



Intelligent Reasoning System - Group 5 Final Report

A Hotel Recommender System

Group Member:

Li Qiuxian / e1351736@u.nus.edu / A0297598X (Group Leader)

Gao Yunjia / e1349640@u.nus.edu / A0295502E

Zhao Lanting / e1351814@u.nus.edu / A0297676A

Liu Weixuan / e1349820@u.nus.edu / A0295682M

Huang Yongle Joshua / u0737921@u.nus.edu / A0028090M

CONTENTS

Intelligent Reasoning System - Group 5 Final Report.....	1
1. Introduction.....	3
2. Market Research	3
2.1 Industry Overview	3
2.2 Current Trends in Hotel Recommend Systems.....	3
2.3 Competitive Landscape.....	4
2.4 Opportunities for Differentiation	4
3. Business Value	5
4. Objectives of Project.....	5
5. Overview of Dataset	5
5.1 Data type and source identification.....	5
5.2 Dataset Structure	6
6. System Design	7
6.1 System Architecture.....	7
6.2 Technology Stack.....	8
6.3 System Components.....	9
6.4 System Workflow	11
7. System Development & Implementation.....	15
7.1 Development Process.....	15
7.2 Implementation Details.....	16
7.3 System Integration & Testing	21
8. Findings and Discussion	22
8.1 High Matrix Sparsity.....	22
8.2 Collaborative Filtering Outcomes.....	22
8.3 Prediction Accuracy.....	23
8.4 Discussion	23
Appendix 1: Project Proposal	24
Appendix 2: Mapped System Functionalities Against Knowledge	26
Appendix 3: Installation & User Guide	28

1. Introduction

In the modern travel industry, online booking platforms have greatly facilitated travelers' accommodation choices. However, in the face of the vast amount of hotel resources, it is often difficult for users to make the most appropriate choice based on their own preferences. Traditional hotel search methods often rely on basic filters such as price, location, star rating, and so on. However, these approaches tend to ignore more personalized factors such as a user's historical choices or preferences for certain hotel amenities (e.g. gym, swimming pool, etc.).

To solve this problem, hotel recommender systems have become an important means of providing personalized advice, able to make customized recommendations based on the user's needs. By leveraging data on both hotel facility characteristics and user preferences, the system can offer more accurate and targeted recommendations, ultimately simplifying the decision-making process and enhancing the user experience. With the increasing demand for personalized recommendations from travelers, the design of an efficient hotel recommendation system has become a key research direction to enhance the overall customer travel experience.

2. Market Research

2.1 Industry Overview

The online hotel booking market is growing rapidly, driven by advancements in technology, increasing internet penetration, and shifting consumer preferences toward online services. The global online travel market was valued at USD 354.2 billion in 2021. And the market is projected to grow USD 1,835.6 billion in 2030, at a compound annual growth rate of 14.8% from 2022 to 2030. The hotel recommendation system market is an integral part of this industry, facilitating better customer engagement and decision-making.

2.2 Current Trends in Hotel Recommend Systems

1. Personalization:

Modern consumers expect highly personalized recommendations, tailored to their preferences, past bookings, and travel history. Personalization is becoming a core feature in hotel recommendation systems, with companies using machine learning and AI based methods to analyze user data.

2. Smart Filters & Recommendation:

Travelers seek easy-to-use filters to sort hotels based on their specific requirements, which includes price, proximity, amenities, ratings, etc. Providing smart search features is a key demand from users.

3. User-generated Content:

Reviews, ratings, and user-generated content heavily influence booking decisions. Integrating customer reviews and social media feedback into the recommendation engine can significantly improve customer trust and engagement.

4. NLP Search Integration:

The rising popularity of NLP search with tools like Amazon Alexa and Google Assistant means that hotel recommendation systems need to be adaptable to natural language queries.

2.3 Competitive Landscape

1. Booking.com:

Known for its comprehensive filtering system and tailored recommendations. It relies heavily on user reviews and past booking behavior to drive personalized suggestions.

2. Airbnb:

Though primarily a home-sharing platform, Airbnb provides recommendations based on user preferences and past stays, including personalized collections.

3. TripAdvisor:

Focuses on leveraging user-generated content (reviews, ratings) to provide hotel suggestions. It combines collaborative filtering with user feedback.

2.4 Opportunities for Differentiation

Despite strong competition from established hotel recommend system, several opportunities still exist to create an advanced hotel recommend system.

1. AI-Driven Personalization:

While most systems offer basic personalization, leveraging learning-based models for more sophisticated recommendations based on past interactions, user reviews, and ratings can differentiate your system.

2. Specialized Market Segments

Tailoring the system to cater specific traveler segments will allow the system to differentiate from mainstream players.

3. Focus on User Trust

Show transparency in recommendations. Transparency regarding how recommendations are generated (e.g., showing why a particular hotel is suggested) can build trust with users, especially

in a market where consumers may feel overwhelmed by sponsored content or biased suggestions.

3. Business Value

The recommendation system significantly enhances user experience through personalized hotel recommendations. By analyzing user behavior data and preferences, the system can quickly identify user needs and recommend hotels that best match their expectations. It generates a personalized recommendation list for each user, greatly reducing the time and effort required for them to manually search for hotels. This recommendation approach, based on individual interests and needs, not only allows users to feel that the system understands their preferences but also increases their trust in the platform, thereby improving overall satisfaction.

Additionally, the system can promptly respond to users' filtering criteria and display hotels that meet their requirements. This feature helps users effectively narrow down their choices and enhances decision-making efficiency. By improving the user experience, the recommendation system not only boosts booking conversion rates but also increases user satisfaction, fostering greater loyalty and repeat bookings.

4. Objectives of Project

The main objective of this project is to significantly enhance the accuracy and user satisfaction of the hotel recommendation system. By integrating the item-based collaborative filtering and the content-based recommendation approaches, the goal is to create a more comprehensive and personalized recommendation list. This involves leveraging the users' rating data on various aspects of hotels, such as cleanliness, comfort, location, and the attributes of the hotels themselves, like breakfast provision and swimming pool facilities. Through this combination, the system aims to provide recommendations that not only match the users' preferences but also take into account the similarity between hotels. Ultimately, the objective is to improve the users' decision-making process, enabling them to find hotels that precisely meet their needs more efficiently, thereby increasing their satisfaction with the recommendation system.

5. Overview of Dataset

5.1 Data type and source identification

Hotel Information:

Booking platforms provide hotel details, which includes name, location, available facilities, and user ratings. Opensource Datasets can also provide a combination of hotel details and user ratings.

User Information:

Collect data about user preferences, behavior, or interactions from surveys or user history.

Look for user demographic information, including travel frequency, preferred available facilities.

Feedback Data:

We set a survey after each success for each user to check if the user is satisfied with the recommended result, which can help us to refine our model.

5.2 Dataset Structure

The main dataset we use is the Tripadvisor-review-dataset, which includes more than 500 records of different users staying in different hotels. Each entry of these records contains both user information and hotel information. Below is an example of a particular entry in this dataset:

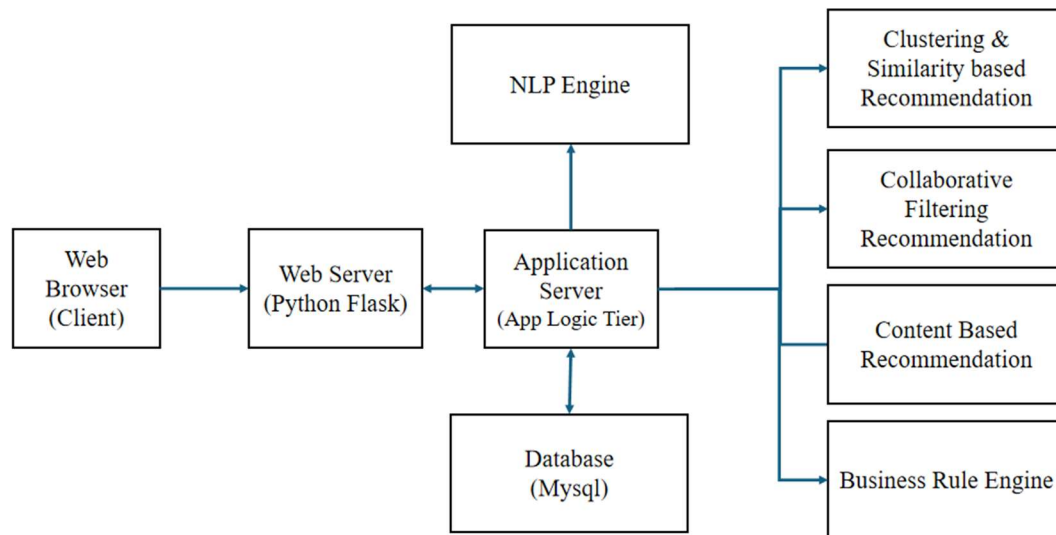
```
{
  User Country: USA
  Nr. reviews: 11
  Nr. Hotel reviews: 4
  Helpful votes: 13
  Score: 5
  Period of stay: Dec-Feb
  Traveler type: Friends
  Swimming pool: NO
  Exercise room: YES
  Basketball court: NO
  Yoga Classes: NO
  Club: YES
  Free Wifi: YES
  Hotel Name: Circus Hotel & Casino Las Vegas
  Hotel Stars: 3
  Nr. Rooms: 3773
  Member years: 9
  Review month: January
  Review weekday: Thursday
}
```

Score	Period of stay	Traveler type	Swimming Pool	Exercise Room	Basketball Court	Yoga Classes	Club	Free Wifi	Hotel name
5	Dec-Feb	Friends	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
3	Dec-Feb	Business	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
5	Mar-May	Families	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
4	Mar-May	Friends	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
4	Mar-May	Solo	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
3	Mar-May	Couples	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
4	Mar-May	Couples	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
4	Mar-May	Families	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
4	Mar-May	Friends	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
3	Mar-May	Families	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
2	Jun-Aug	Families	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
3	Jun-Aug	Friends	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas
2	Jun-Aug	Friends	NO	YES	NO	NO	YES	YES	Circus Circus Hotel & Casino Las Vegas

Part of the structure of hotel dataset

6. System Design

6.1 System Architecture



1. Web browser (client):

This is the user-facing interface where the user interacts with the system. It is the entry point for the user to make ratings and desirable holiday descriptions.

2. Web Server (Python Flask):

The web server is built by the Flask framework and acts as middleware to handle user requests from

the front-end. It processes these requests and forwards them to the application server.

3. Application Server (App Logic Tier):

This is the core processing unit of the system. It:

Receives and processes user inputs.

Calls the relevant recommendation modules.

Retrieves necessary data from the database.

Delivers final recommendations to the user interface.

4. Database (MySQL):

The database is responsible for storing all persistent data, including user information, hotel data, ratings, and historical interactions. When the application server needs data, such as a user's past hotel ratings or hotel information, it queries the MySQL database.

5. NLP Engine:

The NLP (Natural Language Processing) engine is integrated to process user-provided text descriptions (e.g., "I want a hotel with a swimming pool and sea view"). The NLP engine extracts key preferences from the text input, which the application server then uses to generate content-based recommendations.

6. Clustering & Similarity-based Recommendation:

This module groups hotels into clusters based on their features and similarities. Once a user interacts with or rates certain hotels, the system uses these clusters to suggest similar hotels that match the user's preferences. This reduces dimensionality and narrows down the hotel selection.

7. Collaborative Filtering Recommendation:

This recommendation engine suggests hotels based on similarities between hotels, using item-based collaborative filtering. The system analyzes historical rating data and recommends hotels that users with similar preferences have rated highly.

8. Content-Based Recommendation:

This module focuses on matching user preferences with specific hotel features. Based on the provided text input (processed by the NLP engine), this engine finds hotels that closely match the user's expressed desires.

9. Business Rules Engine:

The business rules engine integrates predefined business logic into the recommendation process. It will adjust the hotel booking price based on external business factors.

6.2 Technology Stack

Scikit-learn:

- Used for implementing the item-based collaborative filtering algorithm.

- Provides tools for similarity calculation, including cosine similarity, to measure how closely hotels match user preferences.

Surprise:

- A specialized library for collaborative filtering, used to train and evaluate recommendation models.
- Facilitates **data handling** and provides various algorithms to optimize the system's recommendation accuracy, ensuring personalized hotel suggestions based on user ratings.

Business Rule engine:

- Business-rule-engine 0.2.0 (from Pypi)

Vue.js:

- Vue.js is responsible for the user interface (UI) of the application. It provides an interactive and dynamic experience by handling user input, rendering views, and updating the UI in real-time.
- Automatically synchronizes the UI with the underlying data model.
- Used for routing and navigation between different views/pages in the application.

Axios:

- Used to make HTTP requests from the Vue.js frontend to your Flask backend. This is crucial for fetching data (e.g., user-specific recommendations) and submitting user inputs to the server.

Flask:

- Flask is a micro web framework for Python that handles all the backend logic. It serves API endpoints to communicate with the Vue.js frontend, processes requests, and interacts with the MySQL database.
- Flask-SQLAlchemy makes it easier to interact with MySQL using Python objects instead of raw SQL queries.
- Flask-CORS allows for handling cross-origin resource sharing (CORS).
- Integrate middleware for authentication, logging, or other request/response processing.

MySQL:

- MySQL is a relational database used to store and retrieve structured data for your recommendation system.
- User data, preferences, recommendation metadata, and other system-related data are stored in tables.
- Tables can be linked using foreign keys, which allows for efficient query execution across related tables.

6.3 System Components

6.3.1 Web Server, Application Server, Database Server

Frontend development: The frontend focus on providing a user-friendly interface for the system. Web Server applies Vue framework to provide a responsive interface, enabling UI design and interaction so that users can view recommendations, filter results, and make bookings. Use Vue Router to handle page navigation logic, ensuring smooth navigation between different parts of the

website. Use Axios to communicate with the backend, sending requests to fetch hotel and user rating data, enabling recommendation and booking features.

Backend development: The Application Server uses Flask for backend integration with the hotel recommendation engine, providing personalized hotel recommendations based on user preferences and history. Flask offers RESTful API endpoints to handle user identification and booking processes. The API follows a clear modular architecture, separating business logic into different services.

Database Development: The database will serve as the backbone of the hotel recommendation system, storing essential data related to hotels, users, reviews, and recommendations.

The database efficiently stores hotel and user data in a structured way, allowing for quick retrieval of recommendations, user preferences, and hotel information. It is also scalable to accommodate future data growth.

Key Database Entities:

Users: Stores user profiles, preferences, and interaction history.

Hotels: Contains information about each hotel, including name, location, amenities, price range, and star rating.

Bookings: Stores details about user bookings, hotel stays, and transaction history.

Reviews: User-generated content, such as reviews and ratings for hotels.

Recommendations: Stores the results of recommendation algorithms for each user.

6.3.2 Clustering & Similarity based Recommendation

The system leverages K-means clustering to group hotels into distinct categories based on their features, ensuring that similar hotels are clustered together. After clustering, each cluster is reduced to 15 representative hotels to maintain diversity while preventing overloading the user with options. Once the user selects a cluster and rates five hotels, the system uses cosine similarity to compare the features of the highest-rated hotel to others within the same cluster. The two most similar hotels are then recommended, providing personalized suggestions that align with the user's preferences.

6.3.3 Item-Based Collaborative Filtering Recommendation

Collaborative Filtering Algorithm: Item-Based Collaborative Filtering works by analyzing users' hotel ratings and calculating the similarity between different hotels. The system compares hotel features like amenities, services, and location, alongside the ratings users provide, to build a hotel-to-hotel similarity matrix. Similarity between hotels is measured using techniques such as Cosine Similarity, which identifies how closely their features and ratings align. When a user rates certain hotels highly, the system finds other hotels with similar features and suggests them to the user. This method helps predict which hotels a user might like based on their past ratings.

The recommendation process in the system is executed through a series of steps. First, the user provides ratings for one or more hotels. Next, the system calculates the similarity between the rated hotels and other hotels in the dataset based on those ratings. Finally, the system generates a recommendation list of hotels that are most similar to the highest-rated ones, considering shared features such as amenities and user ratings.

6.3.4 Content-Based Recommendation with NLP

The part of the system that uses NLP for content-based recommendations is designed to handle user-inputted descriptions of their ideal hotel preferences. This component allows the system to extract important features from user text (e.g., "I want a clean hotel with a gym and pool") and match them to hotels that possess those features. The system uses SpaCy for processing the text input, breaking it down into key elements, such as hotel amenities and services, which are then compared with the feature sets of hotels in the database. By doing this, the system can understand the user's preferences and offer more personalized hotel recommendations based on the specific facilities and services they are looking for.

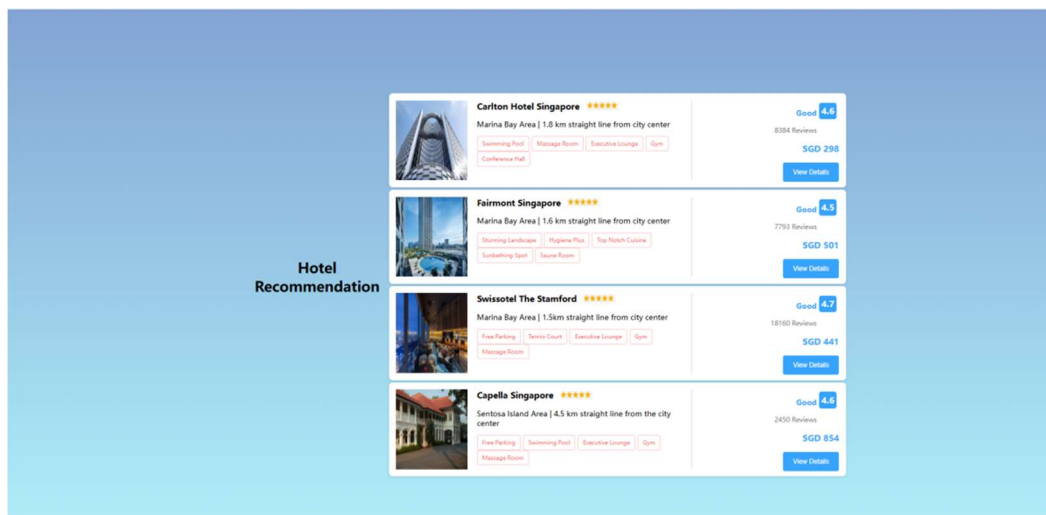
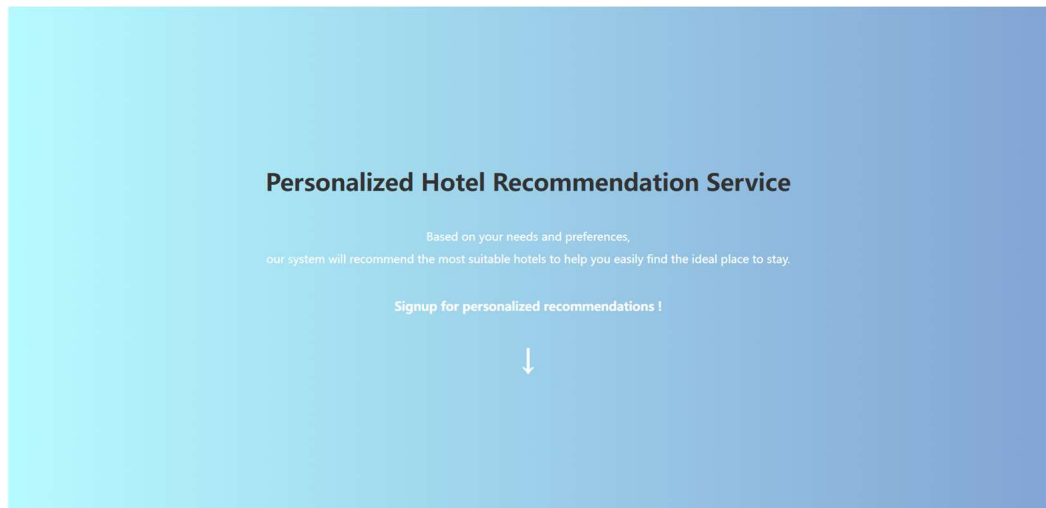
In addition to the textual analysis, this component interacts with the hotel feature matrix, allowing the system to suggest hotels based on matching attributes. The NLP component works hand-in-hand with other recommendation methods like collaborative filtering to deliver a more holistic recommendation experience by incorporating both user input and past behavior.

6.3.5 Business Rule

The business rule engine will implement business logic to calculate the final price of the hotel room for the user. This is based on a set of rules determined by various business division such as marketing and sales so that there is price discrimination determined by various factors such as customer loyalty, business partnerships with hotels, and promotion event run during special roadshows and events. The logic applied will return the final price which will then be displayed to the customer on the web front end.

6.4 System Workflow

Briefly, our hotel recommendation system has a linear workflow that collects various types of information from users, including their preferences, ratings, check-in information, etc. It provides users with a list of multiple hotels for rating and combines natural language processing analysis and collaborative filtering algorithms to provide users with a series of recommendation results. And provide hotel booking information such as discounts to users based on their check-in information and business rules to better assist them. The following will provide a detailed explanation of our system's workflow from three sections: user information collection, recommendation algorithms, and business rules.



• User Information Collection

In order to provide personalized hotel recommendations and booking services for users, we need to collect a series of user data. It includes basic user information such as username, age, vacation frequency, etc. We ask and record this information from the user when they first enter the system. In addition to basic information, we also need to collect information about users' preferences for the hotel. In order to accurately and comprehensively collect this information, we first allow users to have conversations with our system's chatbot and freely express their expectations for the target hotel under the guidance of our chatbot. At the same time, we will also use NLP technology to process the text input by users and provide several preliminary hotel recommendations for users to rate. At this point, we have obtained basic information about user preferences and behavioral information about user ratings for certain hotels. We store this information in a database in a one-to-one correspondence with the user's basic information, in order to further recommend more accurate hotels to the user in the future. Besides, once the user gets recommendation result from our system, they can input their planned check-in date and other information, and our system can use business rules to provide users with a discount information as a reference.

User Signup

Name :

Age :


Gender :

Address :

Country :

Holiday Frequency :

Do you want to sign up for Gold Membership? : ☐ Yes ☐ No



• Recommendation System

In fact, we have two different recommendation algorithms. This is mainly to solve the cold start problem caused by the lack of rating data during the first deployment of our system and the lack of historical behavior of new users entering the system. Previously, we mentioned that we provided several sets of hotels for users to rate based on their input. In order to ensure the diversity of rated hotels, we cluster all hotels and randomly output several hotels from each cluster when providing rating items to users. Our first stage recommendation algorithm calculates and recommends the most similar results from the clustering of the highest rated hotels. We use cosine similarity to calculate the recommendation results in this stage. This step helped us solve the cold start problem and collected more user history behavior and rating data for our system. The second stage recommendation algorithm is a item based collaborative filtering algorithm. We first create a rating matrix and calculate its sparsity. When the sparsity reaches a threshold, we start executing collaborative filtering algorithms to provide users with new recommendation results. Once the recommendation results, it will be displayed on the webpage for user to look through and book.

Chat Window

Hello! Tell us about your ideal vacation!

Swimming Pool Free Parking Free Breakfast

I need a hotel that serves free breakfast.

Based on your ideal vacation you would be most interested in:



Heaven on Hollywood Furnished Apartments



Hotel Solaire Los Angeles



EXQUISITE MARINA 2BDR SUITE WITH BALCONY & RELAXING VIEWS



Penthouse Highrise



Sunset Strip 1-Bedroom with Balcony, Pool and Views



Charming and Lovely apt!

Please confirm your selection

For the selection of



Beverly Hills Marriott

☆☆☆☆☆



Urban Downtown LA Pool Table Penthouse


☆☆☆☆☆

• Business Rules

We have pre-set several different business rules, including checking whether the check-in date falls on holidays and whether the booked room type and quantity meet the hotel's discount requirements. Through these business rules, our system can provide discount information to users.

Your personalized recommendations are ready!

We recommend that this hotel for you:



SLS Hotel, a Luxury Collection Hotel, Beverly Hills,

WiFi: Available
Swimming Pool: Available
Free Parking: Not Available
Facilities for Disabled Guests: Not Available

Previous Next

Book your hotel

Check-in Date:

Check-out Date:

Number of rooms:

Calculate Discount

7. System Development & Implementation

7.1 Development Process

The development process of our recommendation system can be mainly divided into two parts: front-end and back-end. The front-end mainly includes components for user input and result display, used for interacting with users. The backend mainly includes recommendation algorithm functions, databases, and different interfaces for data exchange with the frontend.

• Frontend Development

For the front-end development part, we first design several necessary pages such as user login, recommendation result display, etc. When we have a set of structured front-end pages, we call the components in VUE to implement these functions, and first use some static data to test the effectiveness. Then we just need to beautify these front-end pages.

• Recommendation function

The recommendation function is the core content of the entire recommendation system. When developing recommendation functions, we first need to process the hotel dataset. We use the K-means algorithm to cluster the hotel dataset into six clusters. Based on these six clusters, we can further process the dataset. In addition, we use an expert rating system to generate a random number of ratings for each hotel in the hotel dataset to train the collaborative filtering algorithm model. After obtaining these data, we construct a rating matrix and set different thresholds for the sparsity, and compare the effectiveness of collaborative filtering algorithms under different thresholds. Finally, export the best results for future use in recommendation systems.

• Backend development

For backend development, we need to integrate the trained recommendation function into the backend interface and allow the frontend to call these interfaces when users need to view recommendation results to complete hotel recommendations. At the same time, we need to design a reasonable database structure to efficiently store data. This not only effectively improves the speed of running recommendation algorithms in our recommendation system, but also facilitates our continuous expansion of user historical behavior data and improvement of our recommendation algorithm model in the future.

7.2 Implementation Details

7.2.1 Data processing and storage

The data processing and storage design of the system is centered around user behavior data and hotel rating data to provide support for the real-time calculation and accurate recommendation of the recommendation engine. The specific implementation details are as follows:

First, implement the collection of user behavior and hotel rating data: When a user performs actions such as browsing, rating or booking in the front-end, these actions are sent to the back-end through the API. The specific implementation is to package user actions (such as clicking on a hotel, submitting a rating, etc.) into a JSON-formatted request through the Axios library in Vue, which is passed to the Flask backend for processing.

Second, implement back-end data processing and storage: Flask back-end receives these data, first of all, data validation and processing. After ensuring that the data is in the correct format, the data is saved to the database using an ORM (e.g. SQLAlchemy). The user behavior data is stored in association with the user ID, while the hotel rating data is stored in association with the hotel ID for subsequent queries.

Third, the database is designed with an efficient query mechanism to realize the real-time acquisition of user and hotel data by the recommendation engine, and to accelerate the correlation query between user preference data and hotel data.

7.2.2 Front-end implementation

The front-end interface is developed using the Vue framework, which supports responsive design and user-friendly interactive interface. Data interaction with the backend via API.

First, choose Vue as the front-end framework to build a responsive user interface. Vue's componentized structure allows each UI module (e.g., hotel list, rating module, etc.) to be developed and maintained independently. Implementing a multi-page application based on Vue Router ensures that users can navigate smoothly between different pages. And based on Vue Router to realize multi-

page application, to ensure that users can smoothly navigate between different pages.

Second, the Axios library is used to interact with the RESTful API of the Flask backend. The front-end will send requests through Axios, such as getting a list of hotels or submitting reservation information. Each request carries the necessary parameters to ensure that the requested data is accurate and secure.

Third, perform front-end performance optimization. Through Vue's keep-alive component and browser caching mechanism, cache the pages and data that the user has already visited. When users switch to different recommended hotels, the loaded content does not need to be requested repeatedly, improving the page response speed. Also, optimize the static resources of the front-end and provide resources through CDN to speed up the loading speed of resources and improve user experience.

7.2.3 Clustering & Similarity based Recommendation

This project utilizes clustering techniques to organize hotels into meaningful groups, facilitating personalized hotel recommendations. The clustering process follows several steps to ensure that users receive relevant suggestions based on their preferences.

- **Clustering Methodology**

The first step involves data standardization using 'StandardScaler', which ensures that all features are on the same scale, improving the accuracy of the clustering algorithm. After scaling the data, K-means clustering is applied, grouping the hotels into four distinct clusters. Each cluster contains hotels with similar characteristics, making it easier for users to explore hotels that meet their preferences.

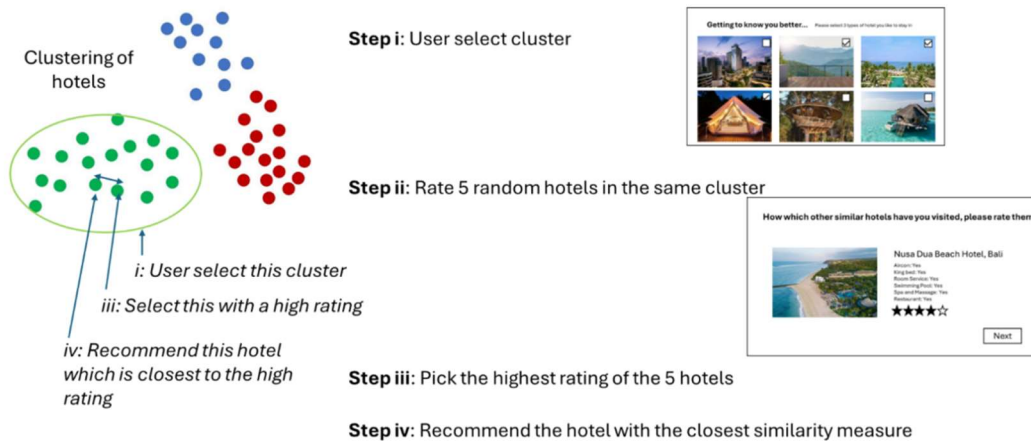
To keep the dataset manageable, the system randomly selects 15 hotels from each cluster, ensuring diversity while preventing overwhelming the user with too many options. This step is essential for streamlining the recommendation process, as it reduces the overall number of hotels while maintaining a variety of choices within each cluster.

- **User Interaction and Ratings**

Once the clustering is complete, users are prompted to select one of the four clusters that best matches their preferences. Within the selected cluster, the system randomly selects five hotels for the user to rate. These ratings are crucial in understanding the user's preferences and determining which hotel aligns most closely with their tastes.

- **Recommendation Mechanism**

After the user provides ratings for the five hotels, the system identifies the highest-rated hotel and uses it as the reference point for further recommendations. The system calculates cosine similarity between the highest-rated hotel and other hotels within the selected cluster. Based on this similarity, the two most similar hotels are recommended to the user. This approach ensures that the recommendations are personalized and directly related to the user's preferences.



7.2.4 Item-Based Collaborative Filtering Recommendation

To implement collaborative filtering, the system first creates a hotel-feature matrix using hotel characteristics like air conditioning, swimming pools, and WiFi. Each hotel is represented by a feature vector, making it possible to compare hotels based on shared attributes. The system also builds a user rating matrix, where users rate different hotels, with higher scores indicating stronger preferences. Using these matrices, the system applies Cosine Similarity to calculate how similar two hotels are, based on their features and user ratings.

For the cold-start problem, when a new user signs up, they are asked to select a hotel category (e.g., city hotels, beach hotels). Then, the system shows them five randomly selected hotels from that category to rate. Based on these initial ratings, the system can start making recommendations by identifying hotels similar to those the user rated highly. This allows the system to offer personalized recommendations, even for users who have no previous interaction data.

Finally, the recommendation process works by first collecting user ratings, then calculating the similarity between the rated hotels and others in the system, and generating a list of recommended hotels that match the user's preferences and past behavior.

	Hotel Pennsylvania	Radisson Martinique on Broadway	Residence Inn by Marriott New York Manhattan/Times Square	MADE Hotel	LUMA Hotel - Times Square	The Standard, High Line New York	citizenM New York Times Square	Gansevoort Meatpacking	Courtyard by Marriott New York Manhattan/Central Park	...
User 1	3			5						
User 2	4					5		4		
User 3	2	3			4					
User 4										
User 5			4		2					
User 6		3		5						
User 7	3		5		2					
User 8		2		1						

How which other similar hotels have you visited, please rate them

Nusa Dua Beach Hotel, Bali

Aircon: Yes
King bed: Yes
Room Service: Yes
Swimming Pool: Yes
Spa and Massage: Yes
Restaurant: Yes

★★★★☆

Next

7.2.5 Content-Based Recommendation with NLP

In the NLP section of the hotel recommendation system, we implemented hotel recommendations based on the user's text descriptions by following a structured process.

First, we used SpaCy, a natural language processing library, to preprocess the user's input. The text, which might describe hotel preferences (e.g., "I want a clean, comfortable hotel with a swimming pool and a gym"), was cleaned by converting it to lowercase, removing stop words (common words like "and" or "the"), and eliminating punctuation. We also applied lemmatization, which reduces words to their base forms, so variations of the same word (e.g., "clean" and "cleanliness") are treated as the same word.

After preprocessing, we used TF-IDF (Term Frequency-Inverse Document Frequency) to convert both the user's input and the hotel descriptions from the dataset into numerical vectors. This step assigns importance to words based on their frequency within the text, ensuring that more significant words have a greater impact in the comparison process.

Next, we applied cosine similarity to compare the user's input with each hotel's description. Cosine similarity calculates the angle between the vector representing the user's preferences and the vectors representing the hotels' features. The smaller the angle, the closer the match between the user's input and the hotel's features.

Finally, based on the similarity scores, the system generates a list of hotels that best match the user's preferences. This process allows for personalized recommendations tailored to what the user is looking for. By integrating these NLP-based content recommendations with the collaborative filtering results, the system delivers a more comprehensive and personalized recommendation experience.

7.2.6 Business Rule

• Business Rule Engine Overview

The business rule engine is crucial to the recommendation system of an e-commerce platform specializing in hotel bookings. Its main function is to determine the final room price presented to customers by integrating essential business logic. The engine processes various rules, including sale discounts, customer membership types, room specifications, and seasonal considerations, ensuring the final price reflects applicable discounts or surcharges based on customer profiles and booking timing.

The output from the recommendation system serves as input for the business rule engine, which applies relevant adjustments before displaying the price to customers. Although scalability strategies are not currently addressed, there is recognition of the need for future expansion to manage increased traffic and rule complexity. Plans to empower business users to modify or add rules are also in consideration, ensuring the engine remains aligned with evolving business needs.

• Selection of Business Rule Engine

The business rule engine was selected based on ease of implementation and cost, leading to the choice of the open-source business-rule-engine package from PyPI. Its compatibility with the

existing Python-based architecture facilitated straightforward integration. Previous attempts with other engines, like Drools and Decisionrules.io, posed significant challenges, either due to integration issues or restrictions on functionality.

While documentation and community support for the chosen engine are limited, its user-friendly syntax for defining rules makes it accessible for business users. Future considerations include developing scalability strategies to efficiently manage an increasing number of rules and adapting the engine as business needs evolve.

• **Designing Business Rules**

The business rules use a clear "when X, then Y" syntax, stored in plain text files for easy updates. The focus is on modifying hotel pricing based on various factors to provide tailored offers. Future rules will include surcharges based on hotel ratings and referral code discounts.

To ensure input integrity, the user interface conducts validation checks; however, improved error handling is necessary. Implementing exception management strategies could enhance user experience by logging errors and notifying users of input issues.

Currently, only unit testing is performed, but adopting a combination of integration testing and behavior-driven development (BDD) would better ensure complex rule sets behave as expected. A centralized repository for documenting business rules should be established, providing stakeholders with essential information.

For modifying or adding rules, a user-friendly interface should be developed, enabling business users to define rules without programming knowledge.

• **Testing the Business Rule Engine**

Given the project's simplicity, only basic unit testing has been implemented. However, a comprehensive testing strategy should be established, utilizing tools like unittest or pytest. This framework should allow for test cases covering individual rules and their interactions, ensuring all logic paths are verified. Automation will be prioritized to facilitate regular testing in a dedicated environment, minimizing disruptions.

• **Conclusion for Business Rule Engine**

The business rule engine has significantly improved the pricing strategy of the e-commerce platform by functioning as an AI expert system. By encapsulating domain knowledge into actionable rules, the engine eliminates the need for human consultants, enabling customized pricing and price discrimination across market segments, which drives profitability.

The challenges faced in selecting the right tool highlight the importance of integrating expert systems as a critical component of AI initiatives. This project demonstrates how expert systems complement machine learning and deep learning, providing structured, rule-based decision-making. As enhancements are planned, the flexibility and scalability of the business rule engine will be vital in adapting to evolving business needs and delivering continued value to customers.

7.3 System Integration & Testing

The testing strategy encompassed three main categories to ensure comprehensive coverage:

7.3.1. Unit Tests

Unit tests were conducted on individual components, including:

- **Web Server:** Validated that the user interface correctly processes profile setups and hotel ratings. This included testing form submissions and data retrieval.
- **Application Server:** Ensured that API calls to the SaaS LLM function correctly, verifying data handling from user input to recommendation requests.
- **Recommendation Systems:**
 - **Collaborative Filtering:** Tested for accurate recommendations based on user similarity, including edge cases for new users facing the cold start problem.
 - **Cosine Similarity:** Confirmed that the system correctly identified and recommended hotels similar to user preferences.
- **Business Rule Engine:** Rigorously assessed the application of business rules and the accuracy of pricing calculations against various input scenarios.

7.3.2. Integration Tests

Integration tests were crucial in validating the interactions between system components:

- **Web Server and Application Server:** Tests ensured that the web server accurately communicated with the application server, effectively passing user inputs and receiving the expected responses. Scenarios included both valid and invalid inputs to confirm robust error handling.
- **Application Server and SaaS LLM:** The interaction between the application server and the SaaS LLM was thoroughly tested. This included sending diverse user queries and validating that the responses were accurate and formatted correctly.
- **Recommendation Systems:** Both recommendation engines were tested to verify that they sent accurate recommendations back to the user interface. Integration tests focused on ensuring that data flowed smoothly from user preferences to recommendations displayed on the front end.
- **Business Rule Engine:** The integration of the business rule engine with the recommendation systems was verified. Tests confirmed that the engine accurately processed the outputs from both recommendation systems to calculate the final pricing, taking into account various business rules.

7.3.3. End-to-End Tests

End-to-end testing simulated real user scenarios, covering the complete workflow from profile setup to hotel recommendation and pricing display. Specific areas tested included:

- User profile creation, including the validation of input data.
- Hotel rating submissions to ensure accurate data entry and processing.
- Full simulation of the user journey, where a user receives hotel recommendations based on their preferences and sees the final pricing calculated by the business rule engine. Test cases

encompassed a range of user profiles and preferences, including edge cases for new users with no prior data. Each final output was validated for accuracy and correctness, ensuring that error handling worked as intended.

7.3.4. Performance Testing

Performance testing evaluated the system's responsiveness and behavior under load. Key areas of focus included:

- Simulating concurrent users to determine the system's capacity and response times.
- Assessing how well the architecture managed simultaneous requests, particularly during peak usage scenarios.

7.3.5 Conclusion for System Integration Testing

The system integration testing conducted for the e-commerce platform's recommendation system successfully validated the robustness and functionality of each component, both individually and collectively. The comprehensive testing strategy ensured that the system meets user needs and business objectives, establishing a solid foundation for ongoing development and enhancement. Through thorough component testing, rigorous integration testing, realistic end-to-end simulations, and performance evaluations, the project is well-positioned for a successful deployment and future scalability.

8. Findings and Discussion

8.1 High Matrix Sparsity

The rating matrix exhibited significant sparsity, with many missing ratings—a common challenge in recommendation systems, where users typically rate only a limited number of items. In this project, the matrix sparsity exceeded 50%, triggering the collaborative filtering algorithm. This step was crucial in ensuring that the system could handle incomplete data efficiently while still providing relevant recommendations.

8.2 Collaborative Filtering Outcomes

The KNN Basic algorithm successfully generated item-based recommendations. By identifying the hotels that users rated the highest, the system was able to suggest similar hotels using cosine similarity between items. This approach effectively tailored recommendations to users' preferences based on their previous ratings.

8.3 Prediction Accuracy

The system's accuracy was assessed using the RMSE (Root Mean Square Error) metric, which evaluates the accuracy of predictions. Although specific RMSE values are not provided, the system demonstrated reasonable accuracy in predicting user preferences, confirming the suitability of the item-based collaborative filtering method.

8.4 Discussion

The findings from this project underscore both the strengths and limitations of item-based collaborative filtering. The high sparsity of the rating matrix posed a typical challenge, but it was effectively managed by activating the filtering process only when necessary. This strategy helped optimize computation without compromising the quality of recommendations.

The KNN-based item similarity algorithm performed well, enabling the system to recommend hotels that aligned with users' preferences. By focusing on item similarities rather than user similarities, the system avoided common issues associated with the variability of user tastes. This is particularly advantageous in the context of hotel recommendations, where user preferences are highly diverse, but similar hotels can still be effectively identified.

However, the system encountered difficulties in generating recommendations for users with very few ratings. The lack of sufficient data for these users limited the system's ability to offer personalized suggestions. To address this issue, future enhancements could involve incorporating hybrid models that combine collaborative filtering with content-based approaches or matrix factorization techniques to mitigate data sparsity.

In conclusion, this project successfully demonstrated the effectiveness of an item-based collaborative filtering recommendation system. Despite the challenge posed by data sparsity, the system was able to deliver personalized and relevant hotel suggestions. Future improvements should focus on enhancing recommendations for users with limited data, thereby improving the overall robustness of the system.

Appendix A: Project Proposal

Date of proposal: 15 Sep. 2024
Project Title: Intelligent Reasoning System - Group 5 Project – A Hotel Recommendation System
Group ID (As Enrolled in Canvas Class Groups): Group 5 Group Members (name , Student ID): Li Qiuxian / A0297598X (Group Leader) Gao Yunjia / A0295502E Zhao Lanting /A0297676A Liu Weixuan /A0295682M Huang Yongle Joshua / A0028090M
Sponsor/Client: (Company Name, Address and Contact Name, Email, if any) <i>Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore NATIONAL UNIVERSITY OF SINGAPORE (NUS)</i>
Background/Aims/Objectives: The main objective of this project is to significantly enhance the accuracy and user satisfaction of the hotel recommendation system. By integrating the item-based collaborative filtering and the content-based recommendation approaches, the goal is to create a more comprehensive and personalized recommendation list. This involves leveraging the users' rating data on various aspects of hotels, such as cleanliness, comfort, location, and the attributes of the hotels themselves, like breakfast provision and swimming pool facilities. Through this combination, the system aims to provide recommendations that not only match the users' preferences but also take into account the similarity between hotels. Ultimately, the objective is to improve the users' decision-making process, enabling them to find hotels that precisely meet their needs more efficiently, thereby increasing their satisfaction with the recommendation system.
Project Descriptions: 1.Web browser (client): This is the user-facing interface where the user interacts with the system. It is the entry point for the user to make ratings and desirable holiday descriptions.

2.Web Server (Python Flask):

The web server is built by the Flask framework and acts as middleware to handle user requests from the front-end. It processes these requests and forwards them to the application server.

3.Application Server (App Logic Tier):

This is the core processing unit of the system. It:

Receives and processes user inputs.

Calls the relevant recommendation modules.

Retrieves necessary data from the database.

Delivers final recommendations to the user interface.

4.Database (MySQL):

The database is responsible for storing all persistent data, including user information, hotel data, ratings, and historical interactions. When the application server needs data, such as a user's past hotel ratings or hotel information, it queries the MySQL database.

5.NLP Engine:

The NLP (Natural Language Processing) engine is integrated to process user-provided text descriptions (e.g., "I want a hotel with a swimming pool and sea view"). The NLP engine extracts key preferences from the text input, which the application server then uses to generate content-based recommendations.

6.Clustering & Similarity-based Recommendation:

This module groups hotels into clusters based on their features and similarities. Once a user interacts with or rates certain hotels, the system uses these clusters to suggest similar hotels that match the user's preferences. This reduces dimensionality and narrows down the hotel selection.

7.Collaborative Filtering Recommendation:

This recommendation engine suggests hotels based on similarities between hotels, using item-based collaborative filtering. The system analyzes historical rating data and recommends hotels that users with similar preferences have rated highly.

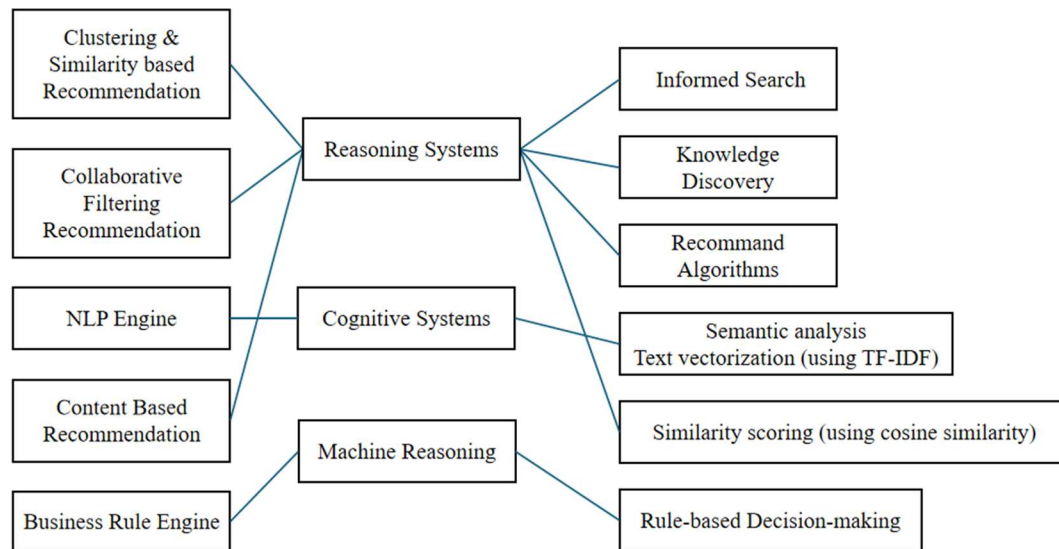
8.Content-Based Recommendation:

This module focuses on matching user preferences with specific hotel features. Based on the provided text input (processed by the NLP engine), this engine finds hotels that closely match the user's expressed desires.

9.Business Rules Engine:

The business rules engine integrates predefined business logic into the recommendation process. It will adjust the hotel booking price based on external business factors.

Appendix B: Mapped System Functionalities Against Knowledge



1. Reasoning Systems

Reasoning systems are used in recommender systems to analyse the relationship between user behaviour, rating data and hotel characteristics, and generate recommendations by reasoning and decision making through predefined rules or algorithms. In the hotel recommendation system, the following modules are related to the inference system:

•Clustering & Similarity-based Recommendation

This part uses Informed Search to classify hotels into different clusters and filter suitable hotels based on the user's preferences or behaviours towards specific categories of hotels. Through the knowledge discovery process, the system is able to identify which hotels are closest to the user's needs and recommend these hotels. For example, a user can select a certain hotel category based on preferences, and the system makes recommendations through clustering methods to help users quickly narrow down their choices. This clustering-based recommendation is essentially searching and reasoning about specific relationships between hotels and users to optimise the recommendation.

•Collaborative Filtering Recommendation

Collaborative Filtering is a typical recommendation algorithm (Recommendation Algorithms), which is based on the user's historical rating data of hotels, by comparing the preferences of different users, it finds other users with similar preferences to the current user, and thus recommends hotels that they had rated highly. This is a process of knowledge discovery from historical data, whereby the inference system identifies and processes the user's preference characteristics to derive hotels that may be of interest to the user in the future.

2. Cognitive Systems

Cognitive systems simulate the human cognitive process of information processing and decision making. In recommend systems, it helps the system to understand the user's needs and preferences and make recommendation decisions accordingly. The following functional modules are related to cognitive systems:

•NLP Engine

The Natural Language Processing (NLP) module allows users to input their hotel needs or preferences through free text. Through Semantic Analysis, the system is able to understand the user's text input, for example "I want a hotel with a swimming pool and a gym". The NLP module helps the cognitive system to mimic human language comprehension through semantic analysis and feature extraction to generate appropriate hotel recommendations for the user by transforming the user's language into processable feature vectors through TF-IDF vectorisation.

•Content-Based Recommendation

Content-based recommendation uses similarity measures such as Cosine Similarity to calculate the similarity in features between the hotels that the user has rated and other hotels, so that hotels with similar features can be recommended. For example, if the user prefers hotels with air-conditioning and gym facilities, the system will recommend other hotels with similar facilities. Here the user behaviour is analysed by way of cognitive system to extract the user's needs and similarity scoring is done based on the hotel features. This approach is based on the understanding of the content and information processing, which directly corresponds to the cognitive process that humans use in making decisions.

3. Machine Reasoning

Machine reasoning is a key part of the overall system for making complex decisions through rules and algorithms. It allows the system to dynamically adjust based on a set of predefined conditions and logic to ensure that recommendations are accurate and tailored to the user's needs.

• Business Rule Engine

The business rules engine allows the system to dynamically adjust to different market conditions and user behaviour. For example, in low season the system may recommend discounted hotels, while in high season it may prioritise hotels with more availability. Such rules are implemented through Rule-based Decision-making, where the inference system generates dynamic recommendation strategies based on the inference results of the rule engine, thus ensuring that the recommended hotels not only meet the user's needs, but also conform to the current market environment. This is a typical rule-based reasoning system that adapts to changing external conditions and user behavior.

Appendix C: Installation & User Guide

Installation

1. Environment Requirement

pip install requirements

2. Run the system on local machine

Open terminal in our local fold file

```
$ git clone https://github.com/A9gust/HotelRecommendationSystem.git
```

```
$ cd HotelRecommendationSystem/backend
```

```
$ python app.py
```

```
$ cd HotelRecommendationSystem/hotel-recommendation
```

```
$ npm run serve
```

Go to URL using web browser <http://0.0.0.0:8080> or <http://127.0.0.1:8080>

User Guide

1. Home Page

The Home Page provides a clear and intuitive interface, allowing users to explore the core functionalities of the hotel recommendation system. Upon entering the Home Page, users can click the “Signup for personalized recommendations!” button to be redirected to the sign-up and login page, where they can start customizing their personalized recommendation experience.

Scrolling down the page, the system presents a few highly-rated hotels selected based on overall user reviews. Users can click on any hotel name to view detailed information, including ratings, prices, and amenities.

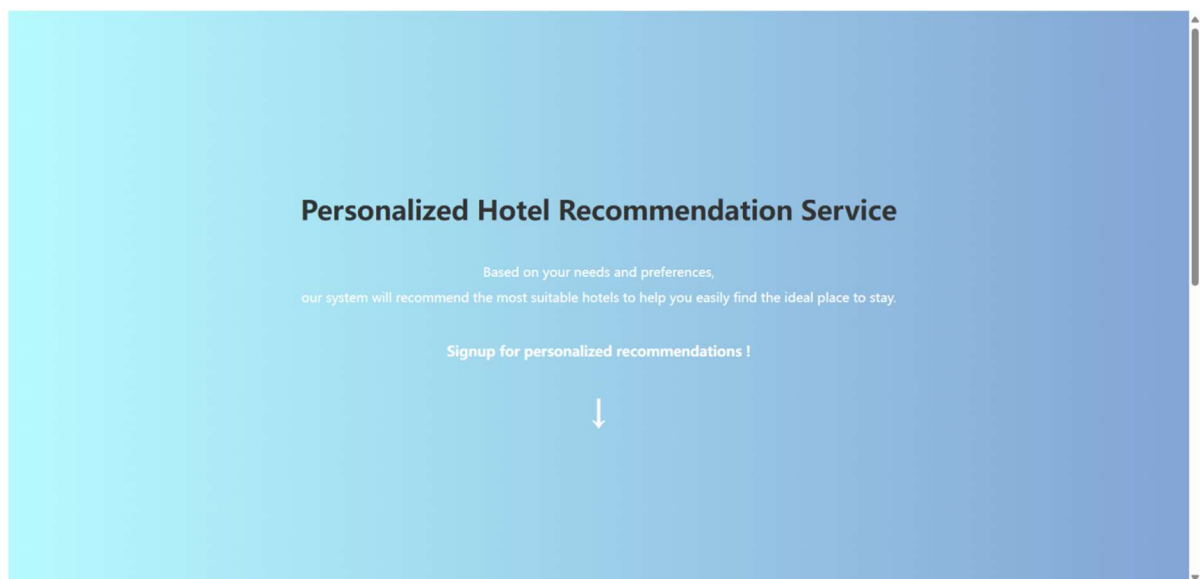


Figure 1: HomePage

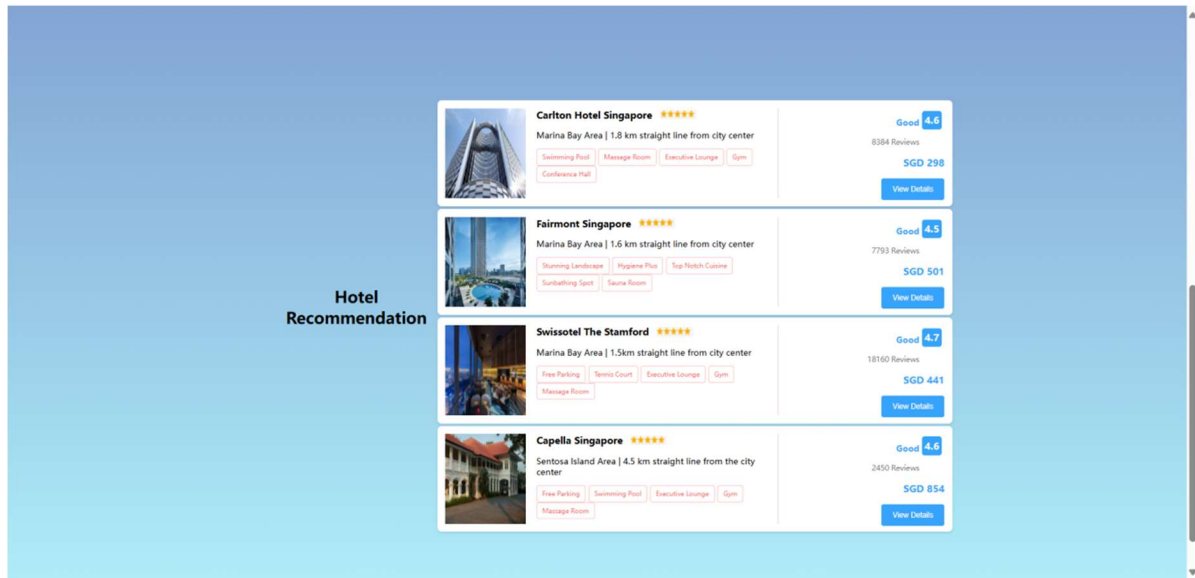


Figure 2: HomePage Recommendation

2. Sign-up and Login Page

On the Sign-up and Login Page, users are required to fill in personal details such as name, age, gender, address, country, holiday frequency per year, and whether they wish to become a member. This information helps the system tailor more personalized recommendations for each user.

Once all the information is entered, click “Submit” to complete the registration process.

User Signup

Name :

Age :

Gender :

Address :

Country :

Holiday Frequency :

Do you want to sign up for Gold Membership? : ☐ Yes ☐ No

Figure 3: SignUp Page

3.Chat Recommendation Page

On the Chat Recommendation Page, users can input simple keywords or phrases to ask the system for hotel recommendations. For example, you can type “luxury hotel near the airport” or “budget hotel near the beach”, and the system will provide suggestions based on your query.

Additionally, users can use the quick-select buttons above the input box to quickly choose their preferences without typing, such as "swimming pool", "free parking" and "free breakfast". After entering the request, click the "Send" button to confirm your input and click "Confirm" to proceed to the recommendation page.

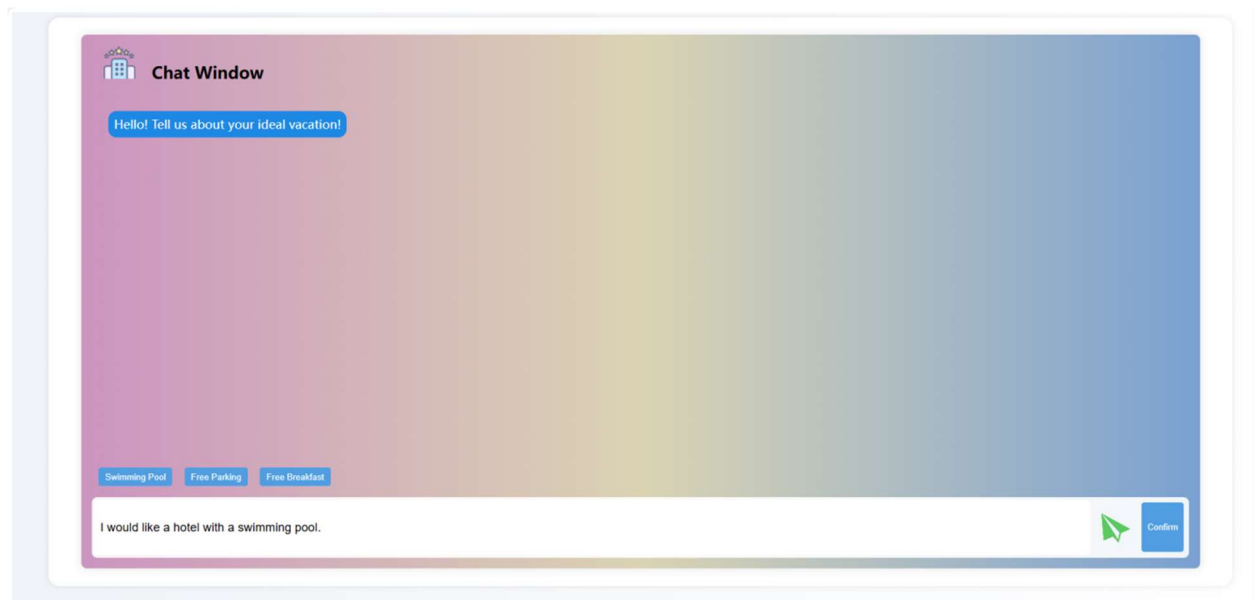


Figure 4: Chat Page

4. Preferences Selection Page

On the Preferences Selection Page, the system offers six hotel categories for users to choose from. Users can select one or more hotel types that they prefer by checking the boxes next to options like city hotel, beach resort, glamping, or mountain view. These preferences will help the system better understand user needs and provide more accurate recommendations.

After selecting your preferences, click the "Confirm Selection" button to submit your choices, and the system will generate personalized recommendations based on user's selections.

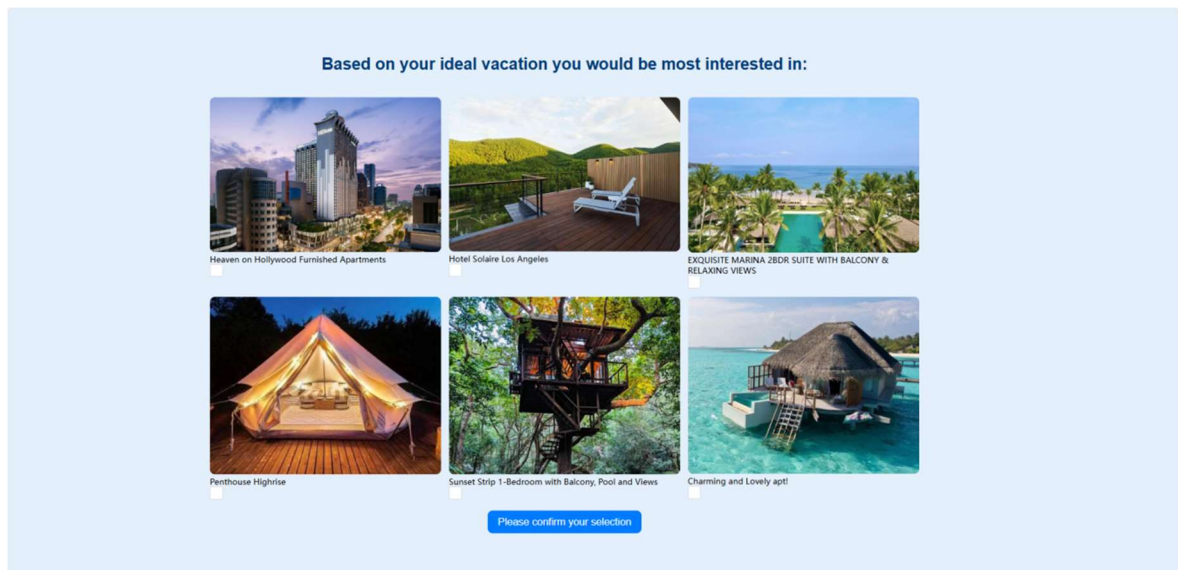


Figure 5: Preferences Selection Page

5. Rating Page

Based on the hotel types you selected on the previous page, the system will present several hotels that match your preferences. On this page, users are asked to rate each hotel on a scale of 1 to 5 stars. These ratings will help the system further refine its recommendations and improve the overall experience for future interactions.

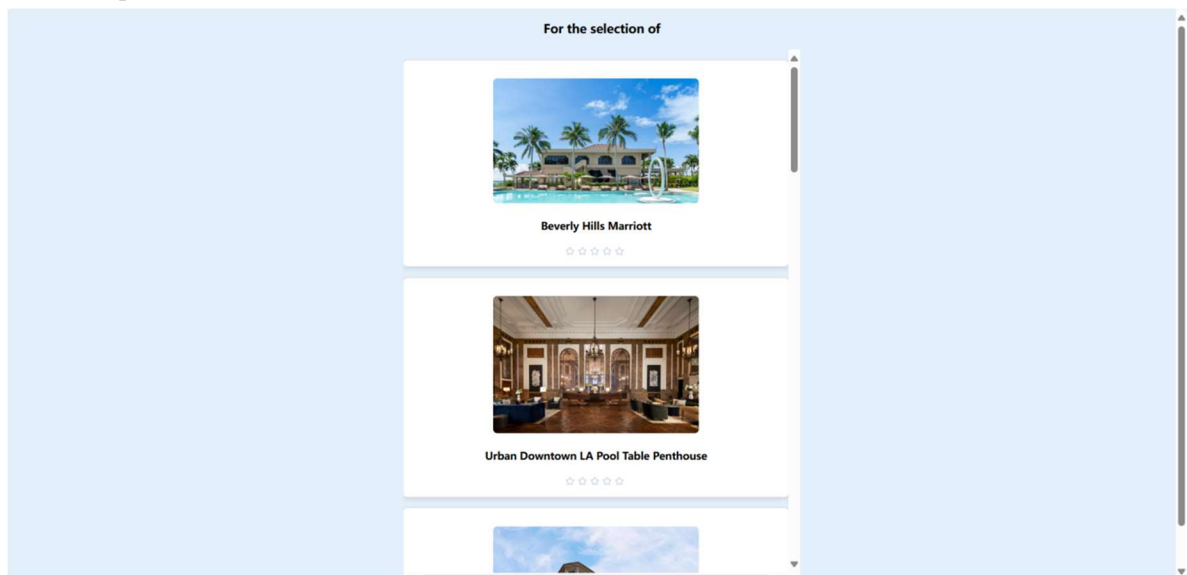


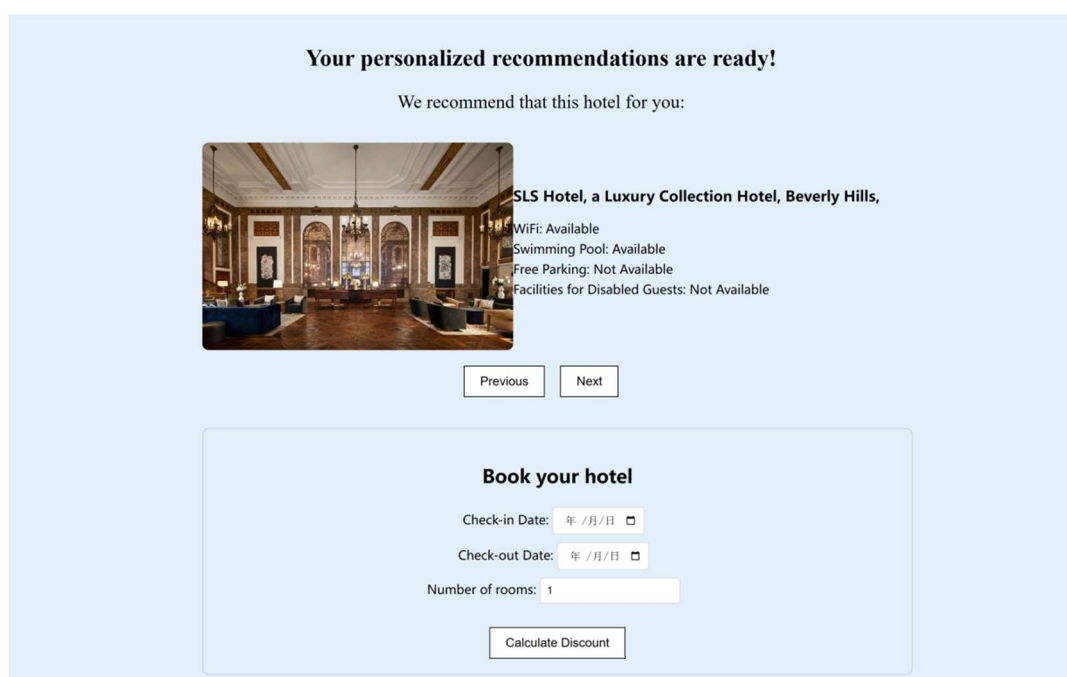
Figure 6: User Rating Page

6. Recommendation Results Page

This is the final recommendation results page, where the system provides a personalized list of hotels based on the user's preferences, past behavior, and ratings. Each recommended hotel entry includes hotel images, name, rating, price, and other key details.


While the system is generating your recommendations, a loading screen will appear with the message: "Give us a moment, we are finding a suitable recommendation for you...". Once the recommendations are ready, the results will be displayed.

Users can click on any hotel to view more detailed information and can proceed to book directly from this page. During the booking process, users can input their booking dates and number of guests, and by clicking the "Calculate Discount" button, the system will apply relevant business rules to calculate any available discounts (e.g., time-based promotional offers).



Your personalized recommendations are ready!

We recommend that this hotel for you:

 **SLS Hotel, a Luxury Collection Hotel, Beverly Hills,**

- WiFi: Available
- Swimming Pool: Available
- Free Parking: Not Available
- Facilities for Disabled Guests: Not Available

[Previous](#) [Next](#)

Book your hotel

Check-in Date:

Check-out Date:

Number of rooms:

[Calculate Discount](#)

Figure 7: Result Page

7. System Rating Page

On the System Rating Page, users can rate the system based on the recommendation result and discount information we provided. The system will allow user to rate from 1 to 5, and a big thanks information will be shown to user after rating.

The page will collect the user rate to our recommendation result, which will help us to promote our system.

Rate Our System



Thank You for Your Rating!

Your feedback helps us improve our service.

Figure 8: System Rating Page

Appendix D: Individual Report

Name: Li Qiuxian	Student ID: A0297598X
<p>1. Personal Contribution to group project: I have mainly participated in four modules of our project:</p> <p>a) Flask backend development</p> <p>The backend is equally important in our system as recommendation algorithms, as it helps us integrate user data, hotel datasets, and recommendation algorithms into a system that is convenient for users to use and manage the entire system. During the development process, I used Flask as a framework and wrote multiple interfaces to help implement different functions in the system. For example, I chose to import the trained recommendation algorithm model into the backend code instead of running the entire recommendation algorithm in the backend. This greatly saves the cost of our system and also makes the recommendation results appear faster. At the same time, I choose to directly retrieve user data from the database, avoiding the need for users to input data multiple times to obtain recommendation results. This process also allows users to better experience the intelligence of our system.</p> <p>b) Debug recommendation algorithm</p> <p>My next main job is to debug recommendation algorithms to ensure they run correctly in the backend. Because our recommendation algorithm is developed independently of the backend, I need to test it and package it for integration into the backend. I need to adjust the input and output formats of the recommendation algorithm and set detailed parameters in the recommendation algorithm to ensure the best performance in the current environment of the entire system. Specifically, I trained a pre trained model based on recommendation algorithms. And I also manually stored some edge data in the database to test whether our algorithm can handle these scenarios correctly, such as the cold start problem in collaborative filtering algorithms and whether it can successfully switch between different recommendation algorithms when there are insufficient scoring algorithms in the database.</p> <p>c) Demo Presentation</p> <p>In addition, I recorded a demonstration video of the system. In this part of the work, I presented in detail the entire operation process and various details of our system, helping users better understand and use our recommendation system.</p> <p>d) Write Report</p> <p>I also participated in the writing of the report, and my main responsibility was to write the parts I was involved in, such as data collection, backend development, recommendation algorithms, etc. In this section, I guarantee that the entire report has clear logic and standardized format, enabling readers to better understand the details and features of our system</p>	

2. What I have learnt:

- a) This recommendation system is my first time applying recommendation algorithms to practical scenarios. This project helped me better understand that developing a mature project not only requires ensuring smooth code execution, but also needs to bring in the user's perspective and modify system details to ensure user experience. Secondly, integrating these recommendation algorithms into the backend also helped me understand the importance of system overhead issues such as runtime and memory usage in practical development. I have also tried different ways to make our system as stable and smooth as possible.
- b) This project was developed by five team members, and as the team leader, I learned how to arrange tasks for different team members and communicate efficiently with them to ensure smooth and efficient project development. At the same time, I also realized that developing a mature project requires teamwork to efficiently complete it.

3. How can I apply the knowledge and skills in other situations or workplaces:

After this project development, I realized the importance of optimizing the project to adapt to actual industrial production. In future work and courses, I will also utilize my experience in backend development and recommendation algorithm debug to handle new tasks. For example, I want to try applying some of the technologies used in this project, such as Vue, Flask, etc., to my previous projects to make them more user-friendly.

Name: Gao Yunjia	Student ID: A0295502E
<p>In this project, our group developed and designed a hotel recommendation system. The system is designed to provide personalized hotel recommendations by analyzing user preferences and hotel characteristics.</p> <p>Main components I worked on:</p> <p>1. Content-Based Recommendation with NLP</p> <p>One of the core modules of the project is content-based recommendation, which uses Natural Language Processing to analyse user input and recommend suitable hotels based on their descriptions. For this module, I used SpaCy to process user input, including converting text to lowercase, removing stop words and punctuation, and applying lemmatization to ensure that words in different forms are recognized consistently. Then, I used the TF-IDF algorithm to vectorise the user's needs with the hotel's description for more accurate matching. For instance, when a user enters, 'I want a hotel with a swimming pool and free breakfast,' the system extracts 'swimming pool' and 'free breakfast' as keywords, then compares them with hotel descriptions in the dataset.</p> <p>2. Item-based collaborative filtering recommendation</p> <p>Another module of the system involves item-based collaborative filtering, where the system recommends hotels similar to those the user has rated highly, based on user rating data. Similarity between hotels is measured through user ratings for various hotels. For example, when users give high ratings to a few hotels, the system uses cosine similarity to calculate the feature similarity between these hotels and other hotels to recommend hotels that are similar to these highly rated hotels. In this way, the system helps users to discover more hotels that match their preferences and increases their satisfaction.</p> <p>In the implementation, there is a need to overcome the cold-start problem, i.e., new users do not have historical rating data, and I also designed an initial user interaction process that requires new users to rate several hotels so that the system can quickly model their preferences and generate recommendations.</p> <p>3. Front-end and back-end development</p> <p>In the front-end and back-end development of the project, I designed and implemented the interaction flow between the user interface and back-end logic. On the front end, users can enter their hotel preferences through a chat window by typing, for example, "I want a hotel near the beach." The system processes this input and recommends relevant hotels based on the description. I also designed a rating page where users can rate the recommended hotels, allowing the system to continuously refine its recommendation results. In addition to creating these specific pages, I worked on achieving a consistent style across all pages, optimizing layout, and making user experience improvements to ensure seamless navigation and interaction. For the back-end, I used the Flask framework to handle user inputs from the front end, invoke recommendation algorithms, and return results. User preference data, hotel information, and rating data are stored in a database, enabling real-time retrieval and updates</p>	

as the system operates.

Learning and Growth

In developing both content-based and item-based collaborative filtering recommendations, I solidified my understanding of recommendation algorithms and how they can be effectively applied to real datasets. This process helped me become proficient in various similarity metrics and taught me how to address challenges like the cold-start problem, deepening my understanding of the complexities of recommendation systems in real-world applications.

Throughout the project, I utilized various tools and technologies, including NLP, recommendation algorithms, web frameworks, and database management. By integrating these technologies, I developed a robust, multi-functional recommendation system, which improved my skills in handling and analyzing real-world data effectively.

I designed the front-end interfaces and implemented backend logic for the project. Leveraging the Flask framework, I built an efficient backend to handle user requests and return recommendation results. Additionally, I designed user-friendly front-end pages to facilitate preference input and recommendation display. This hands-on experience not only strengthened my web development skills but also deepened my understanding of how to bridge complex machine learning algorithms with practical applications to create a seamless user experience.

The project encompassed a wide range of technical domains, including machine learning, NLP, and collaborative filtering. To complete the design and implementation on time, I had to independently learn and master various new concepts. This experience honed my time management skills and highlighted the importance of self-directed learning and quickly acquiring new knowledge.

By working with these technologies, I gained not only technical proficiency but also a comprehensive understanding of the end-to-end recommendation system development process, from algorithm selection to front-end/back-end integration and user experience optimization, laying a solid foundation for future system development projects.

Name: Zhao Lanting	Student ID: A0297676A
<p>In our hotel recommendation system project, I was responsible for front-end development, focusing on creating an intuitive and visually appealing user interface to enhance user interaction with the system. Throughout the development process, I paid close attention to user experience design principles, ensuring that the interface was not only functional but also accessible and responsive across various devices. This experience significantly deepened my understanding of user-centered design and the importance of building interfaces that adapt to user preferences and behavior.</p> <p>I also contributed to the Proposal by writing the Introduction, Market Research, and Business Rule sections. This allowed me to engage in thorough research, helping me better understand market demands and the importance of business rules in driving effective recommendations.</p> <p>For the project video, I handled the System Design, Structure, and Technology sections, providing detailed explanations of our technical choices, system components, and functional implementation. This required a clear, structured approach to presenting complex technical information, which was both a challenge and an opportunity to improve my ability to communicate technical concepts in an accessible way.</p> <p>For the Final Report, I wrote the Technology Stack, System Components, Implementation Details, Introduction, and Business Rule sections. Additionally, I compiled the Appendix of the Report: Installation and User Guide to ensure users can easily install and navigate the system.</p> <p>Overall, my involvement in both the technical development and the project's documentation helped me strengthen my skills in front-end design, technical communication, and project documentation. This project has underscored the importance of balancing technical proficiency with clear communication, as both are vital to the success of a user-focused application. Moving forward, I am motivated to build on these skills, aiming to contribute to projects that demand a blend of technical and communicative expertise.</p>	

Name: Zhao Lanting	Student ID: A0297676A
<p>1. Your personal contribution to the project.</p> <p>In this project, I was mainly responsible for the backend development of the hotel recommendation system, including data preprocessing, cluster analysis, user rating simulation, and item based collaborative filtering recommendation function. I have developed a clustering analysis module to cluster hotels based on different features, providing a data foundation for recommendation systems; And by simulating user ratings and collaborative filtering algorithms, a complete personalized recommendation system was constructed. In addition, I have also implemented cosine similarity calculation and similar hotel recommendation logic for the highest rated hotel by the user, ensuring that the system can make accurate recommendations based on user preferences.</p> <p>2. What you have learnt from the project.</p> <p>In this project, I learned how to use K-Means clustering algorithm and data standardization techniques to process and analyze data, enhancing my ability in data preprocessing and feature engineering in recommendation system development. At the same time, I gained a deep understanding of collaborative filtering algorithms, especially item based collaborative filtering and its application in recommendation systems. In addition, I have also mastered how to use the surprise library to implement recommendation algorithms and learned how to train and test models on sparse datasets.</p> <p>3. How you can apply this in future work-related projects.</p> <p>In the future, in my work, I can apply the clustering analysis, collaborative filtering, and recommendation algorithms learned in this project to other personalized recommendation projects, such as e-commerce product recommendation or content push. My experience in data standardization, user needs analysis, and algorithm optimization will help me better handle complex datasets and build accurate recommendation systems. At the same time, I can use a combination of NLP and collaborative filtering methods to further optimize the user experience and enhance the commercial value of the project.</p>	