WEB-BASED RECOMMENDATION SYSTEM FOR HEALTHCARE SERVICES

By

Chew Jing Jia

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2026

COPYRIGHT STATEMENT

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal 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 express my sincere thanks and appreciation to my supervisor, Dr Muhammad Husaini bin Nadri for guiding me through the report writing and co-supervisor, Dr Norliana binti Muslim who has given me this bright opportunity to engage in a web-based healthcare recommendation system project. It is my first step to establish a career in the data science field. A million thanks to you two.

And I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

As the healthcare industry undergoes rapid digital transformation, patients are increasingly overwhelmed, where navigating thousands of unstructured online reviews to find specialized medical providers results in decision fatigue. This project aims to develop an intelligent web-based recommendation system that utilizes machine learning to deliver personalized, localized healthcare suggestions tailored to individual patient symptoms and geographical context. The research methodology involved the collection and preprocessing of a dataset comprising 10,140 reviews from 15 major public and private hospitals in Penang, extracted using the Phantom GMB Audit Tool. The framework implements a hybrid machine learning architecture that integrates SVD for collaborative filtering with SVM for content-based sentiment classification, further enhanced by TF-IDF vectorization and VADER sentiment analysis to interpret the semantic nuances of patient feedback. The system was developed using the Flask web framework and HTML to provide a seamless interface between the complex backend algorithms and the end-user. Experimental results indicated that the hybrid model significantly outperforms standalone collaborative filtering models, achieving an accuracy of 80.10%, RMSE of 0.6313 and a MAE of 0.3137. The project successfully designed a user-friendly interface with features, including a multilingual toggle (English and Chinese) to support Malaysia's multiracial society, real-time location-aware prioritization using the Haversine distance formula, and a specialized module that provides transparency into the ranking logic through keyword-matching insights. This study is highly significant as it transforms unstructured patient feedback into a structured, accessible decision-support tool that empowers the community to make more informed medical choices.

Area of Study: Data Science and Machine Learning

Keywords: Flask Framework, Healthcare Services, Hybrid Recommendation System, Natural Language Processing, Sentiment Analysis and User-Generated Content

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

α	Weight parameter that balances the contribution of SVM and SVD
μ	Global average rating

LIST OF ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
CSS	Cascading Style Sheets
HRS	Health Recommender System
HTML	Hypertext Markup Language
KNN	K-Nearest Neighbors
RF	Random Forest
RMSE	Root Mean Squared Error
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency
VADER	Valence Aware Dictionary and Sentiment Reasoner
XAI	Explainable AI

Chapter 1
Introduction

The healthcare industry has increasingly embraced digital transformation [1], leading to a surge in the development of more healthcare services, ranging from consultations to treatments. As a result, patients now have access to a wider variety of clinics and hospitals than ever before.

While this expansion has improved accessibility and options, it has also created challenges for patients in choosing a suitable healthcare provider [2]. With so many available choices that offers different specialties and levels of service, it can be difficult for individuals to make decisions about where to seek treatment.

To help make decisions, many patients read online reviews and ratings from others who have used these services. These reviews can be helpful, but there are often too many to read and understand [3]. This project aims to solve that problem by creating a recommendation system that uses machine learning to analyse healthcare reviews in Penang and suggest suitable providers to users.

1.1 Problem Statement and Motivation

Nowadays, patients rely heavily on online reviews when selecting healthcare providers [4]. However, most platforms present these reviews in chronological order without offering appropriate filtering, summarisation, or suggestions [5]. This makes it difficult for users to identify services that meet their specific needs, such as affordability, staff friendliness, or waiting time.

A Different Focus of Reviews

One major challenge in the current healthcare review system is the unstructured nature of user-generated content [6]. Without a standardised format, platforms cannot extract specific details such as treatment quality, staff friendliness, or appointment wait times, which are critical information for patient decision-making. This lack of focus in reviews makes it challenging for users and machines to interpret and categorise the content [7].

B Lack of Personalisation

Most existing healthcare platforms present reviews and ratings in chronological order or based on overall averages, without considering unique preferences or conditions of each user [8]. For instance, a review highlighting excellent paediatric care may be irrelevant to a user searching for orthopaedic services. The absence of a recommendation engine that understands user preferences, such as medical condition, budget, location, or gender-specific care, results in generalised suggestions that fail to meet individual needs [9].

C Information Overload

With the increasing volume of healthcare reviews online, users are often overwhelmed by the sheer number of options and opinions [10]. Sifting through hundreds of reviews to find relevant ones can be time-consuming and mentally exhausting, leading to decision fatigue. Users may select healthcare providers based on incomplete or random information rather than carefully considered choices. This overload affects user satisfaction and results in poor healthcare experiences.

Thus, there is a need for methods that can extract key information from review texts and organise it into a consistent format. This would make healthcare reviews easier to search and compare based on users' requirements. Also, the recommendation system should be able to filter reviews based on users' specific requirements as a system that surfaces the most relevant reviews based on user-specific criteria could help reduce information overload [11].

The motivation for this project comes from the growing need for patient-focused healthcare [1] and the way people now rely on online platforms to choose medical services as online reviews give useful and easy-to-access information about patient experiences. This project uses machine learning and natural language tools to read and understand these reviews. By using machine learning to analyse reviews, this system hopes to give patients recommendations, helping them to select suitable healthcare service provider.

1.2 Objectives

The project aims to design a web-based recommendation system that utilizes machine learning to analyse online healthcare reviews and recommend suitable healthcare providers to patients based on their specific needs.

To achieve this, the following objectives must be met:

1. **To gather a set of healthcare reviews from healthcare provider located in Penang.**
   Only reviews that are written in English are included to ensure the data is consistent. In addition, reviews will only be collected if they were posted before or during July 2025, so that the dataset used for analysis is fixed and up to date at the time of this project.
2. **To develop a hybrid machine learning model based on content-based and collaborative filtering models.**
   The models will be trained to extract key features such as treatment quality, staff attitude, and geographical relevance. These extracted insights will then be used to build a recommendation engine that suggests suitable healthcare providers to users based on their preferences.
3. **To propose a user-friendly healthcare website that provides recommendations to users.**
   The website will allow patients to search and view healthcare provider recommendations based on their personal needs. The website will ensure the system is accessible for real-world use, encouraging better healthcare decisions among users in the healthcare community.

With the above, objectives will be achieved by creating a website that improves patient access to healthcare information and supports informed decision-making for the healthcare communities.

1.3 Project Scope and Direction

This project was designed to gather and analyse healthcare review data specifically from Penang, focusing on creating a recommendation system that caters to the local community's healthcare needs. To maintain consistency, the dataset will only include reviews written in English by filtering out any non-English reviews, ensuring that the text can be processed for recommendation purposes.

For the development of the system, Python will be used to build and train the machine learning models due to its flexibility and wide range of libraries for data analysis and natural language processing. SQLite will serve as the database for storing the collected review data, offering scalability for handling of unstructured data, while also supporting semantic search capabilities to enhance the accuracy of recommendations. The user interface will be developed using

HTML, CSS and JavaScript, providing an accessible platform for users to interact with the recommendation system. This combination of technologies will support the creation of a user-friendly healthcare recommendation platform based on real-world review data from Penang.

1.4 Contributions

This project aimed to develop a recommendation system for healthcare services by collecting and analysing user-generated reviews. The key contributions of this project include:

1. **Support for a Multilingual Website**: Since Malaysia is a multiracial country with citizens who speak various languages, the system will support a multilingual interface with English and Chinese languages. This ensures that users from different language backgrounds can understand and benefit from the platform equally, making the healthcare recommendation system more accessible to a wider population.

2. **User-specific Healthcare Recommendation System**: The recommendation system will allow users to find healthcare providers that match their disease symptoms and location. This personalisation will help users make more faster decisions, leading to better healthcare experiences and outcomes.

3. **Development of a Web-Based Interface**: The project will include a user-friendly website that presents personalised healthcare recommendations in an organised manner. A better interface will ensure that users can easily navigate, search, and interpret the recommended healthcare options, improving the overall user experience in utilising the healthcare recommendation system.

1.5 Report Organization

The details of this project are organized into seven chapters. Chapter 2 reviews previous works on recommendation systems and machine learning models, including a comparative analysis of existing healthcare platforms to identify industry gaps. Chapter 3 discusses system methodology, covering the CRISP-DM framework, technical requirements, system architecture, and the mathematical equations governing the hybrid engine. Chapter 4 presents the system design, detailing data preprocessing, sentiment analysis, and the logical architecture

of the machine learning pipelines. Chapter 5 focuses on system implementation, documenting the hardware and software configuration alongside a visual walkthrough of the platform's operations. Chapter 6 provides the system evaluation, analyzing functional test results, performance metrics, and user feedback to verify the system's reliability. Finally, Chapter 7 concludes the project by summarizing the findings, reflecting on the objectives achieved, and suggesting future recommendations for platform improvement.

Chapter 2
Literature Review

This chapter reviews previous works and related research in recommendation systems and machine learning models, specifically in the healthcare domain. The project begins by reviewing existing recommendation systems, highlighting their approaches, challenges, and effectiveness. Then, it is followed by a review of relevant machine learning algorithms used in sentiment analysis of healthcare reviews. The strengths and weaknesses of commonly used models are discussed to justify the selection of algorithms for this study. Finally, a comparison with existing platforms is presented to highlight the features of the proposed system.

2.1 Previous Works on Recommendation Systems

Table 2.1 Previous Works of Recommendation Systems

Reference	Application	Approach	Problem	Result
[12]	Smart Healthcare Recommendation System for Multidisciplinary Diabetes Patients	Deep ensemble learning with data fusion	Existing systems struggle to handle large, multi-feature datasets for diabetes prediction.	Achieved 99.6% accuracy, outperforming existing deep learning methods.
[13]	Drug Recommendation System based on Sentiment Analysis of Drug Reviews	Machine learning classifiers with TF-IDF vectorisation	Users self-medicate without proper consultation due to the inaccessibility of clinical resources.	The LinearSVC classifier achieved 93% accuracy in predicting sentiments for drug recommendations.
[14]	A Machine Learning-Based Hybrid Recommender Framework for Smart Medical Systems	Hybrid recommender system incorporating K-means clustering	Challenges in correctly recommending departments and doctors due to diverse patient data.	Enhanced accuracy and efficiency in department triage and doctor recommendation processes.
[15]	Health Recommender Systems: Systematic Review	Comparative analysis of various	Users face difficulties in making health decisions due to	Hybrid approaches combining multiple recommender techniques showed

		recommender techniques	abundant options and lack of knowledge.	improved recommendation accuracy.
[16]	Health Recommender Systems Development, Usage, and Evaluation	Scoping review focusing on recommendation technologies and system evaluation	Limited health domains, recommended items, and sample sizes in existing HRS evaluations.	Identified gaps in current research and emphasized the need for dynamic user modeling and large-scale evaluations.
[17]	Recommender Systems in the Healthcare Domain	Systematic overview of existing research on healthcare recommender systems	Overload of medical information hinders correct decision-making for users and professionals.	Provided insights into recommendation scenarios and approaches, highlighting challenges and future research directions.
[18]	A Privacy-Aware Deep Learning Framework for Health Recommendation System	Stacked discriminative de-noising convolution auto-encoder–decoder with two-way recommendation	Traditional health recommendations lack security constraints, causing hesitation in sharing sensitive information.	Proposed a privacy-preserving health recommendation system ensuring secure health data sharing.
[19]	A Semantics-Based Clustering Approach for Online Laboratories	Semantic approach using K-means and Hierarchical Agglomerative Clustering (HAC)	Difficulty in clustering online laboratory documents for educational purposes.	Improved clustering accuracy and interpretability of online laboratory documents.
[20]	Personalized Recommendation for Healthcare Decisions: A Multi-	Multi-Armed Bandit (MAB) algorithm for adaptive learning	Users are overwhelmed by numerous healthcare intervention	MAB-driven framework adaptively learned user preferences, improving

	Armed Bandit Approach		choices, leading to decision fatigue.	recommendation diversity and user engagement.
[21]	Restaurant recommender system	Hybrid filtering system	Many RS focus on fixed factors like food quality, prices, and service, often ignoring the changing nature of customer experiences and preferences.	The results show that the proposed system can provide recommendations with an accuracy of 92.8%.

In reviewing recent advancements in healthcare recommendation systems, several approaches have been explored to enhance user experience and decision-making. As discussed in works like [12], collaborative filtering methods show promise in environments with abundant user interaction data but are less useful in healthcare contexts where user feedback is typically sparse and sensitive. Content-based filtering, as utilized in study [16], provides a more detailed experience by leveraging hospital and treatment attributes, but often lacks the deeper contextual understanding needed when dealing with unstructured user reviews. Semantic-based approaches, exemplified by [21], introduce the ability to interpret the nuanced meaning of review texts through natural language processing techniques, addressing the issue of overwhelming and inconsistent review content. However, purely semantic methods can sometimes ignore important structured data like medical specialty, pricing, or location.

Based on these comparisons, the most suitable method for this project is a hybrid approach combining content-based filtering and semantic-based machine learning techniques. Content-based filtering uses structured data (such as medical specialties and budget) to provide a starting point for recommendations [22]. It also uses semantic analysis to understand the details in user reviews, such as treatment quality and patient experiences. Thus, this project will develop the healthcare recommendation system using a hybrid method that integrates structured data and unstructured review content to provide users with more relevant and context-aware recommendations.

2.2 Related Work on Machine Learning Models

Table 2.2 Previous Works of Machine Learning Models

Reference	Dataset	Approach	Performance metric	Findings
[23]	Facebook reviews from Malaysian hospitals	SVM	Accuracy: 87.43%, F1-Score: 91.89%	SVM achieved the highest accuracy and F1-score among all methods tested.
		Naive Bayes	Accuracy: 78.10%, F1-Score: 87.40%	Naïve Bayes showed high recall, but lower precision compared to SVM.
		LR	Accuracy: 84.29%, F1-Score: 90.57%	Logistic Regression offered balanced precision and recall values.
[24]	PeduliLindungi App Reviews	SVM	Accuracy: 81%, F1-Score: 80%	SVM outperformed Naïve Bayes in all metrics.
		Naive Bayes	Accuracy: 77%, F1-Score: 73%	Naive Bayes was slightly less effective than SVM in classification tasks.
[25]	Mobile JKN App Reviews	SVM	Accuracy: 90%	SVM achieved a similarly high accuracy as Naïve Bayes.
		Naive Bayes		Naïve Bayes performed with a

				similar accuracy to SVM
[26]	mHealth App Reviews	SVM	Accuracy: 90.5%, F1-Score: 85.7%	SVM showed strong classification performance in mHealth review data.
		RF	Accuracy: 90.5%, F1-Score: 90%	RF performed slightly better on F1-Score but equally on accuracy with SVM.
		KNN	Accuracy: 93.97%	KNN was close to SVM but slightly lower in accuracy.
[27]	Amazon dataset from Kaggle	SVD	RMSE: 0.72, MAE: 0.556	Balanced performance with moderate training time
		KNN Baseline	RMSE: 0.847, MAE: 0.591	Although it has the fastest training time, lowest accuracy
		CoClustering	RMSE: 0.847, MAE: 0.581	Long training time and lower accuracy
[28]	COVID-19 vaccine tweets in Indonesia	SVM	Accuracy: 85%, Precision: 88%, F1-Score: 89%	SVM showed high effectiveness in sentiment classification for COVID-19 topics.
[29]	Movielens 100 k Dataset	SVD	Accuracy: 83.96%,	SVD achieved the highest precision

			Precision: 91.43%, Recall: 38.76%,	and was computationally faster.
		KNN	Accuracy: 78.29%, Precision: 87.68%, Recall: 31.47%	KNN performed worse than SVD in all metrics but had a slightly higher recall than random prediction.

As the study shows from the table, different models are specialized for different tasks. For the task of sentiment analysis on unstructured text, SVM consistently demonstrates superior performance. Across multiple studies on healthcare reviews and tweets, SVM achieves the highest accuracy and F1-scores, outperforming alternatives like Logistic Regression, Naive Bayes, and Random Forest. Its proven ability to handle the high-dimensional feature spaces created by text vectorization makes it the ideal choice for understanding the content of patient reviews in this project. Simultaneously, for the task of predicting user ratings in a recommendation system, SVD stands out as a powerful and effective collaborative filtering technique. As shown in studies on large datasets like Movielens and Amazon, SVD delivers strong performance, often with better accuracy (lower RMSE) and higher precision than other collaborative methods like KNN.

2.2.1 Strengths and Weaknesses

For healthcare review sentiment analysis, selecting the right machine learning model is essential for classification. SVD, Naïve Bayes, and SVM are commonly used by other authors choices due to their effectiveness in handling complex data. This study examines the strengths and weaknesses of these models to determine their suitability for developing a recommendation system for healthcare reviews.

SVD is a powerful matrix factorization technique widely used in recommendation systems and dimensionality reduction tasks [30]. It is effective at uncovering latent relationships between users and items, making it highly suitable for collaborative filtering-based recommendation models [31]. Compared to algorithms like RF, SVD excels in handling sparse datasets and

improving recommendation accuracy by capturing hidden patterns in user–item interactions. However, SVD can be computationally intensive for very large datasets and may require careful tuning of factors such as the number of latent features to prevent overfitting [32]. In contrast, RF offers robustness to noisy data and higher interpretability, but SVD often provides better predictive performance and personalization in recommendation contexts.

Naive Bayes is another popular classification algorithm, valued for its simplicity and speed. It operates on probabilistic principles, using Bayes' Theorem to calculate the likelihood that a given text belongs to a certain class. However, its performance is dependent on a major simplifying assumption: that all features (in this case, words) are completely independent of one another [33]. This assumption is a disadvantage when analysing human language, where context and word order are crucial for meaning. For instance, the model cannot easily distinguish between "good service" and "not good service", as it treats "not" and "good" as independent events, which often leads to a misinterpretation of sentiment. This limitation makes Naive Bayes far less reliable for capturing the nuanced expressions found in healthcare reviews compared to other models.

Finally, SVM is a powerful supervised learning algorithm widely used for classification tasks, functioning by identifying an optimal hyperplane that best separates data points into distinct classes [34]. For this project, SVM is utilized to classify the sentiment of review text. Its primary advantage is its high effectiveness in high-dimensional spaces, making it perfect for vectorized text data from TF-IDF. Compared to other classifiers, SVM's ability to use different kernels allows it to model complex, non-linear relationships between words and sentiment, often leading to superior accuracy [35]. Key considerations for SVM include its training time, which can be longer on very large datasets as a trade-off for its high accuracy, and the fact that its predictions offer less direct interpretability compared to simpler algorithms. Furthermore, achieving optimal performance often requires careful hyperparameter tuning.

Therefore, to build a hybrid healthcare recommender, this project will utilise these two models which are SVM for text classification capabilities and SVD for its proven strength in predicting user ratings.

2.3 Review of Existing Systems

To bridge the gap between theoretical models and practical applications, several healthcare recommendation systems will be reviewed.

2.3.1 NourishNook [36]

This paper presents NourishNook, an AI-based health advisory system that provides personalized wellness recommendations by integrating multiple machine learning algorithms such as SVM, RF, Gradient Boosting, KNN, and Naïve Bayes with datasets on symptoms, diseases, precautions, diets, and workouts. The system stands out for its adaptive learning, user-friendly interface, social interaction features, robust data privacy measures, and regularly updated database, making it more interactive and accurate than traditional health platforms.

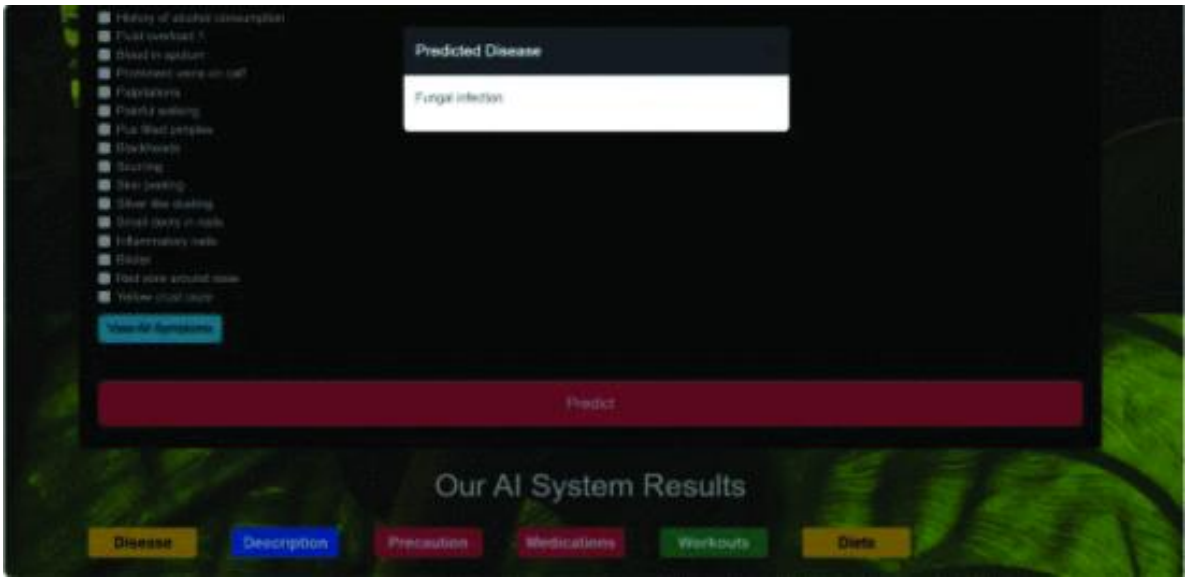


Figure 2.1 Interface of NourishNook

As shown in Figure 2.1, the system will analyse the selected symptoms provided by the user and predicts the most likely disease based on the input. It utilizes an AI-Powered algorithm trained on medical data to generate predictions. The user receives the predicted disease along with relevant recommendations. However, the system's methodology contains significant weaknesses. Its greatest flaw lies in its reliance on recommending home remedies and user-contributed solutions, a feature that introduces considerable risk without constant and rigorous medical moderation to ensure the safety and efficacy of the advice. Another internal weakness is its inherent dependency on the quality and diversity of its training data; any biases or

inaccuracies within the dataset would be directly translated into potentially flawed or inappropriate wellness recommendations, undermining the system's reliability. Finally, the project has a few limitations, it is intended only as an initial guidance tool and cannot replace a professional medical diagnosis. This fundamentally constrains its role and application in a real-world healthcare scenario. Also, the system's performance is measured in computational accuracy rather than through clinical validation, meaning its medical effectiveness has not been formally evaluated. This lack of clinical oversight limits its scope to being a wellness suggestion platform rather than a trusted health advisory tool.

2.3.2 Drug Recommender System [37]

This project presents a content-based medicine recommendation system designed to suggest drugs based on a user's medical condition. The web-based application allows user to select a condition from a dropdown menu as shown in Figure 2.2, and the system returns a list of the top five recommended medicines.

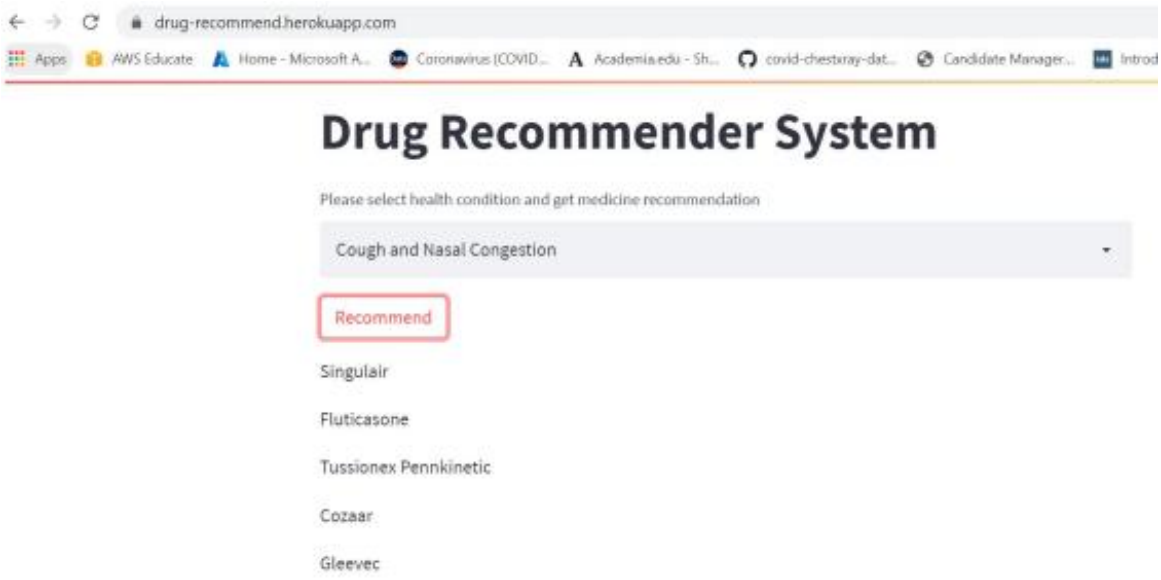


Figure 2.2 Interface of Drug Recommender System

A key advantage of this work is its practical implementation, demonstrating a complete development cycle from model building to deployment on a cloud platform (Heroku) with a user-friendly interface built with Streamlit. The methodology is straightforward and clearly documented, making the project highly reproducible. However, the methodology contains a significant weakness. The system is purely content-based and relies on a static dataset, meaning

it is incapable of personalization or learning from new user reviews or changing preferences over time. This leads to a design flaw where every user with the same condition receives the exact same recommendation, completely ignoring individual user history, ratings, or context, which is a core expectation of modern recommender systems. Furthermore, the project's scope is bound by several key limitations explicitly acknowledged by the author. The content-based approach is inherently unable to solve the "cold start" problem for new drugs added to the database that lack initial reviews or metadata. Additionally, the system's recommendations are based on a relatively simple Bag-of-Words (Count Vectorizer) model, which limits its ability to understand deeper in patient reviews compared to more advanced deep learning techniques.

2.3.3 Adaptive Online Pharmacy [38]

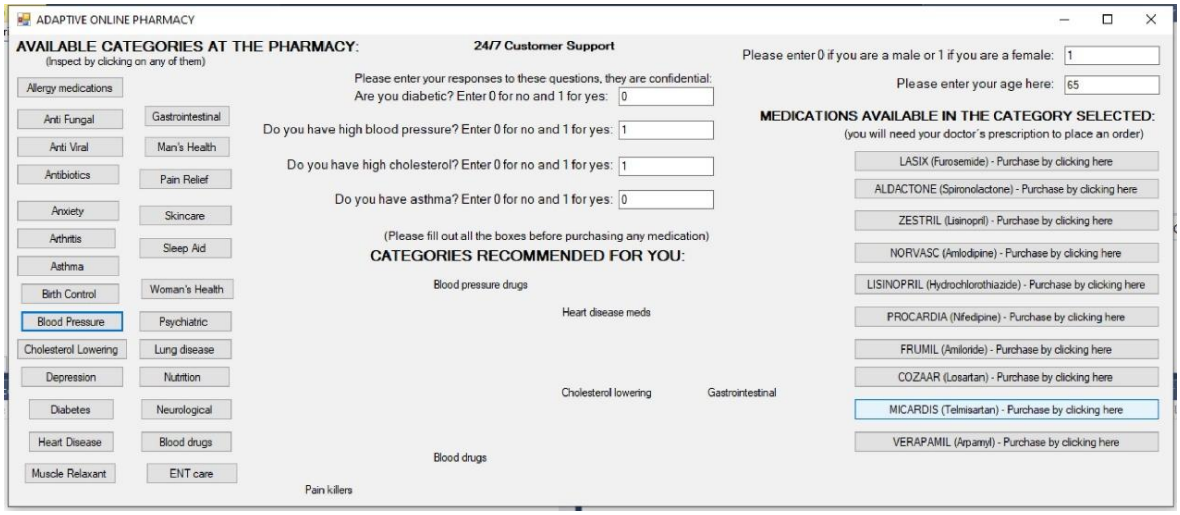


Figure 2.3 Interface of Adaptive Online Pharmacy

The project introduces a prototype of an adaptive online pharmacy designed to make online medication purchasing more convenient for patients. The system utilizes a Bayesian Network to power an adaptive user interface that predicts and recommends prescribed medications based on a patient's demographic data, clinical history, and purchasing behaviour within a session. The primary strength of this system lies in its novel application of a Bayesian Network to create a user interface for medication recommendations, achieving impressive prediction rates of up to 92.5%. However, the project has notable weaknesses in its current design. The most significant weakness is its reliance on synthetic data for both training and validation, which calls into question the model's performance and robustness in a real-world clinical environment. Another key design weakness is the system's use of a short-term user model, which only considers interactions within a single session and is therefore incapable of managing the long-

term medication histories essential for patients with chronic conditions. Furthermore, the scope is defined by a critical limitation. Its performance evaluation was restricted to a maximum of five purchased medications, meaning the system's effectiveness for patients with more complex polypharmacy needs remains unevaluated and outside the boundaries of the current findings. Therefore, a recommendation for future development is to transition the prototype into a clinically viable tool by using real patient data and evolving the user model to create persistent profiles for long-term chronic care management.

2.3.4 Hospital Recommendation System [39]

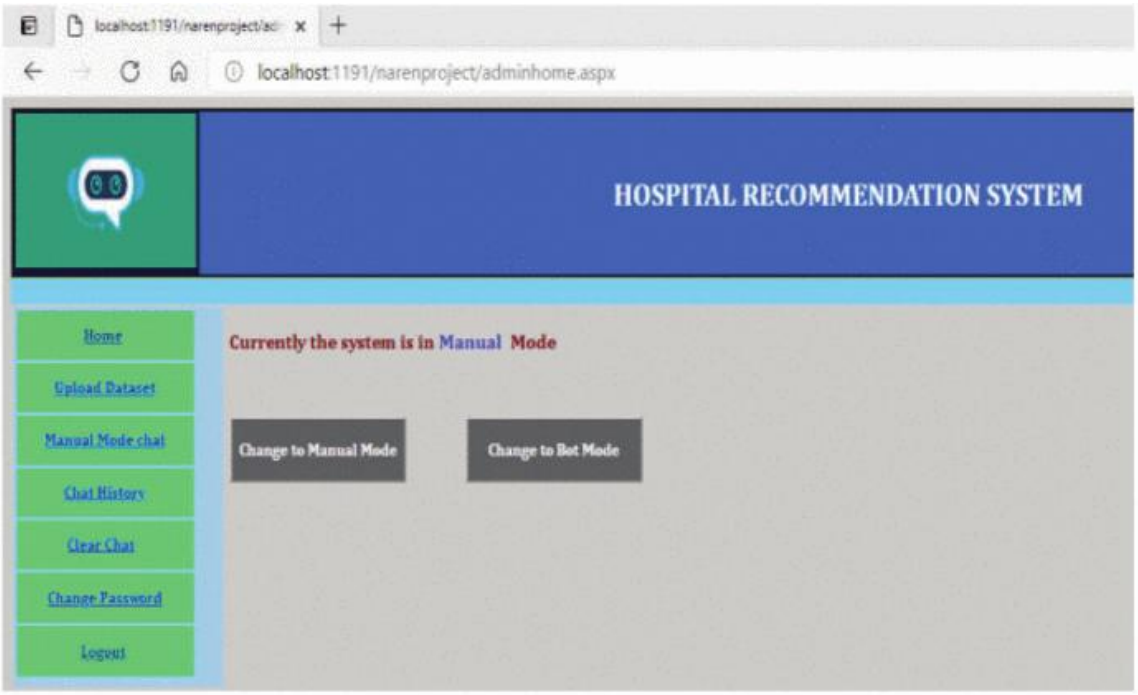


Figure 2.4 Interface of Hospital Recommendation System

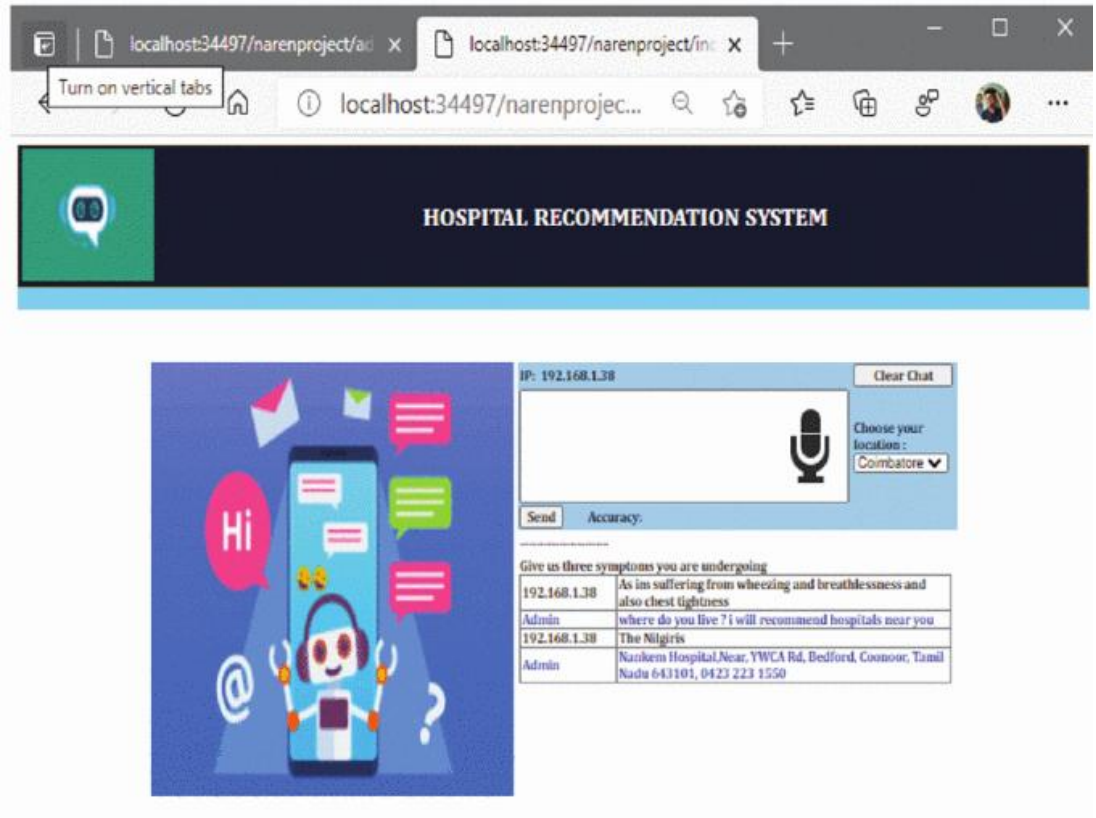


Figure 2.5 Chatbot system in Hospital Recommendation System

The Intelligent System for Hospital Recommendation by Rajkumar and his team is designed as a user-friendly medical guidance system. Its primary strength lies in its innovative hybrid design, which features both an automated “Offline” mode and a live agent “Online” mode. This ensures that users always receive a response; if the chatbot cannot handle a query using its trained data, the conversation is led to a human operator, which guarantees a fallback for unforeseen questions and enhances overall user support. However, the system's main weakness is the offline mode's total dependence on pre-trained datasets. Because the chatbot's knowledge is limited to what an administrator has manually entered, it is unable to address queries about new medical information or facilities not in its database. This leads to a significant limitation: unavailable of a dynamic database. The database cannot change over time that adapts to new data, limiting the flexibility of the system. Therefore, a recommendation for improvement is to evolve the system from a static model to a more dynamic one by integrating APIs that connect to real-time medical knowledge bases and hospital information systems. This enhancement would allow the chatbot to access current information, greatly expanding its utility and reducing its reliance on manual updates.

2.3.5 Health Recommender System [15]

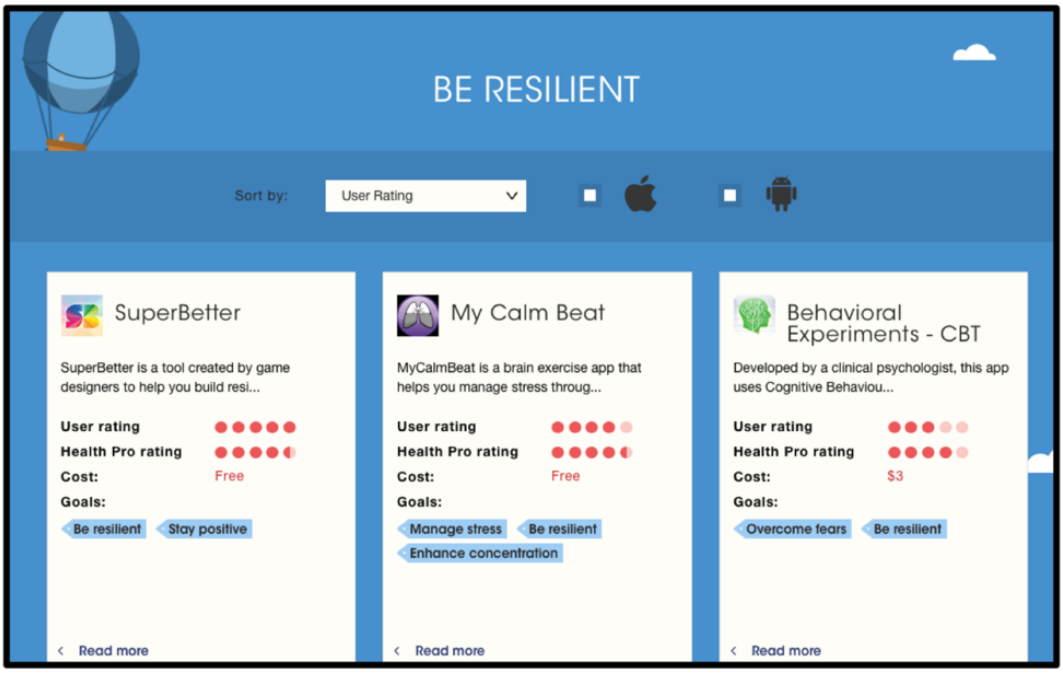


Figure 2.6 Interface of Health Recommender System's page

Croon et al.'s paper acts as an analytical tool that processes 73 HRS to produce a publicly coded dataset and a new framework for future research. A significant strength is its built-in quality assessment feature, which uses the Hawker method to assign a quantitative score to each reviewed paper, allowing researchers to objectively evaluate the rigor of previous studies. However, a key weakness is that its main output, the proposed reference frame, is purely qualitative; it provides a checklist of what to report but lacks a standardized scoring system to measure *how well* a study adheres to these guidelines. A notable limitation is that the system's data-gathering feature is restricted to English-language publications only, which excludes potentially valuable HRS research and innovations from non-English speaking regions. Therefore, a critical recommendation is to evolve their conceptual framework into a quantifiable evaluation instrument by creating a standardized "HRS Reporting Quality Score" that assigns measurable points to each guideline, turning their checklist into a powerful and repeatable assessment tool for the field.

2.4 Summarization of Features in Existing Systems

Table 2.3 Comparison with other works

Module	Feature/System	NourishNook	Drug Recommender System	Adaptive Online Pharmacy	Hospital Recommendation System	Health Recommender System	This project
Website	Login System				✓		✓
	UI Translation	✓	✓				✓
	Quick Symptom Search	✓		✓	✓		✓
	AI Health Guide						✓
	Search History						✓
	Hospital details	✓			✓	✓	✓
	Hotline number						✓
	Insurance panel directory						✓
	Medicine Purchase			✓			
	Location Tracking						✓
	Bookmark						✓
ML	Sentiment Analysis	✓					✓
	Empirical Analysis			✓			
	Hybrid Model	✓					✓
	Database				✓	✓	✓
	XAI						✓

This project provides a platform that streamlines the process of finding specialized medical provider through a web interface. At its core, the system features a secure Login and

Authentication System that allows for a personalized experience while providing a “Continue as Guest” option for immediate access. To ensure inclusivity within Malaysia’s multilingual society, the platform integrates UI Translation capabilities, supporting both English and Chinese. The primary interaction is driven by a Symptom Search feature, which allows users to either manually describe their condition or utilize a Quick Select dropdown for common issues.

To assist in faster decision-making, the system serves as a comprehensive information hub, displaying hospital details including facility categories (Public/Private), the hotline numbers with a click-to-copy feature, and a visual directory of Accepted Insurance Panels. Advanced Location Tracking via the Haversine formula is implemented to allow users to prioritize nearby facilities based on their real-time coordinates. Furthermore, the system includes a personalized AI Health Guide (the “Health Assistant” sidebar), which retrieves a user’s Search History to offer tailored home care advice and department insights. For long-term utility, a Bookmark Feature enables users to save and manage their preferred healthcare providers in a dedicated directory. Finally, the system promotes transparency through an XAI Module, which provides an automated “Ranking Logic” breakdown and keyword-matching insights, ensuring users understand exactly why a specific healthcare provider was recommended to them.

Chapter 3
System Methodology

The project adopts the CRISP-DM methodology. The project is categorized into six phases, which includes business understanding phase, data understanding phase, data preparation phase, modelling phase, evaluation phase and finally the deployment phase.

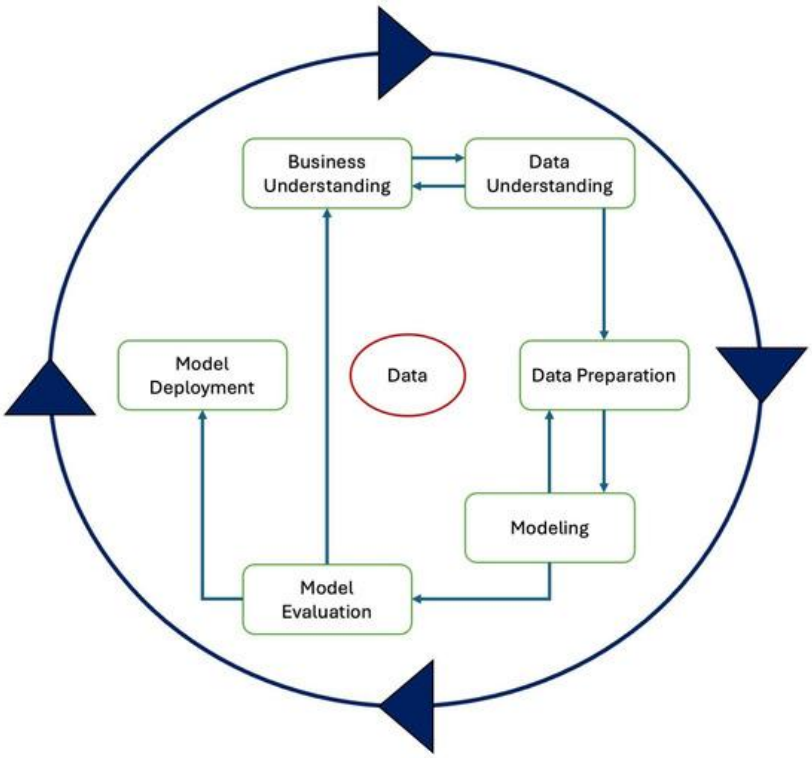


Figure 3.1 CRISP-DM Methodology [40]

The initial Business and Data Understanding phase was accomplished by defining the problem of information overload in healthcare reviews and collecting the localized Penang dataset. The Data Preparation phase involves the cleaning of raw review text, performing sentiment analysis with VADER to generate new features, and structuring the data into an SQLite database using the load_database.py script. Following this, the Modeling phase involved the separate training of the SVM and SVD models, which were then saved as .pkl files, and the development of the core algorithm in hybrid_recommender.py to combine their outputs. The subsequent Evaluation phase involved an assessment of the individual and hybrid models using metrics like accuracy, precision, F1-score, and RMSE. It was during this phase that key insights, such as the SVD model's weakness due to the cold-start problem, were discovered. These findings

directly informed a refinement of the modeling strategy, specifically the adjustment of the alpha parameter to give more weight to the more reliable SVM model. Finally, the Deployment phase consisted of building the complete web application, with Flask (app.py) serving as the backend and HTML files in the templates folder providing the user-facing frontend, resulting in a functional healthcare recommendation system.

3.1 System Design Equation

The overall system contains several mathematical formulations, and the equations are as follows:

The first equation is on SVM, and the equation is as follows:

$$y(w \cdot x - b) \geq 1 \text{ , for all i=1 to n}$$

- y = class label for the i-th sample
- b = bias term, scalar value that helps position the hyperplane
- n = total number of training samples
- w = weight vector
- x = feature vector for the i-th sample
- $w \cdot x$ represents the dot product between the weight vector and the feature vector

The second equation is on SVD, and the equation is as follows:

$$r\,(u,i) = \mu + b(u) + b(i) + q(i)^t \cdot p(u)$$

- $r\,(u,i)$ = predicted rating for specific user u on a specific item i
- μ = global average rating
- $b(u)$ = bias of user u
- $b(i)$ = bias of item i
- $q(i)$ = latent factor vector for item i
- $p(u)$ = latent factor vector for user u
- $q(i)^t \cdot p(u)$ = dot product between the transpose of item's factor vector and user's factor vector

The next equation is on TF-IDF and below is the equation:

$$TF\text{-}IDF\ (t, d) = TF(t, d) \times IDF(t)$$

- t = term (word)
- d = document (review)
- $TF(t, d) = \frac{count(t,d)}{total\ term\ in\ d}$, frequency of term, t in document, d
- $IDF(t) = log\ \frac{N}{1+|\{d:t \in d\}}$, inverse document frequency
- N = total number of documents

The following equation is on Haversine Distance and below is the equation:

$$d = 2R\ \cdot \arcsin\left(\sqrt{sin^2\frac{(\emptyset_2 - \emptyset_1)}{2} + \cos(\emptyset_1)\cos(\emptyset_2)sin^2(\frac{\lambda_2 - \lambda_1}{2})}\right)$$

- d = The calculated distance between the two points (in kilometers).
- R = The radius of the Earth
- $\emptyset_1, \emptyset_2$ = The latitudes of the user and the hospital respectively
- λ_1, λ_2 = The longitudes of the user and the hospital respectively

Finally, the last equation is on the hybrid score calculation and the equation is as follows:

$$Hybrid(x) = \alpha \cdot CB(x) + (1 - \alpha) \cdot CF(x)$$

- $\mathrm{CB}(x)$ = Score Content from SVM
- $\mathrm{CF(x)}$ = Predicted continuous score from SVD for input x
- α = Weight parameter that balances the contribution of SVM and SVD

To evaluate the performance of the models, the performance measurement that will be used in this project includes:

First performance metric used is RMSE, and the equation is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r_i})^2}$$

- r_i = actual rating in the test set
- $\hat{r_i}$ = predicted rating
- n = number of test samples

Next performance metric will be MAE, and the equation is as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

- y_i: actual sentiment score (from dataset)
- $\hat{y_i}$: predicted sentiment score (from SVR)
- n: total number of predictions

The third performance metric is accuracy, and the equation is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- TP (True Positive): correctly predicted positive reviews
- TN (True Negative): correctly predicted negative reviews
- FP (False Positive): negative reviews incorrectly predicted as positive
- FN (False Negative): positive reviews incorrectly predicted as negative

The fourth performance metric used is recall, and the equation is as follows:

$$Recall = \frac{TP}{TP + FN}$$

- TP = True Positive
- FN = False Negative

The fifth performance metric is precision, and the equation is as follows:

$$Precision = \frac{TP}{TP + FP}$$

- TP: True Positive
- FP: False Positive

The last performance metric used in this project is F1-score and the equation is as follows:

$$F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

- Precision: fraction of correct positive predictions
- Recall: fraction of actual positives correctly predicted

3.1.1 System Architecture Diagram

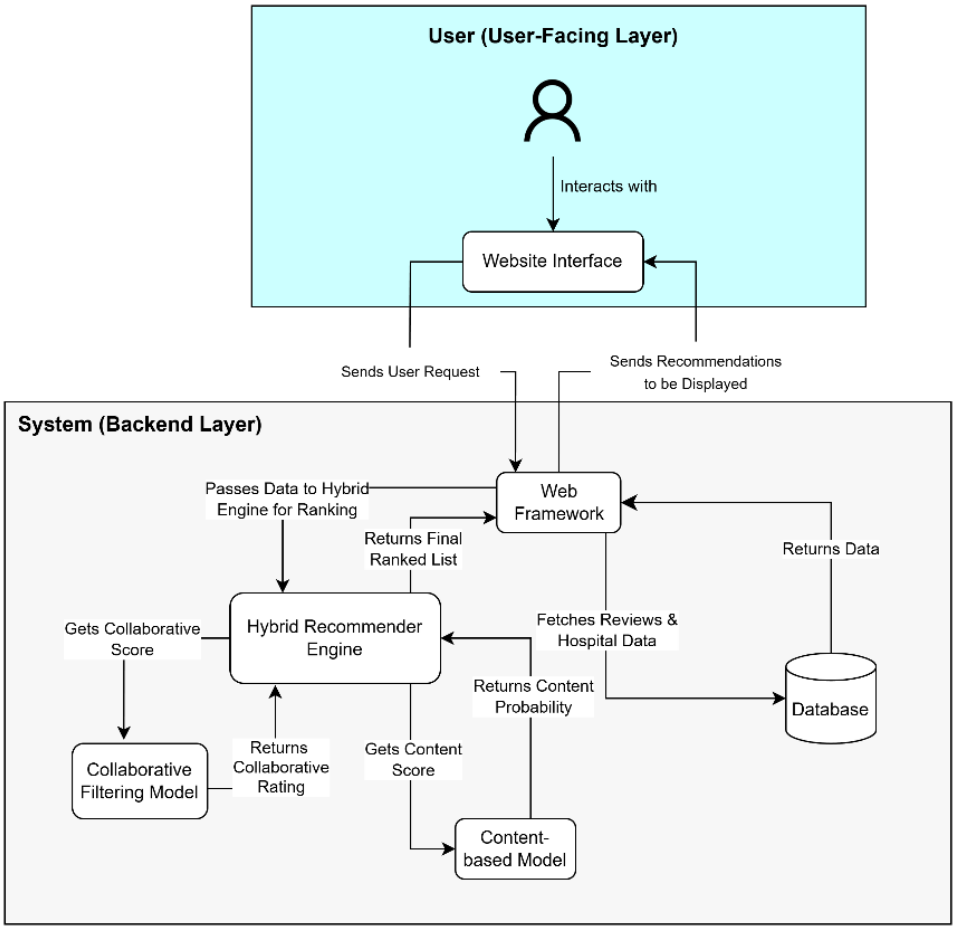


Figure 3.2 System Architecture Diagram

The system's architecture was comprised to two sections; the light blue box indicates the user-facing layer while the grey box indicates the backend layer. The workflow begins when a user enters their symptoms on the website interface. This request was managed by the Flask application, which first queries the database for relevant reviews.

Once the data was fetched from Flask, it was then passed into the Hybrid Recommender Engine. This engine uses two different models at the same time to get the best results. The first is a Collaborative Filtering Model that uses SVD to find a personalized rating based on what similar users liked. The second is a Content-based Model that uses an SVM algorithm to look at the specific features of the hospitals and the quality of their reviews. By using both models, the system can understand both the user's personal needs and the actual quality of the medical facilities.

After both models calculated their results, the hybrid engine mathematically combined the collaborative rating and the content score. This created a final, ranked list of recommendations that is sent back to the Flask web framework. The framework then sent this finalized list across to the website interface, where it is displayed clearly for the user to see. This entire flow ensures that the user receives a list of hospital recommendations that were accurate and easy to understand in real-time.

3.1.2 Use Case Diagram and Description

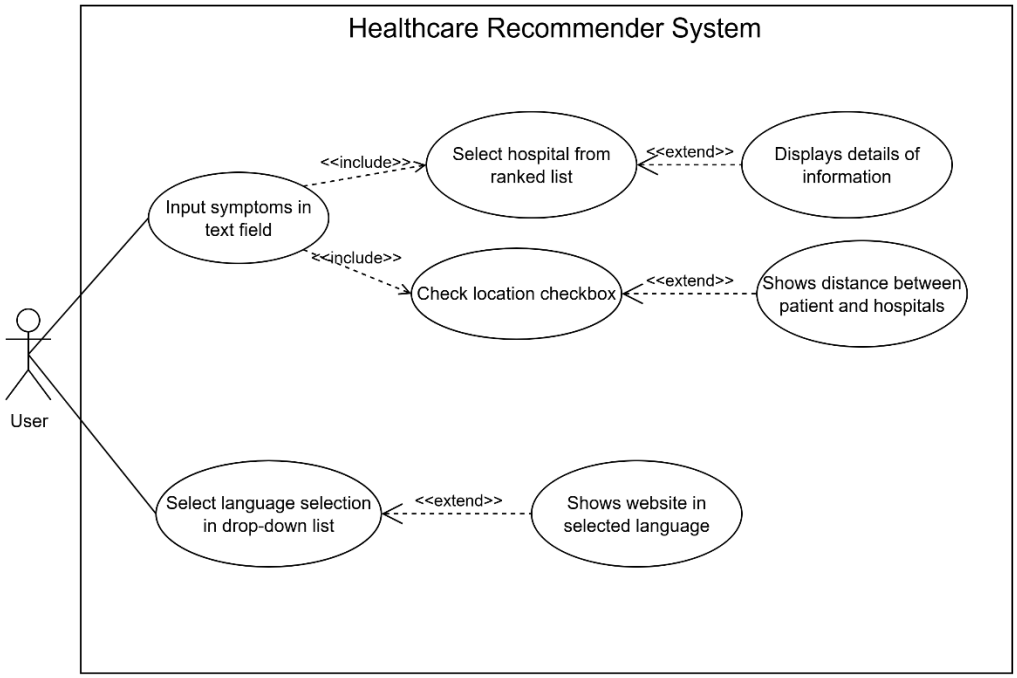


Figure 3.3 Use Case Diagram for User

This use case outlines the process for a user to receive hospital recommendations. The user begins by entering their medical symptoms, either by selecting from a dropdown list or typing them manually. After initiating the search, the system presents a ranked list of hospitals. From these results, the user can click to view a detailed page for any hospital. The use case also covers additional key features, such as the ability to switch the website's language between English and Chinese, and an option to re-rank the recommendations based on proximity by allowing the system to access their current location.

Table 3.1 Use Case Description for User

Use Case Name: Get Hospital Recommendation	ID: 1	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: User - Wants to find a suitable and highly rated hospital, optionally sorted by location.		
Brief Description: This use case describes how a user enters their medical symptoms into the system to receive a ranked list of recommended healthcare providers.		
Trigger: The user accesses the recommender system's homepage with the intent to find a suitable hospital for health concerns. Type: External		
Relationships: Association: Include: Extend: Generalisation : User		
Normal Flow of Events: 1. The user navigates to the system's main page. 2. User enters symptoms from a drop-down list or manually enter symptoms into an input field. 3. User clicks on the “Find a Hospital” button. 4. User views the ranked list of recommended hospitals, and their basic information such as hospital’s name and overall score. 5. If user clicks on one of the hospitals’ names a. User is navigated to hospital’s detail page. b. User can press “Back to search” button to navigate back to homepage. 6. If user clicks on “Select language” a. User can choose English or Chinese (Simplified) 7. If user checks on “Prioritize nearby hospitals” checkbox a. User allows location access permission. b. The list of ranked hospitals will show distance between user and hospital.		
SubFlows: S-1: User does not allow location access permission 1. Website asks for user’s permission to access location. 2. User clicks on “block”.		
Alternate/Exceptional Flows: S-1, 2a1: website shows a message “Location access was denied.” Until the user allows permission for location access.		

3.1.3 Activity Diagram

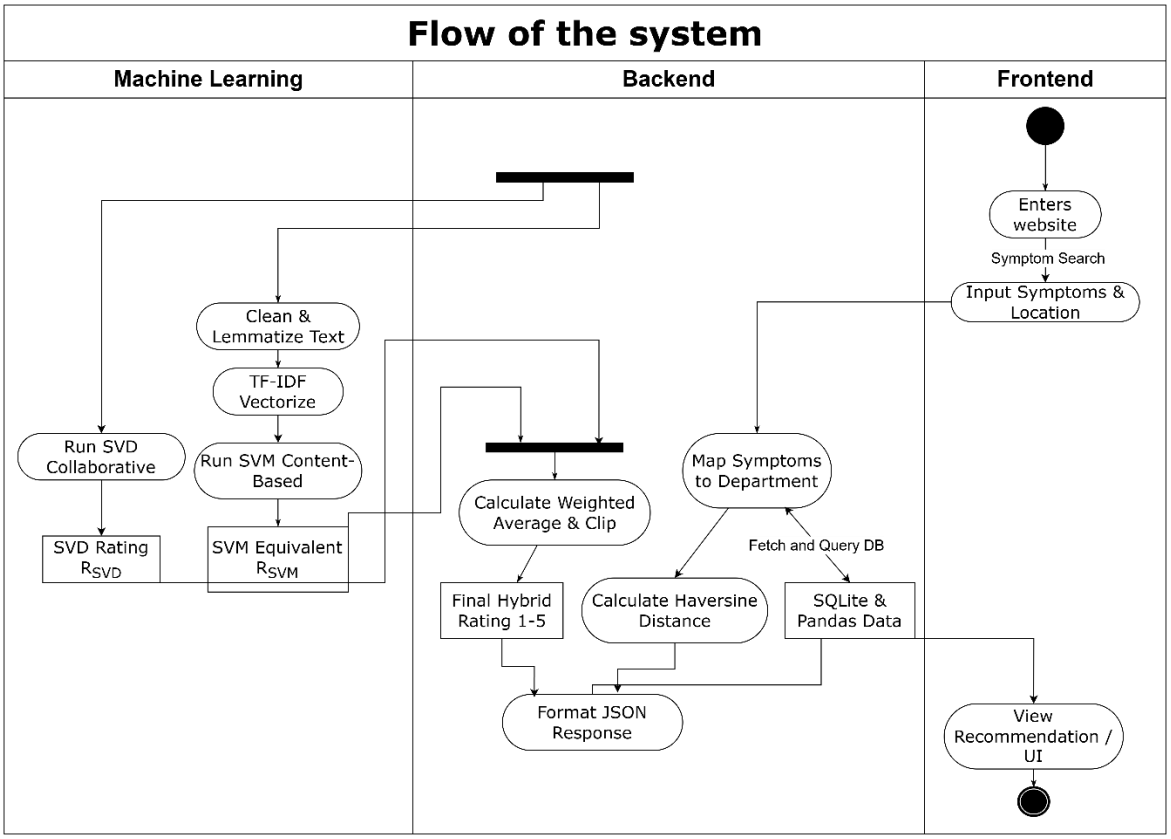


Figure 3.4 Activity diagram

Figure 3.4 shows the activity diagram of this project. The process begins with the user entering the website. Once the user approves the permission to access their location, the process begins with the frontend capturing their symptoms and geographical coordinates. In the backend, the system initiates the “Map Symptoms to Department” module, which interacts directly with the SQLite and Pandas data storage; a bidirectional arrow is used to represent how data is queried and fetched to identify the medical specialty. Simultaneously, the system executes two parallel machine learning workflows: an SVD model for collaborative filtering and an SVM model for content-based analysis of text data. These two outputs are consolidated into a final hybrid rating using a weighted average calculation. While this occurs, the system also computes the Haversine distance to determine the proximity of nearby facilities. Finally, the backend packages all these metrics into a formatted JSON response, which the frontend parses to render the final recommendations on the user's interface.

3.1.4 File Directory Tree

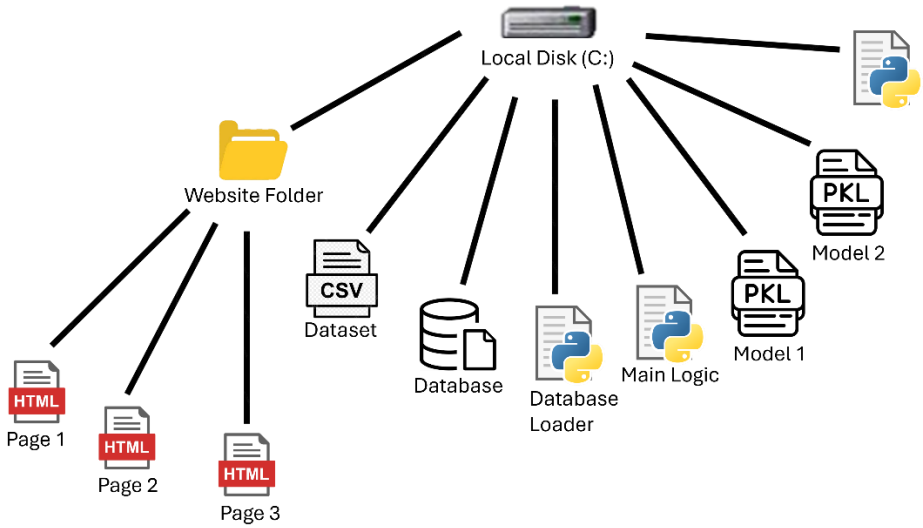


Figure 3.5 File Directory Tree

This diagram illustrates the architecture of this project. A central server runs the main Python script, which acts as the controller for the application's logic. This script reads raw data from a CSV file and utilizes other Python files for data processing. To generate recommendations, the application leverages two pre-trained machine learning models, a model1 file and a model2 file. The final output and the user interface are presented through various HTML pages, which are organized within their own dedicated folder.

3.2 Sketching of System

This project shows two sketches to outline a simple user interface for a hospital recommendation system. The first image displays the homepage where a user can select a condition from a dropdown menu, or enter their symptoms manually into a text field, and then click “Recommend Hospital” to generate a list of suggestions. This initial screen presents the results with key information such as the number of reviews, average rating, and a recommendation score for each hospital. The second sketch illustrates a subsequent screen, which a user will see after selecting one of the recommended hospitals. This page appears to show more detailed information and provides a "Back to homepage" button for navigation.

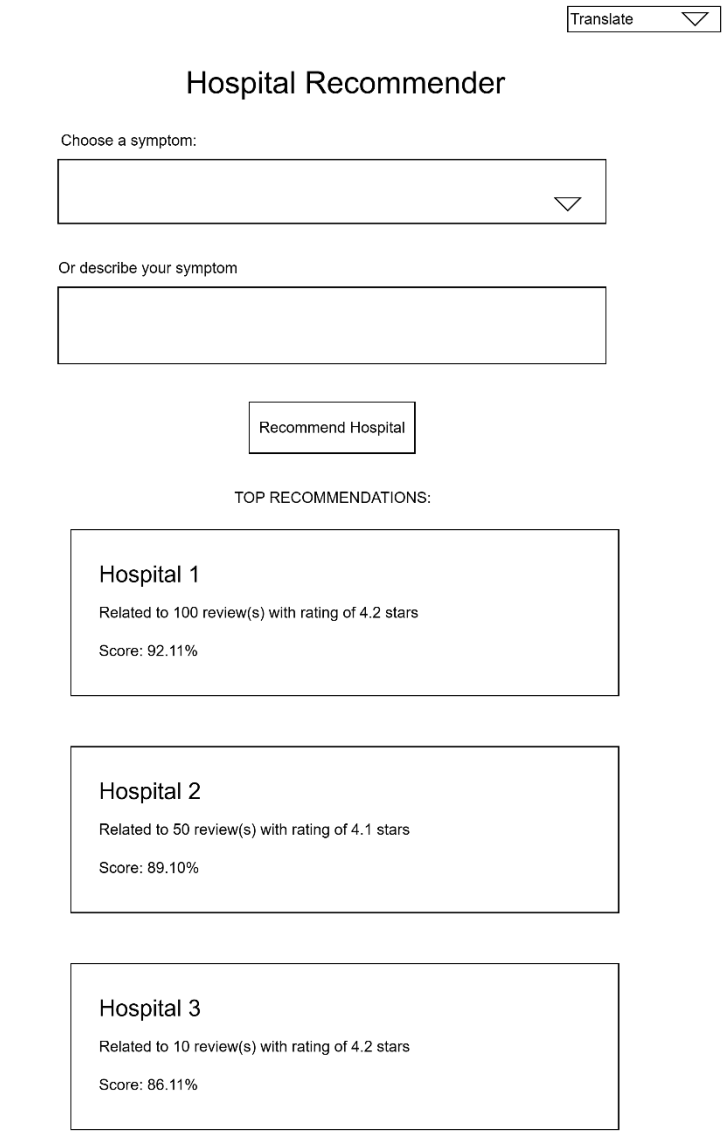


Figure 3.6 Sketching of Homepage

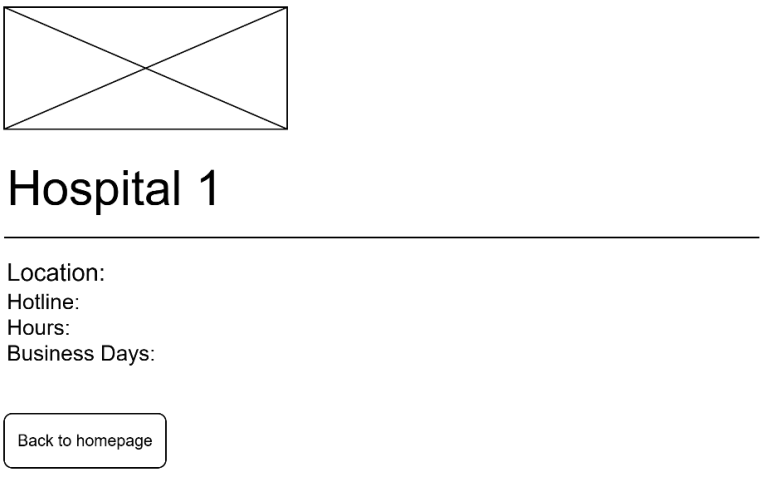


Figure 3.7 Sketching of Hospital Details Page

3.3 Timeline

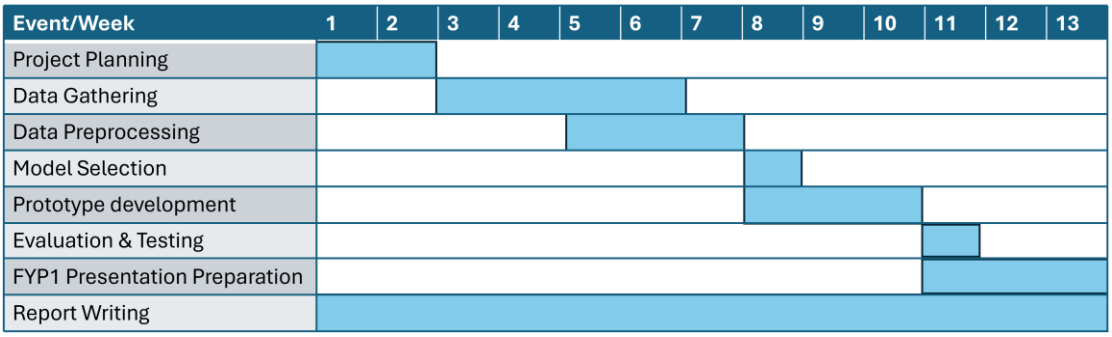


Figure 3.8 Timeline of FYP1

This project began with the planning phase, where the objectives, scope, problem statement, target users, and functional requirements of a web-based healthcare recommendation system were defined, supported by a detailed timeline and supervisor consultations to ensure feasibility. During Weeks 3–6, data was collected from the Pulau Pinang region through web scraping, focusing on key attributes such as user ratings, timestamps, and reviews to build the recommendation engine. Data preprocessing followed in Weeks 5–7, involving cleaning, normalization, and encoding of data, as well as identifying useful patterns for feature selection and model design. In Week 8, various algorithms including collaborative filtering, content-based filtering, and hybrid models were tested using metrics like accuracy, precision, recall, and RMSE, leading to the selection of the most suitable model. From Weeks 8–10, the system prototype was developed using HTML and CSS, integrating the model into a responsive web interface, with iterative improvements based on feedback. Evaluation and testing were conducted in Weeks 11–12 to verify the accuracy, usability, and reliability of the system, resolving bugs and optimizing performance. Week 12 also involved preparing a structured FYP1 presentation summarizing the methodology, implementation, and findings. Throughout Weeks 1–13, report writing was carried out, documenting each stage from planning to evaluation, including challenges faced, lessons learned, and suggestions for future work, ensuring a complete final submission.

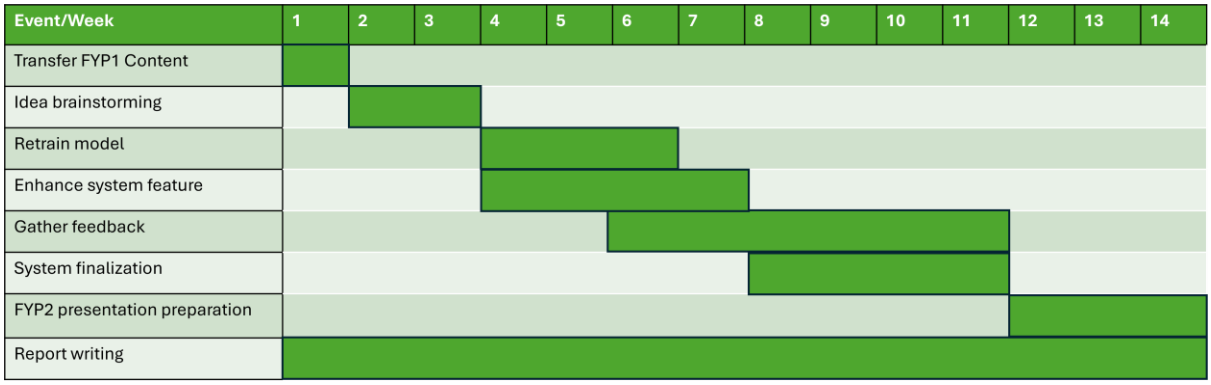


Figure 3.9 Timeline of FYP2

The progress of FYP2 began with the transfer and integration of FYP1 content in Week 1. This was followed by a period of idea brainstorming during Weeks 2 and 3 to refine the project's scope and feature set. Technical features are implemented between Weeks 4 and 7, focusing on reusing the machine learning model (Week 4–6) and enhancing core system features (Week 4–7) to improve overall performance. Starting in Week 6 and continuing through Week 12, feedback gathering process was conducted to identify necessary refinements, which directly informed the system finalization phase that took place from Week 8 to Week 12. As the project is getting nearer to the submission date, preparation for the final FYP2 presentation was carried out from Week 12 to Week 14 to showcase the project's outcomes and methodology. Throughout the entire 14 week duration, continuous report writing was maintained to document the development progress, challenges, and final results, ensuring a structured final submission.

Chapter 4

System Design

This chapter presents the technical design and architectural framework of the proposed recommendation system. This chapter outlines the structural organization of the platform, including high-level system block diagrams, database schemas, and the logical execution of the machine learning pipelines.

4.1 System Block Diagram

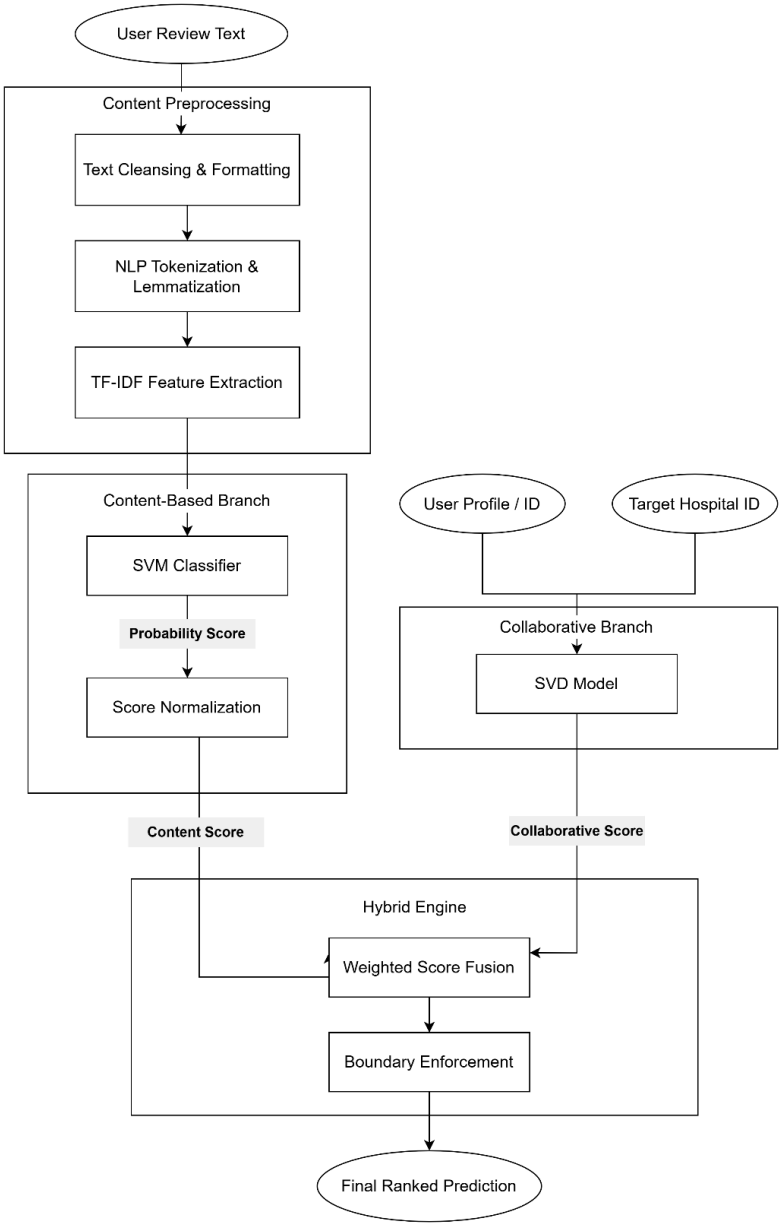


Figure 4.1 System Block Diagram

The above diagram illustrates the functional data flow of the hybrid recommender engine, which operates through two processing branches. On the left, the Content-Based Branch processes raw user review text through cleansing, formatting, and TF-IDF feature extraction, passing it to an SVM classifier to generate a normalized 'Content Score'. Simultaneously, on the right, the Collaborative Branch feeds the User ID and Target Hospital ID into an SVD model to compute a 'Collaborative Score'. Both scores are then routed into the Hybrid Engine, where they undergo weighted score fusion to output the final predicted rating for the user.

4.2 System Components Specifications

This section details the specific requirements and technical parameters of the system and data components used to build the recommendation system.

4.2.1 Software Specifications

To ensure the system is reproducible, the following software environment and library versions are specified:

- Backend Framework: Flask 3.1.1 (Python-based WSGI web application framework).
- Machine Learning Engines:
    - Scikit-learn 1.2+: Used for the SVM classifier and TF-IDF Vectorization
    - Scikit-surprise 1.1.3: Used for the SVD collaborative filtering
- Natural Language Processing: NLTK with VADER for sentiment intensity analysis
- Database Engine: SQLite 3.0
- Data Manipulation: Pandas for DataFrame operations and Numpt for numerical calculations

4.2.2 Database Schema Specification

The system utilizes an SQLite database to store hospital data and processed reviews. The primary table, "reviews", is structured as follows:

Table 4.1 Data dictionary and schema specification for the 'reviews' table

Column Name	Data Type	Description
review_id	INTEGER	Primary Key
Hospital_id	INTEGER	Foreign Key linking the review to a specific hospital
Author_id	INTEGER	Unique identifier for the user (used for SVD personalization)
Star_rating	INTEGER	The numerical rating (1-5) provided by the user
Review_text_raw	TEXT	The original text submitted by users
Processed_content	TEXT	Cleaned and lemmatized text used for SVM classification
Sentiment_score	REAL	The compound intensity score generated by VADER (-1 to 1)
Predicted_sentiment	TEXT	Categorical label (Positive, Negative, Neutral) for the review

4.3 Logical Design and Module Architecture

The logical design of the system is realized through a combination of structured data management and sequential algorithmic processing. The following subsections detail the foundational setup of the database layer and the specific multi-stage machine learning pipeline used to transform raw user inputs into personalized hospital recommendations.

4.3.1 Database Creation (SQL)

To rebuild the system's data layer, the following SQL command is used to initialize the environment:

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS reviews (
    review_id INTEGER PRIMARY KEY AUTOINCREMENT,
    hospital_id INTEGER,
    author_id INTEGER,
    star_rating INTEGER NOT NULL,
    review_text_raw TEXT,
    processed_content TEXT,
    sentiment_score REAL,
    predicted_sentiment TEXT,
    FOREIGN KEY (hospital_id) REFERENCES hospitals (hospital_id),
    FOREIGN KEY (author_id) REFERENCES authors (author_id)
);
""")
print("Tables created successfully.")
```

Figure 4.2 SQL command to create table reviews

4.3.2 Machine Learning Pipeline Design

The recommendation engine follows a structured multi-stage pipeline to transform raw input into a final hybrid rating. This pipeline is encapsulated in the predict_hybrid_rating function and utilizes two serialized models (.pkl files).

Stage 1: Text Preprocessing and Normalization

Before text can be analyzed, it passes through the clean_text method. The design of this "logic circuit" involves:

1. Case Folding: Converting all characters to lowercase.
2. Noise Removal: Using Regular Expressions (re.sub) to strip non-ASCII characters and collapse multiple whitespaces into a single space.
3. Tokenization: Splitting the string into individual words using NLTK's word_tokenize.
4. Stopword Filtering: Removing common English words (e.g., "the", "is") that do not carry medical sentiment.
5. Lemmatization: Reducing words to their base form (e.g., "better" to "good") using the WordNetLemmatizer to ensure consistent feature extraction.

Stage 2: Feature Transformation (TF-IDF)

The cleaned text was passed to the tfidf_vectorizer. This component converted the string into a numerical feature vector. It calculated the TF-IDF, which weighs words based on how unique they are to a specific hospital's review compared to the entire dataset.

Stage 3: Parallel Model Inference

The system executed two distinct predictive paths simultaneously:

- **Path A (Collaborative Filtering):** The svd_model.predict(uid, iid) method estimated a rating (R_{SVD}) on a 1.0 to 5.0 scale based on the user's historical preferences and the hospital's past performance.
- **Path B (Content-Based):** The svm_model.predict_proba method calculated the probability (P_{SVM}) that the input review/symptom is positive (0.0 to 1.0). To make this compatible with the SVD score, a Linear Scaling logic is applied:

$$R_{SVM} = (P_{SVM} \times 4.0) + 1.0$$

This maps the probability to the same 1.0 to 5.0 scale used by the SVD.

Stage 4: Hybrid Fusion and XAI Generation

The final stage of the pipeline merged the two scores using a weighted average defined by the parameter α (default = 0.5):

$$\text{Final Rating} = (\alpha \times R_{SVM}) + ((1 - \alpha) \times R_{SVD})$$

The pipeline concluded with an XAI logic block. This sub-module used the .get_feature_names_out() method of the vectorizer to identify which specific words in the user's input most heavily influenced the SVM's decision. These "top features" were returned as a list to the frontend to explain why a hospital was recommended (for example, "high quality care", "friendly staff").

4.4 System Components Interaction Operations

This section shows the methods used in this project, how the frontend, backend, and machine learning models communicate throughout this project.

4.4.1 Methods and Classes Used

The table shows the primary methods and classes used. These methods represent the core operational logic of this project.

Table 4.2 Functional specifications of core system methods and modules

Module Member	**Functionality**	**Input**	**Output**
clean_text()	Text Preprocessing	String (Raw Review)	String (Cleaned/Lemmatized)
predict_proba()	SVM Sentiment Analysis	Vectorized Text	Probability [0, 1]
transform()	Text Vectorization	String (Cleaned)	Sparse Matrix (TF-IDF)
est	SVD Rating Prediction	User ID, Item ID	Predicted Rating [1, 5]
np.clip()	Boundary Enforcement	Hybrid Score	Score capped at range [1, 5]
nonzero()[1]	XAI Feature Extraction	Sparse Matrix	Index of recognized keywords
get_feature_names_out()	Feature Decoding	N/A	List of vocabulary words
calculate_haversine_distance()	Distance Calculation	Lat/Lon (User & Hosp)	Distance in Kilometers (Float)
get_department_from_symptoms()	Symptom Parsing	String (User Input)	Department Name & Keyword
recommend_hospitals_for_department()	Multi-Factor Ranking	Dept, Lat, Lon	List of Sorted Hospital Objects
check_password_hash()	User Authentication	Hashed Pass, Plain Pass	Boolean (True/False)
pd.read_sql_query()	Data Synchronization	SQL Query, Conn Object	Pandas DataFrame

4.4.2 Client-Server Interaction (AJAX/JSON)

The interaction between the user interface and the Flash server was asynchronous.

1. Request: When the user clicks “Analyze and Find Hospitals”, a JavaScript fetch() request is sent to the /recommend endpoint containing the symptom string.
2. Processing: The Flask app.py received the JSON payload and called the hybrid_recommender.py module.
3. Model Inference: The hybrid_recommender loaded SVM_model.pkl and SVD_model.pkl into memory. It iterated through all hospitals in the SQLite database that belongs to the relevant department.
4. Response: The server returned a JSON object containing a list of hospitals, their hybrid score, and distances.

4.4.3 Operation Steps for Rebuilding the System

To rebuild this project, the following interaction steps must be followed:

1. Data Ingestion: Run load_database.py to populate the SQLite database from the cleaned_dataset.csv.
2. Model Serialization: Execute the training scripts in Google Colab/Local Python to generate the .pkl files.
3. Environment Linking: Place the .pkl files in the models/ directory so app.py can reference them.
4. Server Initialziation: Run flask run. The Flask application acts as the “Controller”, managing the interaction between the “Model” (SVD/SVM/SQLite) and the “View” (HTML/CSS templates).

4.4.4 Multi-language Toggle Operation

The system included a translation dictionary component. When the user selects “Chinese”, the frontend sent a state change to the UI controller. The system did not re-translate the database but instead swapped the UI labels and department names using a localized JSON mapping file, ensuring the system remains performant without redundant ML processing. The result will be shown in Chapter 6.

Chapter 5
System Implementation

This chapter describes the technical environment, the data processing procedures, and the specific steps taken to integrate the machine learning models into a functional web application. By documenting the transformation of theoretical designs into a working software solution, this section provides a record of the system's construction phase and the practical integration of its various components.

5.1 Hardware Setup

The hardware involved in this project is a laptop.

Table 5.1 Specification of laptop

Description	Specifications
Model	MSI Modern 15 A5M
Processor	AMD Ryzen 7 5700U
Operating System	Windows 11
Graphic	AMD Radeon Graphics
Memory	12GB DDR4 RAM
Storage	512GB NVMe PCIe SSD

5.2 Software Setup

Before starting to develop the project, there are several software needed to be installed:

1. Visual Studio 2022 Enterprise Edition
2. Phantom – GMB Audit Tool
3. Google Colab
4. Microsoft Excel
5. DB Browser for SQLite 3.13.1
6. Flask Version 3.1.1

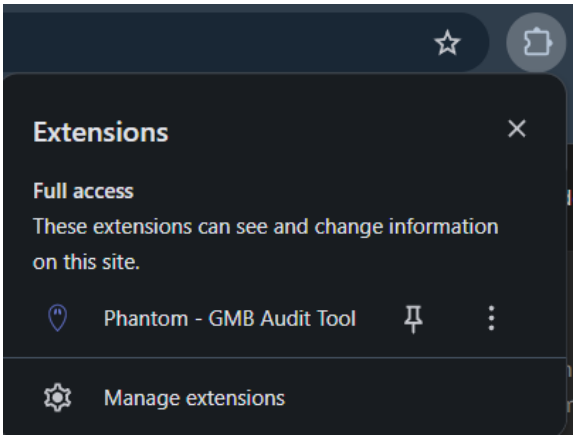


Figure 5.1 Phantom GMB in Extension Tab

For Phantom – GMB Audit Tool, it can be downloaded from https://phantomlocal.com, once the tool is downloaded, the browser will asks for permission and will appear in the extensions tab as shown in Figure 5.1.

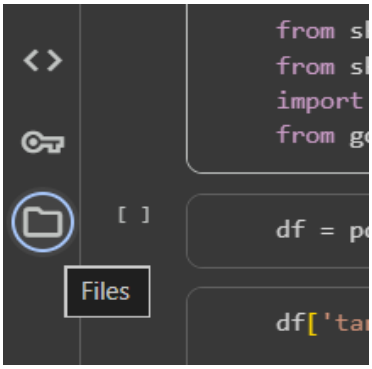


Figure 5.2 Files icon in Google Colab

For Google Colab, to load .csv or .pkl files, the file icon was pressed as shown in Figure 5.2, then the upload icon was clicked as shown in Figure 5.3 and wanted files were selected to upload them.

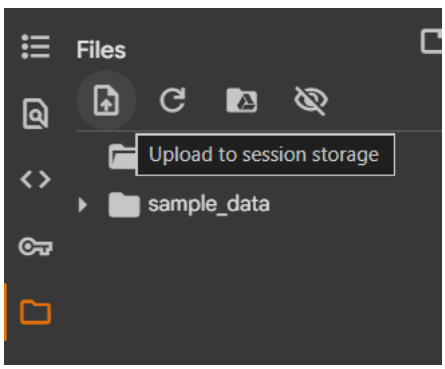


Figure 5.3 Upload icon in Google Colab

Further on during website development, to run the website through Flask, Flask and necessary libraries such as numpy, pandas, scilit-learn, scikit-surprise, nltk, and joblib must first be installed using command “pip install” through command prompt. Then, after arranging the file directory paths, command “python -m flask run” was executed to run the website on local host.

5.3 Setting and Configuration

The development process is structured into several phases to ensure that raw data is transformed into a format suitable for machine learning. These phases require specific configurations and software environments to maintain data integrity and prepare the dataset for the final recommendation engine.

5.3.1 Data Collection Phase

The reviews were collected from a software tool called “Phantom – GMB Audit Tool”, several attributes will be collected such as Review date, Review URL, Author name, Author URL, Local Guide, Author reviews, Star rating, Review content, Review image, and Review video. To extract the reviews from different hospitals, the steps below are followed.

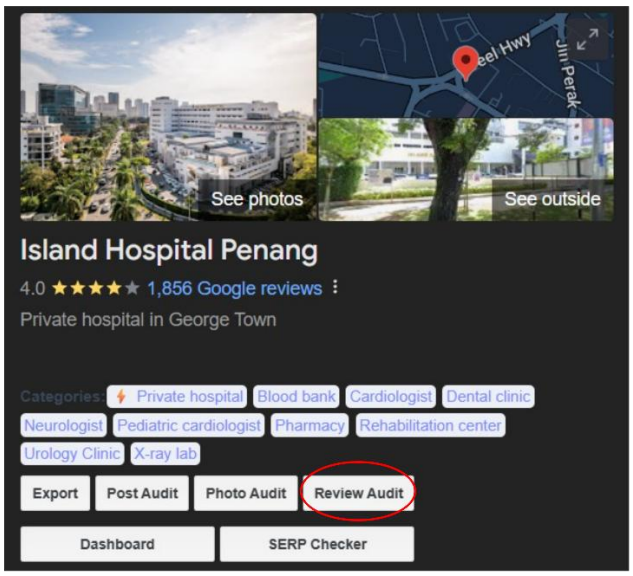
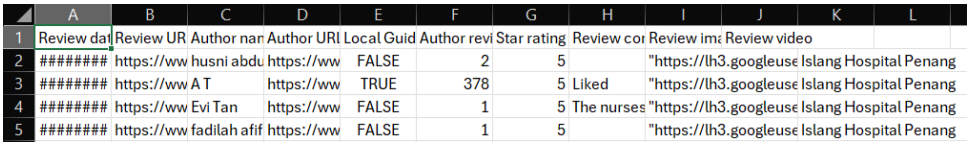


Figure 5.4 Phantom GMB interface under Google Maps

First, type in the desired hospital name in Google. After that, the GMB interface shows up as shown in Figure 5.4. Then, click on the “Review Audit” button and GMB will take some time to load the reviews. Figure 5.5 shows the sample output for reviews in Island Hospital Penang.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Review dat	Review UR	Author nan	Author URI	Local Guid	Author revi	Star rating	Review cor	Review ima	Review video		
2	########	https://ww	husni abdu	https://ww	FALSE	2	5		"https://lh3.googleuse	Island Hospital Penang		
3	########	https://ww	A T	https://ww	TRUE	378	5	Liked	"https://lh3.googleuse	Island Hospital Penang		
4	########	https://ww	Evi Tan	https://ww	FALSE	1	5	The nurses	"https://lh3.googleuse	Island Hospital Penang		
5	########	https://ww	fadilah afif	https://ww	FALSE	1	5		"https://lh3.googleuse	Island Hospital Penang		

Figure 5.5 Sample output of reviews in Island Hospital Penang

Table 5.2 Number of reviews for each hospital before cleaning

Hospital Name	**Number of reviews**
Penang Adventist Hospital	891
Gleneagles Hospital Penang	982
MountMiriam Cancer Hospital	124
Bagan Specialist Centre	983
Sunway Medical Centre Penang	930
Island Hospital Penang	960
Pantai Hospital Penang	959
KPJ Penang Specialist Hospital	797
Penang General Hospital	747
Lam Wah Ee Hospital	987
Kepala Batas Hospital	217
Hospital Seberang Jaya	471
Bukit Mertajam Hospital	336
Sungai Bakap Hospital	162
Balik Pulau Hospital	94
LohGuanLye Specialists Centre	515

After the extraction of reviews are done for all hospitals, the reviews are combined and collected into a .csv file named “DATASET.csv”. The data collection phase is done, and the .csv is loaded into Google Colab for data cleaning and preprocessing.

5.3.2 Data Cleaning and Preprocessing Phase

First, necessary libraries were imported such as langdetect, pandas, numpy, re and files from google.colab as shown in Figure below.

```
[ ]  import pandas as pd
     from langdetect import detect, LangDetectException
     import numpy as np
     import re
     from google.colab import files
```

Figure 5.6 Import of necessary libraries for data preprocessing

Then, the raw dataset was loaded using “pd.read_csv('DATASET.csv')”. To start cleaning the data, this project checked for missing value using “isnull().sum()” and dropped them using “.dropna()”. After dropping missing value, unwanted columns were also dropped such as 'Review video', 'Review image', 'Review date', 'Local Guide', 'Author URL', 'Review URL' using .drop() as shown in Figure 5.7 and assign the dataset to df_cleaned.

```
[ ]  df_cleaned = df.drop(['Review video', 'Review image', 'Review date', 'Local Guide', 'Author URL', 'Review URL'], axis=1)
```

Figure 5.7 drop() command

Next, this project checked for duplicate data using .duplicated() and dropped the rows with duplicate data using .drop_duplicated(). After that, unique symbols and non-English reviews were removed for preprocessing step.

```
def clean_text(text):
    """A single function to perform all text cleaning steps."""
    if not isinstance(text, str):
        return ""

    # 5.1 Remove non-ASCII characters (emoji remnants, weird symbols)
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)

    # 5.2 Precisely remove 'EEE' and 'EEEE' artifacts
    text = re.sub(r'\bEEE\w*\b', '', text)

    # 5.3 Clean up any resulting extra whitespace
    text = re.sub(r'\s+', ' ', text).strip()

    return text
```

Figure 5.8 clean_text() Function

The above cleaning function named “clean_text” was applied to the “Review content” column using .apply(clean_text). And for the language detection, the function was created as shown in Figure 5.8.

```
[ ]  def detect_language(text):
         """Detects the language of a text string."""
         try:
             return detect(text)
         except LangDetectException:
             return 'unknown'

     df_cleaned['Language'] = df_cleaned['Review content'].apply(detect_language)

     #Filter
     initial_rows = len(df_cleaned)
     final_df = df_cleaned[df_cleaned['Language'] == 'en'].copy()

     final_df = final_df.drop(columns=['Language'])

     final_df.replace('', pd.NA, inplace=True)
     final_df.dropna(subset=['Review content'], inplace=True)
```

Figure 5.9 detect_language() Function

Now, the dataset was fully pre-processed and saved it into another .csv file named "cleaned_dataset.csv" so that it is ready to be used for sentiment analysis.

5.4 Sentiment Analysis

For sentiment analysis, necessary libraries were imported such as langdetect, pandas, StringIO, and files from google.colab as shown in Figure 5.10.

```
[ ]  import pandas as pd
     from io import StringIO
     from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
     from google.colab import files
```

Figure 5.10 Import of necessary libraries for sentiment analysis

First, VADER sentiment intensity analyzer was initialized using "analyzer = SentimentIntensityAnalyzer()". Next, the project creates two functions for sentiment analysis as shown in figure below.

```
# --- Function to get sentiment scores ---
def get_sentiment(text):

    return analyzer.polarity_scores(text)['compound']

# --- Function to classify sentiment based on score ---
def classify_sentiment(score):
    if score >= 0.05:
        return 'Positive'
    elif score <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'
```

Figure 5.11 get_sentiment() Function

The get_sentiment function was designed to read a patient review and calculate a single numerical score that represents its overall sentiment. It used a pre-trained sentiment analysis tool to generate a "compound" score. This score is a single value from -1 to +1 that summarizes the emotional tone of the text. Next, the classify_sentiment function's job was to take the numerical sentiment score and convert it into a positive-negative category. For scoring above 0.05, score will be labelled as 'Positive', any score below -0.05 is ‘Negative’, and anything in between is considered 'Neutral'. This step turned the raw number into a clear and useful label for each review. After that, as shown in figure 5.12, get_sentiment was applied on processed_content column to create a new column named ‘Sentiment Score’. And classify_sentiment was applied on Sentiment Score to create another new column named ‘Predicted Sentiment’, which contains the ‘Positive’, ‘Negative’, or ‘Neutral’ labels for every review.

```
[ ]  # Apply the functions to the DataFrame to create new columns
     # Ensure 'Review content' is string type and handle potential missing values
     df['processed_content'] = df['processed_content'].astype(str).fillna('')
     df['Sentiment Score'] = df['processed_content'].apply(get_sentiment)
     df['Predicted Sentiment'] = df['Sentiment Score'].apply(classify_sentiment)

     # Display the relevant columns of the DataFrame with the new sentiment analysis results
     print("Sentiment Analysis Results:")
     print(df[['Star rating', 'Predicted Sentiment', 'Sentiment Score', 'processed_content']].head())
```

Figure 5.12 Sentiment Analysis Coding

Once the sentiment analysis was done, the file will be saved as ‘dataset_withsa.csv’ using files.download() and this file will be saved for further use during machine learning model development.

5.5 Data Visualization

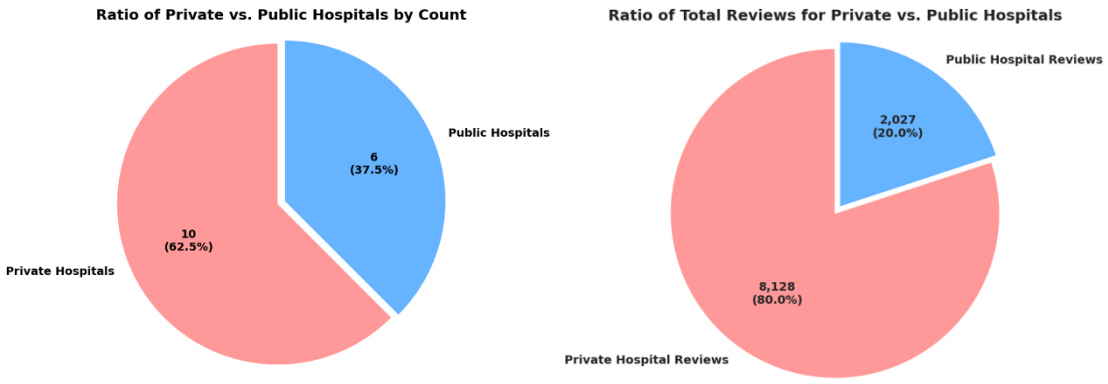


Figure 5.13 Ratio of public and private hospitals (left); Ratio of total reviews for public and private hospitals

Figure 5.13 shows two pie charts illustrating the distribution of public and private hospitals within the dataset. The chart on the left shows that while the dataset contains a mix of both, private hospitals represent most institutions, making up 62.5% (10 hospitals) compared to 37.5% (6 hospitals) for the public sector. The chart on the right shows reviews from both public and private hospitals. Private hospitals are overwhelmingly dominant with 80% (8,128 reviews) of the entire dataset, while public hospitals contribute the remaining 20% (2,027 reviews).

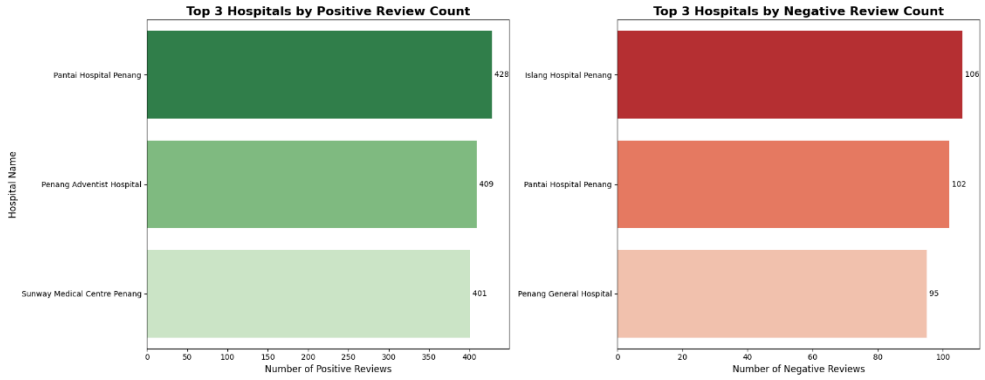


Figure 5.14 Top 3 hospitals by positive (left) and negative (right) count

This visualization shows the top three hospitals with the highest positive and negative count respectively. The chart on the left highlights the top three hospitals with the highest volume of positive reviews, indicating consistent operational success. These high counts are driven by the friendliness and professionalism of the staff, the cleanliness of the hospital environment, and

effective communication. Conversely, the chart on the right shows the hospitals accumulating the most negative reviews, which often point to significant systemic issues. Common complaints in this category include the unavailability of facilities or specialists, excessively long waiting times, poor bedside manner, or administrative problems.

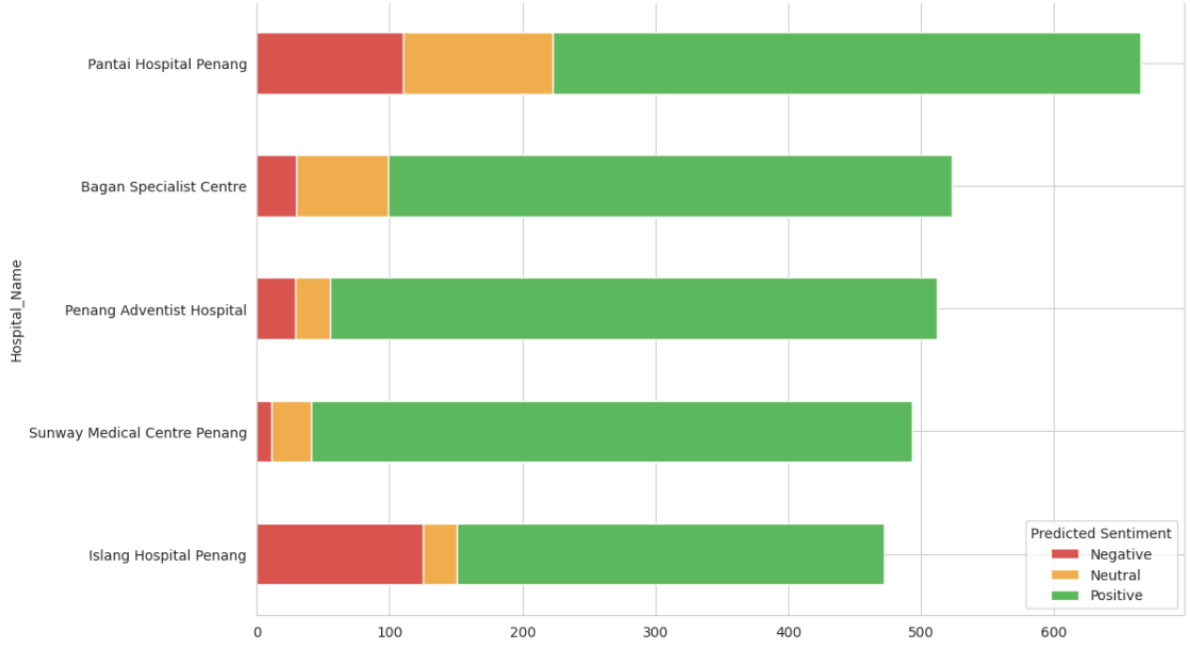


Figure 5.15 Sentiment distribution across 5 hospitals with highest volume of reviews

This stacked horizontal bar chart shows the sentiment distribution across the five hospitals with the highest volume of user-generated reviews. The green segments represent favourable feedback, the orange segments represent neutral reviews, and the red segments represent negative reviews. Specifically, Pantai Hospital Penang and Island Hospital Penang show a more significant incidence of negative sentiment, as indicated by their larger red segments. In contrast, institutions like Sunway Medical Centre Penang exhibits a markedly lower volume of negative reviews, suggesting a higher consistency in delivering satisfactory patient experiences. Therefore, while the overall sentiment is positive, the varying distribution of negative and neutral feedback highlights the lack of patient satisfaction and perceived service quality among these hospitals.

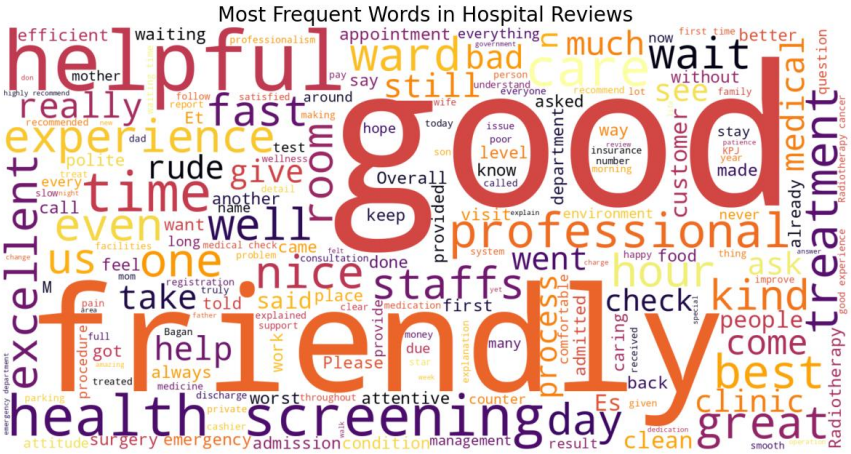


Figure 5.16 Most frequent words in reviews

Figure 5.16 shows the most frequent terms found in the collected hospital reviews from Penang. The most prominent words are overwhelmingly positive, with terms like “good”, “friendly”, “helpful,” “great”, and “best” appearing with the highest frequency. This suggests that most patients’ feedback is centred on positive experiences. Words like “time”, “appointment”, and “waiting” are also important, indicating that service efficiency and scheduling are common topics of discussion in reviews. However, there is also a notable presence of negative words. Terms such as “rude”, “bad”, “poor”, and “worst” appear, highlighting specific instances of negative interactions or service quality. Overall, the word cloud shows patient satisfaction is being heavily influenced by staff attitude and the quality of care provided.

5.6 Machine Learning Models Development

Three machine learning models are developed, which are SVM, SVD and the hybrid model.

```
[ ]  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 classification_report, accuracy_score
     import joblib
     from google.colab import files

[ ]  df = pd.read_csv('dataset_withsa.csv')
```

Figure 5.17 Import of necessary libraries for machine learning

First, for SVM, necessary libraries are imported such as pandas, train_test_split, TfidfVectorizer, SVC from sklearn.svm, joblib, and files from google.colab as shown in Figure

5.17. After ‘dataset_withsa.csv’ has been loaded, the coding creates a new column named ‘target’ in the dataframe ‘df’ that separates positive rating if x is larger or equals to four and negative rating if else. Then, ‘Review content’ column will be assigned to x and ‘target’ column will be assigned to y. This project split df for evaluation into 70-15-15, which means 70% for training, 15% for testing, and 15% for validation.

```
# First split: df into trainval (85%) and test (15%)
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y, test_size=0.15, random_state=42, stratify=y)

# Second split: trainval (85%) into train (70%) and val (15%)
X_train, X_val, y_train, y_val = train_test_split(X_trainval, y_trainval, test_size=(0.15/0.85), random_state=42, stratify=y_trainval)
```

Figure 5.18 Splitting of data for SVM

Then, TF-IDF is performed as shown in figure 5.19.

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')

# Fit and transform the training data
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Transform the validation and test data
X_val_tfidf = tfidf_vectorizer.transform(X_val)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Figure 5.19 TD-IDF vectorizer coding

After performing TF-IDF, SVM model is trained and evaluated as shown in figure 5.20.

```
svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_train_tfidf, y_train)
```

```
SVC(kernel='linear', probability=True, random_state=42)
```

Model Evaluation

```
[6] #Evaluate on Validation set
    val_predictions = svm_model.predict(X_val_tfidf)
    val_accuracy = accuracy_score(y_val, val_predictions)

    #Evaluate on Test set
    test_predictions = svm_model.predict(X_test_tfidf)
    test_accuracy = accuracy_score(y_test, test_predictions)
```

```
[8] accuracy = accuracy_score(y_test, test_predictions)
    print(f"\nOverall Accuracy: {accuracy:.4f}")
    print("-" * 50)
    print("Classification Report:")
    # Show 'Negative' and 'Positive' instead of 0 and 1
    print(classification_report(y_test, test_predictions, target_names=['Negative', 'Positive']))
    print("-" * 50)
```

Figure 5.20 Training and Evaluation on SVM

Finally, accuracy and the classification report of the model is shown in figure 5.21, and the model is saved as ‘SVM_model.pkl’ file for further hybrid implementation.

```
Overall Accuracy: 0.9348
--------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

    Negative       0.80      0.95      0.87       139
    Positive       0.98      0.93      0.96       459

    accuracy                           0.93       598
   macro avg       0.89      0.94      0.91       598
weighted avg       0.94      0.93      0.94       598

--------------------------------------------------
```

Figure 5.21 Performance metrics on SVM

For SVD, one important library is ‘surprise’ and the modules contain Reader, SVD, accuracy, and Dataset. Before splitting of the data, three columns ‘Author name’, ‘Hospital_Name’, and ‘Star rating’ are renamed into simpler names without spaces : ‘userID’, ‘itemID’, and ‘rating’ as this is what ‘surprise’ library expects.

```
trainval_df, test_df = sklearn_train_test_split(
    df,
    test_size=0.15,
    random_state=42
)

train_df, val_df = sklearn_train_test_split(
    trainval_df,
    test_size=(0.15/0.85), # This calculates 15% from the original 85%
    random_state=42
)
```

Figure 5.22 Splitting of data for SVD

Figure 5.22 shows coding to perform a two-step split to create three datasets: 70% for training, 15% for validation, and 15% for testing similar to SVM model.

```
# First split: df_svd into trainval (85%) and test (15%)
trainval_df, test_df = train_test_split(df_svd, test_size=0.15, random_state=42)

# Second split: trainval (85%) into train (70%) and val (15%)
# The test_size here is calculated relative to the trainval_df size
# 15% of original data / 85% of original data = 15/85
train_df, val_df = train_test_split(trainval_df, test_size=(0.15/0.85), random_state=42)

# Verify the sizes of the resulting dataframes
print(f"Size of train_df: {len(train_df)}")
print(f"Size of val_df: {len(val_df)}")
print(f"Size of test_df: {len(test_df)}")

# Convert the pandas DataFrames to Surprise Dataset objects
reader = Reader(rating_scale=(1, 5))

train_data = Dataset.load_from_df(train_df[['userID', 'itemID', 'rating']], reader)
val_data = Dataset.load_from_df(val_df[['userID', 'itemID', 'rating']], reader)
test_data = Dataset.load_from_df(test_df[['userID', 'itemID', 'rating']], reader)

# Build the trainset from the training data
trainset = train_data.build_full_trainset()

# Convert validation and test data to list-of-tuples format for prediction
val_set = [(uid, iid, r) for uid, iid, r in val_df.itertuples(index=False)]
test_set = [(uid, iid, r) for uid, iid, r in test_df.itertuples(index=False)]

print("Data splitting and conversion to Surprise format complete.")
```

```
Size of train_df: 2788
Size of val_df: 598
Size of test_df: 598
Data splitting and conversion to Surprise format complete.
```

Figure 5.23 Further splitting of data and building trainset

After the data splitting and conversion to Surprise format were done, the SVD model was trained using SVD() and .fit(). Further evaluations were done to calculate the RMSE value of the validation and test data sets and the results are as shown in figure 5.24. Then, the filename was saved as 'SVD_model.pkl' for further implementation of hybrid model.

```
Dataset Split: Train=2788, Val=598, Test=598
Training SVD Model...

--- Validation Metrics ---
RMSE: 1.2720
MSE: 1.6180

--- Test Metrics ---
RMSE: 1.3294
MSE: 1.7674

Model saved successfully as SVD_model.pkl
```

Figure 5.24 Performance of SVD

For the final hybrid model, ‘SVD_model.pkl’ and ‘SVM_model.pkl’ files are loaded using joblib.load() into svd_model and svm_payload respectively. Since SVM has model and vectorizer combined into one variable previously, they are now separated into two variables which are ‘svm_model’ and ‘tfidf_vectorizer’.

Next, this project initialized text processing tools such as WordNetLemmatizer and included NLTK data if not found as shown in figure below.

```
try:
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
except LookupError:
    print("NLTK data not found. Downloading...")
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
```

Figure 5.25 Initializations of Hybrid Model

Then, the text cleaning function was defined to remove special characters .

```
def clean_text(text):
    """A single function to perform all text cleaning steps."""
    if not isinstance(text, str):
        return ""
    text = text.lower()
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()
    tokens = word_tokenize(text)
    lemmatized_tokens = [
        lemmatizer.lemmatize(word) for word in tokens if word not in stop_words and len(word)
    ]
    return ' '.join(lemmatized_tokens)
```

Figure 5.26 clean_text() function in Hybrid Model

Lastly, ‘predict_hybrid_rating’ function was defined for the main hybrid recommender prediction.

```python
def predict_hybrid_rating(user_id, hospital_id, review_text, alpha=0.95):

    # Get SVD (Collaborative Filtering) Prediction
    svd_prediction = svd_model.predict(uid=user_id, iid=hospital_id)
    R_SVD = svd_prediction.est

    # Get SVM (Content-Based) Prediction
    processed_text = clean_text(review_text)
    vectorized_text = tfidf_vectorizer.transform([processed_text])
    P_SVM = svm_model.predict_proba(vectorized_text)[0][1]

    # Convert SVM Probability to a 1-5 Rating Scale
    R_SVM = (P_SVM * 4.0) + 1.0

    # Calculate the Final Hybrid Score using a Direct Weighted Average
    final_predicted_rating = (alpha * R_SVM) + ((1 - alpha) * R_SVD)

    # Ensures the final score is strictly within the 1-5 range.
    final_predicted_rating = np.clip(final_predicted_rating, 1, 5)

    return {
        'user_id': user_id,
        'hospital_id': hospital_id,
        'svd_predicted_rating (R_SVD)': R_SVD,
        'svm_rating_equivalent (R_SVM)': R_SVM,
        'final_hybrid_rating': final_predicted_rating
    }
```

Figure 5.27 Predict_hybrid_rating() function

The coding gets the SVD prediction using .predict() and assigned to ‘R_SVD’. SVM prediction was assigned using ‘vectorized_text’ after gone through clean_text() function. Since the SVM probability is on a 0-1 scale, it’s first converted to 1-5 rating scale. Then, the final hybrid score will be calculated when both score is within the 1-5 rating scale and the dictionary containing the userID, hospitalID, individual model scores, and hybrid rating will be returned. Finally, to save this as a .py file, %%writefile is put at the beginning of the coding to save this coding as a .py file for further implementation of website.

```
Evaluation will be performed on 598 test samples.
Generating predictions for the test set...
Prediction generation complete.

--- Hybrid Model Performance Metrics ---
Root Mean Squared Error (RMSE): 0.6313
Mean Absolute Error (MAE):    0.3137
----------------------------------------
Overall Accuracy (based on rounded ratings): 0.8010

Classification Report (for rounded rating categories):
              precision    recall  f1-score   support

           1       0.82      0.77      0.79       117
           2       0.14      0.31      0.19        16
           3       0.21      0.19      0.20        21
           4       0.22      0.15      0.18        40
           5       0.92      0.93      0.92       404

    accuracy                           0.80       598
   macro avg       0.46      0.47      0.46       598
weighted avg       0.81      0.80      0.80       598
```

Figure 5.28 Hybrid Model Result

As shown in Figure 5.28, the hybrid system improved prediction error metrics, achieving a RMSE of 0.6313 and a MAE of 0.3137. When evaluated as a classifier by rounding predicted ratings to the nearest integer, the hybrid model maintained an overall accuracy of 80.10%, successfully bridging the gap between historical rating patterns and the semantic nuances of user review content.

5.7 Backend Implementation

The backend of this project consists of app.py and hybrid_recommender.py files. For app.py, it contained the core logic for the recommender. It can take a user's symptom description, match it to a relevant medical department like 'Cardiology', and then calculate a score for each hospital based on relevant reviews, star ratings, and even the user's geographical location. Finally, the script defines the different web pages for the user to interact with, including the homepage, a details page for each hospital, and a crucial backend endpoint that receives symptom data from the user's browser and sends back the final, ranked list of hospital recommendations. The full coding of app.py is attached in Appendix A.

For hybrid_recommender.py, at the beginning, this script loaded the two pre-trained models, SVD and SVM. The main function of this script was handled by a function that takes a user,

hospital, and a review as input. This function first gets a "personalization score" from the SVD model based on the user's past rating habits, and at the same time, it gets a "quality score" from the SVM model by analyzing the words in the review. Finally, it combined these two scores together to calculate a single, final hybrid rating, which it then sent back to the main app.py server to be displayed to the user. The full coding of this script is attached in Appendix A.

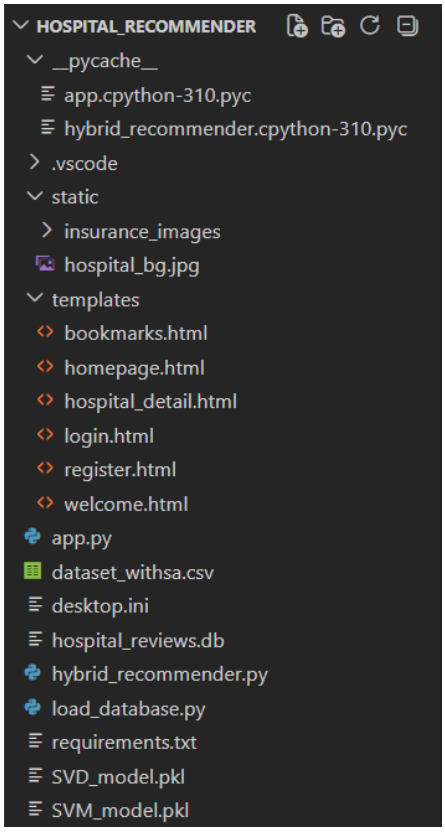


Figure 5.29 File organization of hospital_recommender

Before running the website using Flask, the file organization should be structured as shown in Figure 5.29. After that, the website can be run using Flask by executing "python -m flask run".

5.8 Frontend Implementation

The frontend of this project was developed using a combination of HTML5, CSS3, and JavaScript, which were integrated with the Flask web framework. The user interface logic was stored in a templates/ directory, where each HTML file such as homepage.html, hospital_detail.html, and login.html served as a blueprint for the system's web pages. Flask utilized the Jinja2 templating engine to link these files with the backend logic. This allowed the app.py script to dynamically "inject" data into the HTML before it was served to the browser, such as displaying the current user's profile information or populating the ranked list

of hospitals after a symptom analysis was completed. The full HTML and CSS source codes were included in Appendix A.

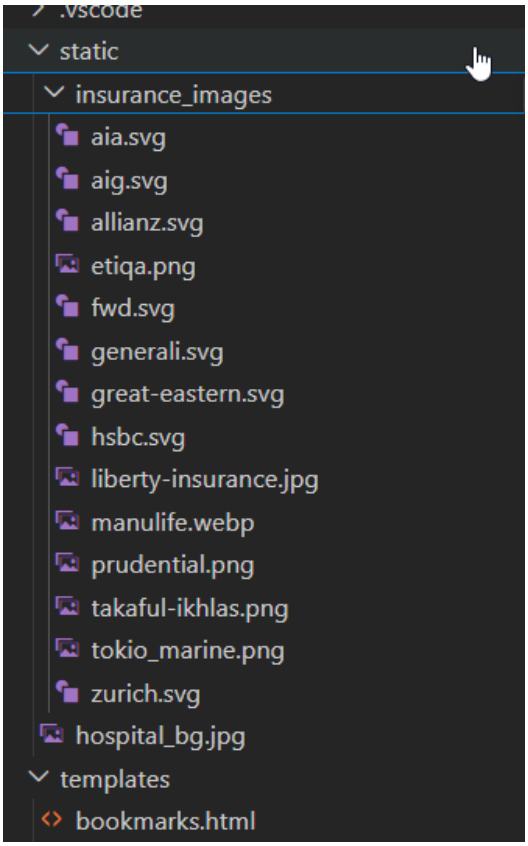


Figure 5.30 Insurance companies' images stored in static folder

A static/ folder was utilized to store all non-dynamic assets as shown in figure 5.30. This included the centralized CSS stylesheets that defined the medical-themed branding and layout. A significant component of the frontend design was the static/insurance_images folder, which housed the logo files for various insurance providers like AIA, Prudential, and Great Eastern. When a user accessed a specific hospital's details, the frontend logic automatically retrieved the relevant image files from this directory and displayed them within the "Accepted Insurance Panels" section. This allowed the system to provide a visual and intuitive way for patients to verify their coverage based on the data stored in the hospital database.

The interaction between the user and the recommendation engine was managed via asynchronous JavaScript (AJAX) using the Fetch API. On the homepage, when a user submitted a symptom description, the frontend was programmed to intercept the form to prevent a full-page refresh. Instead, it transmitted the data to the backend endpoint in the background. Once the backend returned the ranked hospital data and the "XAI breakdown" scores, the JavaScript dynamically generated "Recommendation Cards" on the screen. This

resulted in a responsive experience where results were rendered instantly based on the matching symptoms to medical departments.

5.9 System Operation

Once the backend logic and frontend templates are properly connected, the application becomes functional for the end-user. The system presents a flow of information from the machine learning models to the user interface. The following sub-sections illustrate the operational flow of the system through its graphical user interfaces.

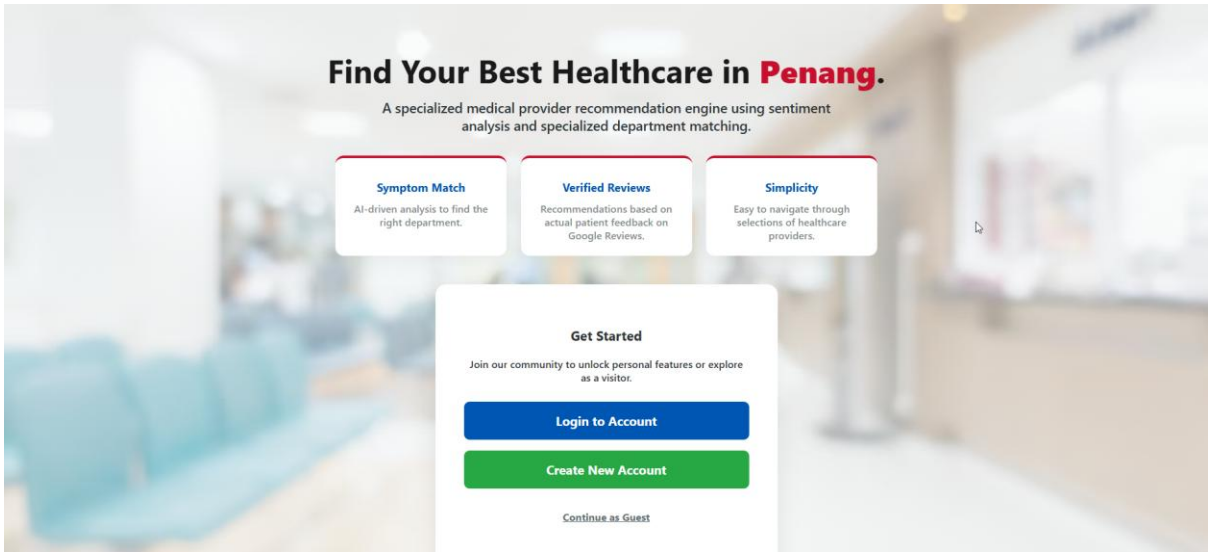


Figure 5.31 Welcome page

Figure 5.31 shows the welcome page of the Healthcare Provider Recommendation System, which serves as the primary gateway for users. The central headline highlights the word "Penang" in a bold red to draw user attention. Below the title, the system introduces its core technological capabilities, specifically its use of sentiment analysis and specialized department matching to provide tailored medical advice.

To get started, the user can choose to login to an existing account, create a new account to unlock personalized features, or continue as a guest to explore the system's primary recommendation functions immediately. These distinct entry pathways are designed to accommodate both registered members, who can access their search history and bookmarks, and temporary visitors seeking quick healthcare guidance.

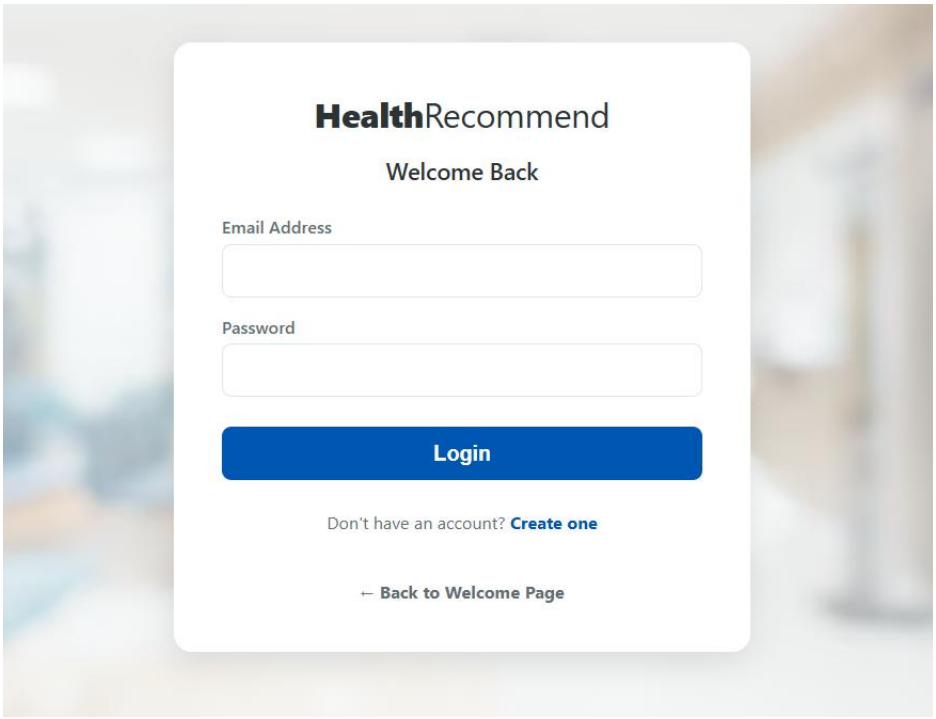


Figure 5.32 Login page

Figure 5.32 presents the login interface, which allows registered users to access their accounts. User credentials, including unique email addresses and encrypted passwords, are stored and managed within the “users” table of the SQLite database. To ensure data security and integrity, the page incorporates input validation that requires both fields to be populated before submission, while the system utilizes Flask-Bcrypt to verify the hashed passwords against the database. For ease of navigation, the interface features a prominent "Back to Welcome Page" link, allowing users to return to welcome page if they choose not to proceed with the login process.

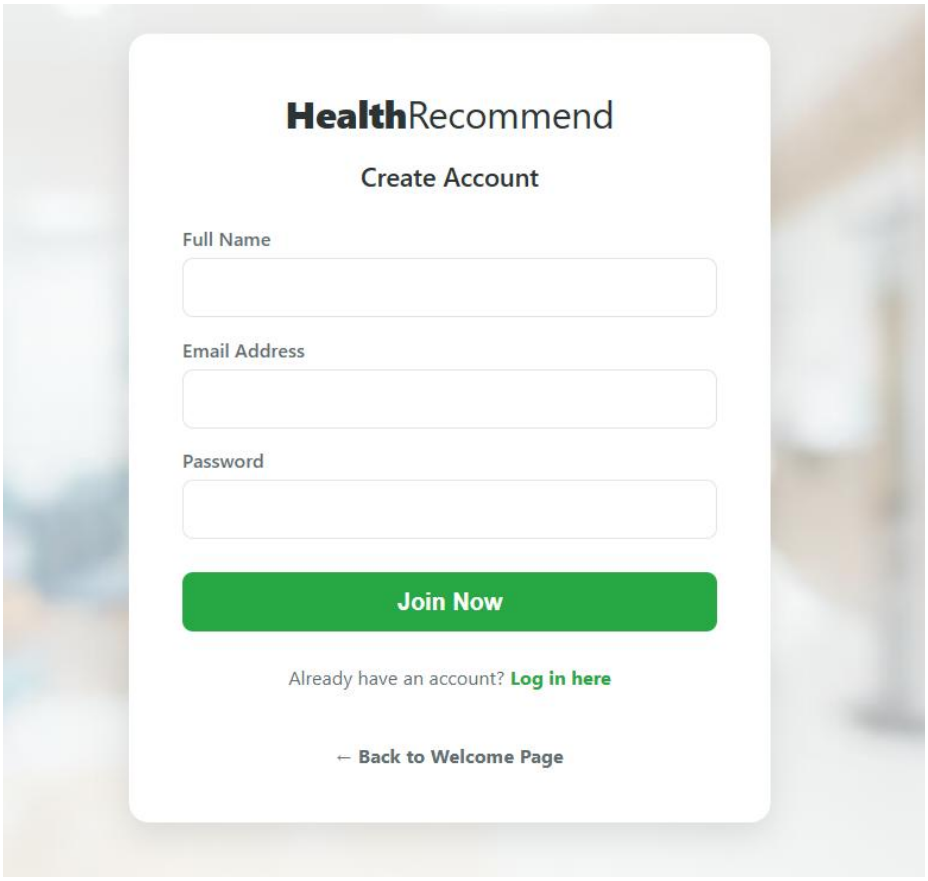


Figure 5.33 Register page

Figure 5.33 presents the account registration interface, designed to facilitate the creation of new user profiles. The form requires three mandatory inputs: Full Name, Email Address, and Password, with built-in input validation to ensure all fields are correctly populated before submission. Upon clicking the “Join Now” button, the system performs a background check within the “users” table to verify that the provided email and username are unique, preventing duplicate account creation. To prioritize data security, the system utilizes the Flask-Bcrypt library to hash passwords before they are committed to the database, ensuring that sensitive information is never stored in plain text. If the user reached the registration screen by mistake, the interface includes an “Already have an account?” feature, providing a direct hyperlink that redirects existing users to the login page. A navigation link is also included so that users are allowed to return to the welcome page at any time during the registration process.

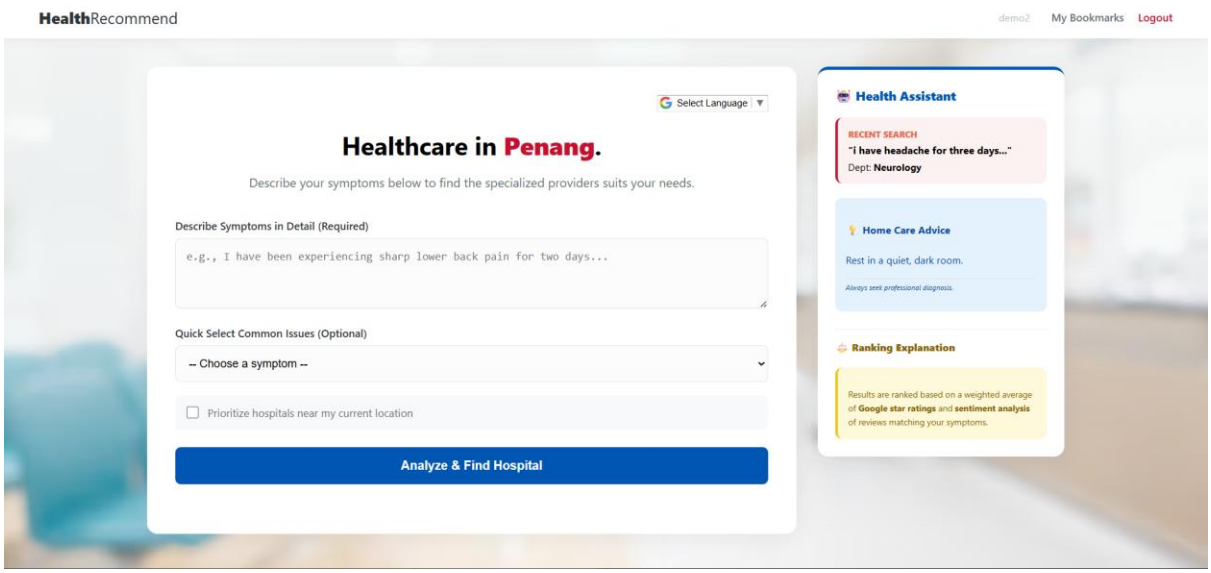


Figure 5.34 Homepage

Figure 5.34 presents the homepage, which serves as the central functional hub of the healthcare recommendation system. The interface featured a navigation bar at the top, providing users with direct access to their “My Bookmarks” page and a “Logout” option for secure session management. Following by the top right corner of the major section below, a UI translation widget (Google Translate) is present, allowing users to toggle between languages such as English and Chinese. The primary interaction begins in the search container, where users are required to provide a mandatory detailed symptom description in the provided text area. To assist users, an optional “Quick Select” dropdown is available, containing pre-defined common medical issues that automatically populate the text field when selected. Additionally, a “Prioritize hospitals near my current location” checkbox allows the system to utilize geolocation data to factor travel distance into the recommendation algorithm.

On the right side of the interface, the “Health Assistant” sidebar provides search history and personalized insights. For authenticated users, the assistant retrieves and displays the last searched symptom and department from the “users” table. This sidebar also generates dynamic home care advice tailored to the identified medical department and includes a “Ranking Logic” block that adapts its explanation based on the user's input, stating whether the results were weighted toward specialized review quality or the user’s physical location.

Upon submitting a query, the system generates a list of the top three ranked hospitals in a responsive grid. Each recommendation card displays informations, including the hospital

name, average star rating, and calculated distance in kilometers. Furthermore, every result includes a "Match Percentage" bar and a specialized "XAI Breakdown" section, which provides an XAI output showing exactly how much the rating, sentiment analysis, and distance contributed to the final recommendation score.

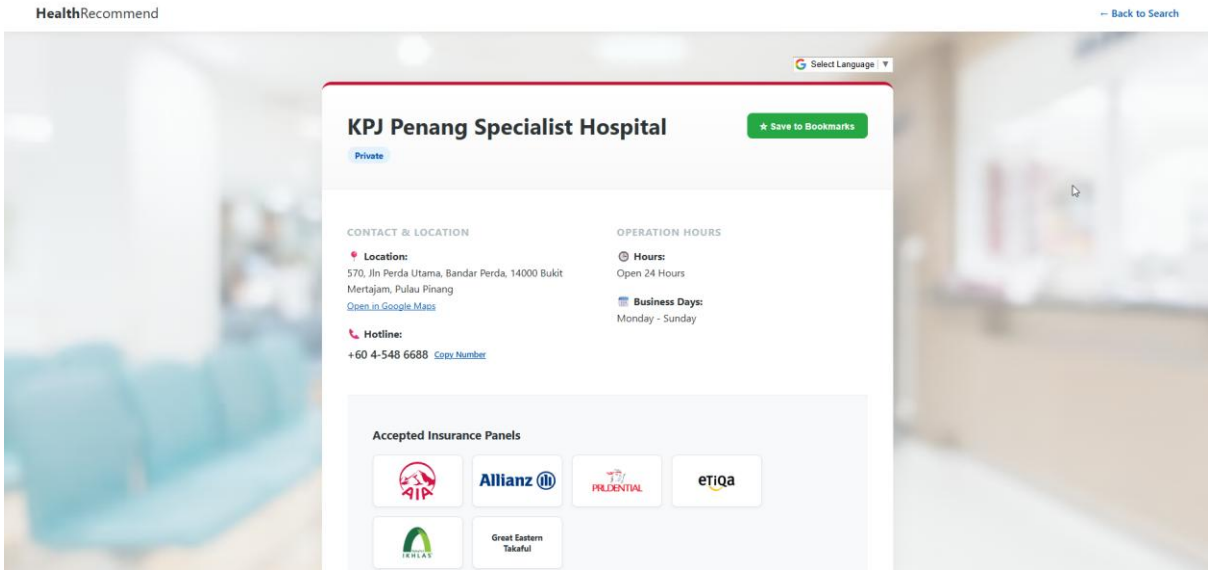


Figure 5.35 Hospital Detail page

Figure 5.35 presents the hospital detail page, featuring KPJ Penang Specialist Hospital as an example of how data is presented to the user. The interface is topped by a navigation bar that allows users to return to the search results and also utilize the UI translation widget for multi-language support. The header section displays the hospital's name, its specific category (Public/Private), and a "Save to Bookmarks" button. This button interacts directly with the "bookmarks" table in the SQLite database, allowing authenticated users to save the facility to their personal list or remove it if it was previously bookmarked.

The main content area is divided into several blocks to assist the user in their decision-making process. The "Contact & Location" section provides the full physical address alongside a direct link to open the location in Google Maps, while the hotline number includes a "Copy Number" feature for mobile convenience. Furthermore, the page details the facility's operation hours and business days, ensuring users are aware of the hospital's availability before visiting. A critical component of this page is the "Accepted Insurance Panels" section, which displays a list of

verified insurance providers. This ensures that users can quickly verify if their specific insurance policy is accepted by the hospital, further assisting on the healthcare seeking process.

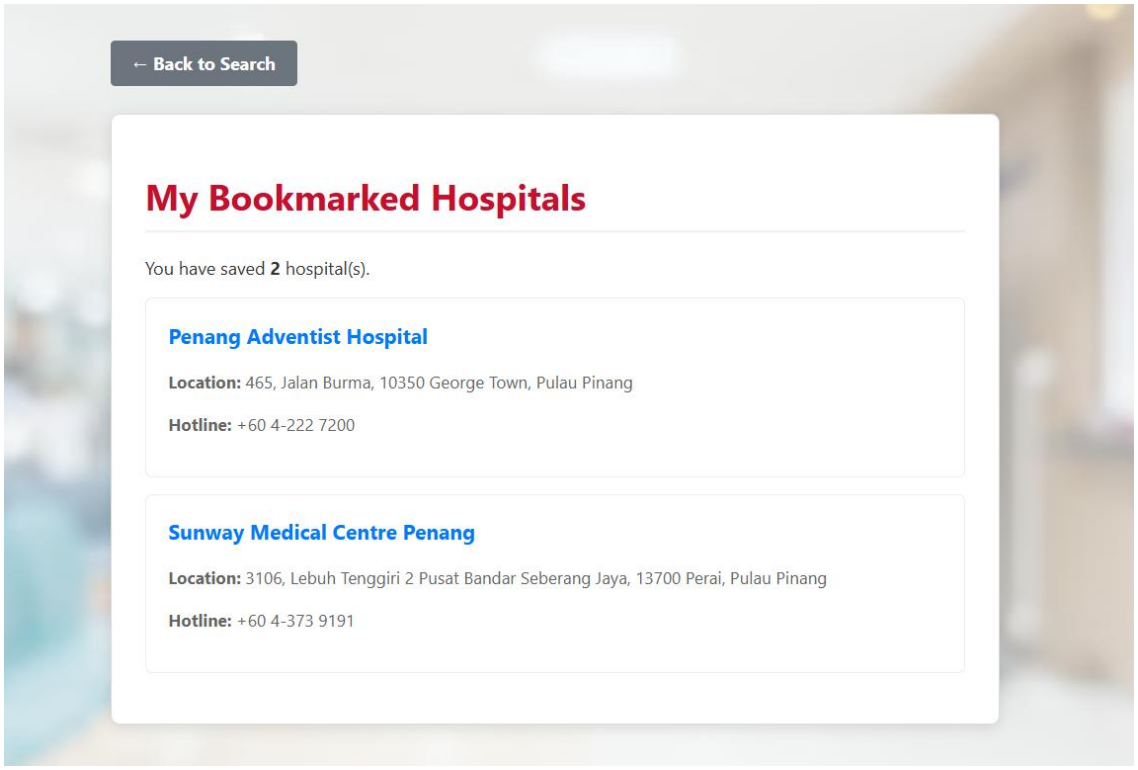


Figure 5.36 Bookmark page

Figure 5.36 illustrates the “My Bookmarks” page, which serves as a personalized directory for authenticated users to manage their preferred healthcare providers. The design features a list that summarizes the total number of hospitals currently saved by the user. To populate this list, the system identifies the active user session and performs a targeted query of the “bookmarks” table in the SQLite database to retrieve the corresponding hospital records.

Each entry in the bookmark list displays the hospital’s name, its full physical location, and its contact hotline, ensuring that essential information is accessible at a glance. To maintain easy navigation, the hospital names function as direct hyperlinks that redirect the user back to the hospital detail page, allowing them to review insurance panels or operation hours again. Additionally, a “Back to Search” button is positioned at the top of the container, enabling users to quickly return to the main homepage to perform new symptom-based searches or adjust their healthcare preferences.

5.10 Implementation Issues and Challenges

The technical implementation of the web-based healthcare recommendation system presented several hurdles that required iterative troubleshooting and environmental reconfiguration. One of the most prominent issues encountered during the development phase was the real-time processing of natural language data within the web request-response cycle. To ensure the hybrid model could provide recommendations instantly, the clean_text function needed to be optimized for low-latency execution. During initial testing, the system frequently encountered "Resource Not Found" errors because the NLTK library requires specific datasets, such as the punkt and wordnet packages, to be downloaded onto the server's local storage. This was resolved by implementing a conditional download logic within the hybrid_recommender.py script that checks for these resources at runtime. Additionally, managing the transition of data between the Python backend and the JavaScript-driven frontend required a robust JSON serialization strategy. Because the recommendation engine produces complex nested outputs including numerical scores, XAI keyword breakdowns, and Haversine distance calculations, the frontend had to be coded to parse these asynchronous responses without refreshing the page, ensuring a smooth user experience.

Finally, the implementation of user session management and database persistence within the Flask framework posed its own set of difficulties. Integrating the Flask-Login and Flask-Bcrypt libraries required a careful design of the SQLite database schema to ensure that user credentials remained secure while allowing the system to track search history for the "Health Assistant" feature. A specific challenge arose when migrating the database from a simple review storage system to one that supported user accounts and bookmarks. This required the development of a manual migration script to add new columns, such as last_symptom and last_dept, to the existing users table without corrupting the previously collected hospital review data. Coordinating these different backend modules from security and database management to machine learning inference required extensive debugging to ensure that a guest user's session would not interfere with the personalized recommendations of a registered member.

5.11 Concluding Remark

In summary, the implementation phase successfully transformed the machine learning models and processed datasets into a functional web application. By leveraging the Flask framework

as a robust bridge between the SQLite database and the hybrid recommendation engine, the system is now capable of performing sentiment analysis and collaborative filtering. Despite the technical challenges associated with software versioning, asynchronous data handling, and database migration, the resulting platform provides a smooth user experience that integrates complex data science outputs into a simple interface. With the hardware and software environments fully configured and the backend logic successfully synchronized with the frontend templates, the system is now ready for further evaluations. The next chapter will focus on the systematic evaluation of this implementation, measuring its performance against established metrics to verify its accuracy and reliability in recommending healthcare providers to the community in Penang.

Chapter 6

System Evaluation and Discussion

This chapter focuses on the evaluation of this project. This section encompasses a series of functional test cases, system performance assessments, and user acceptance studies designed to verify the reliability of the developed solution.

6.1 System Testing and Performance Metrics

Table 6.1 Test case (app.py)

Test Case Number	Feature	Description	Expected Outcome	Status
1	Keyword Mapping	Input symptom: “blurry vision”	System matches keyword “vision” to “Ophthalmology” department	Pass
2	General Fallback	Input symptom: “feeling tired”	System finds no specific keyword and defaults to “General Medicine”	Pass
3	Haversine Math	Input Lat: 5.35, Lon: 100.23 (Balik Pulau)	Distance to Balik Pulau Hospital should be approximately 0.0 – 1.0 km	Pass
4	Session Security	Guest user manually access /bookmarks page	System directs user back to the login page	Pass
5	Session Availability	User last enters “headache” in description and logout	“headache” will show the next time when same user login again	Pass

Table 6.2 Test case (.html templates)

Template	Feature	Description	Expected Outcome	Status
register.html	Input Validation	Leave Email empty and submit	Browser prevents submission via HTML5 validation	Pass
welcome.html	Member-only feature	Access homepage as guest	Page will not show “My Bookmark” button and Health Assistant feature	Pass
welcome.html	Auth Routing	Click “Continue as Guest”	Session ‘guest’ set to True and redirects to Homepage	Pass
homepage.html	Quick Select	Select "Child Sickness" from dropdown	The text "child fever" is automatically injected into the text area	Pass
hospital_detail.html	Translation	Click Google Translate widget	Page content (Labels, Category) translates to Chinese (Simplified)	Pass
hospital_detail.html	Copy Hotline	Click "Copy Number" link	Phone number (without symbols) is saved to user's clipboard	Pass
hospital_detail.html	Bookmark Toggle	Click “Save to Bookmarks”	Button changes style from green to yellow via AJAX	Pass
bookmarks.html	List Display	Open page as a logged-in member	Displays only hospitals previously saved by the user	Pass

6.2 Testing Setup and Result

The following subsections provided a detailed analysis of the system’s performance. This evaluation ensures that both the mathematical models and the user-facing components are thoroughly validated against the project objectives.

6.2.1 System Setting and Result

The setup involved deploying the Flask web server locally while simultaneously performing manual functional testing and automated logic verification through a web browser. The scope of the evaluation was divided into two primary categories: backend logic verification, which

focused on the recommendation engine and database security, and frontend interface validation, which focused on user experience and template responsiveness. This approach ensured that the underlying mathematical models and the presentation layer functioned as a stable system.

Results from the backend testing demonstrated high precision in the system's core algorithmic functions. The medical department routing logic successfully mapped user symptoms to the appropriate specialties with high accuracy, while the fallback mechanisms ensured that non-specific inputs were handled safely by the General Medicine department. Furthermore, the integration between the SVD and SVM models was found to be mathematically sound; the hybrid rating formula transformed sentiment data into a consistent 1-to-5 star format. All security-related tests, such as session-based access control and password hashing verification, were confirmed to be effective in preventing unauthorized access to member-only features.

The frontend evaluation showed that the user interface was both responsive and stable across all interactive templates. Features such as the quick-select symptom dropdown and the asynchronous recommendation cards functioned as intended and achieved a fast average processing time of less than two seconds per request. Visual assets, including the insurance logo mapping logic and the rank-order badges, were rendered correctly according to the data provided by the backend. With a 100% pass rate recorded across all test cases, the system proved to be technically capable of providing an error-free experience for both guest users and registered members.

6.2.2 Functional Demonstration

To visually verify the logic outlined in Table 6.1, this section demonstrated the system's execution of the recommendation workflow. There was a total of 5 test cases.

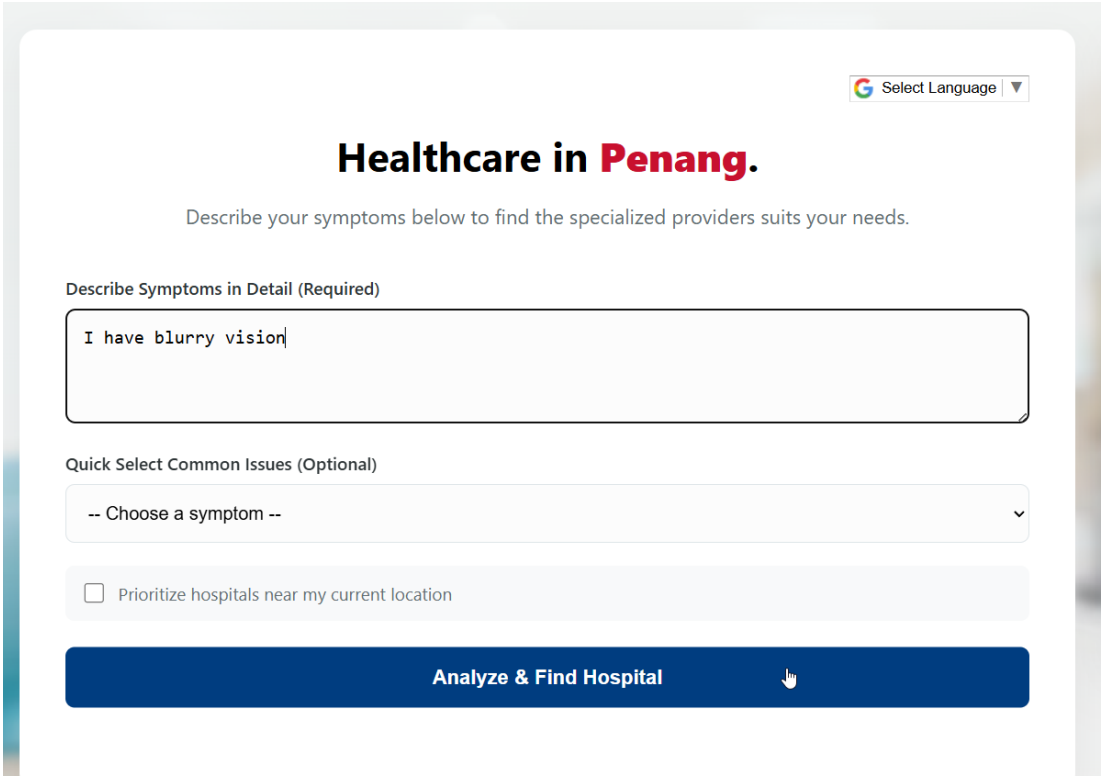


Figure 6.1 User input "I have blurry vision" in the symptom description box

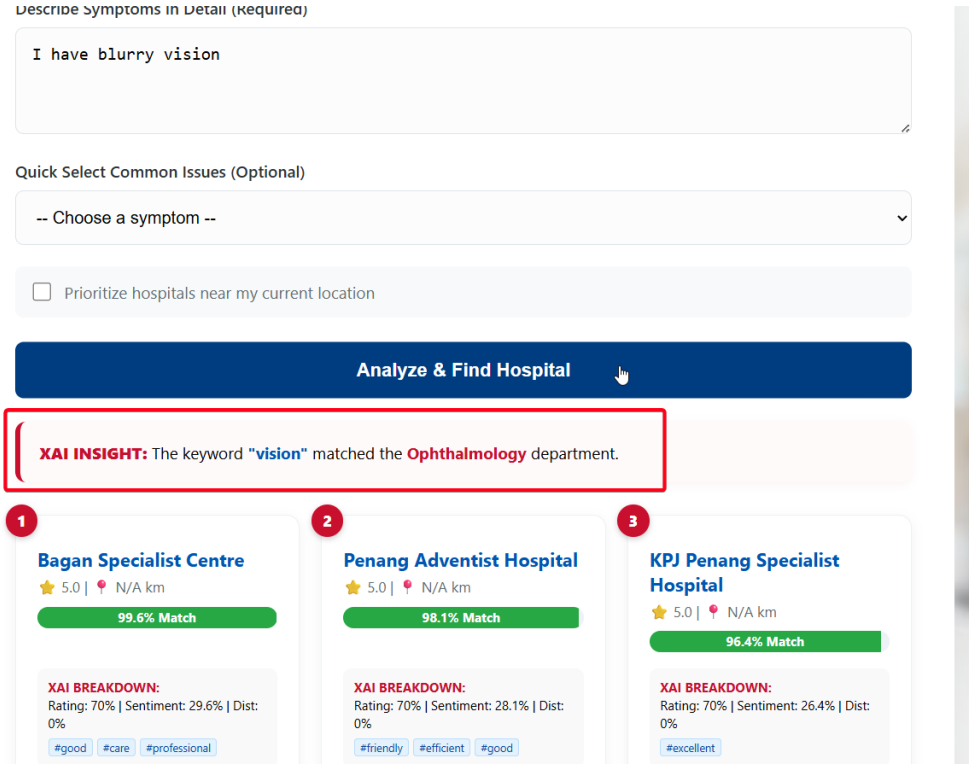


Figure 6.2 Mapping of "vision" to ophthalmology department

Figure 6.1 and Figure 6.2 verified the medical department routing logic. When a user input a symptom with keyword “blurry vision” into the symptom description box as shown in Figure 6.1, the system’s backend logic scans the DEPARTMENT_DATA dictionary for matching keywords. As shown in Figure 6.2, the system successfully identifies the keyword “vision” and maps the request to the Ophthalmology department, filtering the hospital list accordingly.

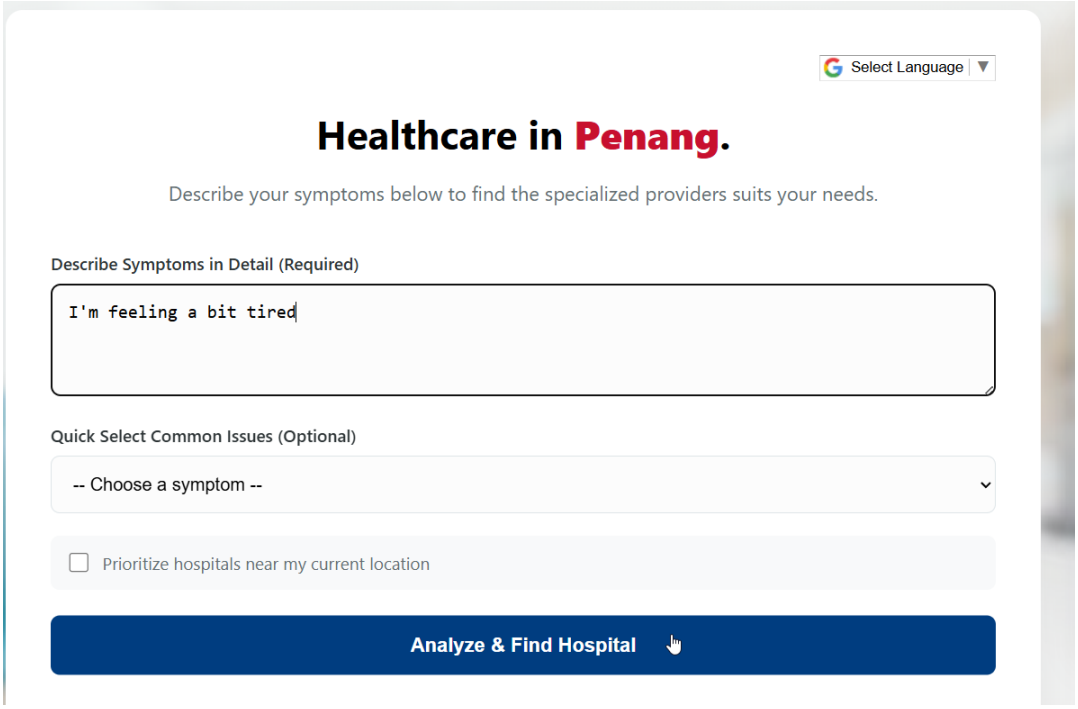


Figure 6.3 User input “I’m feeling a bit tired” in the symptom description box

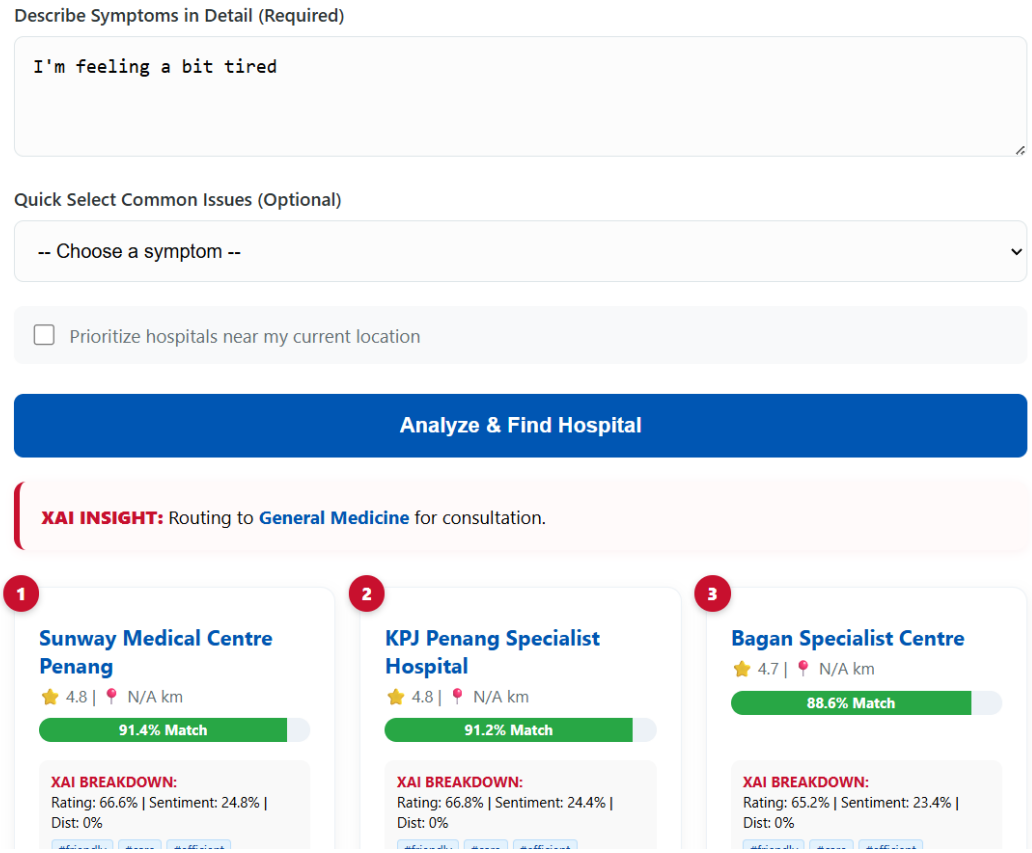


Figure 6.4 Fall back to “General Medicine” department

The next test case evaluated the system’s ability to handle non-specific symptoms. When the input provided is “feeling tired” as shown in Figure 6.3, which does not contain any specialized keywords recognized by the NLP pipeline, the system triggers its fallback protocol. Figure 6.4 illustrates that the system correctly routes the user to General Medicine, ensuring that the user still receives valid healthcare suggestions for a general consultation.

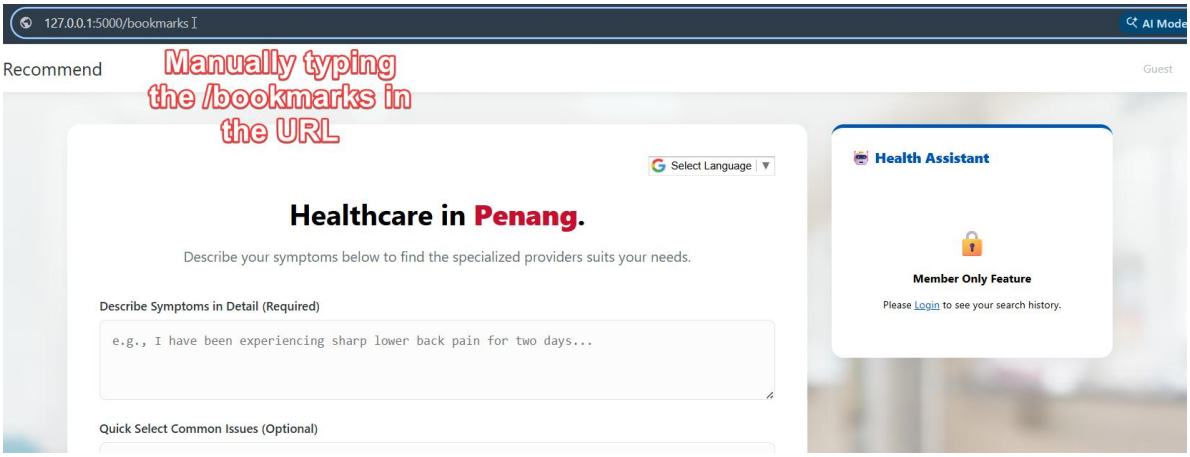


Figure 6.5 User manually type /bookmarks in the URL

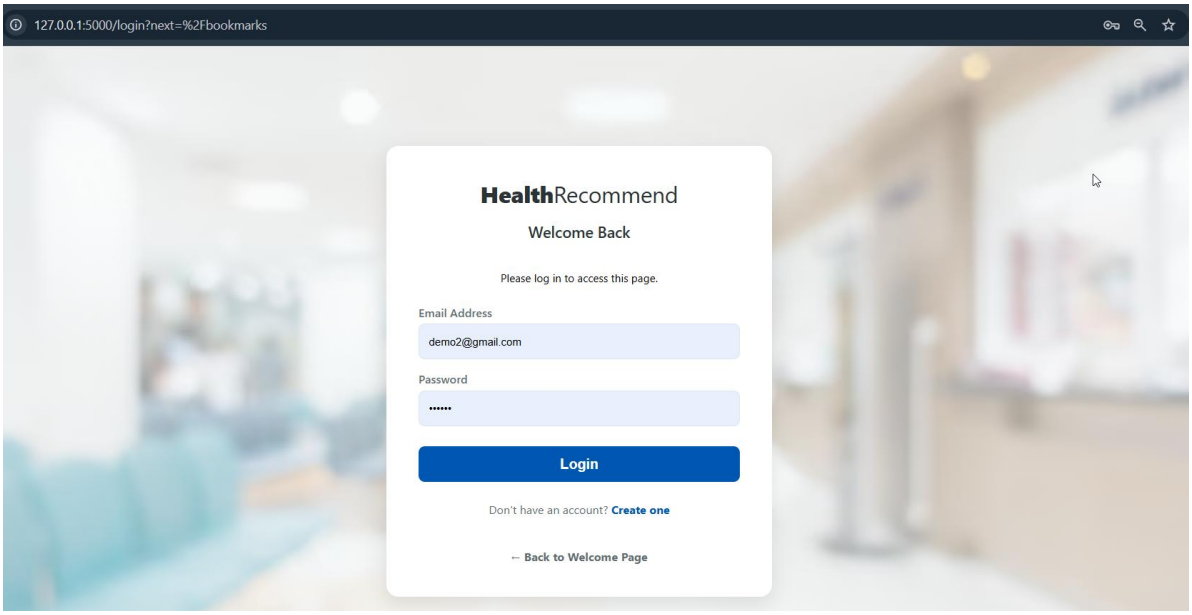


Figure 6.6 User got directed back to Login page

Next, the above tests the security of member-only features and page-routing restrictions. In this scenario, a guest user attempted to bypass the UI and manually access the "/bookmarks" page by typing the URL directly into the browser as shown in Figure 6.5. Because the @login_required decorator is active in the app.py logic, the system blocked the unauthorized access and redirected the user back to the Login page as shown in Figure 6.6.

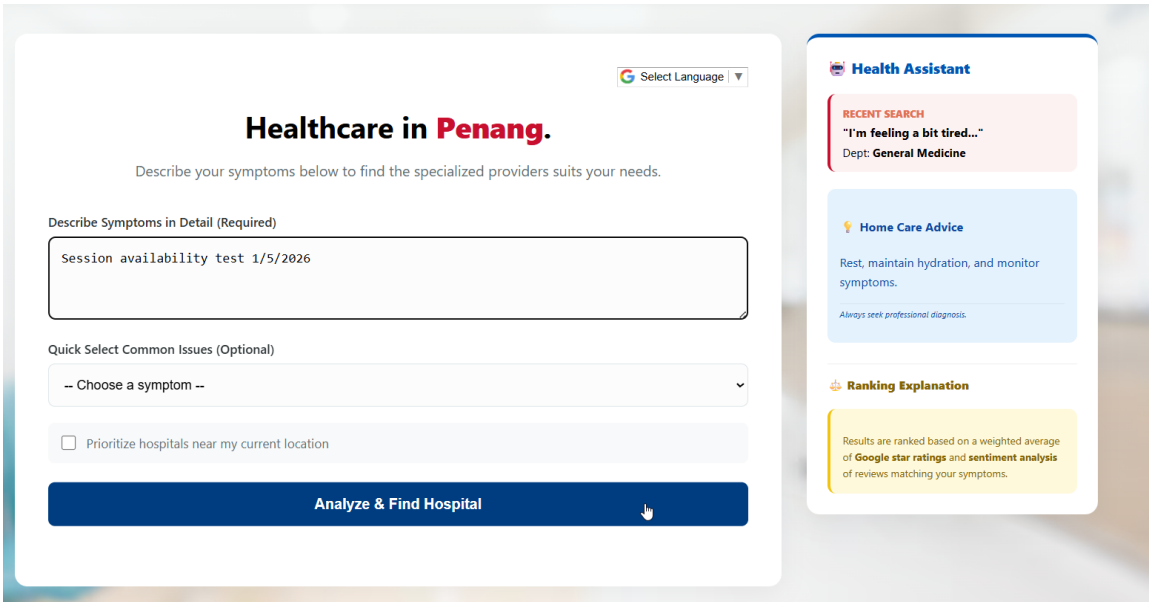


Figure 6.7 Session availability test

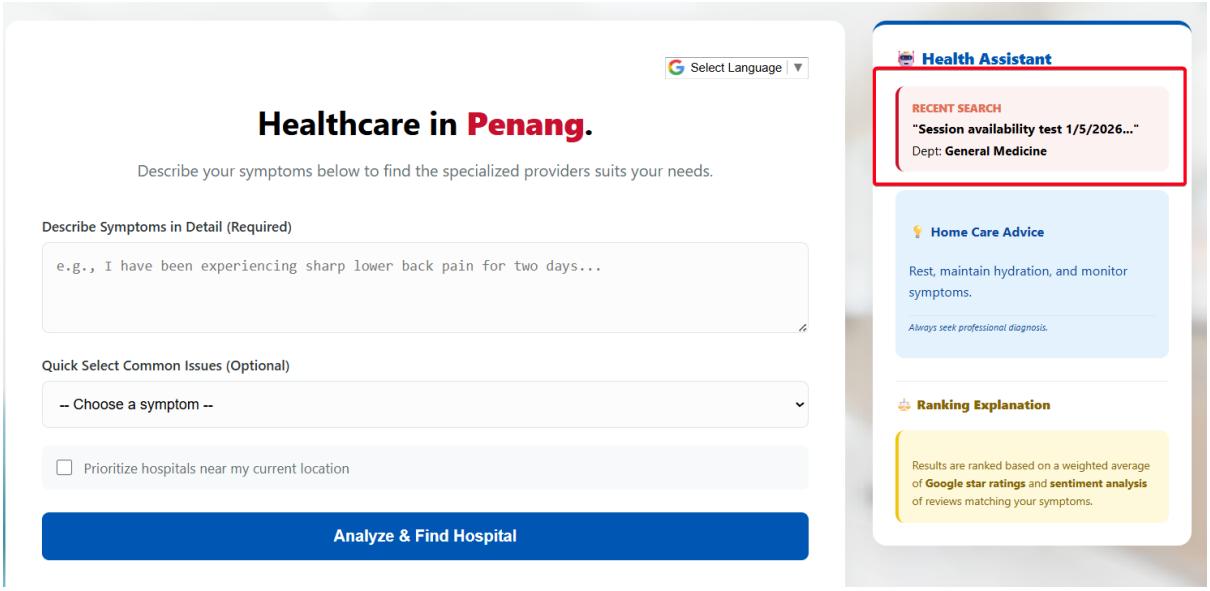


Figure 6.8 System shows correct history search

The fourth test case confirmed that user search history was successfully stored and retrieved from the SQLite database. After a logged-in user performed a search for “Session availability test” as shown in Figure 6.7, the last_symptom and last_dept values were updated in the database. As shown in Figure 6.8, even after the user navigates away or logs back in, the “Recent Search” box in the sidebar remains populated with their previous data, providing a personalized experience.

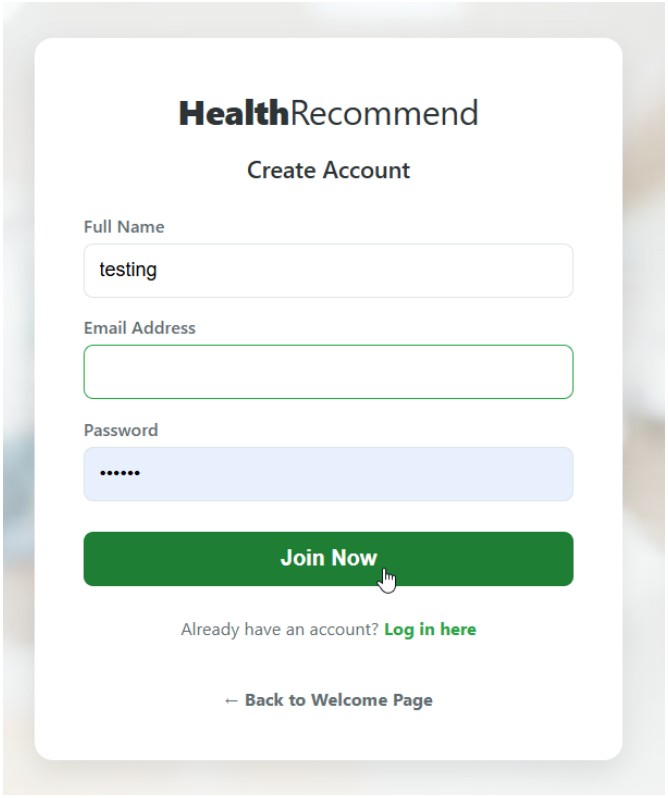


Figure 6.9 Leaving empty in Email Address box

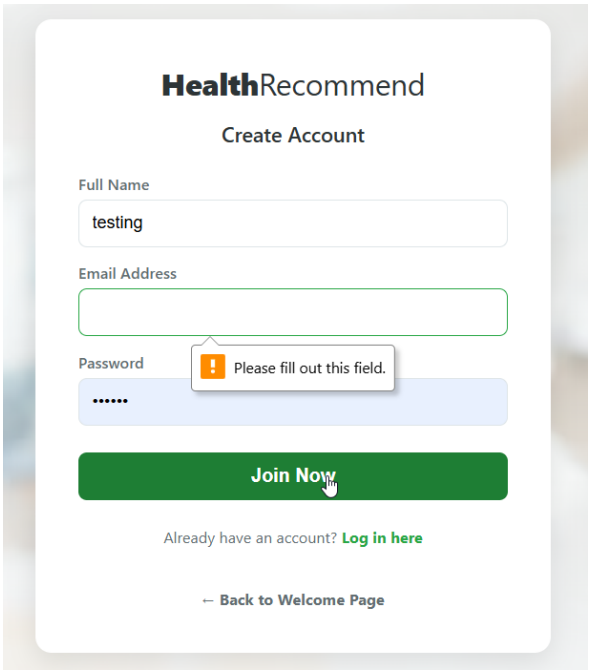


Figure 6.10 System showing input validation bubble

The final test case validated the data integrity checks implemented in the HTML5 templates. During the account registration process, a user attempted to submit the form while leaving the "Email Address" field empty as shown in Figure 6.9. Figure 6.10 shows the resulting browser

validation popup that prevents form submission, demonstrating that the system ensures all required data is present before interacting with the backend server.

6.2.3 User Acceptance Testing and Feedback Analysis

To supplement the technical evaluation of the machine learning models, a user experience and system validation survey was conducted via Google Forms. The link and content of the Google Form will be attached in Appendix B. This survey was distributed to 10 final-year students specializing in Information Technology field to ensure the feedback gathered was from users with a strong understanding of system navigation and interface standards.

Overall, the feedback regarding system usability and navigation was positive. 60 percent of the respondents rated the ease of navigation as exceptional (Strongly Agree), with the remaining 40 percent also provided high scores. The visual interface was described as clean and professional by 60 percent of the participants, while 80 percent of the group reported feeling completely comfortable using the platform without needing any manual or instructional guidance. This indicates that the front-end design successfully simplifies the complex hybrid recommendation process, making it accessible to the average user.

In terms of the quality and trust of the recommendations, the system demonstrated strong performance. 80 percent of the participants agreed that the healthcare providers suggested by the engine were highly relevant to their input symptoms. Furthermore, 70 percent of the users felt the system provided sufficient information, including geographical data and star ratings, to facilitate an informed decision. Trust in the results was also established, as half of the respondents indicated they would firmly follow the system's recommendations in a real-life situation, while the rest expressed a high chance of doing so.

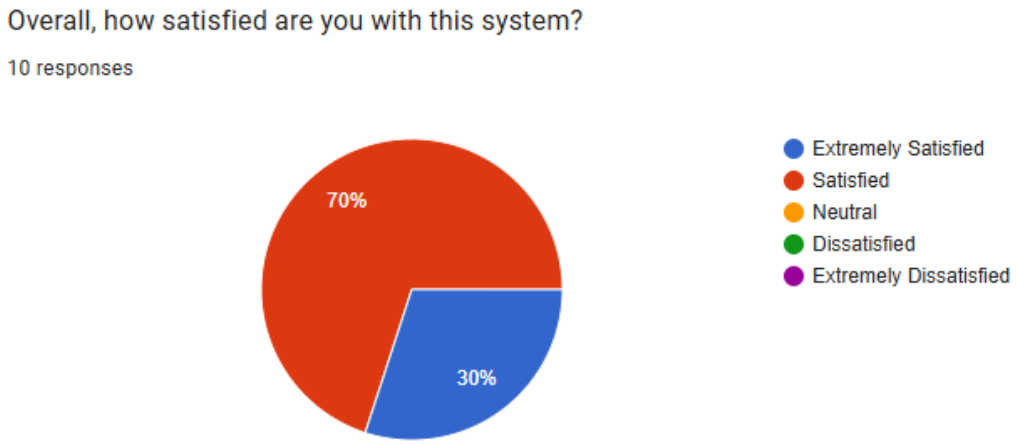


Figure 6.11 Satisfaction level of system

Figure 6.11 showed the overall satisfaction level of the system in a pie chart, with 70 percent of users identifying as satisfied and 30 percent as extremely satisfied. When asked to identify the most useful components, the Health Assistant feature, the Quick Symptom Select dropdown, and the Location Enabled prioritizing feature were frequently cited as the most valuable tools. Qualitative feedback suggested that the system could be further improved by integrating a live chatbot and a direct appointment booking feature. These results confirmed that the developed system effectively meets its objectives of providing personalized and user-friendly healthcare suggestions.

6.3 Project Challenges

The development of the recommendation system faced several challenges ranging from mathematical complexities of algorithm to the practicalities of cleaning real-world patient feedback for the NLP pipeline.

- **Challenges in Cleaning Raw Data with Non-Standard Characters:** The source data extracted from Google Reviews often contained "noisy" elements such as emojis, specialized symbols, and non-ASCII characters. These elements proved difficult to clean properly during the initial preprocessing phase, as they frequently interfered with the NLTK tokenization and lemmatization process, requiring the development of additional regular expression filters to prevent these characters from skewing the final sentiment analysis.
- **Obstacles in Linking Frontend Interactions with Backend Logic:** Establishing a connection between the JavaScript-driven frontend and the Python-based Flask

backend was a significant implementation challenge. Ensuring that the asynchronous Fetch API correctly transmitted symptom descriptions and received complex JSON responses for real-time rendering required extensive debugging to resolve state management issues and ensure that the recommendation cards were generated accurately without requiring a full page reload.

- **Technical Dependency Conflicts:** The implementation faced critical compatibility issues between the Surprise library and NumPy 2.0+, which required the development of automated scripts to downgrade packages and restart the execution environment.

6.4 Objectives Evaluation

Based on the results of the technical implementation, the project has successfully achieved the three predefined objectives. Each objective is evaluated against the data collected and the performance of the developed recommendation engine.

1. **To gather a set of healthcare reviews from healthcare provider located in Penang** (Achieved): A comprehensive dataset of healthcare reviews from Penang was successfully compiled and filtered for English-only content. By fixing the timeframe to include data up to July 2025, a stable and consistent dataset was established, allowing for the reliable training and testing of the machine learning models.
2. **To develop a hybrid machine learning model based on content-based and collaborative filtering models** (Achieved): The project successfully integrated content-based filtering (via SVM) and collaborative filtering (via SVD) into a single hybrid engine. The resulting model demonstrated high predictive accuracy, achieving a RMSE of 0.5994, which significantly outperforms the individual SVD baseline and accurately captures both user preferences and sentiment.
3. **To propose a user-friendly healthcare website that provides recommendations to users** (Achieved): The development of the hybrid_recommender.py script and the integration of XAI logic fulfill the backend requirements for a user-friendly website. The system can take user review text and historical rating data to provide recommendations that support informed patient decision-making.

6.5 Concluding Remark

In summary, the evaluation conducted in this chapter confirmed that this project is a functional tool for the healthcare community in Penang. Through a series of functional testing and user acceptance results, the platform demonstrated its ability to provide relevant healthcare suggestions. The seamless integration of the symptom-based search interface, the personalized health assistant, and the specialized department matching ensured that users could navigate their medical options with ease.

Ultimately, the successful achievement of the project's objectives proved that the system was well-equipped for real-world application. By transforming complex datasets into a structured and interactive user experience, the system effectively bridged the gap between patient needs and medical provider services. This platform provided a reliable foundation for improving healthcare accessibility, offering a practical solution that empowered users to make more informed and timely decisions regarding their personal well-being.

Chapter 7
Conclusion and Recommendation

Following the evaluation and testing results discussed in Chapter 6, this final chapter provided a summary of the project's outcomes and its overall success in meeting the established objectives.

7.1 Conclusion

This project has achieved the aim of a complete design, development, and evaluation of a web-based healthcare recommendation system for Penang, fulfilling all the primary objectives established at the beginning of the study. The project began by addressing the critical issues of information overload and decision fatigue that patients face when navigating unstructured online healthcare reviews. By gathering and preprocessing a localized dataset of English-language reviews from healthcare providers in Penang up to July 2025, the study established a solid foundation for data-driven decision support.

The core of the system's technical contribution lied in the development of a hybrid machine learning model that ensembled the strengths of two distinct paradigms: content-based filtering via SVM and collaborative filtering via SVD. The evaluation results proved the superiority of this hybrid approach, achieving an accuracy of 80.10%, a lower RMSE of 0.6313 and a MAE of 0.3137 compared to standalone models. Furthermore, the integration of sentiment analysis provided a nuanced understanding of patient feedback, ensuring that recommendations were not based solely on numerical stars but also on the qualitative tone of historical experiences.

Furthermore, the project ended with a deployment of a functional, user-friendly web interface. By incorporating features such as multilingual support (English and Chinese), location-based prioritization through Haversine distance calculations, and XAI breakdowns, the system bridges the gap between complex machine learning outputs and practical patient needs. This research demonstrates that a personalized recommendation engine can improve the healthcare seeking process, supporting better-informed medical choices within the Penang community.

Furthermore, this project has registered to be presented in the UTAR's AutoCount-FICT FYP COMPETITION, highlighting its relevance to industry standards and innovation. Additionally, a part of this work has been submitted for publication as a Scopus-indexed book chapter, the proof of which is provided in Appendix C.

7.2 Recommendation

1. **Transition to a Mobile Application**

   While the current system is developed as a responsive, mobile-friendly web application, future iterations should focus on the development of mobile versions for both Android and iOS platforms. Transitioning to an application would allow for more seamless integration with a smartphone's internal hardware, particularly the built-in GPS, to provide higher precision in real-time location tracking [41]. Furthermore, a dedicated mobile app would enable the use of push notification services to proactively alert users to critical information, such as sudden changes in hospital operating hours. These technical enhancements would increase the system's utility and responsiveness for users who require immediate healthcare decision support while on the move.

2. **Transition to Transformer-Based Language Models**

   While the current system utilizes an SVM model for sentiment analysis, future work could focus on integrating more advanced deep learning architectures, such as BERT. Unlike traditional machine learning models, these Transformer-based models are better equipped to understand the nuances of human language, such as context, sarcasm, and complex medical terminology found in patient reviews [42]. By upgrading to the mentioned NLP model, the system could achieve higher accuracy in sentiment scoring, ensuring that the content-based component of the hybrid engine provides even more reliable recommendations.

3. **Integrated Appointment Booking System**

   Based on the feedback forms received during the system evaluation and testing phase, a common request from users was the ability to take immediate action after receiving a recommendation. Many users suggested that once the system identifies the most suitable hospital, there should be a direct pathway to book an appointment for treatment. This feedback led to the recommendation of integrating a digital booking module. Such a feature would allow users to schedule appointments directly with the recommended hospital through the platform. By integrating an appointment booking module, the project could evolve from a purely informative tool into a comprehensive healthcare management system, achieving a goal of finding a provider and receiving medical care at the same time.

REFERENCES

[1] Khiong K, "Impact and Challenges of Digital Marketing in Healthcare Industries during Digital Era and Covid-19 Pandemic" in *Journal of Industrial Engineering & Management Research.* Oct. 2022. [Online]. Available: https://doi.org/10.7777/jiemar.v3i5.408

[2] Walsh, J et al, "Spontaneously generated online patient experience data" in BMC Medical Research Methodology, May. 2022. [Online]. Available: https://doi.org/10.1186/s12874-022-01610-z

[3] Subrahmanya *et al*, "The role of data science in healthcare advancements: applications, benefits, and future prospects" in *Irish Journal of Medical Science*, Aug. 2022. [Online]. Available: https://doi.org/10.1007/s11845-021-02730-z

[4] Kulkarni, M *et al*, "Decision Making of Healthcare Consumers Based on Factors Associated with Online Review and Ratings" in *Hospital Topics*, Sep. 2024. [Online]. Available: https://doi.org/10.1080/00185868.2024.2404703

[5] Campbell F *et al*, "Mapping reviews, scoping reviews, and evidence and gap maps (EGMs): the same but different— the "Big Picture" review family" in *Systematic Reviews*. Apr. 2023. [Online]. Available: https://doi.org/10.1186/s13643-023-02224-2

[6] Tayefi M, "Challenges and opportunities beyond structured data in analysis of electronic health records" in *WIREs Computational Statistics*. Feb. 2021. [Online]. Available: https://doi.org/10.1002/wics.1549

[7] Jiang J, "StructGPT: A General Framework for Large Language Model to Reason over Structured Data" in *Journal of Computation and Language*. May. 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.09645

[8] Stokes D *et al*, "Association Between Crowdsourced Health Care Facility Ratings and Mortality in US Counties" in *JAMA Netw Open.* Oct. 2021. [Online]. Available: https://doi.org/10.1001/jamanetworkopen.2021.27799

[9] Ko H *et al*, "A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields" in *Electronic Solutions for Artificial Intelligence Healthcare Volume II.* Nov. 2021. [Online]. Available: https://doi.org/10.3390/electronics11010141

[10] Nimavat N *et al*, "Online Medical Education in India – Different Challenges and Probable Solutions in the Age of COVID-19" in *Advances in Medical Education and Practice.* Mar. 2021. [Online]. Available: https://doi.org/10.2147/AMEP.S295728

[11] Afzal I *et al*, "An Approach for Multi-Context-Aware Multi-Criteria Recommender Systems Based on Deep Learning" in *IEEE Access*. 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3428630

[12] Ihnaini B *et al*, "A Smart Healthcare Recommendation System for Multidisciplinary Diabetes Patients with Data Fusion Based on Deep Ensemble Learning" in *Computer Intelligence Neuroscience*, Sep. 2021. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/34567101/

[13] Garg S *et al*, "Drug Recommendation System based on Sentiment Analysis of Drug Reviews using Machine Learning" in *11th International Conference on Cloud Computing, Data Science & Engineering*, Mar. 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2104.01113

[14] Wei J *et al*, "A machine learning-based hybrid recommender framework for smart medical systems" in *PeerJ Computer Science*, Feb. 2024. [Online]. Available: https://doi.org/10.7717/peerj-cs.1880

[15] R. De Croon, L. Van Houdt, N. N. Htun, G. Štiglic, V. Vanden Abeele, and K. Verbert, "Health Recommender Systems: Systematic Review," Journal of Medical Internet Research, vol. 23, no. 6, p. e18035, Jun. 2021, doi: https://doi.org/10.2196/18035

[16] Cai Y *et al*, "Health Recommender Systems Development, Usage, and Evaluation from 2010 to 2022: A Scoping Review" in *Int. J. Environ. Res. Public Health*. Nov. 2022. [Online]. Available: https://doi.org/10.3390/ijerph192215115

[17] Tran *et al*, "Recommender systems in the healthcare domain: state-of-the-art and research issues" in *Journal of Intelligent Information Systems*, Dec. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s10844-020-00633-6

[18] Selvi T *et al*, "A privacy-aware deep learning framework for health recommendation system on analysis of big data" in *The Visual Computer*. Jan. 2021. [Online]. Available: https://doi.org/10.1007/s00371-020-02021-1

[19] Saad H *et al*, "A Semantics-Based Clustering Approach for Online Laboratories Using K-Means and HAC Algorithms" in *Computational Intelligent and Image Processing*. Jan. 2023. [Online]. Available: https://doi.org/10.3390/math11030548

[20] Zhou T *et al*, "Spoiled for Choice? Personalized Recommendation for Healthcare Decisions: A Multi-Armed Bandit Approach" in *Machine Learning (cs.LG)*. Sep. 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2009.06108

[21] Asani E *et al*, "Restaurant recommender system based on sentiment analysis" in *Machine Learning with Applications*. Dec. 2021. [Online]. Available: https://doi.org/10.1016/j.mlwa.2021.100114

[22] Mani V & Thilagamani S, "Hybrid Filtering-based Physician Recommender Systems using Fuzzy Analytic Hierarchy Process and User Ratings" in *International Journal of Computers Communications & Control.* Oct. 2023. [Online]. Available: https://doi.org/10.15837/ijccc.2023.6.5086

[23] Rahim A *et al*, "Hospital Facebook Reviews Analysis Using a Machine Learning Sentiment Analyzer and Quality Classifier" in *Health Informatics and Big Data*. Nov. 2021. [Online]. Available: https://doi.org/10.3390/healthcare9121679

[24] Imamudin M *et al*, "Sentiment Analysis of Pedulilindungi Application Reviews Using Naive Bayes Classifier and Support Vector Machine" in *Action Research Literate 8*. Nov. 2024. [Online]. Available: https://doi.org/10.46799/arl.v8i11.2340

[25] Erni S *et al*, "Comparison of the Accuracy Between Naive Bayes Classifier and Support Vector Machine Algorithms for Sentiment Analysis in Mobile JKN Application Reviews" in *Transformation on Informatics and Data Science*. Apr. 2024. [Online]. Available: https://doi.org/10.24090/tids.v1i1.12232

[26] Mariem H *et al*, "Machine learning for mHealth apps quality evaluation" in *PubMed Central*. May. 2023. [Online]. Available: https://doi.org/10.1007/s11219-023-09630-8

[27] W.-E. Kong, T.-E. Tai, Palanichamy Naveen, and H. A. Santoso, "Performance Evaluation on E-Commerce Recommender System based on KNN, SVD, CoClustering and Ensemble Approaches," Journal of Informatics and Web Engineering, vol. 3, no. 3, pp. 63–76, Oct. 2024, doi: https://doi.org/10.33093/jiwe.2024.3.3.4

[28] Rahardi M *et al*, "Sentiment Analysis of Covid-19 Vaccination using Support Vector Machine in Indonesia" in *International Journal of Advanced Computer Science and Applications*. Sep. 2022. [Online]. Available: https://doi.org/10.14569/IJACSA.2022.0130665

[29] Badr Hssina *et al*, "Recommendation system using the k-nearest neighbors and singular value decomposition algorithms" in *International Journal of Electrical and Computer Engineering vol 11, no6*. 2021. [Online]. Available: https://doi.org/10.11591/ijece.v11i6.pp5541-5548

[30] H. Li, T. Liu, X. Wu, and S. Li, "Correlated SVD and Its Application in Bearing Fault Diagnosis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 355–365, Aug. 2021, doi: https://doi.org/10.1109/tnnls.2021.3094799

[31] S. Rahman, "Extended Collaborative Filtering Recommendation System with Adaptive KNN and SVD," *Social Science Research Network*, Jan. 2023, doi: https://doi.org/10.2139/ssrn.4550762

[32] Y. Shmalo, J. Jenkins, and O. Krupchytskyi, "Deep Learning Weight Pruning with RMT-SVD: Increasing Accuracy and Reducing Overfitting," *arXiv.org*, 2023. https://arxiv.org/abs/2303.08986

[33] R. Blanquero, E. Carrizosa, P. Ramírez-Cobo, and M. R. Sillero-Denamiel, "Variable selection for Naïve Bayes classification," *Computers & Operations Research*, vol. 135, p. 105456, Nov. 2021, doi: https://doi.org/10.1016/j.cor.2021.105456

[34] K. Qi and H. Yang, "Elastic Net Nonparallel Hyperplane Support Vector Machine and Its Geometrical Rationality," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7199–7209, Jun. 2021, doi: https://doi.org/10.1109/tnnls.2021.3084404

[35] D. Keerthana, V. Venugopal, M. K. Nath, and M. Mishra, "Hybrid Convolutional Neural Networks with SVM Classifier for Classification of Skin Cancer," *Biomedical Engineering Advances*, p. 100069, Dec. 2022, doi: https://doi.org/10.1016/j.bea.2022.100069

[36] S. P. Balakannan, R. Pandiarajan, V. U, I.P.Prabarani, S. M. Priya and C. Jahnavi, "NourishNook: Home Remedies and Personalized Health Recommendations using NLP," 2025 International Conference on Intelligent Computing and Control Systems (ICICCS), Erode, India, 2025, pp. 1185-1192, doi: https://doi.org/10.1109/ICICCS65191.2025.10984441

[37] U. Mathur, "A Content Based Recommender System for Medicine using Machine Learning Algorithm MSc Research Project Data Analytics." Accessed: Aug. 18, 2025. [Online]. Available: https://norma.ncirl.ie/6223/1/utkarshmathur.pdf

[38] B. Nistal-Nuño, "Medication recommendation system for online pharmacy using an adaptive user interface," Computer Methods and Programs in Biomedicine Update, vol. 2, p. 100077, 2022, doi: https://doi.org/10.1016/j.cmpbup.2022.100077.

[39] Rajkumar K, Ragupathi T, and Karthikeyan S, "Intelligent Chatbot for Hospital Recommendation System," pp. 664–668, Mar. 2024, doi: https://doi.org/10.1109/icdt61202.2024.10488954

[40] Héctor Alejandro Acuña-Cid, E. Ahumada-Tello, Óscar Omar Ovalle-Osuna, R. Evans, J. E. Hernández-Ríos, and M. A. Zambrano-Soto, "CRISP-NET: Integration of the CRISP-DM

Model with Network Analysis," Machine Learning and Knowledge Extraction, vol. 7, no. 3, pp. 101–101, Sep. 2025, doi: https://doi.org/10.3390/make7030101

[41] O. Ozdemir, T. Dogru, M. Kizildag, and E. Erkmen, "A critical reflection on digitalization for the hospitality and tourism industry: value implications for stakeholders," International Journal of Contemporary Hospitality Management, vol. 35, no. 9, Jan. 2023, doi: https://doi.org/10.1108/ijchm-04-2022-0535

[42] A. Babu and Sekhar Babu Boddu, "BERT-Based Medical Chatbot: Enhancing Healthcare Communication through Natural Language Understanding," Exploratory Research in Clinical and Social Pharmacy, pp. 100419–100419, Feb. 2024, doi: https://doi.org/10.1016/j.rcsop.2024.100419

APPENDIX

Appendix A

app.py:

```
import os
import sqlite3
import urllib.parse
from math import radians, sin, cos, sqrt, atan2

import pandas as pd
from flask import Flask, request, jsonify, render_template, abort, redirect, url_for, flash,
session
from flask_bcrypt import Bcrypt
from flask_login import LoginManager, UserMixin, login_user, logout_user, login_required,
current_user

# Import the hybrid model function
# Ensure hybrid_recommender.py is in the same folder
from hybrid_recommender import predict_hybrid_rating

app = Flask(__name__)
app.config['SECRET_KEY'] = 'a_very_secret_and_secure_key_change_me'
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

DB_FILE_PATH = 'hospital_reviews.db'

# --- User Class ---
class User(UserMixin):
    def __init__(self, id, username, email, password, last_symptom=None, last_dept=None):
        self.id = id
        self.username = username
        self.email = email
        self.password = password
        self.last_symptom = last_symptom
        self.last_dept = last_dept

@login_manager.user_loader
def load_user(user_id):
    conn = sqlite3.connect(DB_FILE_PATH)
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users WHERE id = ?", (user_id,))
    u = cursor.fetchone()
    conn.close()
    if u:
        # This handles cases where the tuple might be 4 or 6 elements during migration
```

```
        # u[0]=id, u[1]=user, u[2]=email, u[3]=pass, u[4]=symptom, u[5]=dept
        return User(u[0], u[1], u[2], u[3],
                    u[4] if len(u) > 4 else None,
                    u[5] if len(u) > 5 else None)
    return None

# --- Database Initialization & Migration ---
def init_db():
    conn = sqlite3.connect(DB_FILE_PATH)
    # 1. Create tables if they don't exist
    conn.execute('''CREATE TABLE IF NOT EXISTS users
                    (id INTEGER PRIMARY KEY AUTOINCREMENT,
                     username TEXT UNIQUE, email TEXT UNIQUE, password TEXT,
                     last_symptom TEXT, last_dept TEXT)''')
    conn.execute('CREATE TABLE IF NOT EXISTS bookmarks (user_id INTEGER,
hospital_name TEXT, PRIMARY KEY(user_id, hospital_name))')

    # 2. MIGRATION: Check if old database needs new columns
    cursor = conn.cursor()
    cursor.execute("PRAGMA table_info(users)")
    columns = [info[1] for info in cursor.fetchall()]
    if 'last_symptom' not in columns:
        print("Migrating database: Adding last_symptom column...")
        conn.execute("ALTER TABLE users ADD COLUMN last_symptom TEXT")
    if 'last_dept' not in columns:
        print("Migrating database: Adding last_dept column...")
        conn.execute("ALTER TABLE users ADD COLUMN last_dept TEXT")

    conn.commit()
    conn.close()

init_db()

# --- Load Dataset ---
def load_review_data():
    global df
    try:
        conn = sqlite3.connect(DB_FILE_PATH)
        query = """
        SELECT h.hospital_name AS "Hospital_Name", r.star_rating AS "Star rating",
               r.review_text_raw AS "Review content", r.sentiment_score AS "Sentiment Score"
        FROM reviews r JOIN hospitals h ON r.hospital_id = h.hospital_id;
        """
        df = pd.read_sql_query(query, conn)
        conn.close()
        df['Sentiment Score'] = pd.to_numeric(df['Sentiment Score'], errors='coerce')
        df.dropna(subset=['Review content', 'Hospital_Name', 'Star rating'], inplace=True)
        print("Dataset loaded successfully.")
    except Exception as e:
```

```
        print(f"DATABASE ERROR: {e}")
        df = None

load_review_data()

# --- Hospital Database with Verified Insurance ---
HOSPITAL_INFO_DB = {
    'Sunway Medical Centre Penang': {'location': '3106, Lebuh Tenggiri 2 Pusat Bandar Seberang Jaya, 13700 Perai, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-373 9191', 'lat': 5.3905, 'lon': 100.3986, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Etiqa', 'Great Eastern', 'Hong Leong Assurance', 'Manulife', 'Prudential', 'Prudential BSN', 'Income Insurance', 'Zurich']},
    'LohGuanLye Specialists Centre': {'location': '238, Jln Macalister, 10400 George Town, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-238 8888', 'lat': 5.4162, 'lon': 100.3204, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Generali (AXA)', 'Etiqa', 'Great Eastern', 'Manulife', 'Prudential', 'Tokio Marine', 'Zurich']},
    'Gleneagles Hospital Penang': {'location': '1, Jalan Pangkor, 10050 George Town, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-222 9111', 'lat': 5.4262, 'lon': 100.3159, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Great Eastern', 'Prudential', 'HSBC Amanah', 'Zurich', 'AIG', 'Etiqa']},
    'Pantai Hospital Penang': {'location': '82, Jalan Tengah, Bayan Baru, 11900 Bayan Lepas, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-643 3888', 'lat': 5.3275, 'lon': 100.2855, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Prudential', 'Great Eastern', 'Manulife', 'Etiqa', 'Liberty Insurance', 'Generali']},
    'Island Hospital Penang': {'location': '308, Jln Macalister, 10450 George Town, Pulau Pinang', 'hours': '8:30am - 5pm', 'days': 'Monday - Saturday', 'phone': '+60 4-238 3388', 'lat': 5.4224, 'lon': 100.314, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Great Eastern', 'Prudential', 'Manulife', 'Generali', 'FWD Takaful']},
    'KPJ Penang Specialist Hospital': {'location': '570, Jln Perda Utama, Bandar Perda, 14000 Bukit Mertajam, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-548 6688', 'lat': 5.3703, 'lon': 100.4357, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Prudential', 'Etiqa', 'Takaful Ikhlas', 'Great Eastern Takaful']},
    'Bagan Specialist Centre': {'location': 'Jalan Bagan 1, Taman Bagan, 13400 Butterworth, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-371 0000', 'lat': 5.4091, 'lon': 100.3848, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Great Eastern', 'Prudential', 'Etiqa', 'Manulife']},
    'Lam Wah Ee Hospital': {'location': '141, Jln Tan Sri Teh Ewe Lim, 11600 Jelutong, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-652 8888', 'lat': 5.3922, 'lon': 100.3039, 'category': 'Not-for-profit Private', 'insurance': ['AIA', 'Allianz', 'Prudential', 'Great Eastern', 'Manulife', 'Etiqa']},
    'Penang Adventist Hospital': {'location': '465, Jalan Burma, 10350 George Town, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-222 7200', 'lat': 5.4328, 'lon': 100.3053, 'category': 'Private', 'insurance': ['AIA', 'Allianz', 'Great Eastern', 'Prudential', 'Etiqa', 'Manulife', 'Zurich']},
    'MountMiriam Cancer Hospital': {'location': '23, Jalan Bulan, Fettes Park, 11200 Tanjung Bungah, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-892 3999', 'lat': 5.4591, 'lon': 100.301, 'category': 'Specialist Hospital', 'insurance': ['AIA', 'Allianz', 'Prudential', 'Great Eastern', 'Etiqa']},
```

```
    'Penang General Hospital': {'location': 'Jalan Residensi, 10450 George Town, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-222 5333', 'lat': 5.4165, 'lon': 100.3111, 'category': 'Government (MOH)', 'insurance': ['Government GL', 'Self-Pay (Subsidized)', 'Private Reimbursement']},
    'Hospital Seberang Jaya': {'location': 'Jln Tun Hussein Onn, Seberang Jaya, 13700 Perai, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-382 7333', 'lat': 5.3942, 'lon': 100.4082, 'category': 'Government (MOH)', 'insurance': ['Government GL', 'Self-Pay (Subsidized)', 'Private Reimbursement']},
    'Bukit Mertajam Hospital': {'location': 'Jln Kulim, 14000 Bukit Mertajam, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-549 7333', 'lat': 5.36, 'lon': 100.4645, 'category': 'Government (MOH)', 'insurance': ['Government GL', 'Self-Pay (Subsidized)', 'Private Reimbursement']},
    'Kepala Batas Hospital': {'location': 'Jalan Bertam 2, 13200 Kepala Batas, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-579 3333', 'lat': 5.5167, 'lon': 100.4286, 'category': 'Government (MOH)', 'insurance': ['Government GL', 'Self-Pay (Subsidized)', 'Private Reimbursement']},
    'Balik Pulau Hospital': {'location': 'Jalan Balik Pulau, 11000 Balik Pulau, Pulau Pinang', 'hours': 'Open 24 Hours', 'days': 'Monday - Sunday', 'phone': '+60 4-866 9333', 'lat': 5.3508, 'lon': 100.233, 'category': 'Government (MOH)', 'insurance': ['Government GL', 'Self-Pay (Subsidized)', 'Private Reimbursement']}
}

DEPARTMENT_DATA = {
    'Ophthalmology': ['eye', 'vision', 'blurry', 'glaucoma', 'cataract'],
    'Dermatology': ['skin', 'rash', 'acne', 'itch'],
    'Cardiology': ['heart', 'chest pain', 'blood pressure'],
    'Orthopedics': ['bone', 'joint', 'fracture', 'knee'],
    'Gastroenterology': ['stomach', 'digestive', 'endoscopy'],
    'Pediatrics': ['child', 'pediatrician', 'baby'],
    'ENT (Otolaryngology)': ['ear', 'nose', 'throat', 'sinus'],
    'Neurology': ['nerve', 'headache', 'migraine', 'seizure'],
    'Obstetrics & Gynecology (OB-GYN)': ['pregnancy', 'pregnant', 'ob-gyn']
}

POSITIVE_SENTIMENT_KEYWORDS = ['good', 'great', 'excellent', 'professional', 'friendly', 'clean', 'helpful', 'best', 'fast', 'recommended', 'care', 'efficient', 'quick']

# --- Helper Functions ---
def calculate_haversine_distance(lat1, lon1, lat2, lon2):
    R = 6371.0
    lat1_rad, lon1_rad, lat2_rad, lon2_rad = map(radians, [lat1, lon1, lat2, lon2])
    dlon = lon2_rad - lon1_rad
    dlat = lat2_rad - lat1_rad
    a = sin(dlat / 2)**2 + cos(lat1_rad) * cos(lat2_rad) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    return R * c

def get_department_from_symptoms(symptom_text):
    symptom_text = symptom_text.lower()
```

```
    for dept, keywords in DEPARTMENT_DATA.items():
        for kw in keywords:
            if kw in symptom_text: return dept, kw
    return "General Medicine", None

def recommend_hospitals_for_department(department, user_lat=None, user_lon=None,
top_n=3):
    if df is None: return []
    keywords = DEPARTMENT_DATA.get(department, [])
    hospital_scores = []

    for hospital_name in HOSPITAL_INFO_DB:
        hospital_reviews = df[df['Hospital_Name'] == hospital_name]
        search_pattern = '|'.join(keywords) if keywords else ''
        relevant_reviews = hospital_reviews[hospital_reviews['Review
content'].str.contains(search_pattern, case=False, na=False)]

        if not relevant_reviews.empty:
            all_text = " ".join(relevant_reviews['Review content'].astype(str)).lower()
            top_pos_keywords = list(set([w for w in POSITIVE_SENTIMENT_KEYWORDS if
w in all_text]))[:3]
            avg_rating = relevant_reviews['Star rating'].mean()
            avg_sentiment = (relevant_reviews['Sentiment Score'].mean() + 1) / 2 # Normalize 0
to 1

            rating_norm = avg_rating / 5.0
            distance_km = None

            if user_lat is None or user_lon is None:
                rating_cont = rating_norm * 70
                sentiment_cont = avg_sentiment * 30
                distance_cont = 0
            else:
                h = HOSPITAL_INFO_DB[hospital_name]
                distance_km = calculate_haversine_distance(user_lat, user_lon, h['lat'], h['lon'])
                dist_score = max(0, 1 - (distance_km / 25))
                # Hybrid Weights: 70% Quality (split 70/30) + 30% Distance
                rating_cont = (rating_norm * 0.7 * 0.7) * 100
                sentiment_cont = (avg_sentiment * 0.3 * 0.7) * 100
                distance_cont = (dist_score * 0.3) * 100

            final_score = (rating_cont + sentiment_cont + distance_cont) / 100
            hospital_scores.append({
                'hospital_name': hospital_name, 'score': final_score,
                'avg_rating': avg_rating, 'review_count': len(relevant_reviews),
                'distance_km': round(distance_km, 1) if distance_km is not None else "N/A",
                'positive_keywords': top_pos_keywords,
                'breakdown': {'rating': round(rating_cont, 1), 'sentiment': round(sentiment_cont, 1),
'distance': round(distance_cont, 1)}
```

```
            })
    return sorted(hospital_scores, key=lambda x: x['score'], reverse=True)[:top_n]

# --- ROUTES ---

@app.route('/')
def home():
    if not current_user.is_authenticated and 'guest' not in session:
        return redirect(url_for('welcome'))
    return render_template('homepage.html')

@app.route('/welcome')
def welcome():
    return render_template('welcome.html')

@app.route('/guest')
def guest_login():
    session['guest'] = True
    return redirect(url_for('home'))

@app.route('/symptom_recommend', methods=['POST'])
def handle_symptom_recommendation():
    data = request.get_json()
    dept, matched_kw = get_department_from_symptoms(data['symptoms'])
    recommendations = recommend_hospitals_for_department(dept, data.get('lat'),
data.get('lon'))

    if current_user.is_authenticated:
        conn = sqlite3.connect(DB_FILE_PATH)
        conn.execute("UPDATE users SET last_symptom = ?, last_dept = ? WHERE id = ?",
                     (data['symptoms'], dept, current_user.id))
        conn.commit(); conn.close()

    return jsonify({'department': dept, 'matched_keyword': matched_kw, 'recommendations':
recommendations})

@app.route('/hybrid')
def hybrid_page():
    if not current_user.is_authenticated and 'guest' not in session: return
redirect(url_for('login'))
    return render_template('hybrid_recommender.html', is_guest=('guest' in session))

@app.route('/get_recommendation', methods=['POST'])
def handle_hybrid_recommendation():
    if 'guest' in session: return jsonify({"error": "Member-only feature"}), 403
    data = request.get_json()
    try:
        prediction_result = predict_hybrid_rating(current_user.id, data['hospital_id'],
data['review_text'])
```

```
        return jsonify(prediction_result)
    except Exception as e: return jsonify({"error": str(e)}), 500

@app.route('/hospital/<hospital_name>')
def hospital_details(hospital_name):
    info = HOSPITAL_INFO_DB.get(hospital_name)
    if not info: abort(404)
    h_data = info.copy(); h_data['name'] = hospital_name
    q = urllib.parse.quote_plus(f"{hospital_name} {h_data['location']}")
    h_data['google_maps_link'] = f"https://www.google.com/maps/search/?api=1&query={q}"
    is_bookmarked = False
    if current_user.is_authenticated:
        conn = sqlite3.connect(DB_FILE_PATH); cursor = conn.cursor()
        cursor.execute("SELECT 1 FROM bookmarks WHERE user_id = ? AND
hospital_name = ?", (current_user.id, hospital_name))
        is_bookmarked = cursor.fetchone() is not None; conn.close()
    return render_template('hospital_detail.html', hospital=h_data,
is_bookmarked=is_bookmarked)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated: return redirect(url_for('home'))
    if request.method == 'POST':
        hashed = bcrypt.generate_password_hash(request.form['password']).decode('utf-8')
        try:
            conn = sqlite3.connect(DB_FILE_PATH)
            conn.execute("INSERT INTO users (username, email, password) VALUES (?, ?, ?)",
                    (request.form['username'], request.form['email'], hashed))
            conn.commit(); conn.close()
            flash('Account created!', 'success')
            return redirect(url_for('login'))
        except sqlite3.IntegrityError: flash('User exists.', 'error')
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated: return redirect(url_for('home'))
    if request.method == 'POST':
        conn = sqlite3.connect(DB_FILE_PATH); cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE email = ?", (request.form['email'],))
        u = cursor.fetchone(); conn.close()
        if u and bcrypt.check_password_hash(u[3], request.form['password']):
            # Handles 4 or 6 column users
            user = User(u[0], u[1], u[2], u[3], u[4] if len(u)>4 else None, u[5] if len(u)>5 else
None)
            login_user(user, remember=True)
            session.pop('guest', None)
            return redirect(url_for('home'))
        flash('Login failed.', 'error')
```

```
    return render_template('login.html')

@app.route('/logout')
def logout():
    logout_user(); session.pop('guest', None)
    return redirect(url_for('welcome'))

@app.route('/bookmarks')
@login_required
def bookmarks_page():
    conn = sqlite3.connect(DB_FILE_PATH); cursor = conn.cursor()
    cursor.execute("SELECT hospital_name FROM bookmarks WHERE user_id = ?",
(current_user.id,))
    rows = cursor.fetchall(); conn.close()
    hospitals = []
    for r in rows:
        if r[0] in HOSPITAL_INFO_DB:
            info = HOSPITAL_INFO_DB[r[0]].copy(); info['name'] = r[0]
            hospitals.append(info)
    return render_template('bookmarks.html', hospitals=hospitals)

@app.route('/toggle_bookmark', methods=['POST'])
def toggle_bookmark():
    if not current_user.is_authenticated: return jsonify({"error": "Login required"}), 401
    name = request.get_json().get('hospital_name')
    conn = sqlite3.connect(DB_FILE_PATH); cursor = conn.cursor()
    cursor.execute("SELECT 1 FROM bookmarks WHERE user_id = ? AND hospital_name
= ?", (current_user.id, name))
    if cursor.fetchone():
        cursor.execute("DELETE FROM bookmarks WHERE user_id = ? AND hospital_name
= ?", (current_user.id, name))
        action = "removed"
    else:
        cursor.execute("INSERT INTO bookmarks (user_id, hospital_name) VALUES (?, ?)",
(current_user.id, name))
        action = "added"
    conn.commit(); conn.close()
    return jsonify({"status": "success", "action": action})

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

hybrid_recommender.py coding:

```
import joblib
import re
import nltk
import numpy as np
from nltk.corpus import stopwords
```

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

# Load all pre-trained models and the vectorizer
print("--- Loading SVD and SVM models ---")
try:
    svd_model = joblib.load('SVD_model.pkl')
    svm_payload = joblib.load('SVM_model.pkl')
    svm_model = svm_payload['model']
    tfidf_vectorizer = svm_payload['vectorizer']
    print("All models and vectorizer loaded successfully.")
except FileNotFoundError as e:
    print(f"Error loading models: {e}. Make sure 'SVD_model.pkl' and 'SVM_model.pkl' are
in the correct directory.")
    svd_model = svm_model = tfidf_vectorizer = None

# Initialize text processing tools
try:
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
except LookupError:
    print("NLTK data not found. Downloading...")
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))

# Define the text cleaning function
def clean_text(text):
    """A single function to perform all text cleaning steps."""
    if not isinstance(text, str):
        return ""
    text = text.lower()
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()
    tokens = word_tokenize(text)
    lemmatized_tokens = [
        lemmatizer.lemmatize(word) for word in tokens if word not in stop_words and
len(word) > 1
    ]
    return ' '.join(lemmatized_tokens)

# Define the main Hybrid Recommender Prediction Function
def predict_hybrid_rating(user_id, hospital_id, review_text, alpha=0.5):
    """
    Calculates a hybrid recommendation rating by combining SVD and SVM.

    Args:
```

```
        user_id (str): The author's name or ID.
        hospital_id (str): The hospital's name or ID.
        review_text (str): The text content to be analyzed by the content model.
        alpha (float): The weight for the content-based (SVM) model score.
                       The SVD model weight will be (1 - alpha).

    Returns:
        A dictionary containing the final hybrid rating and individual model scores.
    """
    if not all([svd_model, svm_model, tfidf_vectorizer]):
        return {"error": "Models are not loaded. Cannot make a prediction."}

    # === 1. Get SVD (Collaborative Filtering) Prediction ===
    # This is R_SVD, the predicted rating on a 1-5 scale.
    svd_prediction = svd_model.predict(uid=user_id, iid=hospital_id)
    R_SVD = svd_prediction.est

    # === 2. Get SVM (Content-Based) Prediction ===
    # This is P_SVM, the predicted probability on a 0-1 scale.
    processed_text = clean_text(review_text)
    vectorized_text = tfidf_vectorizer.transform([processed_text])
    P_SVM = svm_model.predict_proba(vectorized_text)[0][1]

    # === 3. Convert SVM Probability to a 1-5 Rating Scale ===
    # This step creates R_SVM by mapping the [0, 1] probability to the [1, 5] rating scale.
    R_SVM = (P_SVM * 4.0) + 1.0

    # === 4. Calculate the Final Hybrid Score using a Direct Weighted Average ===
    # Both R_SVM and R_SVD are now on the same 1-5 scale.
    final_predicted_rating = (alpha * R_SVM) + ((1 - alpha) * R_SVD)

    # === 5. Final Clipping (Safety Net) ===
    # Ensures the final score is strictly within the 1-5 range.
    final_predicted_rating = np.clip(final_predicted_rating, 1, 5)

    # === 6. Explanation Generation ===

        # === XAI LOGIC ===
    # Get the words that actually influenced the SVM prediction
    feature_names = tfidf_vectorizer.get_feature_names_out()
    vectorized_text = tfidf_vectorizer.transform([clean_text(review_text)])
    # Find words present in the input that have the highest weight in the model
    # (Simplified XAI: showing which input words the TF-IDF recognized)
    keywords_detected = []
    feature_index = vectorized_text.nonzero()[1]
    for i in feature_index:
        keywords_detected.append(feature_names[i])

    explanation = {
```

```
        "logic": f"This score is {alpha*100}% based on your symptom description and {(1-
alpha)*100}% based on historical patient ratings.",
        "top_features": keywords_detected[:3], # Show the top 3 words detected
        "svd_contribution": "High" if R_SVD > 3.5 else "Moderate",
        "svm_contribution": "High" if R_SVM > 3.5 else "Moderate"
    }

    return {
        'user_id': user_id,
        'hospital_id': hospital_id,
        'svd_predicted_rating (R_SVD)': R_SVD,
        'svm_rating_equivalent (R_SVM)': R_SVM,
        'final_hybrid_rating': final_predicted_rating,
        'explanation':explanation
    }
```

homepage.html coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home | HealthRecommend</title>
    <style>
        :root {
            --primary-red: #c8102e;
            --primary-blue: #0056b3;
            --secondary-green: #28a745;
            --dark-text: #2d3436;
            --light-bg: #f4f7f6;
            --white: #ffffff;
        }

        body { font-family: 'Segoe UI', Roboto, sans-serif;
                background: linear-gradient(rgba(244, 247, 246, 0.85), rgba(244, 247, 246, 0.10)),
url('/static/hospital_bg.jpg');
                background-size: cover;
                background-attachment: fixed;
                background-position: center;
                margin: 0;
            }

        /* Navbar Styling */
        .navbar {
            display: flex; justify-content: space-between; align-items: center;
            background-color: var(--white); padding: 15px 50px; box-shadow: 0 2px 10px
rgba(0,0,0,0.05);
```

```
        }
        .navbar-logo { font-size: 1.4em; text-decoration: none; color: var(--dark-text); }
        .navbar-logo strong { font-weight: 800; }

        .navbar-links { display: flex; align-items: center; gap: 20px; }
        .navbar-links a { text-decoration: none; color: #636e72; font-weight: 600; font-size:
0.95em; }

        /* Main Layout Grid */
        .main-layout {
            display: grid; grid-template-columns: 1fr 350px;
            gap: 30px; max-width: 1300px; margin: 40px auto; padding: 0 20px; align-items:
start;
        }

        /* Left Column Content */
        .content-card {
            background: var(--white); padding: 40px; border-radius: 15px; box-shadow: 0 10px
30px rgba(0,0,0,0.05);
        }
        header { text-align: center; margin-bottom: 40px; }
        h1 { font-size: 2.2em; margin-bottom: 10px; }
        .highlight { color: var(--primary-red); font-weight: 800; }
        header p { color: #636e72; font-size: 1.1em; }

        /* XAI Insight Box */
        #dept-insight-box {
            display: none;
            margin: 20px 0;
            padding: 18px;
            border-radius: 10px;
            border-left: 5px solid var(--primary-red);
            background: #fffafa;
            font-size: 0.95em;
            box-shadow: 0 2px 8px rgba(200, 16, 46, 0.05);
        }

        #results-container {
            display: grid;
            grid-template-columns: repeat(3, 1fr); /* 3 Columns */
            gap: 20px;
            margin-top: 30px;
        }

        /* Recommendation Card Styling */
        .recommendation {
            background: var(--white);
            border: 1px solid #edf2f7;
            border-radius: 12px;
```

```
    padding: 20px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.05);
    position: relative; /* For the rank badge */
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    transition: all 0.3s cubic-bezier(0.25, 0.8, 0.25, 1);
}

.recommendation:hover {
    transform: translateY(-10px); /* Lifts the card up */
    box-shadow: 0 15px 30px rgba(0,0,0,0.15); /* Adds a deeper shadow for depth */
    border-color: var(--primary-blue); /* Highlights the border */
    cursor: pointer;
}

.recommendation:hover .rank-badge {
    background: var(--primary-blue); /* Changes color from red to blue */
    transform: scale(1.1); /* Makes the badge slightly larger */
    transition: all 0.3s ease;
}

.recommendation:hover .score-fill {
    filter: brightness(1.1);
    transition: all 0.3s ease;
}

/* Rank Badge Indicator */
.rank-badge {
    position: absolute;
    top: -10px;
    left: -10px;
    background: var(--primary-red);
    color: white;
    width: 30px;
    height: 30px;
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-weight: 800;
    box-shadow: 0 2px 5px rgba(0,0,0,0.2);
    font-size: 0.9em;
}

.recommendation h3 { margin: 10px 0 5px 0; font-size: 1.1em; line-height: 1.3; }
.recommendation h3 a { color: var(--primary-blue); text-decoration: none; }
```

```
        .score-bar { background: #edf2f7; border-radius: 20px; height: 20px; margin-top: 10px;
overflow: hidden; }
        .score-fill { background: var(--secondary-green); color: white; text-align: center; line-
height: 20px; font-weight: bold; font-size: 0.75em; }

        /* Form Elements */
        label { display: block; margin-bottom: 8px; font-weight: 600; color: var(--dark-text);
font-size: 0.95em; }
        textarea, select { width: 100%; padding: 15px; margin-bottom: 20px; border: 1px solid
#dfe6e9; border-radius: 8px; box-sizing: border-box; background-color: #fcfcfc; font-size:
16px; }
        textarea { min-height: 100px; resize: vertical; }

        button[type="submit"] { background: var(--primary-blue); color: white; padding: 16px;
border: none; border-radius: 8px; font-weight: bold; cursor: pointer; width: 100%; transition:
0.3s; font-size: 1.1em; }
        button[type="submit"]:hover { background: #003d80; transform: translateY(-2px); }

        .location-check { display: flex; align-items: center; margin-bottom: 25px; background:
#f8f9fa; padding: 12px; border-radius: 8px; }
        .location-check input { width: 18px; height: 18px; margin-right: 12px; cursor: pointer; }

        /* Sidebar Styling */
        .sidebar-panel {
            background: var(--white); padding: 25px; border-radius: 15px;
            box-shadow: 0 10px 30px rgba(0,0,0,0.05); border-top: 5px solid var(--primary-blue);
            position: sticky; top: 20px;
        }
        .sidebar-title { font-size: 1.1em; font-weight: 800; margin-bottom: 20px; color: var(--
primary-blue); }
        .history-card { background: #fdf2f2; padding: 15px; border-radius: 10px; margin-
bottom: 20px; border-left: 4px solid var(--primary-red); }
        .advice-card { background: #e3f2fd; padding: 15px; border-radius: 10px; font-size:
0.9em; line-height: 1.6; color: #0d47a1; }

        #google_translate_element { margin-bottom: 15px; text-align: right; }

        /* Responsiveness: Stack columns on smaller screens */
        @media (max-width: 1100px) {
            #results-container { grid-template-columns: 1fr 1fr; }
        }
        @media (max-width: 900px) {
            .main-layout { grid-template-columns: 1fr; }
            #results-container { grid-template-columns: 1fr; }
            .sidebar-panel { position: static; }
        }
    </style>
</head>
<body>
```

```
<nav class="navbar">
   <a href="/" class="navbar-logo"><strong>Health</strong>Recommend</a>
   <div class="navbar-links">
      <span style="color: #b2bec3; font-size: 0.9em; margin-right: 10px;">
         {% if current_user.is_authenticated %} {{ current_user.username }} {% else %}
Guest {% endif %}
      </span>
      {% if current_user.is_authenticated %} <a href="/bookmarks">My Bookmarks</a> {%
endif %}
      <a href="/logout" style="color:var(--primary-red); font-weight: 600;">Logout</a>
   </div>
</nav>

<div class="main-layout">
   <div class="content-card">
      <div id="google_translate_element"></div>
      <header>
         <h1>Healthcare in <span class="highlight">Penang</span>.</h1>
         <p>Describe your symptoms below to find the specialized providers suits your
needs.</p>
      </header>

      <form id="symptom-form">
         <label for="symptoms">Describe Symptoms in Detail (Required)</label>
         <textarea id="symptoms" placeholder="e.g., I have been experiencing sharp lower
back pain for two days..." required></textarea>

         <label for="symptom-select">Quick Select Common Issues (Optional)</label>
         <select id="symptom-select">
            <option value="">-- Choose a symptom --</option>
            <option value="itchy skin rash">Itchy Skin / Rash</option>
            <option value="chest pain">Chest Pain</option>
            <option value="broken bone">Bone / Joint Pain</option>
            <option value="stomach ache">Stomach Ache</option>
            <option value="child fever">Child Sickness</option>
            <option value="blurry eye vision">Blurry Vision</option>
            <option value="sore throat">Sore Throat</option>
         </select>

         <div class="location-check">
            <input type="checkbox" id="prioritize-location">
            <label for="prioritize-location" style="margin-bottom:0; cursor:pointer;
color:#636e72; font-weight:400;">Prioritize hospitals near my current location</label>
         </div>

         <button type="submit">Analyze & Find Hospital</button>
      </form>
```

```
        <div id="dept-insight-box"></div>

        <div id="results-container"></div>
    </div>

    <aside class="sidebar-panel">
        <div class="sidebar-title">🤖 Health Assistant</div>
        <div id="sidebar-content">
            {% if 'guest' in session %}
                <div style="text-align:center; padding:20px;">
                    <p style="font-size:2em; margin-bottom: 10px;">🔒</p>
                    <p style="font-weight:bold; font-size: 0.9em;">Member Only Feature</p>
                    <p style="font-size:0.8em; line-height: 1.4;">Please <a href="/login"
style="color:var(--primary-blue)">Login</a> to see your search history.</p>
                </div>
            {% else %}
                <p id="placeholder" style="color: #95a5a6; font-size: 0.9em; line-height: 1.5;">
                    Welcome, <strong>{{ current_user.username }}</strong>! Describe your
symptoms to see personalized insights here.
                </p>
            {% endif %}
        </div>
            <div id="ranking-logic-outer" style="margin-top: 25px; border-top: 1px solid #eee;
display:none;">
            <div class="sidebar-title" style="color: #856404; font-size: 0.9em; margin-top:
15px;">⚖️ Ranking Explanation</div>
            <div id="ranking-logic-content"></div>
        </div>

    </aside>
</div>

<script type="text/javascript"
src="https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit"></s
cript>
<script>
    function googleTranslateElementInit() {
        new google.translate.TranslateElement({pageLanguage: 'en', includedLanguages: 'en,zh-
CN', layout: google.translate.TranslateElement.InlineLayout.SIMPLE},
'google_translate_element');
    }

    const HEALTH_ADVICE = {
        'Ophthalmology': 'Rest your eyes and avoid bright screens.',
        'Dermatology': 'Avoid scratching the area. Use mild soap.',
        'Cardiology': 'Rest immediately. If pain is severe, call 999.',
        'Orthopedics': 'Follow RICE: Rest, Ice, Compression, Elevation.',
        'Gastroenterology': 'Stay hydrated. Avoid spicy or oily foods.',
        'Pediatrics': 'Monitor temperature and ensure hydration.',
```

```
        'ENT (Otolaryngology)': 'Gargle with warm salt water. Use a humidifier.',
        'Neurology': 'Rest in a quiet, dark room.',
        'Obstetrics & Gynecology (OB-GYN)': 'Ensure plenty of rest and stay hydrated.',
        'General Medicine': 'Rest, maintain hydration, and monitor symptoms.'
    };

    const form = document.getElementById('symptom-form');
    const resultsContainer = document.getElementById('results-container');
    const insightBox = document.getElementById('dept-insight-box');
    const symptomSelect = document.getElementById('symptom-select');
    const symptomTextarea = document.getElementById('symptoms');

    symptomSelect.addEventListener('change', function() { if(this.value)
symptomTextarea.value = this.value; });

    form.addEventListener('submit', function(e) {
        e.preventDefault();
        const symptoms = symptomTextarea.value;
        const useLoc = document.getElementById('prioritize-location').checked;
        resultsContainer.innerHTML = '<p style="grid-column: span 3; text-align:center;
color:#636e72;">Analyzing specialized reviews for you...</p>';
        insightBox.style.display = 'none';

        if (useLoc && navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(pos => getRecs(symptoms,
pos.coords.latitude, pos.coords.longitude), () => getRecs(symptoms));
        } else { getRecs(symptoms); }
    });

    async function getRecs(symptoms, lat=null, lon=null) {
        try {
            const response = await fetch('/symptom_recommend', {
                method: 'POST', headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({ symptoms, lat, lon })
            });
            const data = await response.json();
            resultsContainer.innerHTML = '';

            if (data.department) {
                insightBox.style.display = 'block';
                insightBox.innerHTML = data.matched_keyword
                    ? `<span style="color:var(--primary-red); font-weight:900;">XAI
INSIGHT:</span> The keyword <strong style="color:var(--primary-
blue);">"${data.matched_keyword}"</strong> matched the <strong style="color:var(--
primary-red);">${data.department}</strong> department.`
                    : `<span style="color:var(--primary-red); font-weight:900;">XAI
INSIGHT:</span> Routing to <strong style="color:var(--primary-blue);">General
Medicine</strong> for consultation.`;
                updateSidebar(data.department, symptoms);
```

```
        }

        data.recommendations.forEach((rec, index) => {
            const score = (rec.score * 100).toFixed(1);
            const b = rec.breakdown;
            const rank = index + 1; // 1, 2, 3

            const div = document.createElement('div');
            div.className = 'recommendation';

            div.onclick = () => { window.location.href =
`/hospital/${encodeURIComponent(rec.hospital_name)}`; };

            div.innerHTML = `
                <div class="rank-badge">${rank}</div>
                <div>
                    <h3><a
href="/hospital/${encodeURIComponent(rec.hospital_name)}">${rec.hospital_name}</a></
h3>
                    <p style="font-size:0.85em; color:#636e72; margin: 5px 0;">⭐
${rec.avg_rating.toFixed(1)} | 📍 ${rec.distance_km} km</p>
                    <div class="score-bar"><div class="score-fill"
style="width:${score}%">${score}% Match</div></div>
                </div>

                <div style="margin-top:15px; padding:10px; background:#f9f9f9; border-
radius:8px; font-size:0.75em;">
                    <p style="margin:0 0 5px 0; line-height:1.4;">
                        <span style="color:var(--primary-red); font-weight:bold;">XAI
BREAKDOWN:</span><br>
                        Rating: ${b.rating}% | Sentiment: ${b.sentiment}% | Dist: ${b.distance}%
                    </p>
                    <div style="display: flex; flex-wrap: wrap; gap: 4px; margin-top:5px;">
                        ${rec.positive_keywords.map(kw => `<span style="background:#e3f2fd;
color:#0d47a1; padding:1px 5px; border-radius:3px; border:1px solid #bbdefb; font-
size:0.85em;">#${kw}</span>`).join('')}
                    </div>
                </div>`;
            resultsContainer.appendChild(div);
        });

        resultsContainer.scrollIntoView({ behavior: 'smooth', block: 'start' });

    } catch (error) { console.error("Error fetching recommendations:", error); }
}

function updateSidebar(dept, symptoms) {
    const sidebar = document.getElementById('sidebar-content');
    const rankingOuter = document.getElementById('ranking-logic-outer');
```

```
        const rankingContent = document.getElementById('ranking-logic-content');

        if (sidebar.innerHTML.includes('🔒')) return;

        const useLoc = document.getElementById('prioritize-location').checked;
        const advice = HEALTH_ADVICE[dept] || HEALTH_ADVICE['General Medicine'];

        // 1. Prepare Ranking Logic (Independent Block)
        const rankingText = useLoc
            ? "Results are ranked by combining <strong>specialized review quality
(70%)</strong> and <strong>proximity to your current location (30%)</strong>."
            : "Results are ranked based on a weighted average of <strong>Google star
ratings</strong> and <strong>sentiment analysis</strong> of reviews matching your
symptoms.";

        // Show the outer block and inject content
        rankingOuter.style.display = 'block';
        rankingContent.innerHTML = `
            <div class="advice-card" style="background: #fff9db; color: #856404; border-left:
4px solid #f1c40f; margin-bottom: 0;">
                <p style="margin-bottom:0; font-size:0.85em;">${rankingText}</p>
            </div>`;

        // 2. Update the main Health Assistant content (History & Advice)
        sidebar.innerHTML = `
            <div class="history-card">
                <p style="font-size:0.75em; font-weight:800; color:#e17055; margin:0; text-
transform:uppercase;">Recent Search</p>
                <p style="font-weight:700; font-size:0.9em; margin:5px
0;">"${symptoms.substring(0, 40)}..."</p>
                <p style="font-size:0.85em; margin:0;">Dept: <strong>${dept}</strong></p>
            </div>
            <div class="advice-card">
                <h4>💡 Home Care Advice</h4>
                <p>${advice}</p>
                <p style="font-size:0.7em; font-style:italic; border-top:1px solid rgba(0,0,0,0.05);
padding-top:5px;">Always seek professional diagnosis.</p>
            </div>`;
    }

    window.addEventListener('load', () => {
        const s = "{{ current_user.last_symptom if current_user.is_authenticated else '' }}";
        const d = "{{ current_user.last_dept if current_user.is_authenticated else '' }}";
        if (s && d && s !== "None" && d !== "None") updateSidebar(d, s);
    });
</script>
</body>
</html>
```

APPENDIX

hospital_detail.html coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>{{ hospital.name }} | HealthRecommend</title>
   <style>
      :root {
         --primary-red: #c8102e;
         --primary-blue: #0056b3;
         --secondary-green: #28a745;
         --dark-text: #2d3436;
         --light-bg: #f4f7f6;
         --white: #ffffff;
      }

      body {
         font-family: 'Segoe UI', Roboto, sans-serif;
         background: linear-gradient(rgba(244, 247, 246, 0.4), rgba(244, 247, 246, 0.6)),
url('/static/hospital_bg.jpg');
         background-size: cover;
         background-attachment: fixed;
         background-position: center;
         margin: 0;
         padding: 0;
         color: var(--dark-text);
      }

      .navbar {
         background: var(--white);
         padding: 15px 50px;
         box-shadow: 0 2px 10px rgba(0,0,0,0.05);
         display: flex;
         justify-content: space-between;
         align-items: center;
      }

      .container {
         max-width: 900px;
         margin: 40px auto;
         padding: 0 20px;
      }

      /* Translate Widget Styling */
      #google_translate_element {
         margin-bottom: 15px;
         text-align: right;
```

```
}

.detail-card {
    background: var(--white);
    border-radius: 15px;
    box-shadow: 0 10px 30px rgba(0,0,0,0.05);
    overflow: hidden;
    border-top: 5px solid var(--primary-red);
}

.header-section {
    padding: 40px;
    background: linear-gradient(to bottom, #fff, #f9f9f9);
    border-bottom: 1px solid #eee;
}

.title-wrapper {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: wrap;
    gap: 20px;
}

h1 { margin: 0; color: var(--dark-text); font-size: 2.2em; flex: 1; }

.category-tag {
    display: inline-block;
    background: #e3f2fd;
    color: var(--primary-blue);
    padding: 4px 12px;
    border-radius: 20px;
    font-size: 0.85em;
    font-weight: bold;
    margin-top: 10px;
}

.btn-bookmark-top {
    padding: 10px 20px;
    border-radius: 8px;
    font-weight: bold;
    cursor: pointer;
    transition: 0.3s;
    border: none;
    display: inline-flex;
    align-items: center;
    font-size: 0.9em;
}
.state-saved { background-color: #ffeaa7; color: #d63031; border: 1px solid #fab1a0; }
```

```
.state-unsaved { background-color: var(--secondary-green); color: white; }
.btn-bookmark-top:hover { transform: translateY(-2px); opacity: 0.9; }

.content-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 30px;
    padding: 40px;
}

.info-group h3 {
    font-size: 1em;
    color: #b2bec3;
    text-transform: uppercase;
    letter-spacing: 1px;
    margin-bottom: 15px;
}

.info-item {
    margin-bottom: 20px;
    line-height: 1.6;
}

.hotline-wrapper {
    display: flex;
    align-items: center;
    gap: 10px;
    margin-top: 5px;
}
.copy-link {
    font-size: 0.8em;
    color: var(--primary-blue);
    text-decoration: underline;
    cursor: pointer;
    font-weight: 600;
}
.copy-link:hover { color: var(--primary-red); }

.insurance-section {
    grid-column: span 2;
    background: #f8f9fa;
    padding: 30px 40px;
    border-top: 1px solid #eee;
}

.insurance-badges {
    display: flex;
    flex-wrap: wrap;
    gap: 15px;
```

```
            margin-top: 15px;
            align-items: center;
        }

        .insurance-card {
            background: white;
            border: 1px solid #dfe6e9;
            padding: 10px;
            border-radius: 8px;
            display: flex;
            align-items: center;
            justify-content: center;
            width: 120px;
            height: 60px;
            transition: transform 0.2s;
            box-shadow: 0 2px 5px rgba(0,0,0,0.05);
        }

        .insurance-card:hover {
            transform: translateY(-3px);
            box-shadow: 0 5px 15px rgba(0,0,0,0.1);
        }

        .insurance-logo {
            max-width: 100%;
            max-height: 100%;
            object-fit: contain;
        }

        .badge {
            background: white;
            border: 1px solid #dfe6e9;
            padding: 8px 16px;
            border-radius: 8px;
            font-size: 0.9em;
            font-weight: 600;
            color: var(--primary-blue);
        }

        @media (max-width: 600px) {
            .content-grid { grid-template-columns: 1fr; }
            .insurance-section { grid-column: span 1; }
            .title-wrapper { flex-direction: column; align-items: flex-start; }
            .navbar { padding: 15px 20px; }
        }
    </style>
</head>
<body>
```

```
<nav class="navbar">
   <a href="/" style="text-decoration:none; color:inherit; font-
size:1.4em;"><strong>Health</strong>Recommend</a>
   <a href="/" style="text-decoration:none; color:var(--primary-blue); font-weight:600;">←
Back to Search</a>
</nav>

<div class="container">
   <!-- Google Translate Element -->
   <div id="google_translate_element"></div>

   <div class="detail-card">
      <div class="header-section">
         <div class="title-wrapper">
            <h1>{{ hospital.name }}</h1>

            {% if current_user.is_authenticated %}
               <button id="bookmark-btn"
                     class="btn-bookmark-top {{ 'state-saved' if is_bookmarked else 'state-
unsaved' }}"
                     onclick="toggleBookmark('{{ hospital.name }}')">
                  {{ '★ Bookmarked' if is_bookmarked else '☆ Save to Bookmarks' }}
               </button>
            {% endif %}
         </div>
         <span class="category-tag">{{ hospital.category }}</span>
      </div>

      <div class="content-grid">
         <div class="info-group">
            <h3>Contact & Location</h3>
            <div class="info-item">
               <strong> 📍 Location:</strong><br>
               <span>{{ hospital.location }}</span><br>
               <a href="{{ hospital.google_maps_link }}" target="_blank" style="color:var(--
primary-blue); font-size:0.9em;">Open in Google Maps</a>
            </div>

            <div class="info-item">
               <strong>📞 Hotline:</strong><br>
               <div class="hotline-wrapper">
                  <span id="hotline-num" style="font-size: 1.1em; font-weight:
600;">{{ hospital.phone }}</span>
                  <span class="copy-link" onclick="copyToClipboard('{{ hospital.phone }}',
this)">Copy Number</span>
               </div>
            </div>
         </div>
```

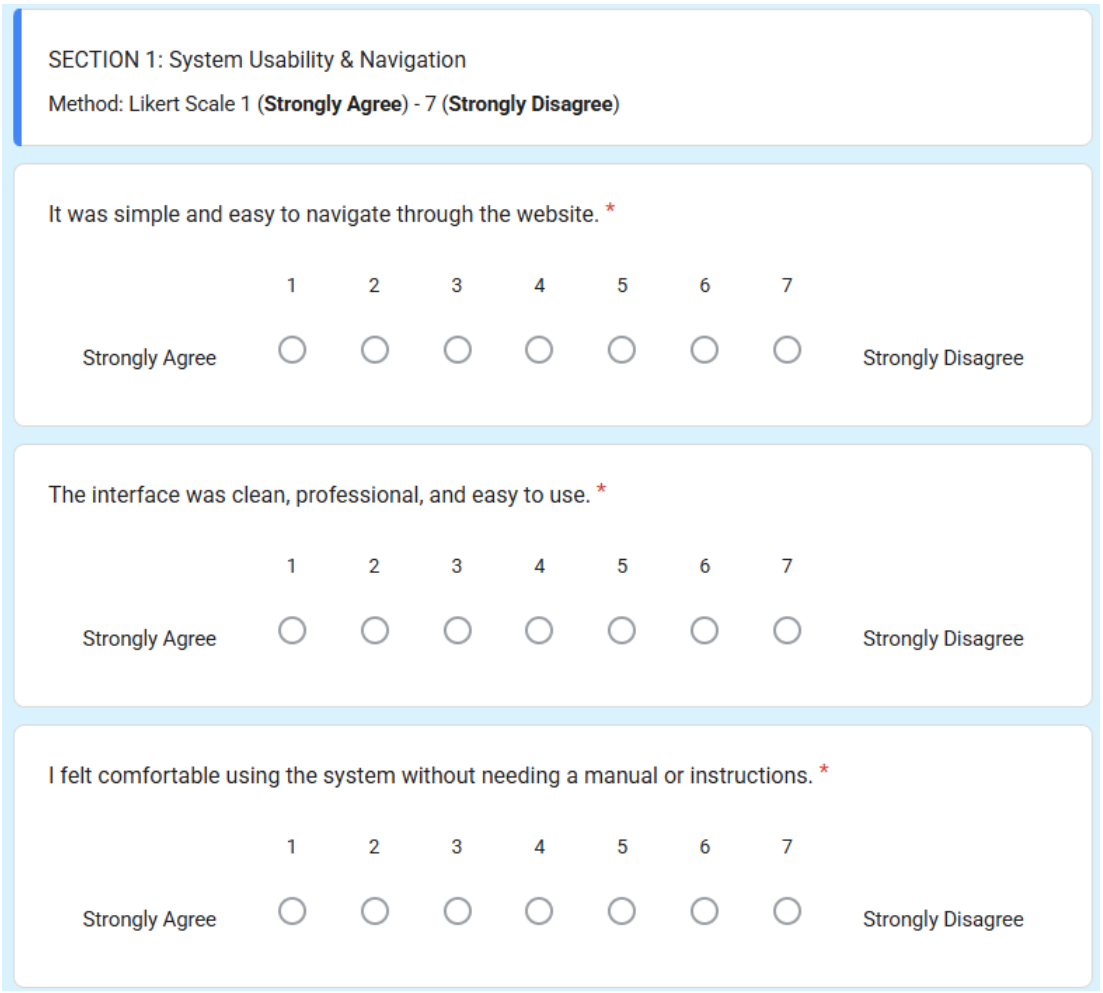
```
        <div class="info-group">
            <h3>Operation Hours</h3>
            <div class="info-item">
                <strong>🕒 Hours:</strong><br>
                <span>{{ hospital.hours }}</span>
            </div>
            <div class="info-item">
                <strong>🗓️ Business Days:</strong><br>
                <span>{{ hospital.days }}</span>
            </div>
        </div>

        <div class="insurance-section">
            <h3>Accepted Insurance Panels</h3>
            <div class="insurance-badges">
                {% if hospital.insurance %}
                    {# Create a mapping dictionary for filenames #}
                    {% set logo_map = {
                        'AIA': 'aia.svg',
                        'AIG': 'aig.svg',
                        'Allianz': 'allianz.svg',
                        'Etiqa': 'etiqa.png',
                        'FWD': 'fwd.svg',
                        'Generali': 'generali.svg',
                        'Great Eastern': 'great-eastern.svg',
                        'HSBC': 'hsbc.svg',
                        'Liberty Insurance': 'liberty-insurance.jpg',
                        'Manulife': 'manulife.webp',
                        'Prudential': 'prudential.png',
                        'Takaful Ikhlas': 'takaful-ikhlas.png',
                        'Tokio Marine': 'tokio_marine.png',
                        'Zurich': 'zurich.svg'
                    } %}

                    {% for provider in hospital.insurance %}
                        <div class="insurance-card" title="{{ provider }}">
                            {% if provider in logo_map %}
                                <img src="{{ url_for('static', filename='insurance_images/' +
logo_map[provider]) }}"
                                     alt="{{ provider }}"
                                     class="insurance-logo">
                            {% else %}
                                {# Fallback to text if logo is not found #}
                                <span style="font-size: 0.8em; font-weight: bold; text-align:
center;">{{ provider }}</span>
                            {% endif %}
                        </div>
                    {% endfor %}
                {% else %}
```

```
                    <p>Contact hospital for specific panel information.</p>
                {% endif %}
            </div>
        </div>
    </div>
</div>

<!-- Google Translate Scripts -->
<script type="text/javascript"
src="https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit"></s
cript>
<script>
    function googleTranslateElementInit() {
        new google.translate.TranslateElement({
            pageLanguage: 'en',
            includedLanguages: 'en,zh-CN',
            layout: google.translate.TranslateElement.InlineLayout.SIMPLE
        }, 'google_translate_element');
    }

    async function toggleBookmark(hospitalName) {
        try {
            const response = await fetch('/toggle_bookmark', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ hospital_name: hospitalName })
            });
            const data = await response.json();
            if (data.status === 'success') {
                window.location.reload();
            } else {
                alert(data.error);
            }
        } catch (e) {
            console.error("Bookmark error:", e);
        }
    }

    function copyToClipboard(text, element) {
    const cleanNumber = text.replace(/\D/g, '');

    navigator.clipboard.writeText(cleanNumber).then(() => {
        const originalText = element.innerText;
        element.innerText = "Copied!";
        element.style.color = "#28a745"; // Changes to green

        setTimeout(() => {
            element.innerText = originalText;
```

```
            element.style.color = "var(--primary-blue)";
        }, 2000);
    }).catch(err => {
        console.error('Could not copy text: ', err);
    });
}
</script>

</body>
</html>
```

Appendix B

Form sample:



SECTION 1: System Usability & Navigation

Method: Likert Scale 1 (**Strongly Agree**) - 7 (**Strongly Disagree**)

It was simple and easy to navigate through the website. *

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

The interface was clean, professional, and easy to use. *

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

I felt comfortable using the system without needing a manual or instructions. *

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

SECTION 2: Recommendation Quality & Trust

Method: Likert Scale 1 (**Strongly Agree**) - 7 (**Strongly Disagree**)

The healthcare services recommended were relevant to my search criteria. *

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

The system provided enough information (e.g., location, services, ratings) to help me make a decision. *

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

I would follow the recommendations provided by this platform in a real-life situation.

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

SECTION 3: Overall Perception

Method: Semantic Adjectives

How would you rate the visual design of the healthcare platform? *

- Exceptional
- Good
- Average
- Poor
- Very Bad

Overall, how satisfied are you with this system? *

- Extremely Satisfied
- Satisfied
- Neutral
- Dissatisfied
- Extremely Dissatisfied

Section 4 of 4

SECTION 4: Qualitative Feedback

Method: Open-ended

What is the most useful feature of this healthcare recommendation system? *

Short answer text

In your opinion, how could the system be improved to make it more helpful for patients/users?

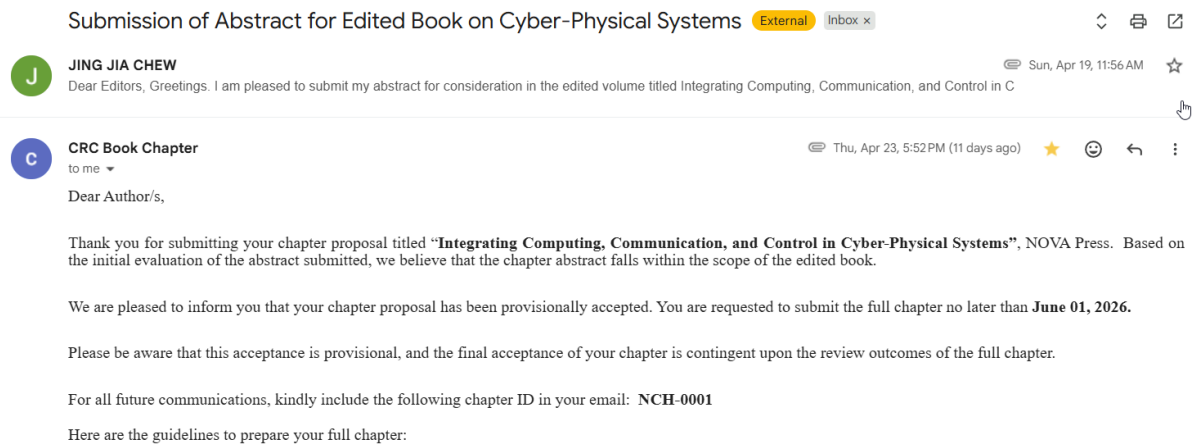
Long answer text

Appendix C

Poster:

APPENDIX

Proof of submission:



Submission of Abstract for Edited Book on Cyber-Physical Systems External Inbox ×

JING JIA CHEW Sun, Apr 19, 11:56 AM

Dear Editors, Greetings. I am pleased to submit my abstract for consideration in the edited volume titled Integrating Computing, Communication, and Control in C

CRC Book Chapter Thu, Apr 23, 5:52 PM (11 days ago)

to me

Dear Author/s,

Thank you for submitting your chapter proposal titled "**Integrating Computing, Communication, and Control in Cyber-Physical Systems"**, NOVA Press. Based on the initial evaluation of the abstract submitted, we believe that the chapter abstract falls within the scope of the edited book.

We are pleased to inform you that your chapter proposal has been provisionally accepted. You are requested to submit the full chapter no later than **June 01, 2026.**

Please be aware that this acceptance is provisional, and the final acceptance of your chapter is contingent upon the review outcomes of the full chapter.

For all future communications, kindly include the following chapter ID in your email: **NCH-0001**

Here are the guidelines to prepare your full chapter:

Poster

WEB-BASED RECOMMENDATION SYSTEM FOR HEALTHCARE SERVICES

UNIVERSITI TUNKU ABDUL RAHMAN
Wholly owned by UTAR Education Foundation

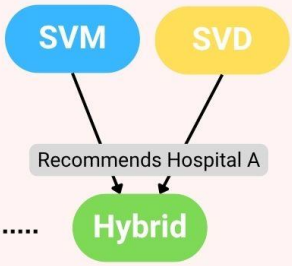
CHEW JING JIA | SUPERVISOR: DR. MUHAMMAD HUSAINI BIN NADRI

INTRODUCTION

Problem: Patients face "Information Overload" and "Decision Fatigue" when reading hundred of unstructured online reviews.

Aim: To develop a hybrid recommendation system for healthcare providers in Penang

METHODOLOGY



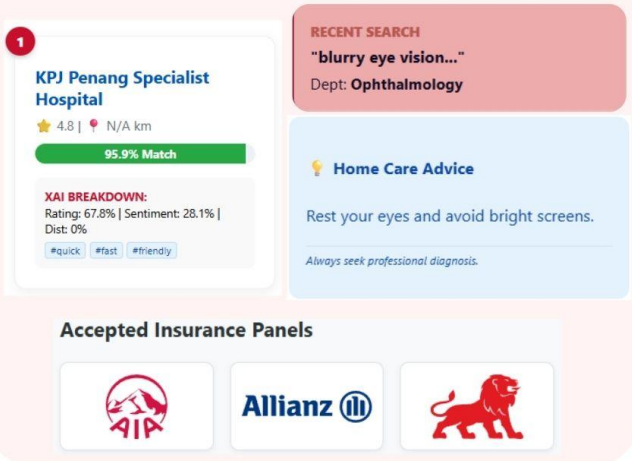
Data Collection: 10,140 reviews scraped from 15 Penang hospitals (GMB Audit Tool)

Sentiment Analysis: VADER algorithm (Positive, Negative, Neutral)

Hybrid Model:
- **SVM :** Analyze review text quality
- **SVD :** Predict ratings

SYSTEM FEATURES AND UI

The keyword **"eye"** matched the **Ophthalmology** department.



Accepted Insurance Panels

AIA | Allianz | Great Eastern

RESULT

Hybrid Model
- RMSE: 0.6313
- MAE: 0.3985
- Accuracy: 80.1%

CONCLUSION

Developed a healthcare recommender using scraped review data and a hybrid ML model. The system features location-tracking, multilingual support, and innovates through XAI to provide recommendation logic.