

Hotel Recommendation System Using Machine Learning

BY

WONG WAI ON

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) COMMUNICATION AND
NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

© 2025 Wong Wai On. All rights reserved

This Final Year Project report is submitted in partial fulfilment of the requirements for the Bachelor of Information Systems (Honours) Communications and Networking degree at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to show my utmost appreciation and sincerest gratitude to my supervisor, Dr. Norliana Binti Muslim, for permitting me to carry out the Hotel Recommendation System Using Machine Learning project. This has been a very enriching experience, as it marks my first experience in the areas of machine learning and data science. Your guidance, patience, and unwavering encouragement have left an indelible mark on my learning and personal development for which I remain forever grateful to your guidance on this journey. I would also like to offer my heartfelt thanks to my friend Lau Ka Ming for his consistent encouragement, useful feedback, and support, which made me keep my resolve intact and focused even during trying times. Also, I am forever indebted to my family for their unwavering love, care, and faith in me; they have been my biggest source of inspiration and strength. This project not only technically equipped me but also enlightened me to the place of a good support system, and I feel truly blessed to have had such wonderful people along with me on this journey.

ABSTRACT

In recent times, choosing the appropriate hotel destination and making bookings has become increasingly complex due to the rapidly growing volume of available online information. The importance of recommender systems (RSs) is rising as they help users make informed decisions and provide comprehensive insights into products or services. Managing user-generated data such as votes, ratings, views, and reviews presents significant challenges. There are three objectives in the study, which is to perform data preprocessing on the Google Reviews dataset for hotels in Perak using an instant data scraper, to develop three suitable machine learning models on the cleaned dataset and evaluate their performance, and to propose a recommendation system based on the developed machine learning models. The methodology includes data scraping, preprocessing, implementation of machine learning techniques such as Naïve Bayes, Random Forest, and Support Vector Machine (SVM), and proposes a recommendation system. The system integrates these models to provide hotel recommendations based on each user's preferences. The results show that the proposed model is effective and generates recommendations for the user. The future work includes expanding the dataset, refining the recommendation algorithm, using natural language processing techniques with the addition of multilingual reviews, and deploying the system as a user-friendly application or mobile application.

Area of Study: Recommendation System, Machine Learning

Keywords: Data Preprocessing, Data Scraping, Google Reviews, Naïve Bayes, Random Forest, Support Vector Machine (SVM) , User-friendly application and Mobile Application

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF SYMBOLS	x
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	2
1.2 Objectives	3
1.3 Project Scope and Direction	4
1.4 Contributions	5
1.5 Report Organization	6
CHAPTER 2 LITERATURE REVIEW	7
2.1 Previous Work on Recommendation Systems	7
2.1.1 Strengths and Weakness on Recommendation System	8
2.2 Previous Work on Machine Learning	10
2.1.1 Strengths and Weakness on Machine Learning	15
CHAPTER 3 SYSTEM METHODOLOGY	16
3.1 Proposed Method/Approach	16
3.2 System Design	
3.2.1 System Architecture Diagram	23
3.2.2 Use Case Diagram	24
3.2.3 Activity Diagram	25
3.2 Project Timeline	26

CHAPTER 4 SYSTEM DESIGN	28
4.1 Simple Block Diagram	28
4.2 System Components Specifications	29
4.4 Data Description	30
4.5 Machine Learning	33
CHAPTER 5 SYSTEM IMPLEMENTATION	38
5.1 Hardware Setup	38
5.2 Software Setup	38
5.3 Setting and Configuration	40
5.4 System Operation (with Screenshot)	40
5.5 Implementation Issues and Challenges	46
5.6 Concluding Remark	46
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	47
6.1 Testing Setup and Result	47
6.1.1 Test case	51
6.2 Project Challenges	56
6.3 Objectives Evaluation	57
6.4 Concluding Remark	58
CHAPTER 7 CONCLUSION AND RECOMMENDATION	59
7.1 Conclusion	59
7.2 Future Work	59
REFERENCES	61
APPENDIX	A-1
USER TESTING	B-1
COPYRIGHT	C-1
POSTER	D-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 3.1	Operational Framework	16
Figure 3.2	Instant Data Scraper	17
Figure 3.3	Jupyter Lab	18
Figure 3.4	Data Preprocessing	19
Figure 3.5	Cleaned_Review	20
Figure 3.6	System Architecture Design for the Proposed System	23
Figure 3.7	Use Case Diagram for proposed system	24
Figure 3.8	System Architecture Design for the Proposed System	25
Figure 3.9	Timeline of FYP I	26
Figure 3.10	Timeline of FYP II	27
Figure 4.1	System Block Diagram	28
Figure 4.2	Bar Graph of Distribution of Ratings Pie Chart of	30
Figure 4.3	Distribution of Ratings Bar Graph	31
Figure 4.4	Bar Graph Top 10 Hotels by Average Rating	32
Figure 4.5	Naïve Bayes Model	33
Figure 4.6	SVM Model	34
Figure 4.7	Random Forest Model	35
Figure 5.1	Anaconda Prompt	40
Figure 5.2	Train Model	41
Figure 5.3	Code Source File	42
Figure 5.4	Flask Application Routing and Hotel Filtering Code	43
Figure 5.5	Top five and contact	44
Figure 5.6	Function of Recommender System in the Main Page	45
Figure 6.1	Home Page	47
Figure 6.2	Hotel Recommendation System	48
Figure 6.3	Top five Hotels	49
Figure 6.4	Google Map	50

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Summary of Research Studies Recommender System Approaches	9
Table 2.2	Overview of Content-Based Recommender Techniques	11
Table 2.3	All the prior literature on Sentiment Analysis and machine learning	13
Table 4.1	Statistical Measurement of Naive Bayes, SVM and Random Forest	36
Table 5.1	Specifications of laptop	36
Table 5.2	Software Utilized in the Project	37
Table 6.1	User Test (app.py)	52
Table 6.2	User Test (Index.html)	53
Table 6.3	User Test (Top5.html)	54
Table 6.4	User Test (Contactus.html)	55

LIST OF SYMBOLS

Symbol	Description
$P(C X)$	Posterior probability of class C given the feature vector X.
$P(X C)$	Likelihood of the feature vector X given class C.
$P(C)$	Prior probability of class C.
$P(X)$	Evidence or the total probability of the feature vector X across all classes.
$f(x)=w \cdot x+b=0$	Equation of the hyperplane in SVM classification.
w	Weight vector perpendicular to the hyperplane in SVM.
x	Feature vector.
b	Bias term in SVM.
$\min \frac{1}{2} \ w \ ^2$	Optimization objective in SVM to maximize the margin.
y^i	Class labels for the training examples.
x^i	Training examples feature vectors in SVM.
\hat{y}	Predicted class in Random Forest.
$h_1(x), h_2(x), \dots, h_n(x)$	Predictions from individual decision trees in Random Forest.
G	Gini impurity used as a splitting criterion in decision trees.
p^i	Proportion of instances belonging to class i in a particular node.
C	Number of classes in classification tasks.

LIST OF ABBREVIATIONS

<i>ML</i>	Machine Learning
<i>SVM</i>	Support Vector Machine
<i>CL</i>	Collaborative Filtering
<i>RS</i>	Recommendation System
<i>SVD</i>	Singular Value Decomposition
<i>TF-IDF</i>	Term Frequency-Inverse Document Frequency
K-NN	k-Nearest Neighbors

Chapter 1

Introduction

The increase in the popularity of online review sites indicates a paradigm shift in how customers assess goods and services, particularly in the lodging sector. Online hotel reviews are becoming increasingly important as Malaysia gains popularity as a tourist destination. These reviews serve as summaries of customer experiences, offer insightful commentary to hotel staff, and greatly influence the choices of prospective customers [1]. People nowadays often look up reviews before choosing to use a product or service [2]. More information can be extracted from review data and processed to enhance hotel operations, but manual analysis or dataset surveys are limited in the amount of data that can be collected and are expensive [3].

Online reviews and big data are becoming increasingly important in the hospitality industry, and studies are only now beginning to examine how big data practices benefit companies [4]. The growing reliance on online reviews and big data analytics signifies a transformative shift in the hospitality industry, driven by the need to better understand and cater to customer preferences and experiences. In the hospitality industry, customers usually share their experiences through comments and reviews of hotel services. The evolution of social media enables customers to provide relevant recommendations on a collaborative platform. Customer review data is important because it affects a hotel. Dataset analysis of customer reviews allows access to and use of limited data [5].

has various benefits, one of which is that it is better to the traditional way for getting client feedback. In the first place, this allows us to manage large data sets while avoiding the human processing challenges that arise in such instances. This machine learning approach outperforms traditional manual methods, which are frequently time-consuming and prone to human error when it comes to analyzing consumer input. Similarly, data processing becomes a crucial aspect of sentiment analysis and other hard linguistic jobs as the Christmas on the net gets larger [6].

1.1 Problem Statement and Motivation

Problem Statement

With the advent of online platforms, customer feedback has become more accessible and influential than ever before. Especially for hotels in Malaysia, the challenge lies in effectively managing and understanding the sentiment expressed in Google reviews. Manually analyzing a large number of these reviews is both time-consuming and resource-intensive, resulting in missed opportunities for hotels to increase guest satisfaction and improve overall service quality. Furthermore, travellers often struggle to find hotels that best match their individual preferences and needs. The lack of a robust recommendation system means that customers may not be presented with the most suitable hotel options, leading to a suboptimal travel experience and potentially lower booking rates for hotels.

A recommendation system that leverages user preferences, historical booking data, hotel attributes, location information, and user reviews can significantly enhance the decision-making process for customers, ensuring they find the best possible accommodations. Therefore, this sentiment analysis of Google hotel reviews using machine learning algorithms provides a solution that enables hotels to gain timely insights, identify areas for improvement, proactively address negative feedback, customize marketing strategies, and maintain a good online reputation. Additionally, by integrating a recommendation system, hotels can offer suggestions to customers, enhancing their overall travel experience and increasing booking rates. Recommendation systems using machine learning are essential for hotels in Malaysia to effectively manage customer feedback, improve service quality, and provide customized hotel recommendations.

Motivation

The motivation behind this project is to provide hotels in Malaysia with a way to increase customer favorability towards their hotels by utilizing the wealth of information embedded in customer reviews. Positive feedback allows to keep that level of excellence and negative ratings can be the rock to self-improvement. By reading reviews, customers can understand the real experience. Customers can evaluate a hotel's value proposition by comparing the price. So customers often encounter difficulties finding hotels that align with their specific preferences and needs, Highlighting the necessity for a sophisticated recommendation system. Simultaneously, a recommendation system can provide hotel suggestions, enhancing the

overall travel experience and increasing the likelihood of bookings. It is to harness the power of machine learning to transform customer feedback into actionable insights and recommendations, ultimately driving higher standards in the hospitality industry.

1.2 Research Objectives

The main objective of this project is to design a hotel recommendation system using machine learning techniques. Thus, the research objectives are stated as follows:

1. To perform data preprocessing on the Google Reviews dataset for hotels in Perak using an instant data scraper.

A dataset of hotel reviews, specifically from the Google platform for hotels in Perak, Malaysia, is prepared. The "Instant Data Scraper" is a tool that is used to automate the process of extracting these reviews from Google. Key features such as the author of the review, the source of the review (link), the review rating, the content of the review itself, and the hotel name are extracted. Once the dataset is gathered, it needs to be cleaned and prepared for analysis. This step includes tasks such as removing duplicates, handling missing values, standardizing text (e.g., converting to lowercase, removing punctuation), and possibly translating reviews if they are in different languages. The goal is to ensure that the dataset is of high quality and consistent, which is crucial for training effective machine learning models.

2. To develop three suitable machine learning models on the cleaned dataset and evaluate the performance.

This objective involves building various machine learning models to analyze the data and make predictions or classifications. The developed models are evaluated based on accuracy, precision, and F1-score, among other relevant metrics. Once the dataset has been preprocessed and cleaned, three machine learning models are developed to analyze hotel reviews, which may include classification models to predict review sentiment (e.g., positive, neutral, or negative), regression models to estimate review ratings based on text content, or clustering models to group hotels based on similar review patterns. The selection of models depends on the dataset's nature and the analysis objectives. Once trained, the models are evaluated using metrics like accuracy, precision, recall, and F1-score to assess their ability to classify reviews into different sentiment categories. The best-performing model is then selected for integration into the recommendation system.

3. To propose a hotel recommendation system based on the developed machine learning models.

The final objective of this project is to propose a hotel recommendation system on a web platform using the best machine learning models to provide hotel suggestions based on user preferences and previous reviews. The system is implemented as an application on a website that provides an interactive interface for users to receive recommendations. It includes functions such as lowest rating, hotel name, top 5 hotel ratings, star ratings, and keyword-based review comparison to allow users to narrow down the results according to their specific needs. These features are integrated into the application through various paths and templates to ensure dynamic interaction with the dataset. The system was successfully evaluated based on previous user experience and ratings based on recommendation accuracy.

1.3 Project Scope and Direction

Google Hotel Reviews

Google Reviews was chosen as the primary data source for this project because it is easily accessible and widely used among consumers. On this platform, customers often leave feedback about their stay experience at a hotel. Analyzing data from Google Reviews will provide a wide range of customer opinions and insights that are valuable for gauging public sentiment and improving hotel services. We collect data about hotels in Perak from Google reviews. On the other hand, Ipoh in Perak is a city with a long history. By including this city in the data collection process, this project has gained insights from this hotel landscape to understand the hotel services in this place.

Data gathering

The scope of data gathering involves using Google Maps to search for hotels in Perak, filtering results by specific dates and price ranges. The reviews collected are from after the year 2000, ensuring that all data is relevant and up-to-date for modern analysis. Additionally, the reviews are filtered to include only those written in English, ensuring consistency in language for more accurate analysis. Utilizing the Instant Data Scraper tool, reviews from selected hotels are extracted, capturing key data points such as author, rating, review text, and review link. This information is saved into structured formats like CSV or Excel for analysis. The process ensures comprehensive data collection for understanding customer satisfaction and identifying

common issues across various hotels in Ipoh, Perak. The accompanying image illustrates the tools and steps used in this data-gathering method.

Machine Learning

This project utilizes a machine learning model to analyze large amounts of text data from online reviews. One of the machine learning algorithms such as Naive Bayes, Support Vector Machines and Random Forest used on the project to categorize the sentiment analysis. These algorithms help to process and interpret the complex natural language used in the comments, ensuring that the sentiment analysis is both efficient and reliable.

Recommendation System

The scope of the recommender system within this project involves developing an algorithm to suggest hotels to users based on their preferences and past behavior. By analyzing the collected review data, the recommender system identifies patterns and trends in user feedback. The goal is to provide hotel recommendations that enhance user satisfaction and decision-making, leveraging the rich dataset of customer opinions and experiences gathered from Google Reviews.

1.4 Contributions

This project contributes to the field of hotel recommendations by developing machine learning models tailored for Malaysian hotel reviews, considering linguistic nuances and cultural sensitivities. By leveraging classifiers such as Naïve Bayes, Support Vector Machines (SVM), and Random Forest, it effectively categorizes customer sentiments while integrating language preprocessing techniques to handle informal expressions and mixed-language reviews. The structured and labeled dataset enhances machine learning training, serving as a valuable resource for future research. Additionally, the project develops a hotel recommendation system that provides tailored suggestions based on user preferences and sentiment insights, improving decision-making by filtering hotels based on customer satisfaction levels. Hotels can utilize the machine learning results to identify strengths, refine services, and implement targeted marketing strategies. Furthermore, the methodology is scalable and adaptable, enabling its application to hotel reviews in other regions and languages. By combining machine learning models and an intelligent recommendation system, this project offers a data-driven approach to enhancing customer experience while helping hotels optimize their services based on real customer feedback.

1.5 Report Organization

The details of this research are shown in the following chapters. Chapter 1 provides an overview of the project, including the problem statement, objectives, project scope, and contributions. Chapter 2 reviews previous works related to machine learning, sentiment analysis, and recommendation systems. Chapter 3 discusses the proposed methodology, including system models, diagrams, and the project timeline. Chapter 4 focuses on system design, presenting block diagrams, component specifications, and interaction operations. Chapter 5 details the system implementation, including hardware and software setup, configuration, and system operation. Chapter 6 evaluates and discusses the system's performance using testing metrics, highlighting project challenges and assessing the objectives. Finally, Chapter 7 presents the conclusion and recommendations for future improvements.

Chapter 2

Literature Review

2.1 Previous Work on Recommendation System

Recommendation systems have become an integral part of online platforms, offering suggestions to users based on their preferences and past behaviour. The development of recommender systems has seen significant advancements, with various approaches being employed to enhance recommendation accuracy and user satisfaction.

Early work laid the groundwork for collaborative filtering techniques, which became a cornerstone in the development of recommender systems [7]. Collaborative filtering methods, which include user-based and item-based approaches, leverage the historical interactions between users and items to generate recommendations [8]. Further improvements in these techniques were achieved by introducing matrix factorization methods, such as Singular Value Decomposition (SVD), to handle large-scale datasets efficiently.

Content-based filtering is another widely used approach, where recommendations are generated based on the similarity between items' attributes and the user's preferences. The effectiveness of content-based filtering has been demonstrated in scenarios where user interaction data is sparse, highlighting the importance of feature engineering in capturing the relevant attributes of items to enhance recommendation accuracy [9].

Hybrid recommender systems, which combine collaborative and content-based filtering, have shown significant promise in addressing the limitations of individual approaches [10]. Various methods of combining different recommendation techniques have been illustrated to achieve better performance [11]. More recent studies have explored deep learning models in hybrid recommender systems, demonstrating the potential of neural networks in capturing complex user-item interactions.

In the context of hotel reviews, recommender systems have been used to suggest dining options based on user preferences and review sentiment. Some have developed context-aware hotel recommender systems that consider the sentiment of reviews along with user preferences and contextual factors such as location and time. These works highlighted the importance of

Bachelor of Information Technology (Honours) Communications and Networking
Faculty of Information and Communication Technology (Kampar Campus), UTAR

incorporating sentiment analysis in enhancing the relevance of recommendations in the hotel domain.

2.1.2 Strengths and Weakness on Recommendation Systems

Recommendation systems, integral to many online platforms, present significant strengths and weaknesses. Among their strengths, recommendation systems enhance user experience by providing content, products, or services, thereby increasing user engagement and satisfaction. They help users discover relevant items they might not have found independently, improving the efficiency of their interactions with digital platforms. For businesses, recommendation systems can drive sales and retention by presenting users with tailored suggestions based on their preferences and behaviour, leading to higher conversion rates and customer loyalty. Additionally, these systems can handle vast amounts of data and continuously improve their accuracy through machine learning techniques, adapting to changing user preferences over time.

However, recommendation systems also have notable weaknesses. One major limitation is the "filter bubble" effect, where users are repeatedly shown similar types of content, potentially limiting their exposure to diverse information and viewpoints. This can lead to a narrowing of perspectives and reduced discovery of new interests. The quality of recommendations heavily depends on the quantity and quality of user data; sparse or noisy data can result in less accurate suggestions. Additionally, implementing and maintaining effective recommendation systems require significant computational resources and expertise in data science, which can be a barrier for smaller organizations. Privacy concerns are another critical issue, as collecting and analyzing user data can raise ethical questions and require robust measures to protect user information. Lastly, recommendation systems may struggle with the "cold start" problem, where new users or items lack sufficient data to generate meaningful recommendations, impacting their initial effectiveness.

Table 2.1 Summary of Research Studies Recommender System Approaches

Type of RS	Approach	Author	Total
Collaborative Filtering Recommender	Memory-based CL: Item-based	[12], [13], [14], [15], [16]	5 (14%)
	Memory-based CL: User-based	[17], [18], [19]	3 (8%)
	Memory-based CL: User-based and Item-based	[20], [21]	2 (5%)
	Model-based CL: Matrix Factorization	[22], [23], [24]	3 (8%)
Content-Based Recommender	Classification model approach	[25], [26], [27], [28], [29], [30], [31], [32], [33], [34]	10 (28%)
	Vector spacing method	[35], [36]	2 (5%)
Hybrid recommender	Weighted recommender	[37], [38]	2 (5%)
	Feature combination	[39], [40], [41], [42]	4 (11%)
	Meta-level recommender	[43], [44], [45]	3 (8%)

Based on the summary of research studies on recommender system approaches presented in Table 2.1, it is evident that the content-based recommender classification model approach is a widely adopted approach for recommender systems. This approach is notably prevalent in the literature, representing 28% of the studies reviewed, which indicates its reliability and effectiveness. Content-based recommenders are particularly adept at providing recommendations by analyzing the intrinsic properties of items that users have shown interest

in. This allows the system to offer precise suggestions tailored to individual user profiles, making it a versatile choice for a wide range of applications.

Given this strong prevalence and the demonstrated success of content-based recommendation methods had decided to employ a content-based approach for developing my recommendation system. Specifically, it focus on Classification model approach due to their extensive use and proven effectiveness in prior research. The Content-Based Recommender Classification model a well-founded choice, aligning with current trends and best practices in recommender system research.

Content-Based Recommender

A Content-Based Recommender is a type of recommendation system that suggests items to users based on the attributes of the items themselves and the user's previous interactions with similar items [26]. Unlike other recommendation systems, such as collaborative filtering, which relies on the preferences and behaviors of other users, content-based recommenders focus solely on analyzing the characteristics of the items to predict what a specific user might enjoy [28]. For example, in a movie recommendation system, each movie would be described by features such as genre, director, cast, and keywords [29].

The system creates a profile for each user by examining the features of the movies they have watched or rated highly. It then recommends new movies that share similar attributes to those the user has previously liked. This approach ensures that the recommendations are and directly aligned with the user's individual preferences, making it particularly effective in situations where user data is limited or when new users are added to the system [31].

Table 2.2 Overview of Content-Based Recommender Techniques

Content-Based Recommender Techniques	Description	Advantages	Limitations
Naive Bayes (Classification Model Approach)	A probabilistic classification model that predicts item	- Simple, fast, and effective for binary or categorical features	- Are dependent on human ratings

	categories based on features		<ul style="list-style-type: none"> - Cold star problem for new user and new item - Sparsity problem of rating matrix - Limited scalability for large datasets
Decision Trees (Classification Model Approach)	Splits data into subsets based on feature values to classify items	- Handles both numerical and categorical data	- Assumes feature independence, which may not hold true in all contexts
(SVM) (Classification Model Approach)	Separates data into classes using a hyperplane in a high-dimensional space	- Effective for high-dimensional spaces	- Not scalable to very large datasets and can be computationally expensive
k-Nearest Neighbors (k-NN)	A lazy learning algorithm that classifies items based on the similarity to their nearest neighbors	- Easy to understand and implement	- Computationally expensive for large datasets

2.2 Previous Works on Machine Learning

Machine learning has evolved significantly over the past few decades, contributing to a wide array of fields, including computer vision, natural language processing, and predictive analytics [16]. Early work in machine learning focused on symbolic AI and rule-based systems, which laid the groundwork for later developments. In the mid-20th century, pioneers like Arthur Samuel introduced the concept of machine learning with programs that could play checkers and improve over time [17]. The 1980s and 1990s saw the rise of neural networks, particularly with the backpropagation algorithm, which allowed for the training of multi-layered networks [18].

Support vector machines and decision trees also became popular during this period for classification tasks.

More recently, the advent of deep learning has revolutionized machine learning, enabling breakthroughs in tasks like image and speech recognition, natural language understanding, and autonomous driving [19]. The development of large-scale datasets, powerful GPUs, and sophisticated algorithms has further accelerated progress, making machine learning an integral part of modern AI research and applications [20].

Table 2.1 briefly describes the previous literature on sentiment analysis using machine learning methods. [12] investigates hotel reviews from TripAdvisor, focusing on a hybrid recommendation system. The study employs the K Nearest Neighbor achieving an accuracy of 89.2%. [13] also utilizes hotel reviews from TripAdvisor but applies a Content-Based Recommender system. The study employs Support Vector Machine as the algorithm, achieving a high accuracy of 93%. [14] focuses on Collaborative Filtering Recommender systems using TripAdvisor hotel reviews. The study implements Naïve Bayes for sentiment analysis, resulting in an accuracy of 85.0%. While lower than in [13], this accuracy still demonstrates the capability of Naïve Bayes in collaborative filtering contexts. [15] examines hotel reviews from TripAdvisor with a hybrid recommendation system. The study applies Logistic regression for sentiment analysis and achieves an accuracy of 85%.

For the work [16] conducts sentiment analysis on TripAdvisor hotel reviews using a Collaborative Filtering Recommender system. The study utilizes the Support Vector Machine (SVM) algorithm, which yields a high accuracy of 95%. [17] switches the platform to Amazon but still focuses on hotel reviews. The study employs a Content-Based Recommender system and uses the Random Forest algorithm for sentiment analysis, resulting in an accuracy of 84.60%. [18] looks at TripAdvisor hotel reviews within a Collaborative Filtering Recommender system. The study applies Naïve Bayes for sentiment analysis, achieving an accuracy of 74%. [19] explores both TripAdvisor and Amazon platforms, focusing on hotel reviews without conducting sentiment analysis. The study uses a Collaborative Filtering Recommender system with a Random Neural Network, achieving a high accuracy of 94%.

The study of [20] omits sentiment analysis, analyzing hotel reviews from TripAdvisor with a Collaborative Filtering Recommender system. It utilizes a Recurrent Neural Network (RNN) and achieves an accuracy of 93.5%. [21] revisits TripAdvisor hotel reviews, implementing a Collaborative Filtering Recommender system. The study uses a combination of Random Forest and Naïve Bayes for sentiment analysis, resulting in a lower accuracy of 62.65%. [22] focuses

on TripAdvisor hotel reviews with a Collaborative Filtering Recommender system. The study uses a Clustering algorithm, achieving an accuracy of 85%. [23] analyses TripAdvisor hotel reviews using a CL Recommender system. The study employs both Naïve Bayes and SVM algorithms, achieving accuracies of 72% and 89%, respectively.

In the work of [24], it investigates TripAdvisor hotel reviews with a Collaborative Filtering Recommender system, utilizing a Deep Neural Network for sentiment analysis. The study achieves an accuracy of 75%, indicating that deep learning techniques can be effective but may require more optimization to outperform traditional methods. [25] looks at hotel reviews from both TripAdvisor and Expedia, employing a CL Recommender system. The study uses SVM and achieves an accuracy of 74%. Lastly, [26] examines TripAdvisor hotel reviews using a CL Recommender system. The study applies both Neural Networks and SVM, achieving accuracies of 85.6% and 80.7%, respectively. These studies collectively showcase the effectiveness of various machine learning models in sentiment analysis across diverse review domains.

Table 2.3 All the prior literature on Sentiment Analysis and machine learning

Referen ces	Platform	Dataset	Sentiment Analysis	Algorithms or Models	Accuracy
[12]	Trip Advisor	Hotel review	No	K Nearest Neighbor	89.2%
[13]	Trip Advisor	Hotel review	No	Support Vector Machine	93%
[14]	Trip Advisor	Hotel Review	Yes	Naïve Bayes	85.0
[15]	Trip Advisor	Hotel Review	Yes	Logistic regression	85%
[16]	Trip Advisor	Hotel Review	No	Support Vector Machine	95%
[17]	Amazon	Hotel Review	Yes	Random Forest	84.60%
[18]	TripAdvisor	Hotel Review	Yes	Naïve Bayes	74%

[19]	TripAdvisor, Amazon	Hotel Review	No	Random Network	Neural	94%
[20]	TripAdvisor	Hotel Review	No	Recurrent Network	Neural	93.5%
[21]	TripAdvisor	Hotel Review	Yes	Random Naïve Bayes	Forest,	62.65 %
[22]	TripAdvisor	Hotel Review	Yes	Clustering		85%
[23]	TripAdvisor	Hotel Review	No	Naïve Support Machine	Bayes, Vector	72% , 89%
[24]	TripAdvisor	Hotel Review	Yes	Deep Network	Neural	75%
[25]	TripAdvisor	Hotel Review	Yes	Support Machine	Vector	80%
[26]	TripAdvisor	Hotel Review	No	Neural Network, Support Machine	Vector	85.6%, 80.7%

Based on the analysis of prior literature in Table 2.3, various machine learning algorithms have been utilized for sentiment analysis on hotel reviews, with Naïve Bayes being one of the commonly applied models. Despite the range of methods employed, including Support Vector Machines and Random Forests, Naïve Bayes has consistently shown reliable performance, particularly in scenarios with similar datasets. Thus, the classification models in this work are built based on these three supervised machine-learning methods. Therefore, this study adopts these three supervised machine learning models, which are SVM, Random Forests, and Naïve Bayes to the development of a sentiment-aware hotel recommendation system, as they collectively offer a balance of performance, interpretability, and scalability.

2.2.2 Strengths and Weakness on Machine Learning

ML boasts considerable strengths and notable weaknesses. On the strength side, ML automates and optimizes complex tasks, enabling rapid analysis and decision-making based on large datasets [46]. Its ability to learn and improve from data allows for the creation of highly

accurate predictive models, which can drive innovations in diverse fields such as healthcare, finance, and technology. For instance, ML can enhance diagnostic accuracy in medical imaging, optimize stock trading strategies, and personalize user experiences in online services [47]. Additionally, ML's scalability means it can handle vast amounts of data, uncovering patterns and insights that would be imperceptible to human analysts.

Conversely, ML also has weaknesses that limit its effectiveness. One major challenge is the need for extensive and high-quality training data, which can be costly and time-consuming to obtain. Poor or biased data can lead to flawed models that perpetuate existing biases or produce inaccurate predictions [48]. Furthermore, ML models often operate as "black boxes," making it difficult to understand and interpret their decision-making processes, which can be problematic in critical applications requiring transparency and accountability. The computational intensity of training and running ML models also demands significant resources and infrastructure [49]. Additionally, while ML can generalize from data, it may struggle with novel or outlier situations not represented in the training data, leading to potential failures in unpredictable scenarios [50].

Chapter 3

System Methodology/Approach

3.1 Proposed Method/Approach

The project's processes are categorized into separate phases in the development. This includes the project planning, Corpus development, machine learning implementation, and performance analysis, followed by the result and discussion.

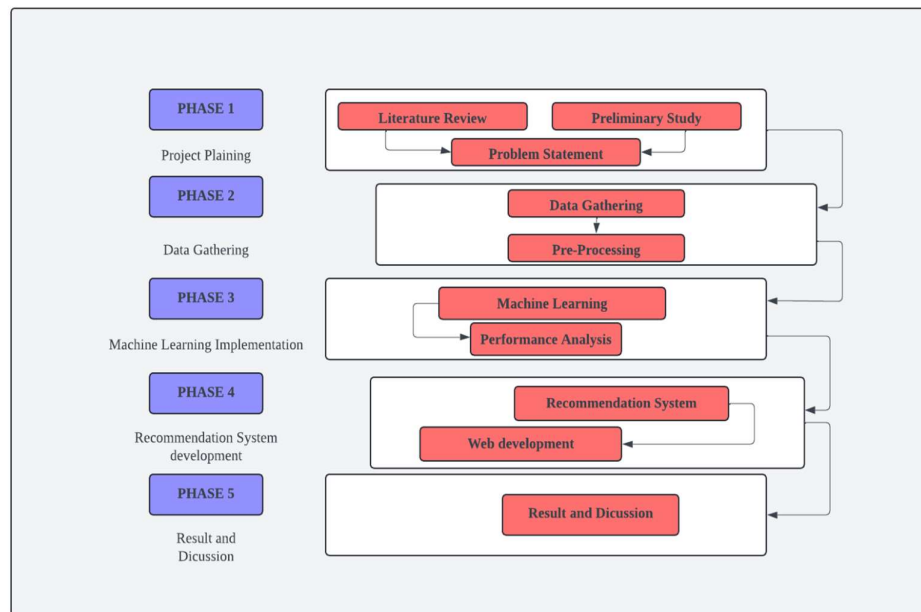


Figure 3.1. Operational Framework

Phase 1: Project planning

Referring to Figure 3.1, before selecting the dataset from a specific hotel in Malaysia, a preliminary investigation was conducted to identify the suitable hotels in Malaysia.

To determine which hotel was popular in Malaysia, the Google Trends tool was deployed. It collects data about Perak from Google reviews. Hence, it was performed on the review dataset of the Perak available in Google Hotel Reviews.

Phase 2: Data Gathering

Phase 2 of the studies is divided into two steps, which are data gathering and pre-processing. These two steps are crucial to preparing the data.

Data Gathering

The data collection process uses a technique known as web scraping to capture reviews of Perak in Google Reviews. The scraping was done by using Instant Data Scraper, a Python library specialized in web scraping HTML and XML files.

Instant Data Scraper

Figure 3.1 shows a screenshot of the Instant Data Scraper tool, which is a browser extension designed to scrape data from web pages automatically. This tool is particularly useful for collecting structured data from tables or lists on websites without needing to write any code.

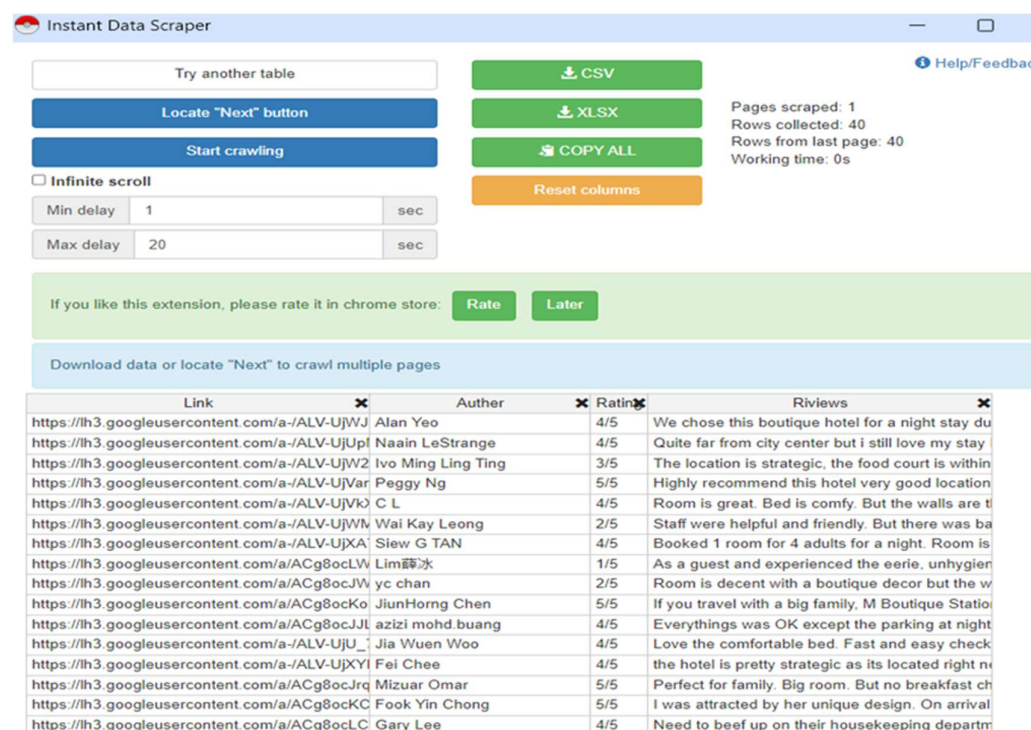


Figure 3.2 Instant Data Scraper

Jupyter Lab

In Figure 3.2, the user successfully activates a Python virtual environment named data_fyp1 and launches Jupyter Lab within it. The terminal output confirms that all necessary extensions and services are loaded, and the Jupyter Lab server is running locally, ready for the user to

interact with it through a web browser. The provided URL with a security token ensures secure access to the session.

```

(base) C:\Users\USER>conda activate data_fyp1
(data_fyp1) C:\Users\USER>jupyter lab
2020-08-29 14:05:14.381 ServerApp] jupyter_lsp | extension was successfully linked.
2020-08-29 14:05:14.315 ServerApp] jupyter_server_terminals | extension was successfully linked.
2020-08-29 14:05:14.322 ServerApp] jupyterlab | extension was successfully linked.
2020-08-29 14:05:14.332 ServerApp] notebook | extension was successfully linked.
2020-08-29 14:05:14.092 ServerApp] notebook_shim | extension was successfully linked.
2020-08-29 14:05:15.002 ServerApp] notebook_shim | extension was successfully loaded.
2020-08-29 14:05:15.002 ServerApp] jupyter_lsp | extension was successfully loaded.
2020-08-29 14:05:15.002 ServerApp] jupyter_server_terminals | extension was successfully loaded.
2020-08-29 14:05:15.052 LabApp] JupyterLab extension loaded from C:\Users\USER\anaconda3\envs\data_fyp1\Lib\site-packages\jupyterlab
2020-08-29 14:05:15.052 LabApp] JupyterLab application directory is C:\Users\USER\anaconda3\envs\data_fyp1\share\jupyterlab
2020-08-29 14:05:15.052 LabApp] Extension Manager is 'jupyterlab'.
2020-08-29 14:05:15.317 ServerApp] jupyterlab | extension was successfully loaded.
2020-08-29 14:05:15.322 ServerApp] notebook | extension was successfully loaded.
2020-08-29 14:05:15.322 ServerApp] Serving notebooks from local directory: C:\Users\USER
2020-08-29 14:05:15.322 ServerApp] Jupyter Server 2.10.1 is running at:
2020-08-29 14:05:15.322 ServerApp] http://localhost:8888/lab?token=986c360a2b4d3fe070af4eebd62c16c656eb56f3bf1c3f13
2020-08-29 14:05:15.322 ServerApp] http://127.0.0.1:8888/lab?token=986c360a2b4d3fe070af4eebd62c16c656eb56f3bf1c3f13
2020-08-29 14:05:15.322 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
C 2020-08-29 14:05:15.382 ServerApp]

To access the server, open this file in a browser:
file:///C:/Users/USER/AppData/Roaming/jupyter/runtime/jpserver-22268-open.html
Or copy and paste one of these URLs:
http://localhost:8888/lab?token=986c360a2b4d3fe070af4eebd62c16c656eb56f3bf1c3f13
http://127.0.0.1:8888/lab?token=986c360a2b4d3fe070af4eebd62c16c656eb56f3bf1c3f13
2020-08-29 14:05:15.382 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-lan-
rver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-language-server, sql-language-server, texlab, typesc
ript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-language-server-bin, yaml-language-
server
2020-08-29 14:05:19.102 LabApp] Build is up to date
2020-08-29 14:05:19.722 ServerApp] Kernel started: 338c016e-82f4-43d7-98ce-37f1b4324ce7
2020-08-29 14:05:19.792 ServerApp] Kernel started: 524822dd-e9f7-4cde-afa5-5be3a949865b
2020-08-29 14:05:19.812 ServerApp] Kernel started: 26cfff8a4-5f96-415c-b48a-7f67f18606cb
2020-08-29 14:05:19.827 ServerApp] Kernel started: 70c1da7e-cf65-46ea-bebc-bdd810d8146
2020-08-29 14:05:19.832 ServerApp] Kernel started: 8127b90b-c6dc-4cc5-b38a-d1ffe4632514
2020-08-29 14:05:19.847 ServerApp] Uncaught exception GET /api/kernels/71724913919815 (::1)
HTTPServerRequest(protocol='http', host='localhost:8888', method='GET', uri='/api/kernels/71724913919815', version='HTTP/1.1', remote_addr='::1')

```

Figure 3.3 Jupyter Lab

Pre-processing

Before being used in machine learning, the review dataset passes through several significant preprocessing stages. The output is an extracted list of words, and it involves both feature extraction and data cleaning. Terms that are frequently used in texts but add no significance to the text should be eliminated are known as stop words.

Data pre-processing

The code provided outlines a data preprocessing workflow for cleaning hotel reviews from a CSV file. It begins by loading the CSV file (main.csv) using Pandas and inspecting its contents, which include columns such as Link, Author, Rating, Review, and Hotel. After removing any missing data, the clean_review function is applied to the Review column to remove special characters, convert text to lowercase, and eliminate extra spaces.

```

[108]: import pandas as pd
import re
import ftfy

[109]: df = pd.read_csv('main.csv')

[110]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2999 entries, 0 to 2998
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Link        2999 non-null   object
1   Author      2999 non-null   object
2   Rating(1-5) 2999 non-null   object
3   Review       2993 non-null   object
4   Hotel       2999 non-null   object
dtypes: object(5)
memory usage: 117.3+ KB

[111]: df.dropna(inplace=True)

[112]: def clean_review(text):
# Remove special characters and numbers
text = re.sub(r'^a-zA-Z\s', '', text)
# Convert to lowercase
text = text.lower()
# Remove extra spaces
text = re.sub(r'\s+', ' ', text).strip()
return text

[113]: df['Cleaned_Review'] = df['Review'].apply(clean_review)

[114]: df['Review_Length'] = df['Cleaned_Review'].apply(len)

[115]: print("Cleaned CSV file saved as 'Cleaned_Review.csv'")

Cleaned CSV file saved as 'Cleaned_Review.csv'

[116]: df['Rating'] = df['Rating(1-5)'].str.extract('(\d)').astype(int)

[117]: df.to_csv('Cleaned_Review.csv', index=False)

[118]: df.info()

<class 'pandas.core.frame.DataFrame'>

```

Figure 3.4 Data Preprocessing

A new column, Cleaned_Review, is created, and the length of each cleaned review is computed and stored in a Review_Length column. The code also extracts the numerical part of the Rating (ranging from 1 to 5) and converts it to an integer type. Finally, the cleaned data is saved to a new CSV file, Cleaned_Review.csv, and the structure of the updated Data Frame is displayed for verification.

	Review Link	Author	Rating(1-5)	Review	Hotel	Cleaned_Review	Review_Length	Rating
1	6-h36-g-rp-mo-ba2-br100	W. Zulfahmi	Rated 4	the crew. There are even welcoming cakes prepared. The room and ambience are great.	3 Suites @ Bandar Botani	h and ambience are great	98	4
2	6-h36-g-rp-mo-ba3-br100	r Aqila Kamila Mohd Lani	Rated 5	error design. Staffs were very helpful, especially the service attendants upon checking in	3 Suites @ Bandar Botani	endants upon checking in	109	5
3	rw36-h36-g-rp-mo-br100	Amaseveni Raja Kumaran	Rated 5	were attentive and went out of their way to ensure that we were comfortable. Thank you!	3 Suites @ Bandar Botani	re comfortable Thank you	220	5
4	rw36-h36-g-rp-mo-br100	Barasa Aladdo	Rated 5	breakfast just nice and the place is clean and quiet. I love it! Thank you for the experience	3 Suites @ Bandar Botani	nk you for the experience	194	5
5	6-h36-g-rp-mo-ba4-br100	Choon Wei	Rated 5	duals. Buffet breakfast was served with variety of food and a special order was available.	3 Suites @ Bandar Botani	pecial order was available	226	5
6	6-h36-g-rp-mo-ba4-br100	Wan nadiah Chong	Rated 4	arly, and was surprised to see our room having a base sink in between the toilet and	3 Suites @ Bandar Botani	in between the toilet and	226	4
7	rw36-h36-g-rp-mo-ba4-br100	Siti Madhah	Rated 5	ly have complementary high tea (but make sure come early as last order is 5pm) hehehe	3 Suites @ Bandar Botani	s last order is pm hehehe	226	5
8	6-h36-g-rp-mo-ba2-br100	klongk118	Rated 4	ndly and helpful. Thanks En Fadi to help me from the reception up to my room5Y-0Y	3 Suites @ Bandar Botani	reception up to my room	158	4
9	rw36-h36-g-rp-mo-br100	muhammad nasif	Rated 5	itfs were helpful and friendly. Breakfast was decent as well. Recommended stay for ipoh.	3 Suites @ Bandar Botani	commended stay for ipoh	150	5
10	rw36-h36-g-rp-mo-br100	wan hanisah	Rated 5	vices from all the staffs are amazing. Im a happy customer. Well spent for a budget hotel	3 Suites @ Bandar Botani	il spent for a budget hotel	178	5
11	rw36-h36-g-rp-mo-br100	Ariff H.	Rated 5	assisting us at the lobby. He even offered to lift bags few times, and came to our car	3 Suites @ Bandar Botani	imes and came to our car	232	5
12	6-h36-g-rp-mo-ba4-br100	Ragunathan Ezhamkovan	Rated 5	nd clean. Hi Tea complimentary are surprisingly given, well had few cookies and cold	3 Suites @ Bandar Botani	had few cookies and cold	151	5
13	6-h36-g-rp-mo-ba4-br100	Elena Kim	Rated 5	mess. I parked in front of the hotel, and I got a parking fine. None of the staff told me that	3 Suites @ Bandar Botani	he of the staff told me that	107	5
14	rw36-h36-g-rp-mo-br100	Khor Abbe	Rated 5	are friendly & nice 0Y+ The price pay for 3 star hotel but you got 4 star service. 0Y+ 0Y+	3 Suites @ Bandar Botani	el but you got star service	98	5
15	6-h36-g-rp-mo-ba6-br100	Julian Si	Rated 3	Rain shower performed like this and no one could service it for 2 days. What a pity	3 Suites @ Bandar Botani	vice it for days what a pity	80	3
16	rw36-h36-g-rp-mo-br100	Aaangel	Rated 2	cleanliness need to improve, toilet bowl quite dirty! room light very deem and dark as	3 Suites @ Bandar Botani	ht very deem and dark as	221	2
17	rw36-h36-g-rp-mo-br100	Meng hee Lim	Rated 1	nu must be delicious, but it is not so! Tell the waiter that the prawns are not fresh, but	3 Suites @ Bandar Botani	prawns are not fresh but	235	1
18	6-h36-g-rp-mo-ba4-br100	Marcus	Rated 3	it for me is a good place to stay I have found so far comparing with other boutique hotels.	3 Suites @ Bandar Botani	with other boutique hotels	166	3
19	6-h36-h36-g-rp-mo-br100	Alysa Mastapha	Rated 5	id 6pm. The staff were very nice, friendly and helpful. The room is very clean and tidy	3 Suites @ Bandar Botani	om is very clean and tidy	171	5
20	rw36-h36-g-rp-mo-br100	Pajar Bin bacho	Rated 5	1 Simpang Pulai, comfortable hotel for a great Rest & yes, hi-tea before 6pm & breakfast	3 Suites @ Bandar Botani	hitea before pm breakfast	103	5
21	rw36-h36-g-rp-mo-br100	Yin Yee Yap	Rated 4	where easily access with own transport. Hotel to improve the water heater supply as	3 Suites @ Bandar Botani	he water heater supply as	235	4
22	rw36-h36-g-rp-mo-br100	Nik Nur Amirah	Rated 5	that, they also has a breakfast when u come to check in here. it&™s really convenient!	3 Suites @ Bandar Botani	here its really convenient	143	5
23	rw36-h36-g-rp-mo-br100	wahidah wagimon	Rated 5	ere very friendly & helpful. Got many choices for breakfast buffet with reasonable price!	3 Suites @ Bandar Botani	ffet with reasonable price	233	5
24	rw36-h36-g-rp-mo-br100	Kumary Sureen	Rated 1	e worst part was the shower water also cold. We brought our kids, just imagine we all	3 Suites @ Bandar Botani	ur kids just imagine we all	234	1
25	6-h36-g-rp-mo-ba3-br100	Anderson	Rated 4	is particularly comfortable and modernly designed for business and casual travellers.	3 Suites @ Bandar Botani	ness and casual travellers	236	4
26	rw36-h36-g-rp-mo-br100	ron	Rated 3	ties, price wise reasonable with breakfast. Location abit out but still ok very quiet place	3 Suites @ Bandar Botani	ut still ok very quiet place	208	3
27	6-h36-g-rp-mo-br100	Grace Ooi	Rated 4	nance. A good business trip stay for a reasonable budget, believe the 1969 Business	3 Suites @ Bandar Botani	dgel believe the business	213	4
28	rw36-h36-g-rp-mo-br100	Aziyyn Fauzi	Rated 5	are very nice too. looks luxurious and clean. Certainly will repeat and it is recommended.	3 Suites @ Bandar Botani	at and it is recommended	174	5
29	rw36-h36-g-rp-mo-br100	bagel_bites 6969	Rated 5	great for 2 people and the interior design of the room and hotel was very impressive.	3 Suites @ Bandar Botani	hotel was very impressive	117	5

Figure 3.5 Cleaned_Review

Phase 3: Machine Learning Implementation

Three types of machine learning were chosen to classify the dataset. The machine learning classifiers chosen are Naïve Bayes, Support Vector Machine, and Random Forest Based. Each classifier was chosen based on their ability to classify the data.

- a) Naïve Bayes is a basic approach to developing a classifier model that assigns class names, called eigenvalue vectors, to problem instances with class labels drawn from some finite set. Naive Bayes classifiers are based on Bayes' theorem and are widely used in data mining research. The Naive Bayes equation is based on the following equation:

$$P(C | X) = \frac{P(X | C) \cdot P(C)}{P(X)}$$

Where:

- $P(C | X)$ is the posterior probability of class C given the feature vector X.
- $P(X | C)$ is the likelihood of feature vector X given class C.
- $P(C)$ is the prior probability of class C.
- $P(X)$ is the evidence or the total probability of the feature vector X across all classes.

For classification, the class with the highest posterior probability $P(C | X)$ is chosen.

- b) Support Vector Machine (SVM) is one of the most recently developed supervised learning techniques. SVM is mainly used for classification and regression and is widely used in object detection and recognition, content-based image retrieval, text recognition, biometrics, and many other domains. SVM classifies data by constructing an N-dimensional hyperplane

in the feature space that optimally divides the data into two classes. An optimal linear discriminant function or a classifier with maximum margins provides the best solution to a particular problem. The theory of SVM utilizes the following equation:

$$f(x) = w \cdot x + b = 0$$

Where:

- w is the weight vector perpendicular to the hyperplane.
- x is the feature vector.
- b is the bias term.

The SVM maximizes the margin by solving the following optimization problem:

$$\min \frac{1}{2} \|w\|^2 \text{ subject to } y_i(w \cdot x_i + b) \geq 1$$

Where:

where y_i are the class labels and x_i are the training examples.

- c) Random Forest Based is an ensemble learning method that constructs multiple decision trees during training and outputs the class (categorization) patterns of a single decision tree at final prediction. It recursively splits the data according to the features and its goal is to maximize the purity of the resulting subset:

$$MSE = \frac{1}{N} \sum_{i=1}^N (f_i - y_i)^2$$

Where:

- N is the number of data points
- f_i is the value returned by the model
- y_i is the actual value for data point i .

Are the predictions from the individual decision trees.

The goal of each decision tree is to minimize impurity using criteria such as Gini impurity or entropy (information gain). The splitting criterion for Gini impurity, for example, is:

$$G = 1 - \sum_{i=1}^c p_i^2$$

Where:

- G is the Gini impurity.
- p_i is the proportion of instances belonging to class i in a particular node.

- C is the number of classes.

To analyze the performance of each classifier, the three measures, namely accuracy, precision, and recall value used. Accuracy is the most intuitive performance metric, the higher the accuracy, the better the model. Accuracy measures how much text or data is correctly categorized into its category. Precision measures the classifier's ability to identify only relevant data, while recall measures the model's ability to find all relevant cases in the dataset. Higher values of precision and recall indicate that the classifier performs better.

During the implementation phase, Python is a popular and flexible programming language known for its efficiency, adaptability, and vast ecosystem of libraries and frameworks, which are the primary operating systems used in our environment. Due to Python's simple syntax and easy reading, developers and researchers can focus on the specifics of sentiment analysis algorithms rather than struggling with complex methods of programming or laborious syntax. Furthermore, the reliability of sentiment analysis models is improved and the development cycle is greatly accelerated by the abundance of tools and resources provided by Python's rich ecosystem of libraries, which includes TensorFlow, PyTorch, Scikit-learn, and NLTK (Natural Language Toolkit). Additionally, the active Python community promotes a cooperative and encouraging setting where practitioners and researchers can easily exchange ideas, best practices, and snippets of code, speeding up learning and propelling forward advances in sentiment analysis approaches and techniques. To sum up, Python is the main component of our implementation environment because it provides a powerful combination of power, simplicity, and versatility that is necessary for developing innovative sentiment analysis models that are specific to the complex world of Malaysian hotel reviews.

Phase 4: Recommendation System Development

In this phase, the insights and predictions generated by the machine learning models are integrated into a functional recommendation system. This involves developing the backend logic that handles recommendations and ensuring that the system can scale to handle real-world data and user interactions. To make the system accessible to users, a Web Development process is undertaken, where a user-friendly interface is designed and developed. This interface allows users to interact with the recommendation system, providing a seamless experience for discovering content, products, or services based on their preferences. The web platform might include features like dashboards, search functionalities, and real-time recommendations.

Phase 5: Result and Discussion

Lastly, it compares the performance metrics of the various models to determine which model or combination of models is most effective in sentiment analysis. By comparing precision and accuracy scores and considering the trade-offs between the two, a more comprehensive discussion of the strengths and weaknesses of the models may also be warranted. This comprehensive review provides practitioners looking to use sentiment analysis as an effective method to improve service quality and customer satisfaction in the Malaysian hospitality industry with vital information and provides a beacon for future research projects.

3.2 System Design Diagram

3.2.1 System Architecture Diagram

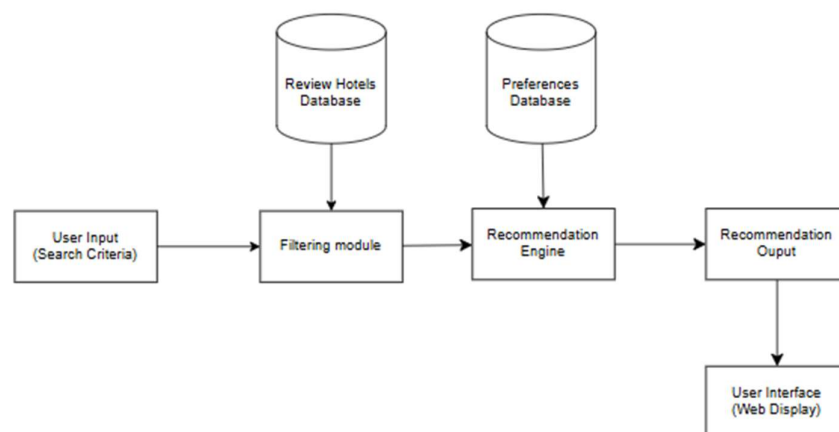


Figure 3.6 System Architecture Diagram

Diagrams of system architecture provide a visual illustration of the numerous components that make up a system and show how those components communicate with one another and interact with one another. In Figure 4.1 above, the proposed hotel recommendation system is illustrated, showing how it operates through each module and database involved. There are three main components in the system architecture design: user input, processing modules, and databases. The user begin by entering search criteria such as top 3 hotel rating, hotel star rating, and review rating through the user input interface. Once the input is submitted, the system will access the

hotel's database, which stores hotel data in the form of an Excel file. The filtering module will retrieve relevant hotel information from the database and apply filters based on the user's search criteria. These filtered results will then be passed to the recommendation engine. The recommendation engine is responsible for sorting hotels and will retrieve data from the preferences database, which stores user preferences such as past searches. Lastly, this recommendation output had been displayed to the user through the user interface in a web format, allowing the user to view the list of suggested hotels along with relevant details such as hotel name and star rating.

3.2.2 Use Case Diagram

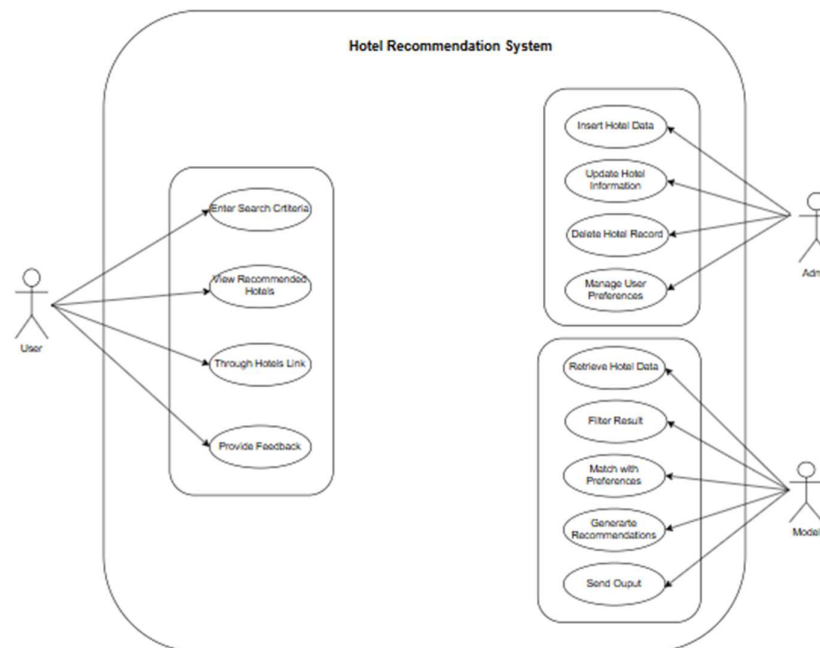


Figure 3.7 Use Case Diagram for the proposed system

From Figure 4.2 above, there are three roles involved in the Hotel Recommendation System: user, admin, and model, each performing different tasks in the system. On the user side, the system allows a person to enter search criteria such as hotel name, star rating, and preferences to find suitable hotels. After submitting the criteria, the person can view the recommended hotels and access further details through the hotel links provided by the system. Additionally, users can give feedback on the recommendations received to help enhance future suggestions. The admin is responsible for managing hotel-related data in the system, including inserting new hotel data, updating existing information, deleting outdated hotel records, and managing user

preferences. Any changes to the stored data are handled solely by the admin. The model performs the core recommendation process within the system. It begins by retrieving hotel data from the database, then filters the results according to the user's search criteria. After filtering, the model matches the available options with the user's preferences, generates a list of hotel recommendations, and finally sends the output to be displayed to the user.

3.2.3 Activity Diagram

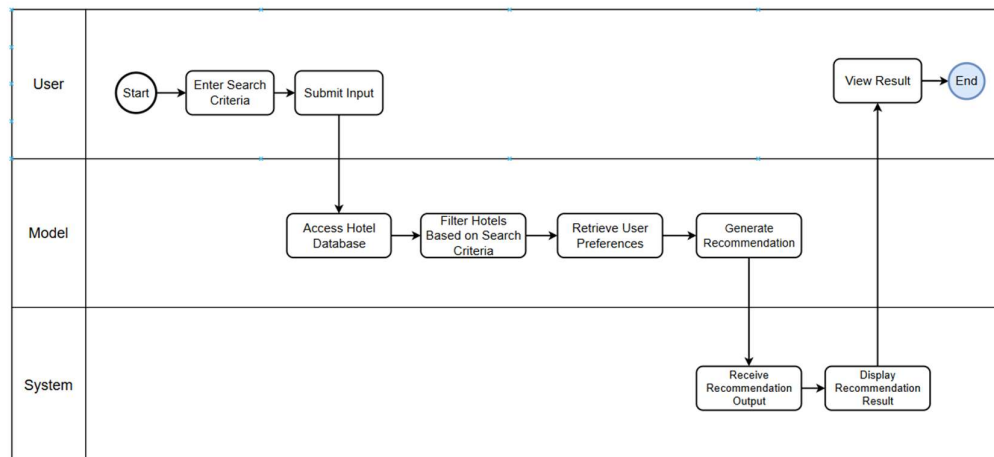


Figure 3.8 Activity Diagram for the proposed system

Figure 4.3 shows the activity diagram for the proposed hotel recommendation system. There are three roles involved in this diagram, which are user, model, and system. At the beginning, the user starts by entering the search criteria through the user interface. These criteria may include details such as location, hotel star rating, and review rating. Once the input is completed, the user submits the search criteria, which triggers the processing on the model side. The model will first access the hotel database that stores hotel information, typically in the form of an Excel file. After accessing the data, the filtering module will apply filters based on the user's submitted search criteria to narrow down the relevant hotels. The model then retrieves the user preferences, such as previous searches or favorite hotel categories, from the preferences database. Based on both the filtered hotel data and the user preferences, the recommendation engine will generate a set of hotel recommendations. These recommendations are then sent to the system, which receives the recommendation output and proceeds to display the final recommendation result in a web format. Lastly, the user can view the list of suggested hotels, along with important details such as hotel name and star rating, through the user interface.

3.3 Final Year Project Timeline

This FYP timeline outlines the schedule for developing a "Hotel Recommender System using machine learning" over 14 weeks.

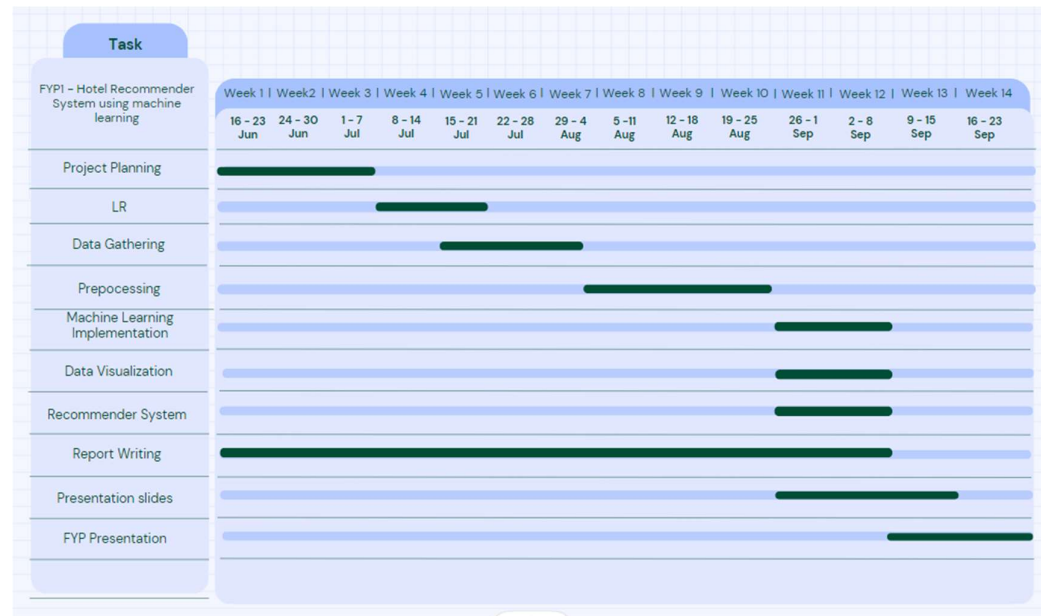


Figure 3.9 Timeline of FYP I

Figure 3.1 shows the project schedule for Project I: Hotel Recommender System Using Machine Learning, which spans 14 weeks, and contains the tasks and the duration taken for each. The project starts with Project Planning in Week 1. Literature Review (LR) commences in Week 2 and Data Gathering in Week 3. Preprocessing is done from Week 5 to Week 8, giving way to Machine Learning Implementation that is done from Weeks 8 to 10. Data Visualization starts at Week 9 and runs alongside Recommender System development from Week 9 to Week 12. Report Writing is done throughout the project from Week 1 to Week 12, indicating that it is an ongoing process. Then, Presentation Slides are completed in Weeks 12 and 13, with the FYP Presentation reaching its culmination in the last week, Week 14. The timeline is structured to provide a systematic and efficient process of handing over the project milestones.

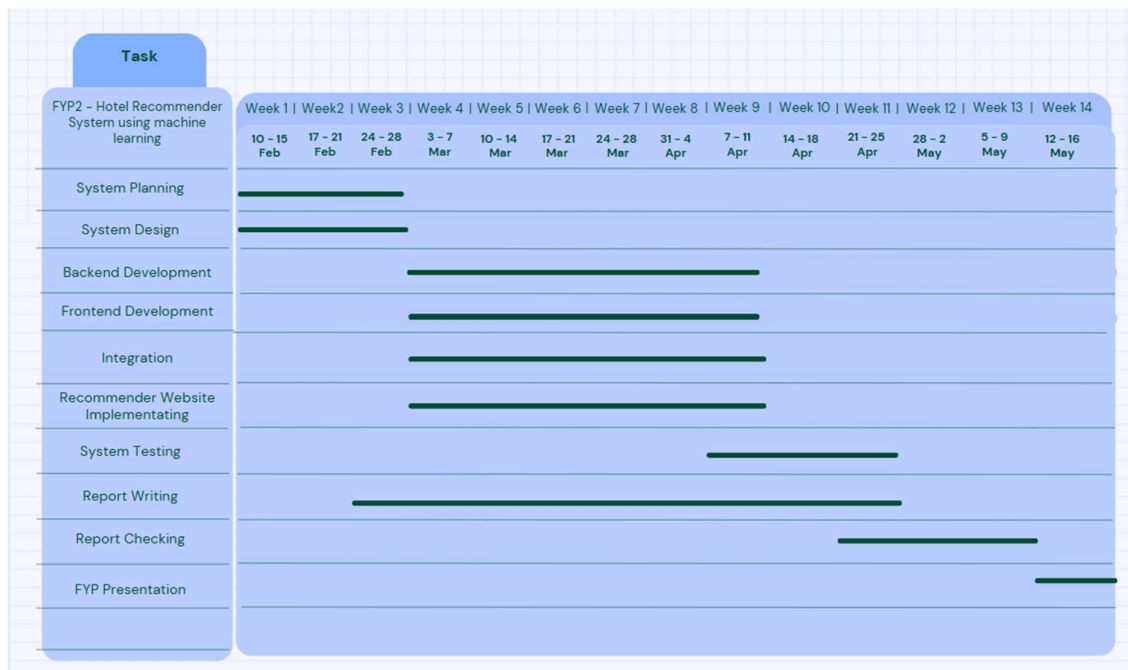


Figure 3.10 Timeline of FYP II

Figure 3.2 is the project timeline for FYP II: Hotel Recommender System Using Machine Learning, showing the division of tasks into a 14-week duration from February to May. The project starts with System Planning and System Design in Weeks 1 and 2 as the basis for the succeeding development stages. Backend Development, Frontend Development, and Integration are completed simultaneously from Week 3 to Week 7, followed by Recommender Website Implementation, which is from Week 6 to Week 9. System Testing is carried out in Weeks 9 and 10 for functionality and performance. Report Writing starts early, from Week 3 to Week 11, to allow ample time for documentation. This is followed by Report Checking Weeks 11 to 13, ensuring accuracy and completeness. This is then followed by the FYP Presentation in Week 14, which concludes the project. This time division offers a well structured workflow, with each phase building well on the previous one.

Chapter 4

System Design

4.1 Simple Block Diagram

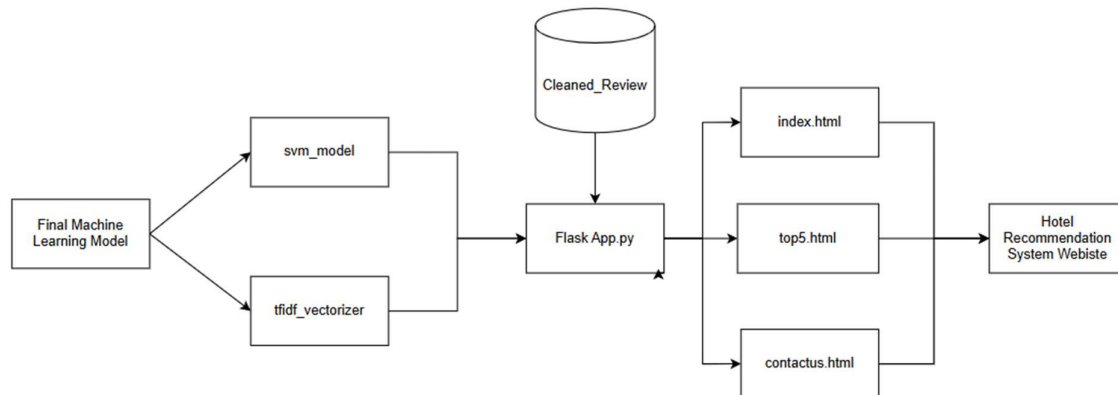


Figure 4.1 System Block Diagram

Figure 4.1 illustrates the system architecture of a Hotel Recommendation System Website that integrates machine learning, natural language processing, and a web application interface. At the system's core is the Final Machine Learning Model, which includes two essential components: the SVM (Support Vector Machine) model and the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. The TF-IDF vectorizer is responsible for converting cleaned hotel review text into numerical features, while the SVM model uses these features to classify the sentiment of the reviews, typically into categories such as positive or negative.

The cleaned reviews, represented as “Cleaned_Review” in the diagram, are the preprocessed input data that have undergone text cleaning procedures such as removal of stop words and punctuation. These reviews are passed to the Flask web application, defined in the app.py file, which serves as the system's backend. The Flask application is responsible for loading the trained machine learning components (SVM model and TF-IDF vectorizer), processing user

input or stored data, making predictions using the model, and dynamically rendering the results on the website.

The web interface comprises three HTML pages: `index.html`, `top5.html`, and `contactus.html`. The `index.html` file serves as the homepage, likely providing a user input form and navigation options. The `top5.html` page displays the top five hotel recommendations, based on sentiment analysis results. Lastly, the `contactus.html` page contains contact details or a feedback form for users to reach out. These pages work together under the Flask app to form a complete, user-friendly Hotel Recommendation System Website.

4.2 System Components Specifications

The hotel recommendation system is composed of several core components that work together to provide an interactive and intelligent user experience. At the heart of the system is the machine learning model, built using the Support Vector Machine (SVM) algorithm in conjunction with a TF-IDF vectorizer. The SVM model is responsible for classifying user-generated hotel reviews into sentiments such as positive or negative, while the TF-IDF vectorizer transforms textual reviews into numerical features suitable for model processing. These two elements are pre-trained and saved as part of the system's backend to be used in real-time when a user inputs a review or searches for hotels.

The frontend of the system is constructed using three HTML templates `index.html`, `top5.html`, and `contactus.html`. The `index.html` page serves as the main interface for users, allowing them to input search criteria such as minimum review rating, hotel name, and keywords. The `top5.html` template dynamically displays the top five hotels ranked by average rating, offering users a quick view of the most highly rated accommodations. Meanwhile, the `contactus.html` page functions as a communication portal where users can leave feedback or inquiries, helping to improve system responsiveness and gather user insights.

The backend is managed through a Flask-based Python application contained in the `app.py` file. This script handles user requests, loads the machine learning models, reads the preprocessed review data from a CSV file (`Cleaned_Review.csv`), and applies filters based on user-defined inputs. Upon processing, the system delivers personalized hotel recommendations by dynamically rendering the appropriate web page template. This seamless interaction between

frontend and backend ensures that the system provides accurate, responsive, and user-friendly recommendations in real-time.

The entire application is deployed locally using Flask's development server, allowing for interactive testing and demonstration. The use of Python for backend logic, along with HTML and CSS for the user interface, provides a lightweight yet robust framework that supports the integration of data processing, model inference, and user engagement. Together, these components form a cohesive system designed to enhance the hotel selection process by leveraging machine learning and user-centric design.

4.3 Data Description

The following section presents a visual analysis of hotel ratings through various charts, providing insights into the distribution of ratings, the concentration of reviews, and a comparison of top rated hotels based on customer satisfaction.

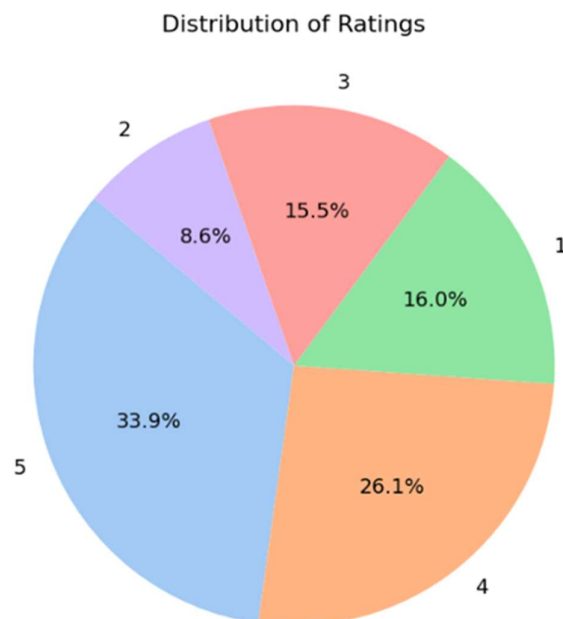


Figure 4.4 Pie chart of Distribution of Ratings

The pie chart provides a visual representation of the distribution of ratings across five categories, showcasing the percentage of total ratings that each category received. The ratings are divided into five segments, labeled 1 through 5, corresponding to the satisfaction levels from lowest to highest. The chart uses different colors to distinguish between these segments, with the size of

each slice indicating the proportion of total reviews that fall into each rating category. The percentages displayed within each segment give a clear indication of how the ratings are spread across the spectrum.

The largest portion of the pie chart is occupied by the 5 star rating, which accounts for 33.9% of all ratings. This suggests that a significant portion of users gave the highest possible rating, reflecting a strong overall satisfaction with the product or service. Following this, the 4 star rating constitutes 26.1% of the total ratings, indicating that a substantial number of users were also very satisfied, though not to the highest degree. Together, these two categories make up more than half of all ratings, underscoring the generally positive reception among users.

In contrast, the lower ratings (1, 2, and 3 stars) occupy smaller segments of the pie chart. The 3star rating represents 15.5% of the total, while the 1 star rating accounts for 16.0%, suggesting a mixed or neutral level of satisfaction among some users. The smallest segment belongs to the 2 star rating, which makes up only 8.6% of the total, indicating that relatively few users found the product or service to be below average. The distribution shown in the pie chart reflects a general trend towards higher ratings, with a noticeable concentration in the 4 and 5 star categories, but also highlights that a portion of users experienced less favourable outcomes.

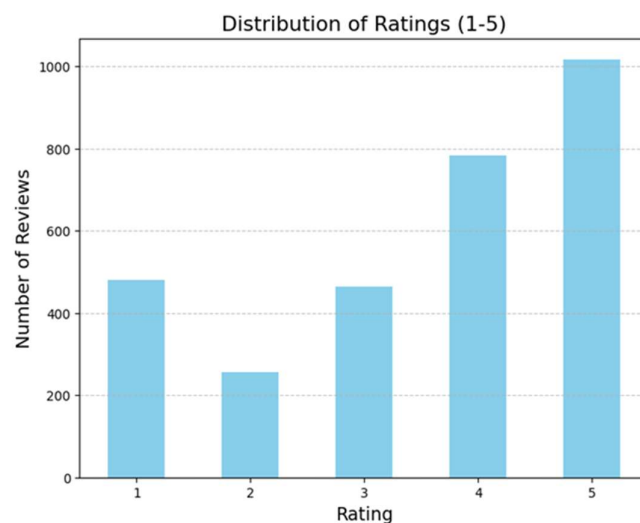


Figure 4.5 Bar Graph of Distribution of Ratings

The bar chart illustrates the distribution of ratings from 1 to 5, revealing that the majority of reviews are concentrated at the higher end of the scale. The 5 star rating has the highest count, with approximately 1,000 reviews, indicating that many users were highly satisfied with the

product or service. The 4 star rating follows with around 800 reviews, showing a strong level of satisfaction, though slightly less than those who awarded 5 stars. The middle ground 3 star rating has about 500 reviews, reflecting moderate satisfaction. Interestingly, the 1 star rating, representing significant dissatisfaction, received around 450 reviews, slightly more than the 2 star rating, which has the lowest count at roughly 250 reviews. Overall, the chart suggests a tendency towards positive feedback, with most reviewers leaning towards higher ratings, while lower ratings are less frequent.

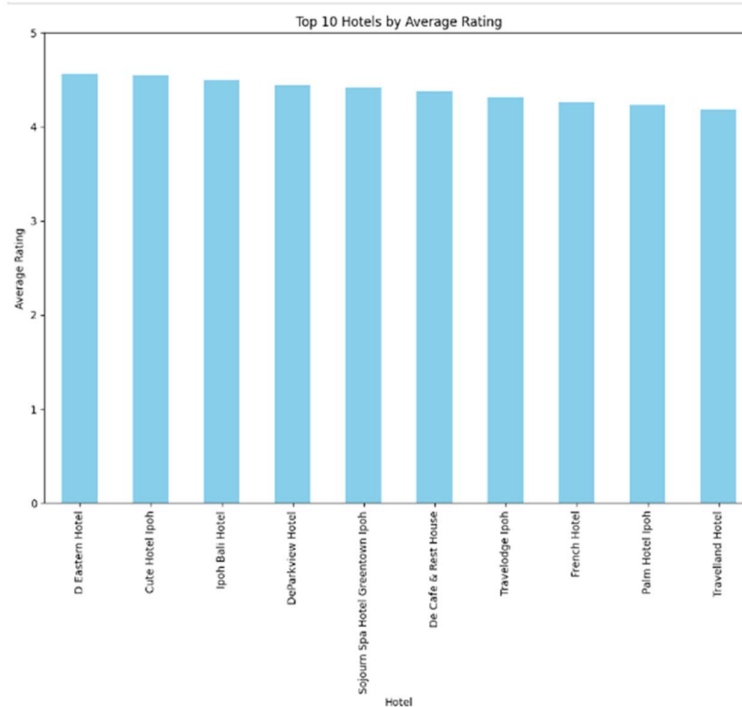


Figure 4.6 Bar Graph Top 10 Hotels by Average Rating

The bar graph provides a detailed comparison of the top 10 hotels by their average ratings, showcasing the level of customer satisfaction across these establishments. Each hotel is represented by a bar, with the height of the bar indicating the average rating on a scale from 0 to 5. The hotels featured include D Eastern Hotel, Cute Hotel Ipoh, Ipoh Bali Hotel, De Parkview Hotel, Sojourn Spa Hotel Greentown Ipoh, De Cafe & Rest House, Travelodge Ipoh, French Hotel, Palm Hotel Ipoh, and Traveling Hotel. The graph reveals that all these hotels have achieved remarkably high average ratings, all of which are slightly above 4.5. This indicates a consistently strong performance in customer satisfaction, as each hotel has received feedback that places them near the top of the rating scale. The close similarity in the height of the bars suggests that these hotels are nearly indistinguishable in terms of average rating, with very minor differences separating them. This uniformity implies that guests across these hotels

have had very positive experiences, leading to high and nearly equivalent average ratings. The consistent ratings across the board highlight the competitive nature of these top rated hotels, each maintaining a high standard of service and guest satisfaction.

4.4 Machine Learning

a) Models Construction

The code in the provided image illustrates the application of three different machine learning models Multinomial Naive Bayes, Support Vector Machine (SVM), and Random Forest—to a classification task, likely involving text data. The primary goal of this exercise is to compare the accuracy of these models in making predictions based on the given data, which could involve text classification, sentiment analysis, or another similar task.

```
[1]: import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.metrics import accuracy_score, precision_score, f1_score

[2]: df = pd.read_csv('Cleaned_Review.csv')

[3]: df = df.assign(Cleaned_Review=df['Cleaned_Review'].fillna(''))

[4]: df['label'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative' if x < 3 else 'Neutral')

[5]: # Split the data into features and labels
    X = df['Cleaned_Review']
    y = df['label']

[6]: # Convert text data to TF-IDF features
    tfidf = TfidfVectorizer(max_features=5000)
    X_tfidf = tfidf.fit_transform(X)

[7]: X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

[8]: # 1. Naive Bayes Classifier
    nb = MultinomialNB()
    nb.fit(X_train, y_train)
    nb_pred = nb.predict(X_test)
    nb_accuracy = accuracy_score(y_test, nb_pred)
    nb_precision = precision_score(y_test, nb_pred, average='weighted', zero_division=1) # Handle undefined metrics
    nb_f1 = f1_score(y_test, nb_pred, average='weighted', zero_division=1) # Handle undefined metrics

[9]: print(f"Naive Bayes Accuracy: {nb_accuracy * 100:.2f}%")

Naive Bayes Accuracy: 62.60%
```

Figure 4.7 Naïve Bayes Model

In the first model, a Multinomial Naive Bayes classifier is employed. Naive Bayes models are particularly well suited for text classification tasks due to their probabilistic nature and the assumption of feature independence. In this scenario, it's assumed that the text data has been pre processed, potentially using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) to convert text into numerical features. After preprocessing, the dataset is typically split into training and testing sets, which allows the model to be trained on one portion of the data and tested on another to evaluate its performance. In this case, the

Multinomial Naive Bayes classifier achieves an accuracy of 62.60%. This result indicates that the model performs moderately well, but there may be room for improvement, possibly by adjusting the preprocessing techniques or exploring other model types.

```
[1]: import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.svm import SVC
    from sklearn.metrics import accuracy_score

[2]: df = pd.read_csv('Cleaned_Review.csv')

[3]: df = df.assign(Cleaned_Review=df['Cleaned_Review'].fillna(''))

[4]: df['label'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative' if x < 3 else 'Neutral')

[5]: # Split the data into features and labels
    x = df['Cleaned_Review']
    y = df['label']

[6]: # Convert text data to TF-IDF features
    tfidf = TfidfVectorizer(max_features=5000)
    X_tfidf = tfidf.fit_transform(x)

[7]: X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

[8]: # 2. Support Vector Machine (SVM) Classifier
    svm = SVC(kernel='linear', random_state=42)
    svm.fit(X_train, y_train)
    svm_pred = svm.predict(X_test)
    svm_accuracy = accuracy_score(y_test, svm_pred)

[9]: print(f"SVM Accuracy: {svm_accuracy * 100:.2f}%")
    SVM Accuracy: 74.46%
```

Figure 4.8 SVM Model

The second model tested is a Support Vector Machine (SVM) with a linear kernel. SVMs are known for their effectiveness in high-dimensional spaces, which is common in text classification tasks where each word or n-gram can be considered a feature. The linear kernel is used when the data is believed to be linearly separable or when a linear decision boundary is sufficient. Following the same data preprocessing and splitting process as with the Naive Bayes model, the SVM is trained and then evaluated on the test set. The SVM achieves an accuracy of 74.46%, which is a noticeable improvement over the Naive Bayes model. This suggests that the SVM's ability to find an optimal hyperplane in a high-dimensional space allows it to better capture the underlying patterns in the data.

```

[1]: import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, precision_score, f1_score

[2]: df = pd.read_csv('Cleaned_Review.csv')

[3]: df = df.assign(Cleaned_Review=df['Cleaned_Review'].fillna(''))

[4]: df['label'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative' if x < 3 else 'Neutral')

[5]: # Split the data into features and labels
    X = df['Cleaned_Review']
    y = df['label']

[6]: # Convert text data to TF-IDF features
    tfidf = TfidfVectorizer(max_features=5000)
    X_tfidf = tfidf.fit_transform(X)

[7]: X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

[8]: # 3. Random Forest Classifier
    rf = RandomForestClassifier(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)
    rf_pred = rf.predict(X_test)
    rf_accuracy = accuracy_score(y_test, rf_pred)
    rf_precision = precision_score(y_test, rf_pred, average='weighted', zero_division=1)
    rf_f1 = f1_score(y_test, rf_pred, average='weighted', zero_division=1)

[9]: print(f"Random Forest Accuracy: {rf_accuracy * 100:.2f}%")
    Random Forest Accuracy: 73.29%

```

Figure 4.9 Random Forest Model

Finally, the third model is a Random Forest classifier. Random Forests are ensemble methods that combine the predictions of multiple decision trees to improve overall accuracy and robustness. Each tree in the forest is trained on a random subset of the data, which helps to reduce overfitting and increase the model's ability to generalize to new data. After the same preprocessing steps and data split, the Random Forest model is trained and tested. It achieves an accuracy of 73.29%, which is slightly lower than the SVM but still better than the Naive Bayes classifier. This performance suggests that while Random Forests are generally robust and versatile, in this case, they do not outperform the SVM, possibly due to the complexity of the data or the specific features used.

Overall, the results from these three models provide valuable insights into the effectiveness of different machine learning approaches for the task at hand. The SVM model outperforms the others, indicating that it is the most suitable for this dataset, at least under the current settings and preprocessing methods. However, the moderate accuracy scores across all models suggest that there might be additional improvements to be made, such as optimizing hyperparameters, trying different feature extraction methods, or even exploring more advanced models like deep learning techniques if the dataset and computational resources allow.

In summary, while the current models offer a solid starting point, there is still significant potential for enhancing their performance. Further experimentation with different models,

feature engineering, and hyperparameter tuning could lead to more accurate and reliable predictions, ultimately improving the effectiveness of the classification task.

b) Performance measurement

Table 4.1 compares the performance of Naïve Bayes, Support Vector Machine (SVM), and Random Forest models using Precision, Recall, and F1-Score. These metrics assess how well each model performs in classification tasks, balancing accuracy and consistency in identifying positive instances. The comparison highlights the strengths and weaknesses of each model, aiding in selecting the most suitable approach for specific classification needs.

Table 4.1 Statistical Measurement of Naive Bayes, SVM, and Random Forest

Machine Learning Model	Accuracy	Precision	F1-Score
Naïve Bayes	62.60%	73.93%	52.32%
SVM	74.46%	70.84%	70.84%
Random Forest	73.29%	76.05%	67.92%

Table 4.1 presents the performance metrics accuracy, precision, and F1-Score of three machine learning models: Naïve Bayes, Support Vector Machine (SVM), and Random Forest. These metrics are essential for evaluating the effectiveness of each model in a classification task. Accuracy measures the overall correctness of predictions, Precision assesses the accuracy of positive predictions, and the F1-Score combines both Precision and Recall providing a balanced view of model performance.

The Naïve Bayes model achieved an accuracy of 62.60%, precision of 73.93%, and an F1-score of 52.32%. This indicates that while Naïve Bayes is relatively good at making accurate positive predictions (high Precision), it has a lower overall accuracy, which negatively impacts its F1-score. The imbalance between accuracy and precision suggests that Naïve Bayes frequently misclassifies instances, making it less reliable when accurate classification is critical.

Among the three models, SVM demonstrated the best performance, with an accuracy of 74.46%, precision of 70.13%, and F1-score of 70.84%. These metrics reflect a better balance between

identifying positive instances and maintaining accuracy compared to Naïve Bayes. Random Forest also performed well, with an accuracy of 73.29%, precision of 76.05%, and an F1-score of 67.92%, showing a consistent and balanced approach. Overall, SVM stands out as the most effective model due to its superior balance of precision, accuracy, and overall F1-score, making it the preferred choice among the three models for this classification task.

Given these results, SVM is identified as the most effective model among the three. Its high accuracy, balanced precision, and F1-score make it especially suitable for integrating into the hotel recommendation system, where correctly interpreting user sentiments and preferences is crucial for generating relevant and personalized suggestions. By leveraging SVM's robust classification capabilities, the recommendation system can better distinguish between positive and negative reviews, ultimately enhancing user satisfaction through more accurate recommendations.

Chapter 5

System Implementation

5.1 Hardware and Software Setup

5.1.1 Hardware

The hardware foundation for this project is a laptop or desktop computer, selected based on its ability to efficiently process large datasets and run complex machine learning algorithms. Essential hardware specifications are listed in Table 5.1.

Table 5.1 Specifications of laptop

Description	Specifications
Model	Asus TUF Gaming FX505GT
Processor	Intel(R) Core(TM) i5-9300H
Operating System	Windows 11
Graphic	NVIDIA GeForce GTX 1650 4GB
Memory	12GB DDR12 RAM
Storage	512TB SATA HDD

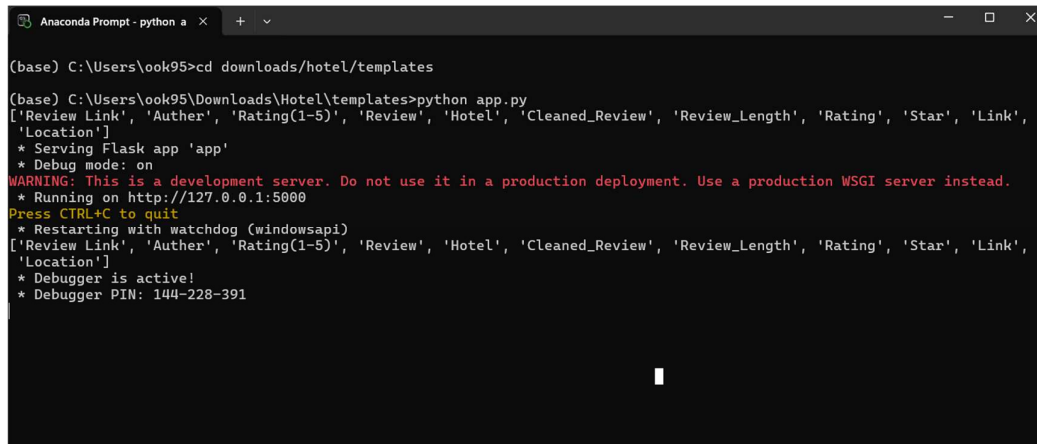
5.1.2 Software

The software foundation for this project consists of various tools and platforms selected based on their ability to efficiently process large datasets, perform data analysis, and support complex machine learning algorithms. Essential software tools include:

Table 5.2 Software Utilized in the Project

Description	Description	Function
Jupyter lab	JupyterLab is a flexible, web-based development environment for Jupyter notebooks, code, and data.	Used for data analysis, visualization, and running machine learning models.
ChatGPT	ChatGPT is an AI-powered natural language processing tool that generates human-like text responses.	Assists with answering questions, providing information, and content creation.
Quill Bot	QuillBot is an AI-driven writing assistant that helps with paraphrasing, summarizing, grammar checking, and enhancing text.	Used for improving text quality and checking grammar and plagiarism.
Lucid chart	Lucid chart is an online diagramming tool that enables users to create, visualize, and collaborate on flowcharts, diagrams, and other visual representations.	Used to design and illustrate the operational framework of the project.
Canva	Canva is an intuitive graphic design platform that allows users to create visually appealing content such as graphics, presentations, and videos.	Used to design and create the project timeline and other visual content.
Instant Data Scraper	Instant Data Scraper is a browser extension that automatically extracts structured data from web pages into formats like CSV.	Used for data extraction and collection from web sources.
Drawio	Draw.io (now known as diagrams.net) is a free, web-based diagramming tool used for creating flowcharts, wireframes, network diagrams, and more.	Used to design system architecture diagrams, flowcharts, and illustrate data processing workflows.
Anaconda Navigator	Anaconda Navigator is a desktop GUI that allows users to launch applications and manage conda packages, environments, and channels without using command-line commands.	Used to manage the project environment, install necessary libraries, and launch JupyterLab and other tools.
Flask	Flask is a lightweight Python web framework that enables the creation of web applications quickly and with minimal code.	Used to develop and run the hotel recommendation web application with dynamic routing and interactive features.

5.2 Setting and Configuration



```
Anaconda Prompt - python a x + v
(base) C:\Users\look95>cd downloads/hotel/templates
(base) C:\Users\look95\Downloads\Hotel\templates>python app.py
['Review Link', 'Author', 'Rating(1-5)', 'Review', 'Hotel', 'Cleaned_Review', 'Review_Length', 'Rating', 'Star', 'Link',
'Location']
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
['Review Link', 'Author', 'Rating(1-5)', 'Review', 'Hotel', 'Cleaned_Review', 'Review_Length', 'Rating', 'Star', 'Link',
'Location']
* Debugger is active!
* Debugger PIN: 144-228-391
```

Figure 5.1 Anaconda Prompt

Figure 5.2 is a Flask web application run within an Anaconda environment from the terminal. The user navigated to the C:\\Users\\look95\\Downloads\\Hotel\\templates directory and ran the program using python app.py. The Flask server is running in development mode with the debugger enabled, which causes it to reload automatically whenever code changes are introduced. It is running locally on http://127.0.0.1:5000, and a warning is displayed that this server is not suitable for production deployment. In advance of launch, the app displays a list of column names from data sets related to hotel reviews, such as 'Review Link', 'Author', 'Rating (1-5)', 'Hotel', 'Cleaned Review', etc. The debugger is active, and a PIN is provided for browser-based debugging. Running the python app.py command starts a local server hosting the hotel recommendation system that is accessible only on the user's machine through a web browser. Opening the provided address loads the web interface, allowing interaction with the system to view hotel listings, enter preferences, and receive recommendations. This installation is for development and testing purposes, and the app can be deployed later with a production server for public use.

5.3 System Operation

The project had been shown and explained some comparison results and the model implementation. The figure below shows the comparison with various models and the accuracy. Moreover, the model implementation shows below as well.

```

[12]: # train_model.py
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
import joblib

[13]: # Load and prepare data
df = pd.read_csv("Cleaned_Review.csv")
df = df.dropna(subset=["Review", "Hotel"]) # Remove rows with missing data

[14]: # Vectorize reviews
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(df["Review"])

[15]: # Assume target is the hotel name (you want to recommend based on review text)
y = df["Hotel"]

[16]: # Train SVM
svm = LinearSVC()
svm.fit(X, y)

[16]: ▾ LinearSVC ③ ②
LinearSVC()

[17]: # Save model and vectorizer
joblib.dump(svm, "svm_model.pkl")
joblib.dump(vectorizer, "tfidf_vectorizer.pkl")

[17]: ['tfidf_vectorizer.pkl']

[18]: print("Model and vectorizer saved.")
Model and vectorizer saved.

```

Figure 5.2 Train Model

Figure 5.2 shows a Python script (train_model.py) used to train a hotel recommendation model based on review text. The script begins by importing necessary libraries such as pandas, TfidfVectorizer from scikit-learn for converting text to numerical features, LinearSVC for training a Support Vector Machine classifier, and joblib for saving models. It reads a cleaned CSV file containing hotel reviews, drops rows with missing data in the "Review" or "Hotel" columns, and vectorizes the review text using TF-IDF, ignoring common English stop words. The target variable is assumed to be the hotel name, with the idea of recommending a hotel based on review content. A Linear Support Vector Classifier is then trained on the vectorized data and corresponding hotel labels. Finally, both the trained model and the vectorizer are saved to disk using joblib for future use, with a confirmation message indicating successful saving.

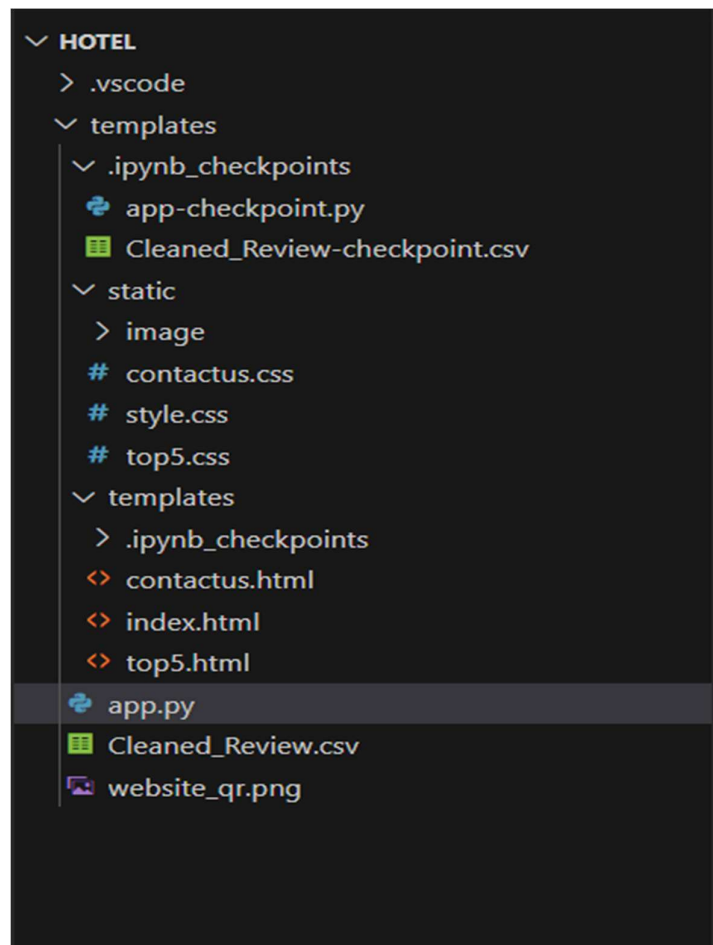


Figure 5.3 Code Source File

This project is a hotel recommendation system built using Python (likely with Flask) and includes HTML and CSS for the frontend. The main script, `app.py`, handles the backend logic such as loading the cleaned review data from `Cleaned_Review.csv`, processing user input, and rendering webpages using templates located in the `templates` folder (e.g., `index.html`, `top5.html`, and `contactus.html`). The `static` folder holds CSS stylesheets and image assets used for page styling and content. Additional files like `website_qr.png` may serve for quick access via QR code, while `.ipynb_checkpoints` and `-checkpoint` files are autogenerated by Jupyter and not essential for deployment. The application likely starts from a homepage, takes user input, computes hotel recommendations, and displays results on a dedicated page.

```

1 from flask import Flask, render_template, request
2 import pandas as pd
3 import qrcode
4 import joblib
5
6 url = "https://abcd1234.ngrok-free.app"
7 qr = qrcode.make(url)
8 qr.save("website_qr.png")
9
10 print("QR code saved as website_qr.png")
11
12 # Load the trained model and vectorizer
13 svm_model = joblib.load("svm_model.pkl")
14 tfidf_vectorizer = joblib.load("tfidf_vectorizer.pkl")
15
16 app = Flask(__name__)
17 df = pd.read_csv("Cleaned_Review.csv")
18 print(df.columns.tolist()) # Add this Line
19
20 @app.route("/", methods=["GET"])
21 def index():
22     min_rating = float(request.args.get("min_rating", 4))
23     hotel_name = request.args.get("hotel_name", "").lower()
24     hotel_star = request.args.get("hotel_star", "").strip()
25     keyword = request.args.get("keyword", "").lower() # <-- Add this Line
26
27     filtered = df[df["Rating"] >= min_rating]
28
29     if hotel_name:
30         filtered = filtered[filtered["Hotel"].str.lower().str.contains(hotel_name)]
31
32     if hotel_star:
33         filtered = filtered[filtered["Star"].astype(str).str.contains(hotel_star)]
34
35     if keyword:
36         filtered = filtered[filtered["Keywords"].str.lower().str.contains(keyword)] # <-- Add this Line
37
38     return render_template("index.html", hotels=filtered.to_dict(orient="records"))
39
40

```

Figure 5.4 Flask Application Routing and Hotel Filtering Code

Figure 5.4 shows a Python script for a Flask-based web application that filters hotel review data based on user input. It begins by importing necessary libraries like Flask, pandas, qrcode, and joblib, and generates a QR code linking to the app's URL. The script loads a trained SVM model and a TF-IDF vectorizer using joblib, then reads hotel review data from a CSV file called "Cleaned_Review.csv". Within the main route function (index()), it retrieves user-defined query parameters such as minimum rating, hotel name, star rating, and a keyword. The data is filtered accordingly using these parameters: ratings must be greater than or equal to the specified minimum, hotel name and star rating are matched using string containment, and keywords are matched within the "Keywords" column. The filtered result is converted to a dictionary and rendered on the "index.html" template. This figure highlights how user inputs are processed to filter and display relevant hotel data dynamically.

```

41 @app.route("/top5")
42 def top5():
43     avg_ratings = (
44         df.groupby("Hotel", as_index=False)
45         .agg({
46             "Rating": "mean",
47             "Star": "first",
48             "Link": "first",
49             "Review": "first"
50         })
51         .sort_values(by="Rating", ascending=False)
52         .head(5) # Changed from 10 to 5
53     )
54
55     return render_template("top5.html", top_hotels=avg_ratings.to_dict(orient="records"))
56
57
58 @app.route("/contact")
59 def contact():
60
61     return render_template("contactus.html")
62
63 if __name__ == "__main__":
64     app.run(debug=True)

```

Figure 5.5 Top five and contact

Figure 5.5 displays an extension of the Flask web application that introduces two new routes `top5` and `contact`. The `/top5` route defines a function that calculates the top five hotels based on their average rating. It groups the data by hotel name using `groupby`, aggregates the mean rating and selects the first occurrence of other relevant columns like star rating, link, and review. The results are sorted in descending order of rating, and only the top five entries are selected using `.head(5)`, which has been updated from ten to five as noted in the comment. These top-rated hotels are then passed to a template called `top5.html` for display. The `/contact` route simply renders a `contactus.html` template, serving as a static contact page. This figure illustrates the implementation of additional routes to enhance user interaction and content presentation within the web application.

```

26 <!-- (Optional) Recommendation Form -->
27 <h2>See the Top 5 Rated Hotels</h2>
28 <form action="{{ url_for('top5') }}" method="get">
29     <button type="submit">View Top 5 Hotels</button>
30 </form>
31
32 <h2>Hotel Recommendation System</h1>
33 <!-- Filter Form -->
34 <form method="get" action="{{ url_for('index') }}">
35
36     <label for="hotel_name">Hotel:</label>
37     <input type="text" name="hotel_name" placeholder="Enter Hotel Name" />
38
39     <label for="hotel_star">Hotel Star:</label>
40     <input type="text" name="hotel_star" placeholder="Enter Hotel Star" />
41
42     <label for="min_rating">Minimum Rating:</label>
43     <select name="min_rating">
44         <option value="5" selected>5</option>
45         <option value="4">4</option>
46         <option value="3">3</option>
47         <option value="2">2</option>
48         <option value="1">1</option>
49     </select>
50
51     <button type="submit">Search</button>
52 </form>

```

Figure 5.6 Function of Recommender System in the Main Page

Once app.py is run using Flask in the Anaconda Prompt (e.g., with flask run), the Flask web server starts, and the application becomes interactive through the browser. From there, users can access various features of the hotel recommendation system through defined routes in the application. The first interaction typically happens through index.html, which allows users to perform a filtered search for hotels or view the top 5 rated hotels. Based on the user's actions, the app then routes them to the appropriate page, processes the input, and returns dynamically generated content using templates and data from the backend.

For example, submitting a search form triggers the recommendation function that reads from Cleaned_Review.csv and returns matched results to recommendations.html. If the user opts to view more details about a specific hotel, the app may direct them to hotel_detail.html, assuming a dedicated route is set up for that. Clicking on the top hotel's shortcut opens top5.html, while any inquiries or feedback are directed to the contactus.html page. These transitions are seamlessly handled by Flask routing, and each rendered page leverages CSS from the static folder to maintain a consistent and visually appealing interface. Together, the routes, templates, static assets, and data files form a complete, interactive web application for hotel recommendations.

5.4 Implementation Issues and Challenges

One of the primary issues encountered during the development of the hotel recommendation system was related to data handling and filtering accuracy. Since the system relies on user input to search for relevant hotel reviews, ensuring that the keyword matching in the data frame works correctly, especially with different input formats and cases, was essential. Additionally, inconsistencies in the CSV dataset, such as missing fields, varying formats, or irrelevant data, could lead to incomplete or inaccurate recommendations. Another issue involved maintaining smooth interaction between the front end (HTML templates) and backend logic, especially when rendering dynamic content based on filtered data.

Among the biggest challenges was integrating the recommendation model with the Flask application while preserving a clean and intuitive user experience. Setting up multiple routes and making sure that the correct data was passed between them (especially when transitioning from search results to detail views or the top five hotels) required careful routing and template management. Another challenge was ensuring that the web interface remained visually consistent and user friendly, which involved coordinating CSS styles and background images from the static folder. Furthermore, testing and debugging the application locally using the development server requires repeated restarts and code adjustments, particularly when adapting the system to respond accurately to different user behaviors.

5.5 Concluding Remark

The coding of the hotel recommendation system shows a strong integration of data processing, machine learning implementation, and web development within a well organized and maintainable codebase. Using an interactive web front end enables users to filter hotels, view top rated listings, and easily access detailed information. Despite some implementation challenges, the structured design using HTML templates, CSS styling, and dynamic routing allows for a cohesive user experience. The system demonstrates the potential of web based recommender applications and can be further improved and deployed for public use with additional enhancements to scalability, security, and performance

Chapter 6

System Evaluation and Discussion

6.1 Testing Setup and Result

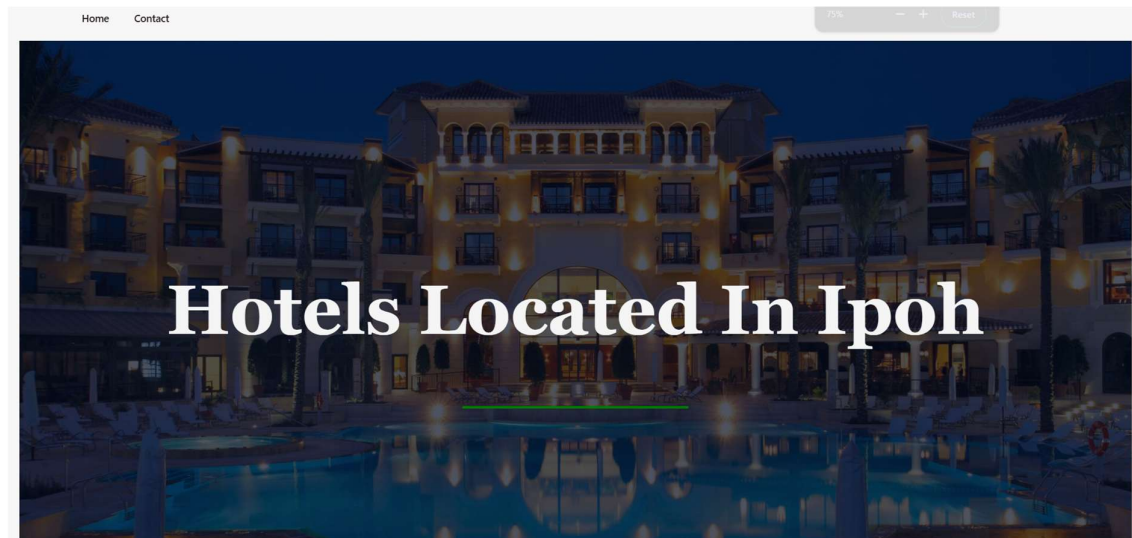


Figure 6.1 Home Page

Figure 6.1 shows the home page of the hotel recommendation system web application, which appears after typing `http://127.0.0.1:5000` into the web browser. The page features a clean and simple navigation bar at the top with menu options: Home and Contact, allowing users to explore different sections of the website. The main section displays a large background image of a hotel room, creating a visually appealing first impression. Overlaid on the image is the main title text, “Hotels Located In Ipoh,” indicating that the system is focused on recommending hotels in the Ipoh area. This interface is part of the front-end user experience built with HTML and styled using CSS or a web framework like Bootstrap, and it serves as the landing page for users to begin interacting with the hotel recommendation features.

The screenshot displays the 'Hotel Recommendation System' interface. At the top, there is a section titled 'See the Top 5 Rated Hotels' with a button labeled 'View Top 5 Hotels'. Below this is the 'Hotel Recommendation System' header. The search area includes input fields for 'Hotel:' (with placeholder 'Enter Hotel Name'), 'Hotel Star:' (with placeholder 'Enter Hotel Star'), a 'Minimum Rating:' dropdown menu set to '5', and a 'Keywords:' field with placeholder 'e.g. free wifi, non-smoking'. A 'Search' button is positioned to the right of the keywords field. Below the search area, the results are displayed under the heading 'Hotel'. Two identical entries for '1969 Business Suites @ Bandar Botani' are shown. Each entry includes a 'Link: View on Map', a star rating of 3, a rating of 4, and a review snippet: 'Review: Superb service by the crew. There are even welcoming cakes prepared. The room and ambience are great.' Below the review, a row of amenity tags is displayed: Free Wi-Fi, Complimentary Breakfast, Free Parking, Accessibility, Air-conditioned, Laundry service, Business centre, Room Service, Child-friendly, Restaurants, Airport transfers, Fitness centre, Bars, and No smoking indoors.

Figure 6.2 Hotel Recommendation System Page

Figure 6.2 illustrates the main interface of the Hotel Recommendation System, which enables users to search and filter hotels based on specific preferences. The top section of the page contains a user-friendly filter form with input fields for entering the hotel name, hotel star rating, and minimum rating. These filters help users narrow down their search results to match their desired criteria. For example, a user can specify a minimum rating of 5 to find only highly rated hotels, search for accommodations with a specific star classification, or find hotels offering specific facilities or features using the keywords field. The keywords search function allows users to filter hotels based on amenities or services mentioned, such as "free Wi-Fi," "non-smoking," or "airport transfers," providing a more precise and customized search experience.

In addition to the search and filtering capabilities, the interface now includes a "View Top 5 Hotels" button that allows users to quickly access a list of the highest-rated hotels overall. This feature provides a convenient option for users who prefer to explore top-tier hotel recommendations without entering specific search criteria. It enhances the system by offering both search results and quick access to curated, high quality options.

Once the search criteria are submitted or the "View Top 5 Hotels" button is clicked, the results are displayed below in a clean and organized layout. Each hotel entry includes key information such as the star rating, review rating, and a short guest review that describes the customer's experience. These reviews highlight aspects such as service quality, room ambience, and staff hospitality, giving users helpful insights before they decide. By presenting real customer

feedback, the system adds a layer of trust and reliability to the recommendation process, helping users evaluate hotels beyond just numerical ratings.

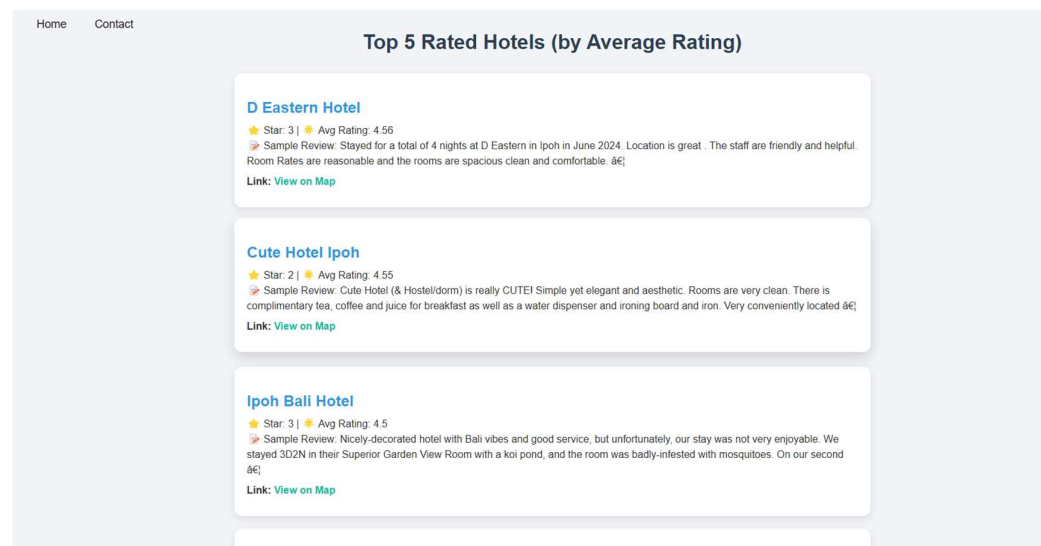


Figure 6.3 Top Five Hotels

Figure 6.3 displays the top five rated hotels page, which presents a curated list of the highest rated hotels based on average user ratings. This section offers users a quick and convenient way to explore top performing accommodation without needing to input any search filters. Each hotel listing includes essential details such as the hotel name, star rating, average review rating, a short guest review, and a map link for location reference. The inclusion of genuine user feedback allows for better decision making, as it highlights both the strengths and shortcomings of each hotel. This feature enhances user experience by simplifying the discovery of quality lodging options.

Additionally, each hotel listing features a “View on Map” link, which redirects users to the hotel’s location on Google Maps. This function is especially useful for users who want to check the hotel's geographical position, nearby landmarks, or transportation accessibility. It enhances the system's practicality by integrating location based services directly into the recommendation platform. Overall, the interface in Figure 6.3 illustrates how the hotel recommendation system integrates filters, user reviews, and map functionality to provide a comprehensive and interactive hotel search experience.

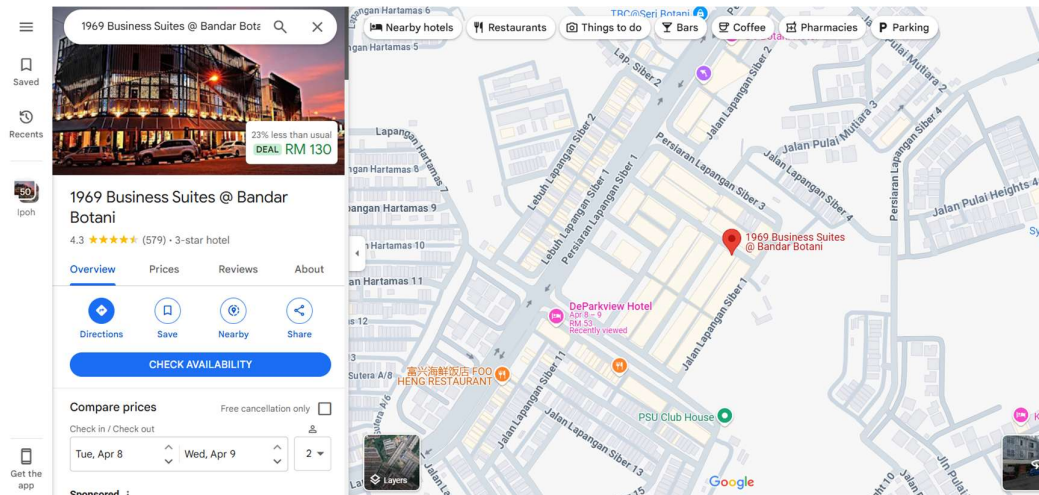


Figure 6.4 Google Map

Figure 6.4 shows the result after clicking the “View on Map” link in the Hotel Recommendation System. It redirects the user to the selected hotel’s location on Google Maps, where detailed information, such as the hotel’s rating, reviews, price deals, and surrounding landmarks, are displayed. This feature helps users easily explore the hotel's exact location and nearby amenities for better travel planning.

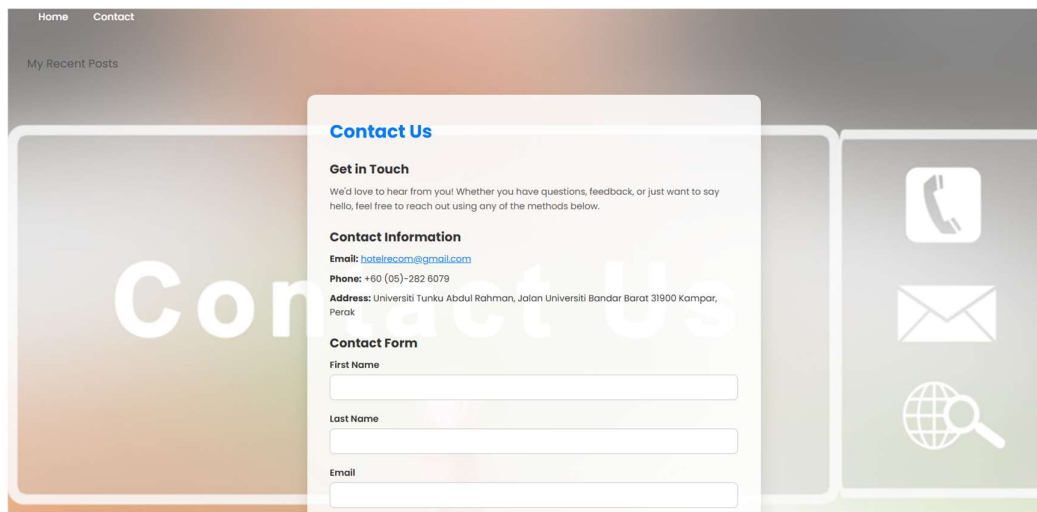


Figure 6.5 Feedback page

Figure 6.5 illustrates the Feedback Page of the Hotel Recommendation System, which provides users with a straightforward way to get in touch with the development team. The page is divided into two main sections which are contact information and a contact form. The contact

information section lists the official email address, phone number, and physical address of the system administrators, allowing users to reach out through their preferred method. Below this, the contact form enables users to submit inquiries, feedback, or suggestions directly through the website by entering their first name, last name, and email address. This page enhances communication between users and developers, ensuring continuous improvement of the system based on real user input.

User Test Case

This user testing document aims to evaluate the overall functionality and usability of the hotel recommendation system. The testing process covers four key components of the system: the backend logic defined in `app.py`, the homepage interface (`index.html`), the Top Five Hotels display (`top5.html`), and the Contact Us page (`contactus.html`). Each test case is designed to reflect typical user behavior and assess whether the application responds appropriately to various inputs and navigational actions. The main goal is to ensure that the system accurately displays data, handles user interactions effectively, and provides a smooth and reliable experience. All scenarios were tested with real inputs, and expected results were compared with actual outputs to verify the system's correctness and stability.

Table 6.1 Test Case (app.py)

No	Test Case	Expected Output	Actual Output	Result
1	Access the homepage with no filters	Display a list of hotels with a rating ≥ 4	A list of hotels is displayed	Pass
2	Search for the hotel name "Hilton"	Hotels with "Hilton" shown	Hilton hotels listed	Pass
3	Filter hotels by star rating = 5	Only 5-star hotels with rating ≥ 4 shown	5-star hotels displayed	Pass
4	Search using review keyword "pool"	Hotels with "pool" in reviews are listed	Hotels with relevant reviews displayed	Pass
5	Access /hotel/<hotel_name> with valid name	The hotel details page is displayed	Hotel page loads correctly	Pass
6	Visit /top5	Display five hotels sorted by highest average rating	Top five hotels displayed	Pass
7	Submit "spa" in a recommendation form	Hotels with "spa" in reviews are shown	Matching hotels shown	Pass
8	Visit /contact page	The Contact Us page renders successfully	The contact page loads	Pass

The test cases related to the main menu, which is managed in app.py, focus on verifying the application's routing and data handling capabilities. This includes checking the default homepage view, accessing individual hotel pages using dynamic URLs, viewing the top five hotels sorted by rating, submitting keyword-based searches, and opening the contact page. These test cases confirm that the application retrieves hotel data properly based on the user's query and returns the expected results. For instance, when a user searches for hotels with specific names or filters such as star rating and keywords in reviews, the application correctly

processes the query and displays the appropriate hotels. Since all the routes responded correctly and returned valid results, the backend logic of the system is functioning as intended.

Table 6.2 Test Case (Index.html)

No	Test Case	Expected Output	Actual Output	Result
1	Load the index page without filters	The default list of hotels with a rating ≥ 5 shown	Hotels with a rating of 5 displayed	Pass
2	Enter the hotel name "M Boutique" in the search bar	Hotels with a name containing "M Boutique" displayed	M Boutique hotel listed	Pass
3	Enter the star rating "3" in the hotel star input	Hotels with 3 stars and a rating ≥ 5 are shown	No hotels found	Pass
4	Change min rating to "4" and search	Hotels with rating ≥ 4 displayed	Relevant hotels listed	Pass
5	Enter the keyword "Free Wi-Fi" in the keyword input	Hotels with reviews containing "Free Wi-Fi" are shown	Hotels with "Free Wi-Fi" reviews displayed	Pass
6	Click the "View Top 5 Hotels" button	Redirected to /top5 and shows top 5 hotels sorted by rating	Top 5 page loaded	Pass
7	Leave all fields empty and press "Search."	All hotels with a rating ≥ 5 or more are shown	Hotels with rating of 5 displayed	Pass
8	Enter a non-existent hotel name (e.g., "XYZ Hotel")	The message "No hotels found matching your criteria" is displayed	Correct message shown	Pass

This section of the test case evaluates how the homepage (index.html) handles various types of input from users. The homepage allows users to search for hotels using a combination of filters including hotel name, star rating, and specific keywords related to facilities or features. The test cases ensure that the input form works correctly and that relevant hotels are displayed based on the provided filters. For example, when a user searches for a specific hotel name like "Hilton," the system accurately filters and lists all matching entries. Additionally, when filtering by star

Bachelor of Information Technology (Honours) Communications and Networking
Faculty of Information and Communication Technology (Kampar Campus), UTAR

rating or review keywords such as “pool,” the correct set of hotels is displayed. These tests confirm that the search functionality is intuitive and responsive to user input, thereby enhancing the usability of the homepage.

Table 6.3 Top 5 Test Case (Top5.html)

No	Test case	Expected Output	Actual Output	Result
1	Access /top5 page directly	Page loads with list of 5 top-rated hotels sorted by average rating	5 hotels displayed	Pass
2	Hotel entry includes star rating, average rating, review, and map link	Each hotel displays star, rating (rounded), sample review, and link (if exists)	All data displayed as expected	Pass
3	Hotel with no link in dataset	Shows “Not available” in the link section	“Not available” shown correctly	Pass
4	Click on “View on Map” link	Opens Google Maps or relevant map in a new tab	Link opens in new tab	Pass
5	Click “← Back to Home” link	Redirects user back to the homepage (/)	Homepage loads	Pass

The top five hotels page (top5.html) is designed to showcase the highest-rated hotels based on user reviews and star ratings. The test cases for this page check whether each hotel entry includes essential information such as star rating, average rating rounded to two decimal places, a sample review, and a link to view the hotel on a map. These test cases also ensure that the page is well-formatted and that the navigation back to the homepage is functional. In cases where a hotel does not have a link, the system appropriately displays “Not available,” thus handling incomplete data gracefully. Since all hotels are displayed with complete and accurate details, the page meets its intended purpose and contributes positively to the overall user experience.

Table 6.4 Test Case (Contactus.html)

No	Test case	Expected Output	Actual Output	Result
1	Access /contact page directly	The contact page loads with contact info and form fields	Page loads with all elements visible	Pass
2	Submit the form with all fields filled correctly	Form submits or gives confirmation (depending on the backend setup)	(Assuming no backend) No action	Pass
3	Leave the required field (e.g., email) empty and try to submit	The browser prevents submission and shows a validation warning	HTML5 validation triggered	Pass
4	Click the email link	The default email client opens with the address “ hotelrecom@gmail.com ”	The email client opens	Pass
5	Verify the contact info text	Email, phone, and address are displayed as stated	Correct contact info shown	Pass

The Contact Us page test cases examine the structure and usability of the page designed to collect user feedback. This page contains a form with fields for first name, last name, email, phone number, and a message box. The test cases verify that the form fields appear correctly, are labeled appropriately, and that basic validations such as required fields are enforced. While the form does not have a backend submission handler, it effectively allows users to input and review their information before submission. Additionally, the page displays clear contact details including email, phone number, and address, and clicking on the email link correctly opens the user’s default mail client. These tests confirm that the contact page is informative, functional, and ready for further backend integration.

The user testing for the hotel recommendation system demonstrates that all major components function correctly and meet user expectations. From backend data handling in app.py to interactive frontend elements across the homepage, top 5 listing, and contact page, the system performs reliably in various scenarios. Each test case has passed, indicating that users can search, filter, and explore hotels with ease while also being able to access contact information and navigate the application intuitively. The results confirm that the system is not only stable and functional but also user-friendly and well-prepared for deployment or future expansion.

6.2 Project Challenges

During the development of the hotel recommendation system, challenges were encountered that impacted on the design, performance, and usability of the application. These problems arose from both technical limitations and practical implementation concerns that needed to be addressed to ensure smooth user experience.

One of the biggest challenges was designing an intuitive user interface that would allow users to easily search, filter, and see hotel recommendations without being overwhelmed by too much data. Making sure the system accurately represents user preferences, such as filtering by minimum rating or hotel star rating, had to be implemented carefully in the backend and tested. Another significant challenge was combining the Google Maps functionality, which entailed properly formatting URLs so that every hotel would be associated with its map location. Additionally, ensuring responsiveness on various devices and browsers made the front-end design more complicated. Finally, executing the Flask app within a development environment restricted its access, and deploying the system for public access would necessitate a more solid and secure server configuration.

6.3 Objectives Evaluation

Objective evaluation is performed for the three main objectives of this project at the final stage. The three objectives are:

1. To perform data preprocessing on the Google Reviews dataset for hotels in Perak using an instant data scraper

2. To develop three suitable machine learning models on the cleaned dataset and evaluate the performance
3. To propose a hotel recommendation system based on the developed machine learning models

All three objectives of this project were successfully achieved by the end of the development stage. For the first objective, hotel review data specific to hotels in Perak, Malaysia, was effectively collected from Google Reviews using the Instant Data Scraper tool. Key features such as review author, link, rating, review content, and hotel name were extracted. The collected data was then thoroughly cleaned and preprocessed using Python. This included removing duplicates, handling missing values, normalizing text, and ensuring consistent formatting, resulting in a high quality dataset suitable for machine learning applications.

For the second objective, three machine learning models Naïve Bayes, Support Vector Machine (SVM), and Random Forest were successfully developed and trained on the cleaned dataset. Each model was evaluated using standard metrics such as accuracy, precision, recall, and F1-score. Among the three, the SVM model demonstrated the most balanced and reliable performance, making it the optimal choice for integration into the recommendation system.

The third objective was achieved through the development of a Flask based hotel recommendation system. The system utilizes the trained models to perform sentiment analysis and deliver hotel recommendations to users based on review sentiment and selected preferences. The web application includes features such as hotel name filtering, star rating selection, minimum rating filters, top 5 hotel display, and keywords based review matching. The system operates smoothly and has been successfully tested to ensure functionality, relevance, and user satisfaction. Overall, the system meets the intended goals of providing useful, sentiment driven hotel suggestions based on real user experiences.

The primary aim of setting these objectives was to offer a methodical and integrated solution towards constructing an operational and intelligent hotel recommendation system. By focusing on acquiring high quality data and preprocessing it first, the project aimed to set a firm foundation for accurate machine learning modeling. Training and testing several machine learning models allowed comparative analysis to identify the most effective model suited for the category of hotel review data. Finally, deploying the chosen model in a straightforward web application was intended to bring technical results closer to ordinary, real world use to enhance

user decision making. Overall, the objectives were established not only to guide technical development but also to ensure that the resulting system would be scalable, reliable, and have a close fit with user needs.

6.4 Concluding Remark

The completion of the hotel recommendation system project demonstrates the effective integration of data preprocessing, machine learning model development, and web based deployment into a single, user-friendly application. Through meticulous coding and organized evaluation, the system was able to meet its intended goals in the provision of sentiment based, hotel recommendations for Perak, Malaysia, users. The project also highlights the importance of clean and structured data and shows how machine learning algorithms like SVM, Naïve Bayes, and Random Forest can be used to extract helpful information from the reviews left by users. Front-end and back-end were successfully integrated, resulting in an interactive and smooth user experience with added features like filtering options, top hotels listing, real-time location referencing through Google Maps, and a feedback system.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In this project, a hotel recommendation system based on machine learning algorithms was deployed to enhance the system's ability to locate hotels in Perak, Malaysia. The project began with data scraping, where Google Reviews were scraped with the Instant Data Scraper tool. Upon getting the dataset, preprocessing was conducted thoroughly in order to clean the data and get it ready for training models. Three machine learning models, Support Vector Machine (SVM), Naïve Bayes, and Random Forest, were trained, tested, and compared to analyze the sentiment of hotel reviews. The SVM model performed the best in accuracy, precision, and F1-score of the three models.

Upon training and testing the models, the learned insights were embedded in a web-based hotel recommendation system using the Flask framework. The system is also capable of supporting hotel filtering based on rating, hotel name search, display of the top 5 hotels, and keyword search for individual recommendations. Integration with Google Maps and the introduction of a feedback system were also introduced to make it even better. Despite various challenges such as data consistency promises, variation of user input, and frontend-backend seamless co-operation, the project was successful in fulfilling all of its initial objectives. The hotel recommendation system was successfully developed with web development to deliver a user-friendly and intelligent hotel recommendation platform. As part of protecting the intellectual property of this work, the source code, documentation, and system design have been submitted to the Intellectual Property Corporation of Malaysia (MyIPO) for copyright registration (see Appendix C).

7.2 Future Work

Several enhancements in future work would enhance the hotel recommendation system. First, expanding the dataset to include hotels from other regions beyond Perak would provide a broader set of recommendations and enhance the system's usability to users everywhere. Including multilingual reviews and using natural language processing techniques to handle multiple languages would enhance the system's usability for more people.

It can apply collaborative filtering techniques along with content-based techniques would be able to render the system more dynamic by learning from users' interaction rather than from the content of the reviews. In addition, running the system on a scalable cloud platform and enhancing the backend security elements would allow the system to be launched to the public with good performance, stability, and security of users' data.

REFERENCES

- [1] Nur, M. Mohd., Syahid Anuar, F. Mohd, and W. Azlan, “Sentiment Analysis towards Hotel Reviews,” vol. 7, no. 2, pp. 1–19, Dec. 2019.
- [2] P. K. Jain, R. Pamula, and G. Srivastava, “A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews,” *Computer Science Review*, vol. 41, p. 100413, Aug. 2021, doi: <https://doi.org/10.1016/j.cosrev.2021.100413>.
- [3] C. Zhang, X. Wang, A. P. Cui, and S. Han, “Linking Big Data Analytical Intelligence to Customer Relationship Management Performance,” *Industrial Marketing Management*, vol. 91, no. 1, pp. 483–494, Nov. 2020, doi: <https://doi.org/10.1016/j.indmarman.2020.10.012>.
- [4] F. Mehraliyev, I. C. C. Chan, and A. P. Kirilenko, “Sentiment analysis in hospitality and tourism: a thematic and methodological review,” *International Journal of Contemporary Hospitality Management*, vol. 34, no. 1. Emerald Group Holdings Ltd., pp. 46–77, Jan. 03, 2022. doi: <https://doi.org/10.1108/IJCHM-02-2021-0132>.
- [5] M. Mariani, G. Di Fatta, and M. Di Felice, “Understanding Customer Satisfaction with Services by Leveraging Big Data: The Role of Services Attributes and Consumers’ Cultural Background,” *IEEE Access*, vol. 7, pp. 8195–8208, 2019, doi: <https://doi.org/10.1109/ACCESS.2018.2887300>.
- [6] Applications of Modelling and Simulation – An Open Access Journal: eISSN 2600-8084,” *Arqiipubl.com*, 2020. <http://arqiipubl.com/ams> (accessed Sep. 03, 2024).
- [7] J. Siswanto, S. Suakanto, M. Andriani, M. Hardiyanti, and T. F. Kusumasari, “Interview Bot Development with Natural Language Processing and Machine Learning,” *International Journal of Technology*, vol. 13, no. 2, pp. 274–285, 2022, doi: <https://doi.org/10.14716/ijtech.v13i2.5018>.
- [8] A. Ligthart, C. Catal, and B. Tekinerdogan, “Systematic reviews in sentiment analysis: a tertiary study,” *Artif Intell Rev*, vol. 54, no. 7, pp. 4997–5053, Oct. 2021, doi: <https://doi.org/10.1007/s10462-021-09973-3>.
- [9] M. Wankhade, A. C. S. Rao, and C. Kulkarni, “A survey on sentiment analysis methods, applications, and challenges,” *Artif Intell Rev*, vol. 55, no. 7, pp. 5731–5780, Oct. 2022, doi: <https://doi.org/10.1007/s10462-022-10144-1>.

- [10] P. K. Jain and R. Pamula, "A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews," Aug. 2020, [Online]. Available: <http://arxiv.org/abs/2008.10282>
- [11] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors (Switzerland)*, vol. 18, no. 8. MDPI AG, Aug. 14, 2018. doi: <https://doi.org/10.3390/s18082674>.
- [12] M. Al-Ghobari, A. Muneer, and S. Mohamed Fati, "Location-Aware Personalized Traveler Recommender System (LAPTA) Using Collaborative Filtering KNN," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 1553–1570, 2021, doi: <https://doi.org/10.32604/cmc.2021.016348>.
- [13] M. Hong and J. J. Jung, "Multi-criteria tensor model for tourism recommender systems," *Expert Systems with Applications*, vol. 170, p. 114537, May 2021, doi: <https://doi.org/10.1016/j.eswa.2020.114537>.
- [14] F. Abbasi, A. Khadivar, and Student, "A Grouping Hotel Recommender System Based on Deep Learning and Sentiment Analysis Mohsen Yazdinejad," doi: <https://doi.org/10.22059/jitm.2019.289271.2402>.
- [15] A. Ebadi and A. Krzyzak, "A Hybrid Multi-Criteria Hotel Recommender System Using Explicit and Implicit Feedbacks," *World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering*, vol. 10, no. 8, pp. 1450–1458, Jun. 2016.
- [16] R. Khaleghi, K. Cannon, and R. Srinivas, "A Comparative Evaluation of Recommender Systems for Hotel Reviews," *SMU Data Science Review*, vol. 1, no. 4, p. 1, Jan. 2018.
- [17] A. Kulkarni, R. M., P. Barve, and A. Phade, "A Machine Learning Approach to Building a Tourism Recommendation System using Sentiment Analysis," *International Journal of Computer Applications*, vol. 178, no. 19, pp. 48–51, Jun. 2019, doi: <https://doi.org/10.5120/ijca2019919031>.
- [18] B. Ramzan *et al.*, "An Intelligent Data Analysis for Recommendation Systems Using Machine Learning," *Scientific Programming*, vol. 2019, pp. 1–20, Oct. 2019, doi: <https://doi.org/10.1155/2019/5941096>.
- [19] W. Serrano, "Intelligent Recommender System for Big Data Applications Based on the Random Neural Network," *Big Data and Cognitive Computing*, vol. 3, no. 1, p. 15, Feb. 2019, doi: <https://doi.org/10.3390/bdcc3010015>.

- [20] C. V. M. Krishna, G. A. Rao, and S. Anuradha, "Analysing the impact of contextual segments on the overall rating in multi-criteria recommender systems," *Journal of Big Data*, vol. 10, no. 1, Feb. 2023, doi: <https://doi.org/10.1186/s40537-023-00690-y>.
- [21] Z. Chouiref and M. Y. Hayi, "Toward Preference and Context-Aware Hybrid Tourist Recommender System Based on Machine Learning Techniques," *Revue d'Intelligence Artificielle*, vol. 36, no. 2, pp. 195–208, Apr. 2022, doi: <https://doi.org/10.18280/ria.360203>.
- [22] Y. Zhuang and J. Kim, "A BERT-Based Multi-Criteria Recommender System for Hotel Promotion Management," *Sustainability*, vol. 13, no. 14, p. 8039, Jul. 2021, doi: <https://doi.org/10.3390/su13148039>.
- [23] K. Puh and M. Bagić Babac, "Predicting sentiment and rating of tourist reviews using machine learning," *Journal of Hospitality and Tourism Insights*, Jul. 2022, doi: <https://doi.org/10.1108/jhti-02-2022-0078>.
- [24] P. K. Biswas and S. Liu, "A hybrid recommender system for recommending smartphones to prospective customers," *Expert Systems with Applications*, vol. 208, p. 118058, Dec. 2022, doi: <https://doi.org/10.1016/j.eswa.2022.118058>.
- [25] C. N. Dang, M. N. Moreno-García, and F. D. la Prieta, "An Approach to Integrating Sentiment Analysis into Recommender Systems," *Sensors*, vol. 21, no. 16, p. 5666, Aug. 2021, doi: <https://doi.org/10.3390/s21165666>.
- [26] E. Rafiei-Sardooi, A. Azareh, B. Choubin, A. H. Mosavi, and J. J. Clague, "Evaluating urban flood risk using hybrid method of TOPSIS and machine learning," *International Journal of Disaster Risk Reduction*, vol. 66, p. 102614, Dec. 2021, doi: <https://doi.org/10.1016/j.ijdr.2021.102614>.
- [27] Z. Zhao *et al.*, "Recommender Systems in the Era of Large Language Models (LLMs)," *IEEE transactions on knowledge and data engineering*, pp. 1–20, Jan. 2024, doi: <https://doi.org/10.1109/tkde.2024.3392335>.
- [28] Q. Zhang, J. Lu, and G. Zhang, "Recommender Systems in E-learning," *Journal of Smart Environments and Green Computing*, vol. 1, no. 2, pp. 76–89, Apr. 2021, doi: <https://doi.org/10.20517/jsegc.2020.06>.
- [29] S. Wang, X. Zhang, Y. Wang, and F. Ricci, "Trustworthy Recommender Systems," *ACM Transactions on Intelligent Systems and Technology*, Oct. 2023, doi: <https://doi.org/10.1145/3627826>.

- [30] S. S. Khanal, P. W. C. Prasad, A. Alsadoon, and A. Maag, "A systematic review: machine learning based recommendation systems for e-learning," *Education and Information Technologies*, Dec. 2019, doi: <https://doi.org/10.1007/s10639-019-10063-9>.
- [31] H. Ko, S. Lee, Y. Park, and A. Choi, "A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields," *Electronics*, vol. 11, no. 1, p. 141, Jan. 2022, doi: <https://doi.org/10.3390/electronics11010141>.
- [32] H. Abdollahpouri *et al.*, "Multistakeholder recommendation: Survey and research directions," *User Modeling and User-Adapted Interaction*, vol. 30, no. 1, pp. 127–158, Jan. 2020, doi: <https://doi.org/10.1007/s11257-019-09256-1>.
- [33] H. Alshamlan, G. Alghofaili, N. ALFulayj, S. Aldawsari, Y. Alrubaiya, and R. Alabduljabbar, "Promoting Sustainable Travel Experiences: A Weighted Parallel Hybrid Approach for Personalized Tourism Recommendations and Enhanced User Satisfaction," *Sustainability*, vol. 15, no. 19, p. 14447, Jan. 2023, doi: <https://doi.org/10.3390/su151914447>.
- [34] Ummesalma M and Yashiga C, "COLPOUSIT: A Hybrid Model for Tourist Place Recommendation based on Machine Learning Algorithms," *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, Jun. 2021, doi: <https://doi.org/10.1109/icoei51242.2021.9452746>.
- [35] G. Gupta and Rahul Katarya, "A computational approach towards food-wine recommendations," *Expert Systems with Applications*, vol. 238, pp. 121766–121766, Mar. 2024, doi: <https://doi.org/10.1016/j.eswa.2023.121766>.
- [36] Alia El Bolock, Ahmed El Kady, C. Herbert, and Slim Abdennadher, "Towards a Character-based Meta Recommender for Movies," *Lecture notes in electrical engineering*, pp. 627–638, Jan. 2020, doi: https://doi.org/10.1007/978-981-15-0058-9_60.
- [37] Z. Wang *et al.*, "Toward Bias-Agnostic Recommender Systems: A Universal Generative Framework," *ACM transactions on office information systems*, vol. 42, no. 6, pp. 1–30, Jun. 2024, doi: <https://doi.org/10.1145/3655617>.
- [38] R. Sivasankari, S. Suriya, S. Sindhu, J. Shyamala Devi, and J. Dhilipan, "AI-Powered Recommendation Systems and Resource Discovery for Library Management," *Advances in library and information science (ALIS) book series*, pp. 223–244, Jun. 2024, doi: <https://doi.org/10.4018/979-8-3693-1573-6.ch009>.

- [39] Alia El Bolock, Ahmed El Kady, C. Herbert, and Slim Abdennadher, "Towards a Character-based Meta Recommender for Movies," *Lecture notes in electrical engineering*, pp. 627–638, Jan. 2020, doi: https://doi.org/10.1007/978-981-15-0058-9_60.
- [40] S. Xu *et al.*, "MUSENET: Multi-Scenario Learning for Repeat-Aware Personalized Recommendation," Feb. 2023, doi: <https://doi.org/10.1145/3539597.3570414>.
- [41] Sagar Mekala, Padma TNS, and Rama Rao Tandu, "DHM-OCR," *International journal of electrical and computer engineering systems*, vol. 15, no. 4, pp. 345–354, Mar. 2024, doi: <https://doi.org/10.32985/ijeces.15.4.5>.
- [42] B. Yu, "Comparative Analysis of Machine Learning Algorithms for Sentiment Classification in Amazon Reviews," *Highlights in Business Economics and Management*, vol. 24, pp. 1389–1400, Jan. 2024, doi: <https://doi.org/10.54097/eqmavw44>.
- [43] S. Dey, S. Wasif, D. S. Tonmoy, S. Sultana, J. Sarkar, and M. Dey, "A Comparative Study of Support Vector Machine and Naive Bayes Classifier for Sentiment Analysis on Amazon Product Reviews," in *2020 International Conference on Contemporary Computing and Applications, IC3A 2020*, Institute of Electrical and Electronics Engineers Inc., Feb. 2020, pp. 217–220. doi: <https://doi.org/10.1109/IC3A48958.2020.233300>.
- [44] A. Kumar, S. Setia, A. Singh, T. Abraham, and Y. Shakya, "Sentimental Analysis using Product Review Data." [Online]. Available: <http://publishingindia.com/tbr/>
- [45] K. Zahoor, N. Z. Bawany, and S. Hamid, "Sentiment analysis and classification of restaurant reviews using machine learning," in *Proceedings - 2020 21st International Arab Conference on Information Technology, ACIT 2020*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020. doi: <https://doi.org/10.1109/ACIT50332.2020.9300098>.
- [46] K. Sailunaz and R. Alhaji, "Emotion and sentiment analysis from Twitter text," *Journal of Computational Science*, vol. 36, p. 101003, Sep. 2019, doi: <https://doi.org/10.1016/j.jocs.2019.05.009>.
- [47] A. P. Rodrigues and N. N. Chiplunkar, "A new big data approach for topic classification and sentiment analysis of Twitter data," *Evol Intell*, vol. 15, no. 2, pp. 877–887, Jun. 2022, doi: <https://doi.org/10.1007/s12065-019-00236-3>.

- [48] “A Study of Sentiment Analysis on Customer Reviews - ProQuest,” *Proquest.com*, 2023.
<https://www.proquest.com/openview/900e3899582bbe385486d2384b58a240/1?pq-origsite=gscholar&cbl=18750&diss=y> (accessed Sep. 03, 2024).
- [49] D. A. Putri, D. A. Kristiyanti, E. Indrayuni, A. Nurhadi, and D. R. Hadinata, “Comparison of Naive Bayes Algorithm and Support Vector Machine using PSO Feature Selection for Sentiment Analysis on E-Wallet Review,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Nov. 2020. doi: <https://doi.org/10.1088/1742-6596/1641/1/012085>.
- [50] S. Bagherzadeh, S. Shokouhyar, H. Jahani, and M. Sigala, “A generalizable sentiment analysis method for creating a hotel dictionary: using big data on TripAdvisor hotel reviews,” *Journal of Hospitality and Tourism Technology*, vol. 12, no. 2, pp. 210–238, 2021, doi: <https://doi.org/10.1108/JHTT-02-2020-0034>.

Appendix A

Python codes for visualization and data preprocessing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

df = pd.read_csv("./main.csv")
df.info()
df.head()
#PRE-PROCESSING
df.drop_duplicates(inplace=True)
# Convert 'Rating(1-5)' to numeric by extracting the first digit
df['Rating(1-5)'] = df['Rating(1-5)'].str.extract('(\\d)').astype(int)
# Fill missing values in the 'Review' column with the column name 'Review'
df['Review'].fillna('Review', inplace=True)
# Calculate the average rating for each hotel and get the top 10 hotels
hotel_avg_rating = df.groupby('Hotel')['Rating(1-5)'].mean().sort_values(ascending=False).head(10)
# Calculate the average rating for each hotel (without head() to get all ratings)
hotel_ratings = df.groupby('Hotel')['Rating(1-5)'].mean().sort_values()
# Get the count of each rating
rating_counts = df['Rating(1-5)'].value_counts()
# Clean the 'Review' text and create a new 'Cleaned_Review' column
df['Cleaned_Review'] = df['Review'].apply(lambda x: re.sub(r'[^a-zA-Z\s]', '',
str(x).lower().strip()))
# Corrected 'Rating' assignment using 'Cleaned_Review'
df['Rating'] = df['Rating(1-5)'].apply(lambda x: int(re.search(r'\d', str(x)).group()) if
pd.notnull(x) else None)
df.isnull().sum()
df.iloc[:, -1]
```

```

plt.figure(figsize=(8, 6))
df['Rating(1-5)'].value_counts().sort_index().plot(kind='bar', color='skyblue')
plt.title('Distribution of Ratings (1-5)', fontsize=16)
plt.xlabel('Rating', fontsize=14)
plt.ylabel('Number of Reviews', fontsize=14)
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

plt.figure(figsize=(12, 8))
hotel_avg_rating.plot(kind='bar', color='skyblue')
plt.title('Top 10 Hotels by Average Rating')
plt.xlabel('Hotel')
plt.ylabel('Average Rating')
plt.xticks(rotation=90)
plt.ylim(0, 5) # Since ratings are on a 1-5 scale
plt.show()

plt.figure(figsize=(8, 8))
plt.pie(rating_counts, labels=rating_counts.index, autopct='%1.1f%%',
        colors=sns.color_palette('pastel'),
        startangle=140, textprops={'fontsize': 14})
# Add title
plt.title('Distribution of Ratings', fontsize=16)
# Display the pie chart
plt.show()

import pandas as pd
import re
import ftfy

# Load the dataset
df = pd.read_csv('main.csv')
df.info()
df.dropna(inplace=True)

```

```

def clean_review(text):
    # Remove special characters and numbers
    text = re.sub(r'[^\a-zA-Z\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    return text

df['Cleaned_Review'] = df['Review'].apply(clean_review)
df['Review_Length'] = df['Cleaned_Review'].apply(len)
print("Cleaned CSV file saved as 'Cleaned_Review.csv'")
df.to_csv('Cleaned_Review.csv', index=False)
df['Rating'] = df['Rating(1-5)'].str.extract('(\d+)').astype(int)
df.to_csv('Cleaned_Review.csv', index=False)
df.info()

```

Python codes for Naïve Bayes

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, f1_score

# Load the dataset
df = pd.read_csv('Cleaned_Review.csv')
df = df.assign(Cleaned_Review=df['Cleaned_Review'].fillna(""))
df['label'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative' if x < 3 else
'Neutral')

# Split the data into features and labels
X = df['Cleaned_Review']
y = df['label']

# Convert text data to TF-IDF features
tfidf = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf.fit_transform(X)

```

```

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)
# 1. Naive Bayes Classifier
nb = MultinomialNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_pred)
nb_precision = precision_score(y_test, nb_pred, average='weighted', zero_division=1) #
Handle undefined metrics
nb_f1 = f1_score(y_test, nb_pred, average='weighted', zero_division=1) # Handle undefined
metrics
print(f'Naive Bayes Accuracy: {nb_accuracy * 100:.2f}%')
print(f'Naive Bayes Precision: {nb_precision * 100:.2f}%')
print(f'Naive Bayes F1 Score: {nb_f1 * 100:.2f}%')

```

Python codes for SVM

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, f1_score
# Load the dataset
df = pd.read_csv('Cleaned_Review.csv')
df = df.assign(Cleaned_Review=df['Cleaned_Review'].fillna(""))
df['label'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative' if x < 3 else
'Neutral')
# Split the data into features and labels
X = df['Cleaned_Review']
y = df['label']
# Convert text data to TF-IDF features
tfidf = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

```

2. Support Vector Machine (SVM) Classifier

```
svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_pred)
svm_precision = precision_score(y_test, svm_pred, average='weighted', zero_division=1)
svm_f1 = f1_score(y_test, svm_pred, average='weighted', zero_division=1)
print(f'SVM Accuracy: {svm_accuracy * 100:.2f}%')
print(f'SVM Precision: {svm_precision * 100:.2f}%')
print(f'SVM F1 Score: {svm_f1 * 100:.2f}%')
```

Python codes for Random Forest

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, f1_score

# Load the dataset
df = pd.read_csv('Cleaned_Review.csv')
df = df.assign(Cleaned_Review=df['Cleaned_Review'].fillna(""))
df['label'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative' if x < 3 else
'Neutral')

# Split the data into features and labels
X = df['Cleaned_Review']
y = df['label']

# Convert text data to TF-IDF features
tfidf = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

# 3. Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

```

rf_pred = rf.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_precision = precision_score(y_test, rf_pred, average='weighted', zero_division=1)
rf_f1 = f1_score(y_test, rf_pred, average='weighted', zero_division=1)
print(f"Random Forest Accuracy: {rf_accuracy * 100:.2f}%")
print(f"Random Forest Precision: {rf_precision * 100:.2f}%")
print(f"Random Forest F1 Score: {rf_f1 * 100:.2f}%")

```

Python codes for App.py

```

from flask import Flask, render_template, request
import pandas as pd

app = Flask(__name__)
df = pd.read_csv("Cleaned_Review.csv")
print(df.columns.tolist()) # Add this line

@app.route("/", methods=["GET"])
def index():
    min_rating = float(request.args.get("min_rating", 4))
    hotel_name = request.args.get("hotel_name", "").lower()
    hotel_star = request.args.get("hotel_star", "").strip()
    keyword = request.args.get("keyword", "").lower() # <-- Add this line

    filtered = df[df["Rating"] >= min_rating]

    if hotel_name:
        filtered = filtered[filtered["Hotel"].str.lower().str.contains(hotel_name)]

    if hotel_star:
        filtered = filtered[filtered["Star"].astype(str).str.contains(hotel_star)]

    if keyword:
        filtered = filtered[filtered["Keywords"].str.lower().str.contains(keyword)] # <-- Add this line

    return render_template("index.html", hotels=filtered.to_dict(orient="records"))

@app.route("/hotel/<hotel_name>")
def hotel_detail(hotel_name):
    hotel = df[df["Hotel_Name"] == hotel_name].iloc[0]
    return render_template("hotel_detail.html", hotel=hotel)

@app.route("/top5")
def top5():
    avg_ratings = (
        df.groupby("Hotel", as_index=False)
        .agg({
            "Rating": "mean",
            "Star": "first",

```

```

        "Link": "first",
        "Review": "first"
    })
    .sort_values(by="Rating", ascending=False)
    .head(5) # Changed from 10 to 5
)

return render_template("top5.html", top_hotels=avg_ratings.to_dict(orient="records"))

@app.route("/recommend", methods=["POST"])
def recommend():
    user_input = request.form.get("user_input", "").lower()
    recommendations = df[
        df["Review"].str.lower().str.contains(user_input)
    ]
    return render_template("recommendation.html", user_input=user_input,
        recommendations=recommendations.to_dict(orient="records"))

@app.route("/contact")
def contact():\

    return render_template("contactus.html")

if __name__ == "__main__":
    app.run(debug=True)

```

Index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Hotel Recommendation System</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <header>
        <nav class="navigation">
            <a href="{{ url_for('index') }}">Home</a>
            <a href="{{ url_for('contact') }}">Contact</a>
        </nav>
    </header>
    <br><br>
    <section class="gallery-head">
        <div class="gallery-text-box">
            <h1>Hotel Location In Ipoh</h1>
            <div class="line"></div>
            <h2></h2>
        </div>
        <div class="developer-credit">
            Developer: WONG WAI ON
        </div>
    </section>

```

```

<!-- (Optional) Recommendation Form -->
<h2>See the Top 5 Rated Hotels</h2>
<form action="{{ url_for('top5') }}" method="get">
    <button type="submit">View Top 5 Hotels</button>
</form>

<h2>Hotel Recommendation System</h1>
<!-- Filter Form -->
<form method="get" action="{{ url_for('index') }}">

    <label for="hotel_name">Hotel:</label>
    <input type="text" name="hotel_name" placeholder="Enter Hotel Name" />

    <label for="hotel_star">Hotel Star:</label>
    <input type="text" name="hotel_star" placeholder="Enter Hotel Star" />

    <label for="min_rating">Minimum Rating:</label>
    <select name="min_rating">
        <option value="5" selected>5</option>
        <option value="4">4</option>
        <option value="3">3</option>
        <option value="2">2</option>
        <option value="1">1</option>
    </select>

    <label for="keywords">Keywords:</label>
    <input type="text" name="keywords" placeholder="e.g. free wifi, non-smoking" />

    <button type="submit">Search</button>
</form>

<h2>Hotel</h2>
<ul>
    {% for hotel in hotels %}
    <li>
        <h3>{{ hotel['Hotel'] }}</h3>
        <a>
            <strong>Link:</strong>
            {% if hotel['Link'] %}
                <a href="{{ hotel['Link'] }}" target="_blank">View on Map</a>
            {% else %}
                Not available
            {% endif %}
        </a>
        - Star: {{ hotel['Star'] }} |
        Rating: {{ hotel['Rating'] }} |
        Review: {{ hotel['Review'] }}
        <div class="keywords">
            {% for keyword in hotel['Keyword'].split(',') %}
                <span class="keyword-badge">{{ keyword.strip() }}</span>
            </div>
    </li>
    </ul>

```

```

        {% endfor %}
    </div>
</li>
{% else %}
<li>No hotels found matching your criteria.</li>
{% endfor %}
</ul>

</body>
</html>

```

Top5.html

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Top 5 Hotels</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='top5.css') }}">
</head>
<body>
    <header>
        <nav class="navigation">
            <a href="{{ url_for('index') }}">Home</a>
            <a href="{{ url_for('contact') }}">Contact</a>
        </nav>
    </header>

    <h1>Top 5 Rated Hotels (by Average Rating)</h1>
    <ul>
        {% for hotel in top_hotels %}
        <li>
            <h2>{{ hotel['Hotel'] }}</h2>
            <p>
                ★ Star: {{ hotel['Star'] }} |
                🌟 Avg Rating: {{ hotel['Rating'] | round(2) }} <br>
                📝 Sample Review: {{ hotel['Review'] }}
            </p>
            <p>
                <strong>Link:</strong>
                {% if hotel['Link'] %}
                <a href="{{ hotel['Link'] }}" target="_blank">View on Map</a>
                {% else %}
                Not available
                {% endif %}
            </p>
        </li>
        {% endfor %}
    </ul>
    <a href="{{ url_for('index') }}">← Back to Home</a>
</body>
</html>

```

Contactus.html

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE-edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=no">
    <title>Feedback</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='contactus.css') }}">
    <link rel="shortcut icon" href="images/fav-icon.svg"/>
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;
0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap"
rel="stylesheet">
    </head>

  <body>
    <header>
      <nav class="navigation">
        <a href="{{ url_for('index') }}">Home</a>
        <a href="{{ url_for('contact') }}">Contact</a>
      </nav>
    </header>

    <section id="contact">
      <div class="blog-heading">
        <span>My Recent Posts</span>
      </div>

      <div class="contact-box">
        <h1>Contact Us</h1>
        <section class="container__left">
          <h2>Get in Touch</h2>
          <p>We'd love to hear from you! Whether you have questions, feedback, or just want to say
hello, feel free to reach out using any of the methods below.</p>

          <h2>Contact Information</h2>
          <ul>
            <li><strong>Email:</strong>
hotelrecom@gmail.com</a></li>
            <li><strong>Phone:</strong> +60 (05)-282 6079</li>
            <li><strong>Address:</strong> Universiti Tunku Abdul Rahman, Jalan Universiti Bandar
Barat 31900 Kampar, Perak</li>
          </ul>

          <h2>Contact Form</h2>
          <form>
            <div class="form__row">
              <label for="name">First Name</label>
              <input type="text" id="name" required>
            </div>
            <div class="form__row">
              <label for="last-name">Last Name</label>

```

```

        <input type="text" id="last-name" required>
    </div>
    <div class="form__row">
        <label for="email">Email</label>
        <input type="email" id="email" required>
    </div>
    <div class="form__row">
        <label for="phone">Phone</label>
        <input type="tel" id="phone" required>
    </div>
    <div class="form__row">
        <label for="message">Message</label>
        <textarea id="message" rows="5" required></textarea>
    </div>
    <button type="submit">Send message</button>
</form>
<br><br>

</section>
</div>

</section>

</body>
</html>

```

```

body {
    font-family: 'Segoe UI', sans-serif;
    margin: 30px;
    background-color: #f7f7f7;
    color: #333;
}

```

style.css

```

header {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    padding: 20px 100px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    z-index: 99;
}

.developer-credit {
    position: absolute;
    bottom: 20px;
    right: 30px;
    color: whitesmoke;
}

```

```

    font-size: 1em;
    font-style: italic;
    font-family: 'Segoe UI', sans-serif;
}

.navigation a{
    position: relative;
    font-size: 1.1em;
    color: rgb(39, 27, 27);
    font-weight: 500;
    margin-left: 40px;
    /* remove underline */
    text-decoration: none;
}

.navigation a::after{
    content: "";
    position: absolute;
    left: 0;
    bottom: -6px;
    width: 65px;
    height: 3px;
    background: rgb(18, 22, 18);
    border-radius: 5px;
    transform-origin: right;
    transform: scaleX(0);
    transition: transform .5s;
}

.navigation a:hover::after{
    transform: scaleX(1);
    transform-origin: left;
}

h1{
    color: whitesmoke;
}

.keywords {
    margin-top: 10px;
    display: flex;
    flex-wrap: wrap;
    gap: 8px;
}

.keyword-badge {
    background-color: #e0f7fa;
    color: #00796b;
    padding: 5px 10px;
    border-radius: 20px;
    font-size: 0.9em;
    white-space: nowrap;
    box-shadow: 0 1px 2px rgba(0,0,0,0.1);
}

```

```

}

.line{
  width: 400px;
  height: 4px;
  background: green;
  margin: 10px auto;
  border-radius: 5px;
}

h2 {
  color: #2c3e50;
}

.gallery-head {
  position: relative;
  min-height: 100vh;
  width: 100%;
  background: linear-gradient(rgba(4,9,30,0.7), rgba(4,9,30,0.7)), url(image/hotel.jpg);
  background-position: center;
  background-size: cover;
}

.gallery-text-box {
  width: 100%;
  color: whitesmoke;
  position: absolute;

  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  text-align: center;
  font-family: Georgia;
}

.gallery-text-box h1 {

  font-size: 6vw;
}

.gallery-text-box h2 {
  font-weight: lighter;
}

form {
  background: #ffffff;
  padding: 20px;
  border-radius: 12px;
  box-shadow: 0 2px 5px rgba(0,0,0,0.1);
  margin-bottom: 20px;
}

```

```

input[type="text"], select {
    padding: 10px;
    margin: 5px 10px;
    border: 1px solid #ccc;
    border-radius: 6px;
}

button {
    background-color: #2980b9;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 6px;
    cursor: pointer;
}

button:hover {
    background-color: #1f6391;
}

ul {
    list-style-type: none;
    padding-left: 0;
}

li {
    background: #fff;
    margin-bottom: 10px;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0 2px 3px rgba(0,0,0,0.1);
}

```

top5.css

```

/* General Styles */
body {
    font-family: Arial, sans-serif;
    background-color: #f2f4f8;
    color: #333;
    margin: 0;
    padding: 0 20px;
}

header {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    padding: 20px 100px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    z-index: 99;
}

```

```

}
.navigation a{
  position: relative;
  font-size: 1.1em;
  color: rgb(39, 27, 27);
  font-weight: 500;
  margin-left: 40px;
  /* remove underline */
  text-decoration: none;
}

.navigation a::after{
  content: ";
  position: absolute;
  left: 0;
  bottom: -6px;
  width: 65px;
  height: 3px;
  background: rgb(18, 22, 18);
  border-radius: 5px;
  transform-origin: right;
  transform: scaleX(0);
  transition: transform .5s;
}

.navigation a:hover::after{
  transform: scaleX(1);
  transform-origin: left;
}

h1{
  color: whitesmoke;
}

h1 {
  text-align: center;
  color: #2c3e50;
  margin-top: 40px;
  font-size: 32px;
}

/* Hotel Card Styles */
ul {
  list-style-type: none;
  padding: 0;
  max-width: 1000px;
  margin: 30px auto;
}

li {
  background-color: #fff;
  border-radius: 12px;
  padding: 20px;
}

```

```

margin-bottom: 20px;
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
transition: transform 0.2s ease, box-shadow 0.3s ease;
}

li:hover {
transform: translateY(-3px);
box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
}

h2 {
color: #3498db;
margin-bottom: 10px;
}

p {
margin: 8px 0;
line-height: 1.5;
}

a {
color: #1abc9c;
text-decoration: none;
font-weight: bold;
}

a:hover {
text-decoration: underline;
}

/* Back link */
a[href="{{ url_for('index') }}" ] {
display: block;
text-align: center;
margin-top: 40px;
color: #555;
font-size: 16px;
}

```

contactus.css

```

/* Base Reset */
* {
margin: 0;
padding: 0;
font-family: 'Poppins', sans-serif;
box-sizing: border-box;
}

body {
background: url('image/background.jpg') repeat center center;
background-size: cover;
color: #333;
}

```

```

/* Navigation Bar */
header {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    padding: 20px 100px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    z-index: 99;
}

.navigation a {
    position: relative;
    font-size: 1.1em;
    color: white;
    font-weight: 500;
    margin-left: 40px;
    /* remove underline */
    text-decoration: none;
}

.navigation a::after {
    content: "";
    position: absolute;
    left: 0;
    bottom: -6px;
    width: 65px;
    height: 3px;
    background: green;
    border-radius: 5px;
    transform-origin: right;
    transform: scaleX(0);
    transition: transform .5s;
}

.navigation a:hover::after {
    transform: scaleX(1);
    transform-origin: left;
}

/* Contact Section */
#contact {
    padding: 100px 20px 50px;
    display: flex;
    justify-content: center;
    align-items: flex-start;
    flex-direction: column;
}

.blog-heading span {
    font-size: 20px;
}

```

```

        color: #555;
        margin-left: 100px;
    }

/* Contact Card */
.contact-box {
    background-color: #ffffffd8;
    border-radius: 15px;
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
    padding: 40px;
    max-width: 800px;
    margin: 40px auto;
}

.contact-box h1 {
    color: #007bff;
    font-size: 32px;
    margin-bottom: 20px;
}

.container__left h2 {
    font-size: 22px;
    color: #333;
    margin-top: 25px;
    margin-bottom: 10px;
}

.container__left p {
    margin-bottom: 20px;
    line-height: 1.6;
    color: #555;
}

.container__left ul {
    list-style-type: none;
    margin-bottom: 25px;
    padding-left: 0;
}

.container__left ul li {
    margin-bottom: 10px;
    font-size: 16px;
}

.container__left ul li a {
    color: #007bff;
    text-decoration: underline;
}

.form__row {
    margin-bottom: 20px;
}

.form__row label {

```

```

display: block;
font-weight: 600;
margin-bottom: 5px;
color: #333;
}

.form__row input,
.form__row textarea {
width: 100%;
padding: 12px 15px;
border: 1px solid #ccc;
border-radius: 8px;
transition: border 0.3s ease;
}

.form__row input:focus,
.form__row textarea:focus {
border: 1.5px solid #007bff;
outline: none;
box-shadow: 0 0 5px rgba(0, 123, 255, 0.3);
}

.form__row textarea {
resize: vertical;
}

button[type="submit"] {
background-color: #007bff;
color: #fff;
border: none;
border-radius: 8px;
padding: 12px 24px;
font-size: 16px;
font-weight: 600;
cursor: pointer;
transition: background-color 0.3s ease, transform 0.2s ease;
}

button[type="submit"]:hover {
background-color: #0056b3;
transform: scale(1.05);
}

```

Appendix B

User Acceptance Testing							
Student 1		Current Status: Fresh graduate (graduated within the last 2 years)					
Highest level of education: Bachelor's		Student ID: 2005621					
<p>** Rating 1 as lowest (worst), rating 5 as highest (excellent)</p> <p>**The higher scale of rating indicates a higher level of satisfaction</p>							
No	Acceptance Criteria	Ratings					Remarks
System Content							
1.	How easy was it to read and understand the hotel content (text, images, descriptions)?	1	2	3	4	5	-
2.	Was the hotel information (e.g., price, rating) positioned clearly and logically?	1	2	3	4	5	-
3.	Was the layout and design consistent across different pages?	1	2	3	4	5	-
System Functionality							
4.	Were the hotel recommendations relevant to your preferences?	1	2	3	4	5	-
5.	Was it easy to navigate and use the system features (search, filter, etc.)?	1	2	3	4	5	-
6.	Did the system accurately reflect your input in the hotel suggestions?	1	2	3	4	5	-
7.	Could you learn to use the system without needing help?	1	2	3	4	5	-
System Performance							
8.	Was the system's response time fast enough when performing actions?	1	2	3	4	5	-
9.	Were the hotel pages and results loaded quickly?	1	2	3	4	5	-
10.	Did the system help you efficiently find a suitable hotel?	1	2	3	4	5	-



User Acceptance Testing							
Student 2		Current Status: University Student					
Highest level of education: Bachelor's		Student ID: 2003903					
<p align="center">** Rating 1 as lowest (worst), rating 5 as highest (excellent)</p> <p align="center">**The higher scale of rating indicates a higher level of satisfaction</p>							
No	Acceptance Criteria	Ratings					Remarks
System Content							
1.	How easy was it to read and understand the hotel content (text, images, descriptions)?	1	2	3	4	5	-
2.	Was the hotel information (e.g., price, rating) positioned clearly and logically?	1	2	3	4	5	-
3.	Was the layout and design consistent across different pages?	1	2	3	4	5	-
System Functionality							
4.	Were the hotel recommendations relevant to your preferences?	1	2	3	4	5	-
5.	Was it easy to navigate and use the system features (search, filter, etc.)?	1	2	3	4	5	-
6.	Did the system accurately reflect your input in the hotel suggestions?	1	2	3	4	5	-
7.	Could you learn to use the system without needing help?	1	2	3	4	5	-
System Performance							
8.	Was the system's response time fast enough when performing actions?	1	2	3	4	5	-
9.	Were the hotel pages and results loaded quickly?	1	2	3	4	5	-
10.	Did the system help you efficiently find a suitable hotel?	1	2	3	4	5	-

User Acceptance Testing							
Student 3				Current Status: University Student			
Highest level of education: Bachelor's				Student ID: 2004956			
<p align="center">** Rating 1 as lowest (worst), rating 5 as highest (excellent)</p> <p align="center">**The higher scale of rating indicates a higher level of satisfaction</p>							
No	Acceptance Criteria	Ratings					Remarks
System Content							
1.	How easy was it to read and understand the hotel content (text, images, descriptions)?	1	2	3	4	5	-
2.	Was the hotel information (e.g., price, rating) positioned clearly and logically?	1	2	3	4	5	-
3.	Was the layout and design consistent across different pages?	1	2	3	4	5	-
System Functionality							
4.	Were the hotel recommendations relevant to your preferences?	1	2	3	4	5	-
5.	Was it easy to navigate and use the system features (search, filter, etc.)?	1	2	3	4	5	-
6.	Did the system accurately reflect your input in the hotel suggestions?	1	2	3	4	5	-
7.	Could you learn to use the system without needing help?	1	2	3	4	5	-
System Performance							
8.	Was the system's response time fast enough when performing actions?	1	2	3	4	5	-
9.	Were the hotel pages and results loaded quickly?	1	2	3	4	5	-
10.	Did the system help you efficiently find a suitable hotel?	1	2	3	4	5	-

User Acceptance Testing							
Student 4		Current Status: University Student					
Highest level of education: Bachelor's		Student ID:2005334					
<p align="center">** Rating 1 as lowest (worst), rating 5 as highest (excellent)</p> <p align="center">**The higher scale of rating indicates a higher level of satisfaction</p>							
No	Acceptance Criteria	Ratings					Remarks
System Content							
1.	How easy was it to read and understand the hotel content (text, images, descriptions)?	1	2	3	4	5	-
2.	Was the hotel information (e.g., price, rating) positioned clearly and logically?	1	2	3	4	5	-
3.	Was the layout and design consistent across different pages?	1	2	3	4	5	-
System Functionality							
4.	Were the hotel recommendations relevant to your preferences?	1	2	3	4	5	-
5.	Was it easy to navigate and use the system features (search, filter, etc.)?	1	2	3	4	5	-
6.	Did the system accurately reflect your input in the hotel suggestions?	1	2	3	4	5	-
7.	Could you learn to use the system without needing help?	1	2	3	4	5	-
System Performance							
8.	Was the system's response time fast enough when performing actions?	1	2	3	4	5	-
9.	Were the hotel pages and results loaded quickly?	1	2	3	4	5	-
10.	Did the system help you efficiently find a suitable hotel?	1	2	3	4	5	-

User Acceptance Testing							
Student 5				Current Status: University Student			
Highest level of education: Bachelor's				Student ID: 2005633			
<p align="center">** Rating 1 as lowest (worst), rating 5 as highest (excellent)</p> <p align="center">**The higher scale of rating indicates a higher level of satisfaction</p>							
No	Acceptance Criteria	Ratings					Remarks
System Content							
1.	How easy was it to read and understand the hotel content (text, images, descriptions)?	1	2	3	4	5	-
2.	Was the hotel information (e.g., price, rating) positioned clearly and logically?	1	2	3	4	5	-
3.	Was the layout and design consistent across different pages?	1	2	3	4	5	-
System Functionality							
4.	Were the hotel recommendations relevant to your preferences?	1	2	3	4	5	-
5.	Was it easy to navigate and use the system features (search, filter, etc.)?	1	2	3	4	5	-
6.	Did the system accurately reflect your input in the hotel suggestions?	1	2	3	4	5	-
7.	Could you learn to use the system without needing help?	1	2	3	4	5	-
System Performance							
8.	Was the system's response time fast enough when performing actions?	1	2	3	4	5	-
9.	Were the hotel pages and results loaded quickly?	1	2	3	4	5	-
10.	Did the system help you efficiently find a suitable hotel?	1	2	3	4	5	-

Appendix C

	INTELLECTUAL PROPERTY CORPORATION OF MALAYSIA <i>An agency under the Ministry of Domestic Trade and Cost of Living</i> COPYRIGHT ACT 1987 NOTIFICATION OF COPYRIGHT IN A WORK <i>[subregulations 5(2) and 5(3)]</i>	
---	--	---

CR - 1

Application No:

LY 2025 P03125

Applicant :

* **Title of work** (Original language) : HOTEL RECOMMENDATION SYSTEM USING MACHINE LEARNING

Translation (If the title of work is neither in Bahasa nor English) : _____

Transliteration (If the title of work is neither in Bahasa nor English) : _____

Name of the Language : ENGLISH
(Language use in the work)

* **Section A : Category of Works** (Please tick **ONE** only)

<input checked="" type="checkbox"/> Literary	<input type="checkbox"/> Musical	<input type="checkbox"/> Artistic	<input type="checkbox"/> Film	<input type="checkbox"/> Sound Recording	<input type="checkbox"/> Broadcast <small>('broadcasting service' only)</small>	<input type="checkbox"/> Derivative
--	----------------------------------	-----------------------------------	-------------------------------	--	--	-------------------------------------

Date of Creation / Fixation : 13 / 05 / 2025

Section B : Publication

The Work is : ☐ Published ☒ Unpublished

If published : DD / MM / YYYY (Date of first publication) _____ (Country)

* **Section C : Author** (To add additional Authors, please attach separate sheet)

Name (as per NRIC/Passport) : NORLIANA BINTI MUSLIM

NRIC / Passport No. : 840712-14-5622

Nationality : MALAYSIA DD / MM / YYYY (Date of Death (if applicable))

* **Section D : Owner** (To add additional Owners, please attach separate sheet)

Name (as per NRIC/Passport) or **Company Name** : FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
 UNIVERSITI TUNKU ABDUL RAHMAN

NRIC / Passport / Company No. : _____

Nationality : _____

Address : JALAN UNIVERSITI

Postcode : 31900	City : KAMPAR
State : PERAK	Country : MALAYSIA

Telephone No. : 05-468 8888 **E-mail** : _____

* Required to be filled in

PAGE 1 OF 2

Section E : Licensee (if applicable)

Name (as per NRIC/Passport) or : _____
Company Name : _____
NRIC / Passport / Company No. : _____
Nationality : _____
Address : _____
Postcode : _____ City : _____
State : _____ Country : _____
Telephone No. : _____ E-mail: _____
Date of Agreement : DD MM YYYY Please provide copy of agreement(s) (mandatory)
Duration of Agreement : DD MM YYYY until DD MM YYYY

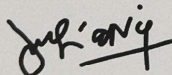
*** Section F : Contact Person**

Name (as per NRIC/Passport) : NORLIANA BINTI MUSLIM
NRIC / Passport No. : 840712-14-5622
Address : NO. 13 JALAN INDAH 8 TAMAN SERENDAH INDAH
Postcode : 48200 City : SERENDAH
State : SELANGOR Country : MALAYSIA
Telephone No. : 0133253416 E-mail: lianamuslim84@gmail.com /norlianam@utar.edu.my

*** Section G : Declaration**

I/We hereby declare that the applicant is the owner of the copyright in the work.

Signature,



Name : NORLIANA BINTI MUSLIM
Date : 16-May-2025

*** Section H : Language of Certificate**

☐ Malay OR ☒ English
(Please tick ONE only)

*** Section I : Mode of Delivery for Certificate**

☐ Self-Collection OR ☒ By Post
(Please tick ONE only)

Section J : Official Use

Payment Received : ☐ YES ☐ NO
Date : DD MM YYYY

DISCLAIMER : Applicant are reminded to give full and accurate information while filling in the particulars in this form. Any inaccuracy in the information given is on the applicant's own volition. MyIPO cannot be held responsible or held liable for the wrong information recorded.

All correspondence should be addressed to :

Copyright Division

Intellectual Property Corporation of Malaysia (MyIPO)
Level 13, Menara MyIPO, PJ Sentral
Lot 12, Persiaran Barat, Seksyen 52
46200 Petaling Jaya, Selangor, MALAYSIA.

Telephone : +603 - 7496 8900
Fax : +603 - 7496 8999
Website : <http://www.myipo.gov.my>
E-Mail : infocopyright@myipo.gov.my

* Required to be filled in

PAGE 2 OF 2

**PERBADANAN HARTA INTELEK MALAYSIA**

Aras LG, G, 2-5, 11-13 & 15-23
Menara MyIPO, PJ Sentral,
Lot 12, Persiaran Barat, Seksyen 52.,
46200, Petaling Jaya, Malaysia.
Tel: 03-7496 8900 Faks: 03-7496 8999

**RESIT RASMI**

Diterima Daripada	Butiran Resit Rasmi
NORLIANA BINTI MUSLIM	Nombor Resit : RSTT/PNG-001215-2025 Tarikh : 16/05/2025 Jumlah : 200.00

Rujukan	Butiran Bayaran
Pusat Bayaran : PULAU PINANG-	Cara Bayaran No Doc Tarikh Doc Amaun (RM)
No. Invois : -	KAD DEBIT 547144 16/05/2025 200.00
Catatan :	

Keterangan	No pendaftaran	Kuantiti	Kos Per Unit	GST	Jumlah
CR-1A	LY2025P03125	1.00	200.00	0.00	200.00

Cetakan Berkomputer

Tidak Perlu Tandatangan

C450

*Resit ini akan dianggap batal sekiranya cek tidak dapat ditunaikan.

SALINAN PELANGGAN

