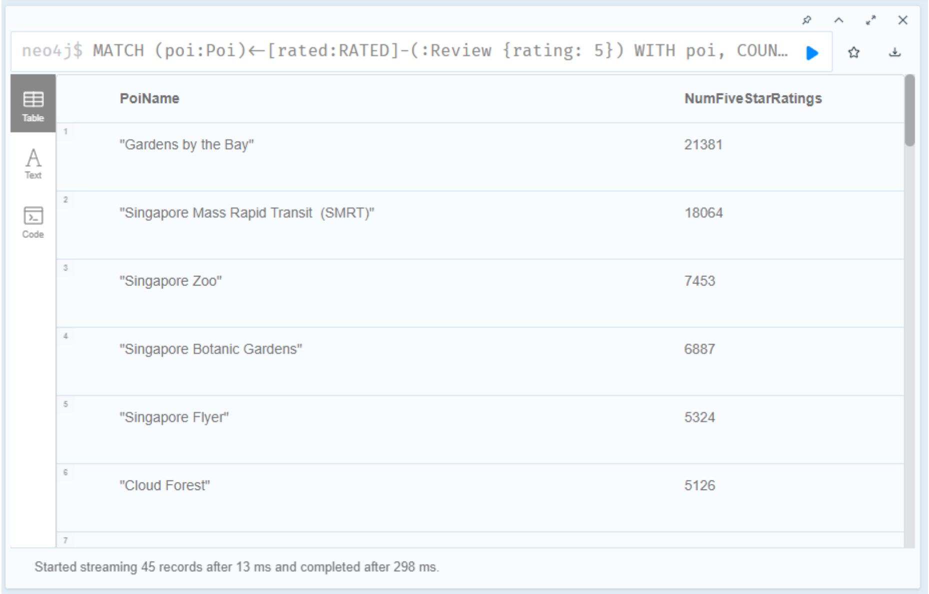
neo4j$ MATCH (poi:Poi)←[r:RATED]-(review:Review) WITH poi, AVG(review.rat…

	PoiName	AverageRating
1	"Gardens by the Bay"	5.0
2	"Cloud Forest"	5.0
3	"Singapore Botanic Gardens"	5.0
4	"Singapore Flyer"	5.0
5	"Sands Skypark Observation Deck"	5.0
6	"Supertree Grove"	5.0
7		

Started streaming 10 records after 13 ms and completed after 733 ms.

2) Find POIs with the Highest Number of 5-Star Ratings

Query	`MATCH (poi:Poi)<-[rated:RATED]-(:Review {rating: 5})`
	`WITH poi, COUNT(rated) AS numFiveStarRatings`
	`ORDER BY numFiveStarRatings DESC`
	`RETURN poi.name AS PoiName, numFiveStarRatings AS NumFiveStarRatings`

Result	

neo4j$ MATCH (poi:Poi)←[rated:RATED]-(:Review {rating: 5}) WITH poi, COUN…

	PoiName	NumFiveStarRatings
1	"Gardens by the Bay"	21381
2	"Singapore Mass Rapid Transit (SMRT)"	18064
3	"Singapore Zoo"	7453
4	"Singapore Botanic Gardens"	6887
5	"Singapore Flyer"	5324
6	"Cloud Forest"	5126
7		

Started streaming 45 records after 13 ms and completed after 298 ms.

Appendix B: Performance Evaluation of Ensemble Learning with all Combinations of Algorithms

1. Ensemble Learning with Algorithms 1,2,3,4

- **Precision Score:** 0.410
- **Recall Score:** 0.308
- **Coverage Score:** 0.681
- **F1 Score:** 0.352

This combination attains intermediate precision and recall, with high scores in coverage metrics, and a relatively higher F1 score. This balanced performance underscores its effectiveness in recommending relevant items while accommodating diverse preferences. Therefore, this combination emerges as a favourable choice.

2. Ensemble Learning with Algorithms 2,3,4

- **Precision Score:** 0.499
- **Recall Score:** 0.120
- **Coverage Score:** 0.522
- **F1 Score:** 0.193

This combination exhibits relatively higher precision but comes with lower recall scores, with a low F1 score suggesting imbalanced performance. Consequently, it may not be suitable for discovering a broader range of relevant items.

3. Ensemble Learning with Algorithms 1,3,4

- **Precision Score:** 0.610
- **Recall Score:** 0.268
- **Coverage Score:** 0.420
- **F1 Score:** 0.372

This combination demonstrates relatively high precision, signifying its effectiveness in recommending highly relevant items. The recall and coverage remain balanced with a relatively high F1 score. Thus, this combination is also a suitable choice.

4. Ensemble Learning with Algorithms 1,2,4

- **Precision Score:** 0.162
- **Recall Score:** 0.058

- **Coverage Score:** 0.536
- **F1 Score:** 0.086

This combination shows better coverage, suggesting its capacity to retrieve a larger proportion of relevant items. However, given its poor precision, extremely low recall and very low F1 score, this combination does not perform well overall.

5. Ensemble Learning with Algorithms 1,2,3

- **Precision Score:** 0.422
- **Recall Score:** 0.280
- **Coverage Score:** 0.667
- **F1 Score:** 0.336

This combination achieves a balanced performance, with a decent precision score, and relatively higher recall, coverage and F1 score, suggesting its effectiveness in providing relevant recommendations across a wide range of preferences. Thus, this is also a good choice.

6. Ensemble Learning with Algorithms 1,2

- **Precision Score:** 0.140
- **Recall Score:** 0.039
- **Coverage Score:** 0.464
- **F1 Score:** 0.061

Despite its lower precision, recall and F1 scores, this combination maintains a moderate coverage range, suggesting its capability to recommend a variety of items. However, due to its poor accuracy, this combination does not perform optimally.

7. Ensemble Learning with Algorithms 1,3

- **Precision Score:** 0.651
- **Recall Score:** 0.236
- **Coverage Score:** 0.377
- **F1 Score:** 0.347

This combination exhibits high precision, recall scores and relatively high F1 score, indicating its effectiveness in recommending relevant items. Additionally, its coverage falls within the upper range, making this combination a good choice.

8. Ensemble Learning with Algorithms 1,4

- **Precision Score:** 0.424
- **Recall Score:** 0.032
- **Coverage Score:** 0.101
- **F1 Score:** 0.059

This combination demonstrates extremely low recall scores and very low coverage and F1 scores, indicating poor performance.

9. Ensemble Learning with Algorithms 2,3

- **Precision Score:** 0.570
- **Recall Score:** 0.078
- **Coverage Score:** 0.450
- **F1 Score:** 0.138

This combination achieves balanced precision and coverage scores but exhibits a lower recall score and F1 score, suggesting that it retrieves a narrower range from the full recommendation list. Consequently, it may not be suitable for comprehensive recommendation tasks.

10. Ensemble Learning with Algorithms 2,4

- **Precision Score:** 0.254
- **Recall Score:** 0.022
- **Coverage Score:** 0.159
- **F1 Score:** 0.041

This combination demonstrates extremely low recall, coverage and F1 scores, coupled with relatively low precision, indicating that it is not a suitable choice for recommendation tasks.

11. Ensemble Learning with Algorithms 3,4

- **Precision Score:** 0.710
- **Recall Score:** 0.062
- **Coverage Score:** 0.174
- **F1 Score:** 0.114

This combination attains the highest precision score, but its recall and F1 scores are very low, with coverage slightly below other combinations. This suggests its effectiveness in recommending highly relevant items, albeit within a narrow scope.

Appendix C: Web Application Use Case Descriptions

Use Case 1: Browse Points of Interest (POIs)

Pre-Conditions:	User accesses the home page of the web application.
Primary Paths:	1. User navigates to the home page. 2. System displays a list of all POIs available in the database.
Alternative Path:	None

Use Case 2: View Details of a Point of Interest (POI)

Pre-Conditions:	User is on the home page of the web application.
Primary Paths:	1. User clicks on a specific POI from the list. 2. System retrieves and displays detailed information about the selected POI.
Alternative Path:	None

Use Case 3: View Personalized Recommendations

Pre-Conditions:	User is logged in and on the home page or POI detail page of the web application.
Primary Paths:	1. User navigates to the home page or POI detail page. 2. System generates and displays personalized recommendations based on user's past interactions and data of POIs.
Alternative Path:	None

Use Case 4: User Login

Pre-Conditions:	User accesses the login page of the web application.
Primary Paths:	1. User enters valid credentials and submits the login form. 2. System verifies the credentials and grants access to the user.
Alternative Path:	• If the user enters invalid credentials, system displays an error message and prompts the user to try again.

Use Case 5: View User Profile

Pre-Conditions:	User is logged in.
Primary Paths:	1. User clicks on the profile icon or navigates to the user profile page. 2. System retrieves and displays the user's profile information.
Alternative Path:	None

Use Case 6: User Logout

Pre-Conditions:	User is logged in.
Primary Paths:	1. User clicks on the logout button. 2. System logs the user out and redirects them to the previously visited page.
Alternative Path:	If user was on user profile page and click on logout button, redirect users to home page.

Appendix D: System Validation Test Cases and Results

1. Unit Testing

1) Recommender Module

Test Case 1

Test Case Title:	Evaluates response when only `poi_id` input is available.
Pre-Conditions:	None.
Test Steps:	1. Provide only the `poi_id` input. 2. Execute the recommender module.
Expected Results:	The recommender module should return recommendations based on the provided `poi_id`.
Actual Results:	Recommendations based on the provided `poi_id` are returned successfully.

Test Case 2

Test Case Title:	Evaluates response when only `user_id` input is available.
Pre-Conditions:	None.
Test Steps:	1. Provide only the `user_id` input. 2. Execute the recommender module.
Expected Results:	The recommender module should return recommendations based on the provided `user_id`.
Actual Results:	Recommendations based on the provided `user_id` are returned successfully.

Test Case 3

Test Case Title:	Evaluates response when both `poi_id` and `user_id` inputs are available.
Pre-Conditions:	None.
Test Steps:	1. Provide both `poi_id` and `user_id` inputs. 2. Execute the recommender module.
Expected Results:	The recommender module should return recommendations based on the provided `poi_id` and `user_id`.
Actual Results:	Recommendations based on the provided `poi_id` and `user_id` are returned successfully.

Test Case 4

Test Case Title:	Evaluates response when no input data is provided.
Pre-Conditions:	None.
Test Steps:	1. Do not provide any input data. 2. Execute the recommender module.
Expected Results:	The recommender module should handle the absence of input data gracefully and return an appropriate response.
Actual Results:	The recommender module handles the absence of input data gracefully and returns no result.

2) Data Loader Module

Test Case 5

Test Case Title:	Validates the process when the Neo4j database is initially initialized and empty.
Pre-Conditions:	Neo4j database is empty.
Test Steps:	1. Initialize the Neo4j database. 2. Execute the data loader module.
Expected Results:	The data loader module should successfully load data into the Neo4j database.
Actual Results:	The data loader module successfully loads data into the Neo4j database.

Test Case 6

Test Case Title:	Ensures proper handling when the Neo4j database already contains pre-loaded data.
Pre-Conditions:	Neo4j database contains pre-loaded data.
Test Steps:	1. Ensure Neo4j database contains pre-loaded data. 2. Execute the data loader module.
Expected Results:	The data loader module should not load duplicated data without causing errors.
Actual Results:	The data loader module does not load duplicated data.

3) Web Application Module

Test Case 7

Test Case Title:	Tests functionality of navigating to the home page.
Pre-Conditions:	Web application is running.
Test Steps:	1. Open the web application. 2. Navigate to the home page.
Expected Results:	The home page of the web application should be displayed correctly.
Actual Results:	The home page of the web application is displayed correctly.

Test Case 8

Test Case Title:	Ensures proper user login and authentication.
Pre-Conditions:	Web application is running.
Test Steps:	1. Open the web application. 2. Log in with valid credentials.
Expected Results:	User should be logged in successfully and authenticated when correct credentials is entered; error message should be shown if user enters wrong credentials.
Actual Results:	User is logged in successfully and authenticated with correct credentials, error message is displayed when user enters wrong credentials.

Test Case 9

Test Case Title:	Validates functionality of viewing user profiles.
Pre-Conditions:	User is logged in.
Test Steps:	1. Navigate to the user profile page.
Expected Results:	User profile information should be displayed correctly.
Actual Results:	User profile information is displayed correctly.

Test Case 10

Test Case Title:	Assesses user logout process.
Pre-Conditions:	User is logged in.
Test Steps:	1. Click on the logout button.
Expected Results:	User should be logged out successfully and redirected to home page.
Actual Results:	User is logged out successfully and redirected to home page.

2. Integration Testing

Test Case 11

Test Case Title:	Assesses system behavior when starting with an empty database.
Pre-Conditions:	Neo4j database is empty.
Test Steps:	1. Start the application.
Expected Results:	The application should start successfully and handle the absence of data gracefully, and start loading the data into neo4j database.
Actual Results:	The application starts successfully and handles the absence of data gracefully, and start loading the data into neo4j database.

Test Case 12

Test Case Title:	Evaluates response when the application begins with a pre-loaded database.
Pre-Conditions:	Neo4j database contains pre-loaded data.
Test Steps:	1. Start the application.
Expected Results:	The application should start successfully and utilize the pre-loaded data without errors.
Actual Results:	The application starts successfully and utilizes the pre-loaded data without errors.

Test Case 13

Test Case Title:	Verifies behavior when a user clicks on a specific POI.
Pre-Conditions:	User is logged in.
Test Steps:	1. Navigate to a specific POI. 2. Click on the POI.
Expected Results:	Details of the selected POI should be displayed correctly.
Actual Results:	Details of the selected POI are displayed correctly.

3. User Workflow Testing

Test Case 14

Test Case Title:	Evaluates the workflow with a user account who has written 34 reviews.
Pre-Conditions:	User is logged in with an account who has written 34 reviews.
Test Steps:	1. Log in with the specified user account.
Expected Results:	User should be able to interact with the application without errors.
Actual Results:	User is able to interact with the application without errors.

Test Case 15

Test Case Title:	Evaluates the workflow with a user account who has written only 5 reviews.
Pre-Conditions:	User is logged in with an account who has written 5 reviews.
Test Steps:	1. Log in with the specified user account.
Expected Results:	User should be able to interact with the application without errors.
Actual Results:	User is able to interact with the application without errors.

Test Case 16

Test Case Title:	Evaluates the workflow with a user account who has written 1 single review.
Pre-Conditions:	User is logged in with an account who has written 1 review.
Test Steps:	1. Log in with the specified user account.
Expected Results:	User should be able to interact with the application without errors.
Actual Results:	User is able to interact with the application without errors.

Test Case 17

Test Case Title:	Evaluates the workflow when accessed as a guest user.
Pre-Conditions:	User is not logged in.
Test Steps:	1. Access the application as a guest user.
Expected Results:	Guest user should be able to browse the application without errors.
Actual Results:	Guest user is able to browse the application without errors.