GRADUATE CERTIFICATE
INTELLIGENT REASONING SYSTEM (IRS)

---------- **PROJECT REPORT** ----------

OhMyFish, Your Friendly Pet Fish Veterinarian

GROUP 8

Lim Eng Hwee Jason (A0180557Y)

Tan Sio Poh (SXXXX916J)

Teo Wei Ming (A0249278M)

Teoh Jeng Wei (A0093899B)

Wang Song (A0026411X)

Table of Contents

1 Executive Summary

Preface

The Group Project is one of the key learning outcomes of the Intelligent Reasoning System (IRS) Graduate Certification and in this report, details the extensive but extremely rewarding learning journey we (***Group 8***) have undertaken to alleviate the impacts brought about by Covid-19 on the pet fish veterinary services in relation to pet fishkeeping hobbyists. This report details the problem statement and our investigative research journey for a solution through the analysis performed, approaches evaluated through trail-and-error, knowledge and insights formed, and observations made.

Problem

Covid-19 in Singapore has had a profound impact on our daily lives, completely upheaving the physical and social interactions that many have relied on all their lives for companionship. This resulting cataclysm has caused many to seek companionship through pet ownership with fishkeeping being highly popular due to the lack of licensing regulations as well as being an affordable hobby when entry-level pet fishes are concerned. Despite all the reasons making pet fishkeeping a popular hobby, the unravelling of ***limited availability*** of professional veterinary services during the pandemic coupled with ***limited published knowledge*** of pet fish care and ***limited of self-service*** options have left many hobbyists frustrated when their beloved pet fishes fall sick and eventually succumb to diseases.

Opportunity

With the pet fish imports and exports in Singapore valued at ***USD$40M in 2020***, the growing fraternity of pet fish hobbyist amidst the highly limited professional veterinary services lies a lucrative business opportunity that can be tapped into with an intelligent reasoning system built to offer self-diagnostic veterinary capabilities. The system built based on knowledge and various techniques acquired throughout the learning journey of the Intelligent Reasoning Systems (IRS) Graduate Certificate, will be able to solve the underlying problems brought about by the impacts of Covid-19 namely availability of veterinary services, accessibility of knowledge of pet fish diseases and symptoms and on-demand self-service diagnostic inquiry to enable hobbyists to triangulate on the most possible disease suffered by their pet fishes.

Solution

The proposed solution in response to the business opportunity is an intelligent reasoning veterinary ***ChatBot*** entitled ***OhMyFish*** that is built upon a knowledge base of pet fish diseases, enabling hobbyists to perform diagnostic inquires for their pet fishes suspected to be suffering from diseases through an iterative set of textual inputs describing the symptoms observed, acknowledging recommended symptoms or a combination of both until OhMyFish is able to triangulate on the most probable diseases with the highest confidence scores accumulated during the inquiry process. OhMyFish will offer two approaches to inquiry, namely ***Guided*** and ***Unguided***. Guided approach is meant for novice hobbyists with OhMyFish taking charge of the inquiry from symptom recommendations to disease triangulation while Unguided approach is meant for expert hobbyists to discover the symptoms leaving only the disease triangulation to OhMyFish.

Design and Techniques

OhMyFish is based on an ***Object-Oriented*** design built on a ***Three-Tier Architecture*** to ensure future development can easily extend the system design and to scale for future growth. Using a combination of ***Natural Language Processing***, ***Natural Language Understanding***, ***Deductive Reasoning***, ***Logic-Based Inference*** and ***Reinforced Learning***, the underlying design of OhMyFish involves a ***Four-Stage Process*** consisting of ***Text Pre-Processing*** (Input Processing), ***Topic Modelling*** (Input Understanding),

Knowledge Base Inquiry (Response Selection) and ***Decisioning*** (Action Plan Generation and Response) [1].

Given OhMyFish is a chatbot that primarily processes textual data where such data in its native form is both sparse and unstructured which will require "*connecting the dots*" [2], a ***Graph Database*** has been determined to be the most suited data model to identify relationships between diseases, symptoms, treatments as well as similarity or dissimilarity [2].

A hybrid exploratory method of ***Investigative Research*** and ***Trial-And-Error*** forms the guiding principle behind an iterative process backed by a data-driven approach was adopted to determine the various techniques in consideration and select the most optimum technique to form the core reasoning intelligence of OhMyFish. To that end, a total of four techniques were comprehensively evaluated and with ***spaCy with Text Pre-Processing augmented with Custom Stopwords*** being selected as one of the core techniques underpinning the first two stages of the four-stage process. The final two stages of the four-stage process are then underpinned by ***Custom Heuristic Model*** that adopts a weighted approach whereby diseases with higher confidence are accorded higher weightages through having more associated symptoms that are unconfirmed returned for confirmation during the diagnostic inquiry.

Closing Thoughts

Through the union of design and techniques adopted to implement this ***Minimum Viable Product (MVP)*** version built upon a limited dataset, OhMyFish has been evaluated to have ***meet its intended purpose*** of being an intelligent reasoning veterinary chatbot to enable hobbyists to perform diagnostic inquires on their pet fishes with reasonable accuracy. Further, OhMyFish solves the three problems of limited availability, limited published knowledge, limited of self-service with being ***always available***, ***published knowledge base*** and ***unlimited self-service*** to all hobbyists.

And while there is good value in this MVP version, there are obvious limitations that inhibits a tremendous amount of untapped potential amidst a lucrative business opportunity in Singapore that can be unlocked with future development on OhMyFish to unlock further capabilities. These future development opportunities include ***Speech Recognition (Speech-to-Text and Text-to-Speech)*** for a genuine conversational capability, ***Computer Vision*** to enable the usage of images and videos of pet fishes, ***Sentiment Analysis*** to evaluate the usefulness of the diagnostic inquiry to the hobbyist, ***3rd Party Integrations*** to applications like Telegram and Whatsapp, ***Plugins to External Knowledge Bases*** such as Quora and ***KPI Measurements*** to measure performances. There remains in OhMyFish and this domain, much room for further research and future development to improve, and limitless untapped potential.

2 Introduction

2.1. Problem Statement

The onset of the once-in-a-lifetime Covid-19 pandemic has brought about a massive wave of border closures and lockdowns. In a similar fashion, the Circuit Breaker which is Singapore's version of a lockdown had all but eliminated everyone's ability to participate in physical social interactions with others. The protracted pandemic lasting two years and counting coupled with the tough laws policing physical social interactions has had many Singapore residents turning to pet ownership as a hobby to seek companionship [3], as alternative to the warmth of human-to-human interaction or to divert their attention from the depressive news on which otherwise would have been spent on face-to-face and social interactions with others.

And among the varying choices of pet ownership for companionship during the socially restrictive pandemic, pet fishkeeping is growing in popularity among the younger as a hobby [4] that has already been widely popular among the older even before the pandemic as illustrated in Figure 2.1-1 below showing the popularity of pet fishkeeping being only second to pet dog-keeping in 2018.

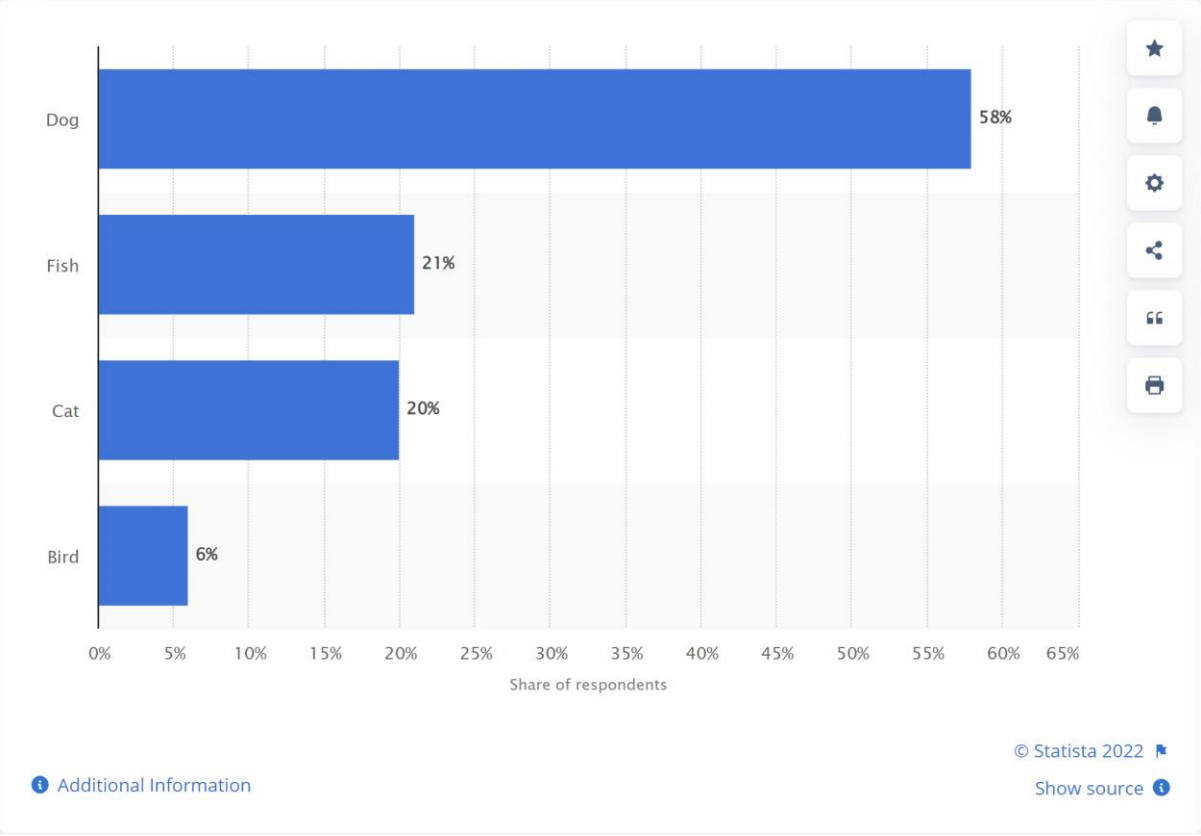


Figure 2.1-1. Bar chart showing the breakdown of pet hobbyists by types in Singapore [5]

Despite the growing fraternity of pet fishkeeping hobbyists in Singapore, a search on Google revealed the existence of only two veterinary clinics offering diagnostic and treatment services for pet fishes. This highly limited availability of veterinary services for pet fishes coupled by the regulations on social distancing and veterinary services [6] has exacerbated hobbyist's access to essential diagnostic and treatment services for their pet fishes.

2.2. Business Opportunity

And with the world still in a laborious push towards a collective declaration of victory against the Covid-19 pandemic and a full return to normalcy in Singapore lacking further behind, the growing and pressing

demand for pet fish veterinary services in Singapore presents a massively lucrative business opportunity to solve the following problems for hobbyists who are willing to pay hefty sums for pets [7]. Figure 2.2-1 and Figure 2.2-2 shows the observed values of pet fish imports and exports in Singapore valued at more USD$10m and USD$30m respectively in 2020.

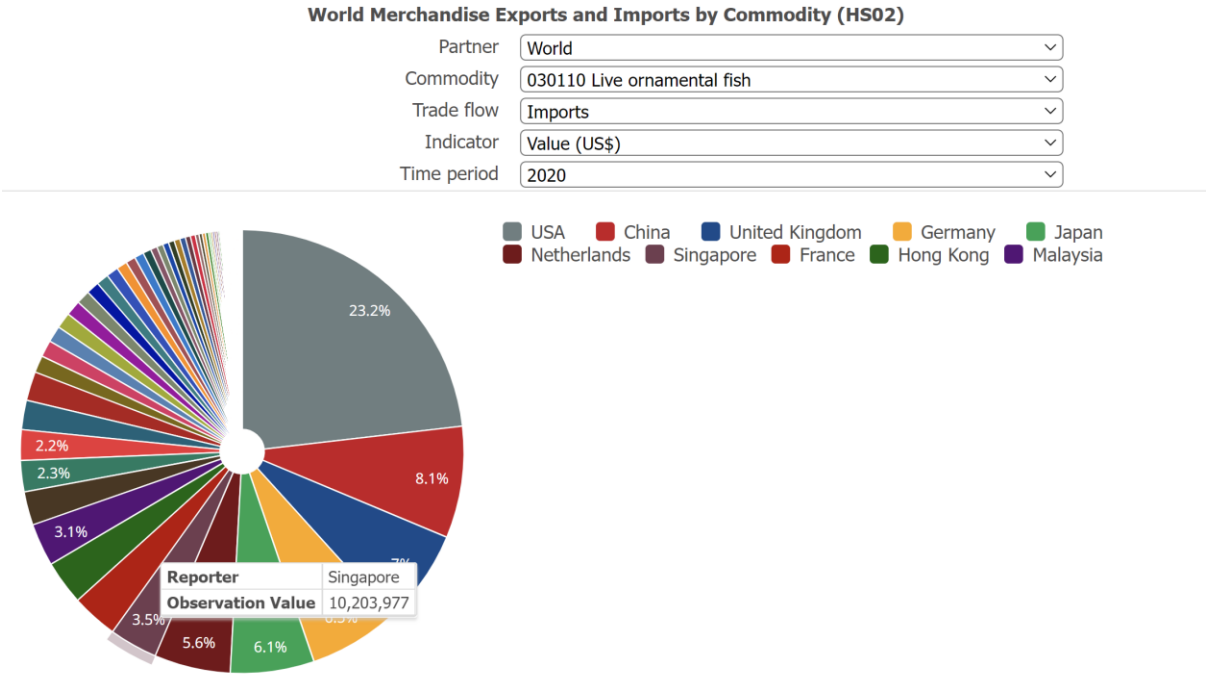


Figure 2.2-1. Pie chart showing the import values in USD$ of pet fish by country in 2020 [8]

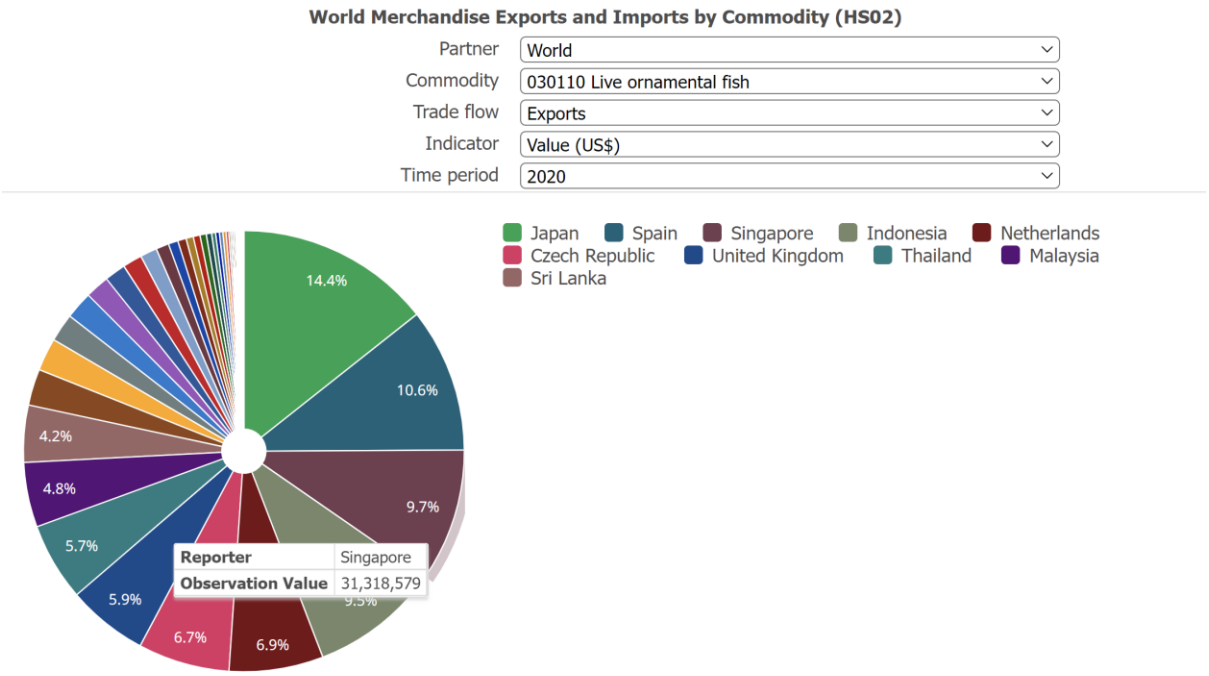


Figure 2.2-2. Pie chart showing the export values of pet fish by country in 2020 [8]

2.2.1. Availability

The costly [9] yet limited availability of pet fish veterinary services against the backdrop of a growing fraternity of pet fishkeeping hobbyists is expected to increase the demand for such services on the backdrop of the on-going Covid-19 pandemic. This growing demand can be addressed with an online digital service that will break down all limits of availability and ensure all hobbyists are always able to obtain access to veterinary services for their pet fishes.

2.2.2. Accessibility

Despite a population of hobbyist, accessibility to essential knowledge on pet fish care is limited due to an overabundance of information without proper organization and management leading to new hobbyist being unable to make appropriate decisions when their pet fishes are sick. This accessibility can be remediated with an intelligent knowledge base that focuses on the continual ingesting, filtering, organizing, aggregating and optimizing essential knowledge on pet fish care. This knowledge base will enable hobbyists to discover diagnostic information for their pet fishes and obtain treatment options.

Easing the accessibility to essential knowledge will also alleviate the pressure of crowd control on the two veterinary clinics and consequently, reducing the potential of Covid-19 transmission chains.

2.2.3. Self-Service

With the complete relaxation of regulations still nowhere in sight and coupled with limited yet costly pet fish veterinary services, hobbyists have had to often resort to trial-and-error methods in treating their pet fishes which may not always yield the intended outcome. The problem can be alleviated with the advent of a self-service expert diagnostic service built on an intelligent knowledge base that will provide hobbyists with a guided approach to conduct a self-service diagnostic analysis on their pet fishes.

2.3. Proposed Solution

In considering the business opportunity presented by the problems faced by pet fishkeeping hobbyists, the proposed solution is the development of an online Chatbot service built with an intelligent reasoning system that will interact with a pet fish disease knowledge base to provide hobbyists with a guided approach in conducting self-service expert diagnostic analysis on their pet fishes to triangulate on the most probable diseases for the evaluation of treatment strategies. This service will be entitled **OhMyFish** to reflect the sentiments of hobbyists when their pet fishes are sick.

OhMyFish will specifically target two groups of hobbyists, namely the novices and experts. The former has been brought about by the Covid-19 pandemic that caused a surge in novice hobbyists of pet fishkeeping [10] while the latter are formed by the steady stream of expert hobbyists. Considering the targeted groups of hobbyists represent two ends of the spectrum in terms of experience on pet fishkeeping, OhMyFish will enable two forms of self-service diagnosis inquiry to cater to novices with a Guided Approach and to experts with an Unguided Approach. The Guided will guide the novices by listing the symptoms systemically during the diagnosis inquiry while UA will allow the experts to articulate the symptoms to chart their own diagnosis.

Novice hobbyists tend to gravitate towards pet fishes at the entry levels in terms of pricing while expert hobbyists gravitate towards the top end. The entry groups typically comprise of Betta Fish and Gold Fish while the top end groups comprise Flowerhorn Cichlid and Asian Arowana as shown in Table 2.3-1 and Table 2.3-2.

Entry	
Betta Fish	Gold Fish
	
Source: https://www.fishbazaar.sg/product/female-betta-fish-assorted/	*Source: https://www.fishbazaar.sg/product/red-oranda-goldfish/*

Table 2.3-1. Table showing entry level pet fishes

Top End	
Flowerhorn Cichlid	Asian Arowana
	
Source: https://www.sunbeamaquarium.com/	*Source: https://www.fishbazaar.sg/product/red-oranda-goldfish/*

Table 2.3-2. Table show top end pet fishes

3. Design

3.1. Conceptual

OhMyFish leverages a four-stage process adopted by many chatbots [1] consisting of Text Pre-Processing (Input Processing), Topic Modelling (Input Understanding), Knowledge Base Inquiry (Response Selection) and Decisioning (Action Plan Generation and Response). Figure 3.1-1 below illustrates the conceptual model of OhMyFish and how the four-stage process works.

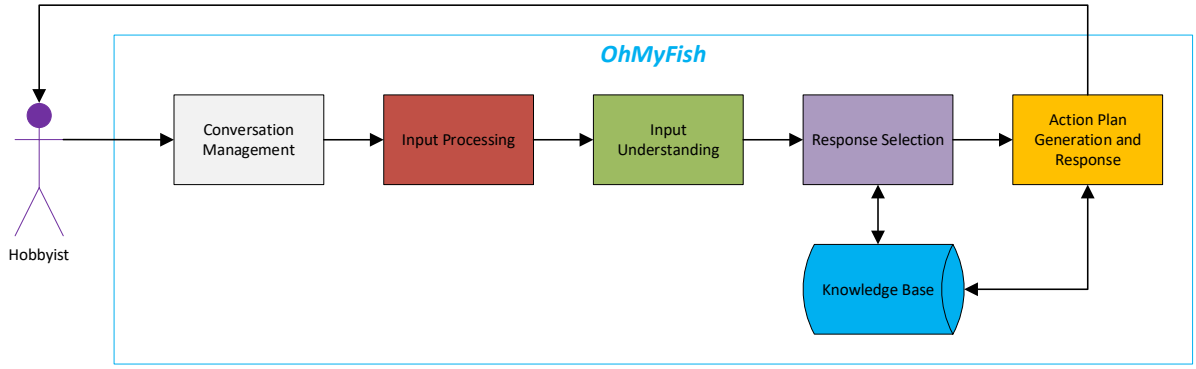


Figure 3.1-1. Figure illustrating the conceptual mode of OhMyFish [1]

3.2. Architecture

With the primary focus to enable application decouplability and potential to scale, OhMyFish adopts an object-oriented approach built upon a Three-Tier Architecture. OhMyFish is organized into three logical tiers consisting of Presentation, Application and Data as illustrated in Figure 3.2-1.

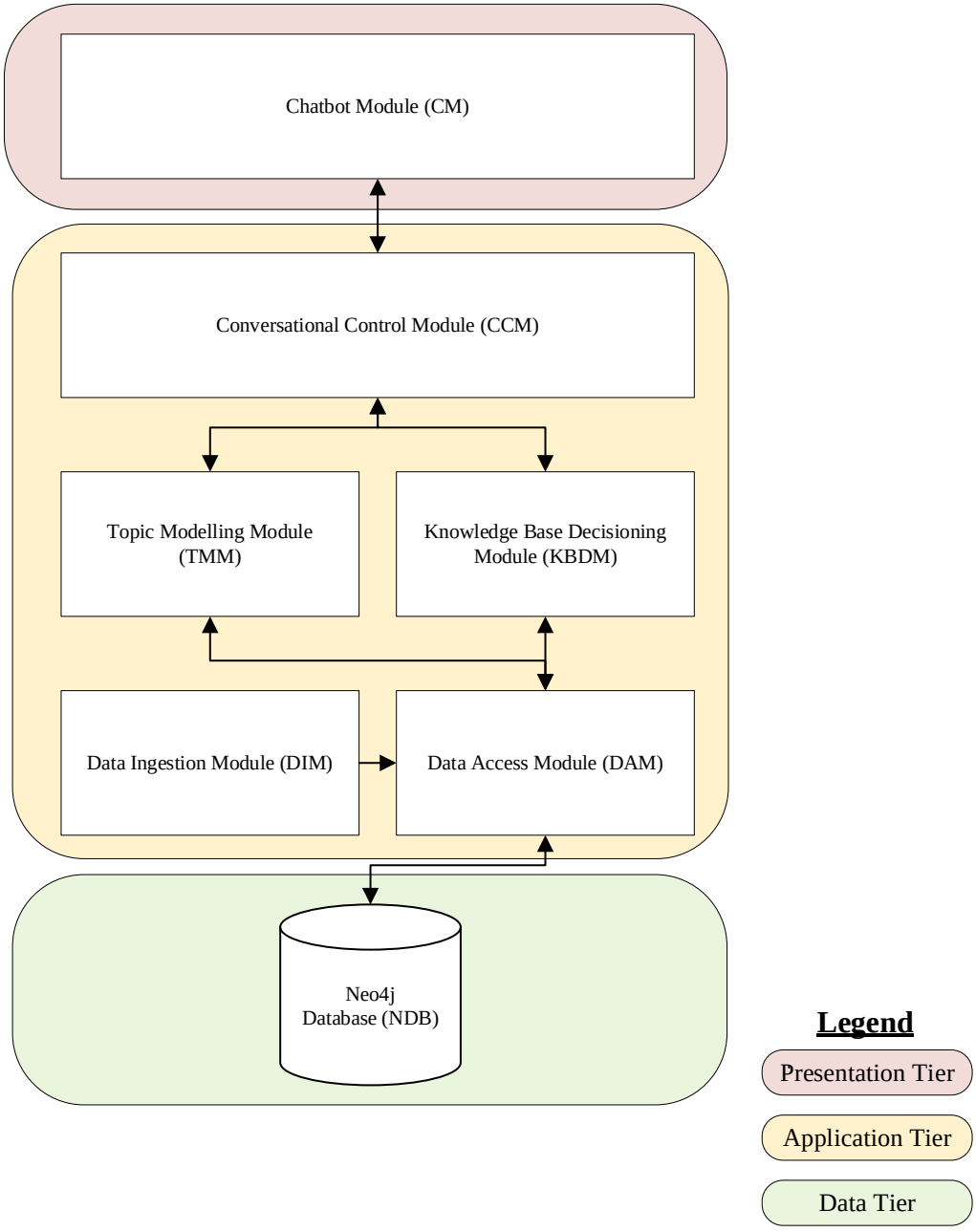


Figure 3.2-1. Diagram depicting the system architecture of OhMyFish

3.2.1. Presentation Tier

The user interface of OhMyFish which is the chatbot, is primarily the Presentation Tier described as follows.

A. Chatbot Module (*CM*) functions as the user interface of OhMyFish. During the diagnostic inquiry, CM processes all user inputs and calls CCM in the Application Tier for processing then perform presentment on responses received from CCM. CM manages each diagnostic inquiry as a case and determines if the diagnostic process follows the Guided or Unguided flow.

B. Flask API Layer - Acts as the logical separation of frontend - backend. Facade for interaction with Conversation Flow module.

3.2.2. Application Tier

All core functions of OhMyFish are built into the Application Tier which is also the primary processing layer of is and described as follows.

A. Conversational Control Module (*CCM*) functions as the gateway between CM and the rest of the modules within the Application Tier. CCM receives the user input from CM then either calls TMM or KBM for processing then returning the responses back to CM. CCM also manages the disease inference of each case through a confidence hyperparameter during the diagnostic process.

B. Topic Modelling Module (*TMM*) has two functions, namely a text pre-processer and the topic modeler. TMM receives the user input from CCM, performs text pre-processing that primarily includes stop-words removal, lemmatization and punctuation removal before calling DAM to query for symptoms. Text pre-processing is intended to reduce noise and enhance the quality of the query built from the user input before topic modelling. Further, the list of stop-words has been augmented further to include common words associated with pet fishkeeping to further optimize the quality of the query.

C. Knowledge Base Decisioning Module (*KBM*) functions as the interference decision engine, creating associations of suspected diseases to a case and applies custom heuristic rules to compute the weightages and penalties of symptoms within a disease then assign confidence scores to suspected diseases followed by retrieving other symptoms associated to the suspected diseases with the highest confidence scores.

D. Data Access Module (*DAM*) functions as the data accessor, manages all CRUD operations between the Data Tier and the rest of the modules within the application tier. DAM abstracts the nature of all nodes and edges within the Graph database then returns data as nodes and edge represented in a Knowledge Graph illustrated in Figure 3.4.2-1.

E. Data Ingestion Module (*DIM*) functions as the data ingestion framework, transforming all raw data from CSV files into the Graph database within the Data Tier. The CSV files are from data extracted via web scraping and manual sanitization.

3.2.3. Data Tier

The Data Tier consists of the following.

A. Neo4j Graph Database (NGDB) functions as the storage medium for the Fish Disease knowledge base and storing all data as nodes and edges.

3.3. Class Diagram

Expounding further beyond the system architecture and object-oriented approach in a Three-Tier Architecture, the following class diagram in Figure 3.3-1 provides a detailed illustration into the structure of OhMyFish's classes along with the associated attributes, methods, and relationships.

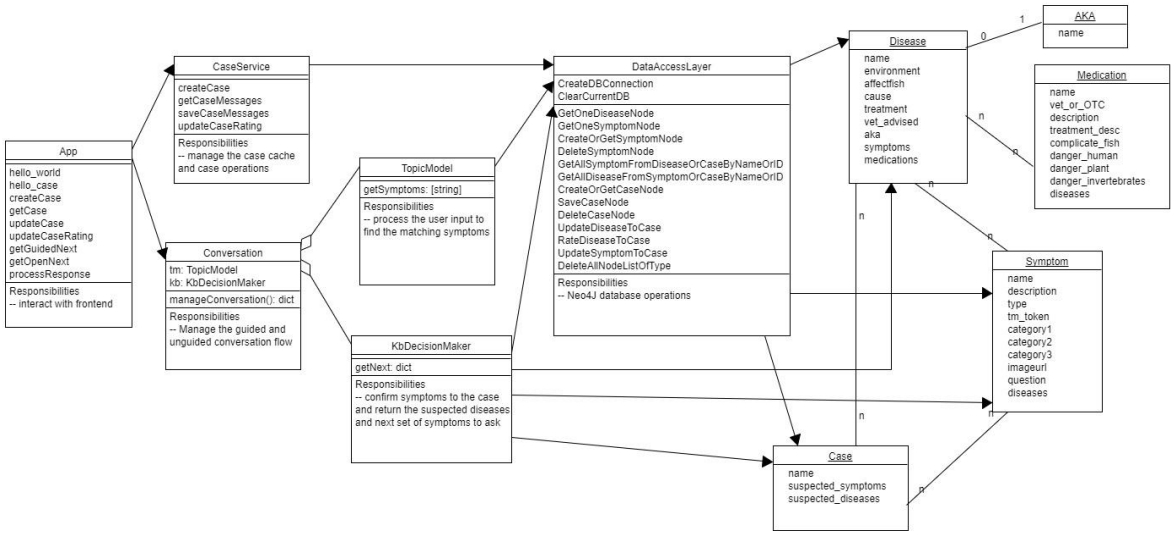


Figure 3.3-1. Class Diagram illustrating the object structure and associated relationships within OhMyFish

3.4. Data

3.4.1. Data Preparation

A knowledge base is an essential component in any successful intelligent reasoning system. For the scope of OhMyFish, the knowledge base is built through data obtained through web-scraping. [11] [12] The knowledge base will be reinforced using user input data which will be stored when a user is using the application. The reinforcement will enable the continuous improvement on the accuracy of subsequent diagnostic inquires.

The data obtained through web-scraping will include medical facts on common fish diseases as well as clinical symptoms, known causes and treatments associated with those diseases. The data will undergo manual sanitization and processed into a tabular format for effective data ingestion into the Neo4j database. Figure 3.4.1-1 below depicts the process flow of the data preparation process.

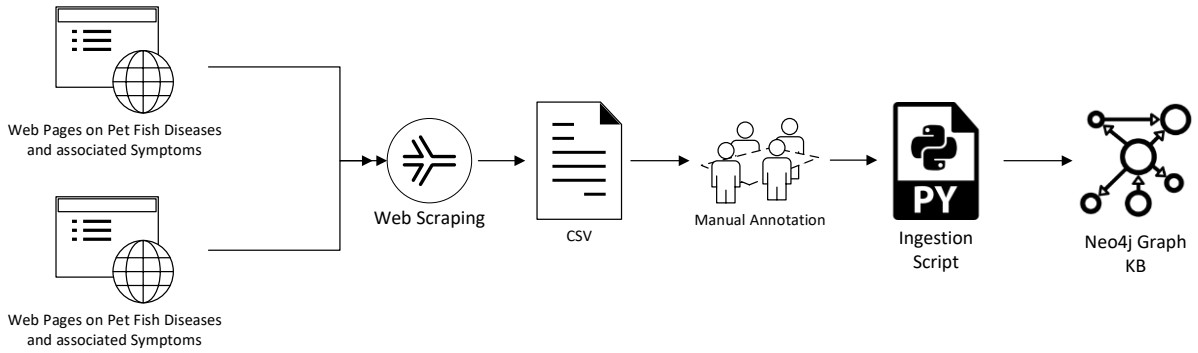


Figure 3.4.1-1. Figure depicting the flow how of the data preparation process

The data is then processed into three datasets - Disease, Symptom and Medication as shown with sample data in Table 3.4.1-1, Table 3.4.1-2 and Table 3.4.1-3.

Disease	
Columns	**Sample Data**
ID	26
Disease1	Chilodonella
Disease2	
Native_Water	Freshwater
Fish	Goldfish
Symptom1	RAPIDBREATH
Symptom2	THICKMUCUS
Symptom3	CLOUDSKIN
Symptom4	CLAMPFIN
Symptom5	LOSSAPPETITE
Symptom6	LETHARGIC
Symptom7	FLASHING
Symptom8	BLUEMUCUS
Symptom9	SWOLLENSKIN
Symptom10	
Symptom11	
Symptom12	
Causes	Like many aquatic parasites, Chilodonella usually gets into your tank on new fish introduction or with live foods.
Treatments	Most common chemicals used to treat fish infected with Chilodonella spp. include formalin (formaldehyde solution) and potassium permanganate.

Vet_Advised	FALSE
Medicine1	FORMALIN
Medicine2	POTAMAGANATE
Medicine3	
Medicine4	
Medicine5	
Medicine6	
Medicine7	
Medicine8	

Table 3.4.1-1. CSV dataset of Disease

Symptom	
Columns	**Sample Data**
ID	31
SymptomID	INFANUS
Symptom	Inflamed anus
SymptomType	Physical
SymptomCategory1	Body
SymptomCategory2	
SymptomCategory3	
Question	Is the anus region inflamed?
ImageURL	infanus.jpg
Weight	1
Penalty	-0.2
Source	https://www.researchgate.net/figure/Common-carp-swollen-anus-and-skin-ulcers_fig3_311867630

Table 3.4.1-2. CSV dataset of Symptom

Medication	
Columns	**Sample Data**
ID	1
Medicine	Acriflavine
MedicineID	ACRIFLAVINE
Vet_Or_OTC	
Treatment_Description	
Complicate_Fish	
IsDangerousHuman	
IsDangerousPlants	
IsDangerousInvertebrates	

Table 3.4.1-3. CSV dataset of Medication

3.4.2. Data Ingestion to Neo4j

The Python pandas library is used to convert the three datasets (in csv format) into Pandas DataFrames, and the native iloc functions in the pandas library are used to search through the DataFrames which are tabular data structures. The creation of nodes and relationships are done using the neo4j python library.

During the graph database setup process, the Python script will iterate through every row (instance) in the Symptom csv dataset and for every instance, creates a node of *Symptom* label in the Neo4j graph database, where the node properties include name, category1, category2, category3, description, question, imageurl and type.

The Disease csv dataset associates and stores all the different foreign keys of other tables. Similarly, the Python script will create a node of *Disease* label in the Neo4j graph database for every instance in the Disease csv dataset, where the node properties include name, affectfish, cause, environment,

treatment and vet_advised. Thereafter, the Python script will create a *isDetectedIn* relationship from the Symptom nodes to the associated *Disease* nodes and *hasSymptom* relationship from the *Disease* nodes to the associated *Symptom* nodes by iterating through the columns of the Disease csv dataset. *IsDetectedIn* relationship has a property called weight which is set to a default value of 1 as there is insufficient data on probabilistic weight or likelihood of occurrence of the symptoms for a given disease. For the same reasons, the *hasSymptom* relationship has a property called weight which is also set to a default value of 1. To improve the reasoning model's performance, an additional property, defined as penalty, is added to the *hasSymptom* relationship and the penalty is set to a default value of -0.2. This property synthetically reduces the confidence of the disease diagnosis when an associated symptom is absent, which helps to improve the accuracy and speed of the diagnosis.

In addition, some diseases have alternative names which are available in the Disease csv dataset. For every alternative disease name, a node of *AKA* (Also-Known-As) label will be created in the Neo4j graph database. An *AKA* relationship from the *Disease* node to the associated *AKA* node will also be created.

Finally, the Python script will iterate through the rows of the Medicine csv dataset and for every instance, creates a node of *Medication* label in the Neo4j graph database, where the node properties includes name, complicate_fish, danger_human, hanger_invertebrates, danger_plant, description, treatment_desc and vet_or_OTC.

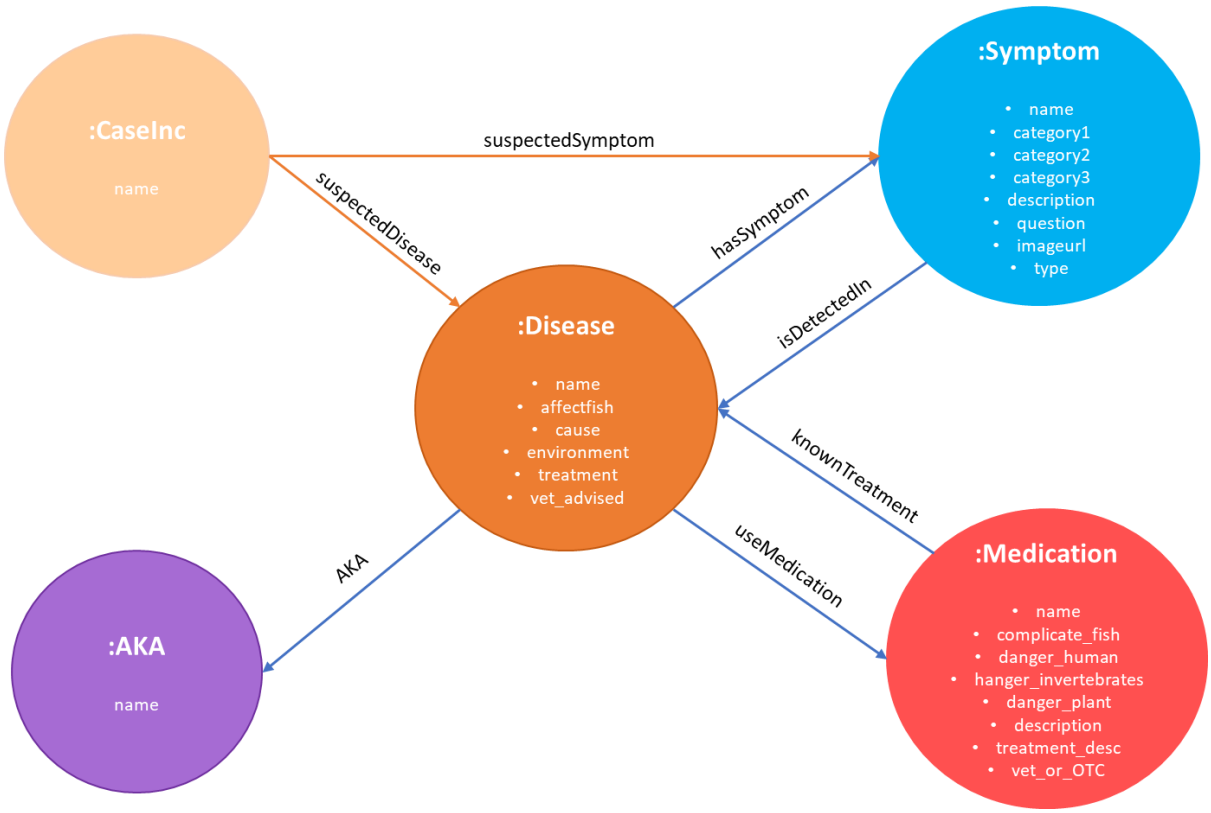


Figure 3.4.2-1. Data Model depicting the Knowledge Graph of OhMyFish

Figure 3.4.2-2. Sample snippet of Knowledge Graph in the Neo4j graph database of OhMyFish

Figure 3.4.2-2 depicts a snippet of the knowledge graph created in the Neo4j graph database. In this single disease-focussed snippet, the *Disease* node is representing "Freshwater Velvet Disease", with an associated *AKA* node with the name "Gold Dust Disease" and it has 9 *Symptom* nodes associated to it, with the corresponding ***isDetectedIn*** and ***hasSymptom*** relationship. The disease node is also associated with three *Medication* nodes, with the corresponding ***knownTreatment*** and ***useMedication*** relationship. Figure 3.4.2-3 depicts the full knowledge graph created in the Neo4j graph database, which shows all *Disease*, *Symptom* and *Medication* nodes and their associated relationships which is parsed from the respective datasets.

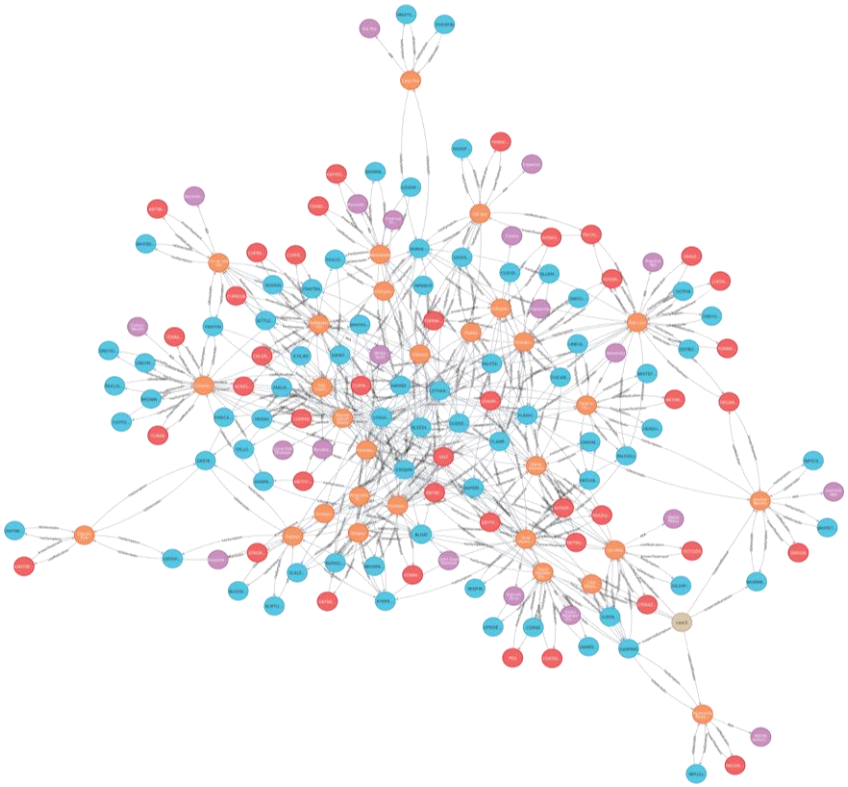


Figure 3.4.2-3. Complete diagram of the Knowledge Graph in Neo4j graph database of OhMyFish

3.4.3. Neo4j to Internal Python Objects

Using Python's Neomodel object mapping libraries, the Neo4j nodes are mapped to class objects which the solution manipulates for logical processing.

3.4.4. Saving User Case Diagnosis to Neo4j for Expansion of Knowledge Graph

One limitation of the current reasoning model is the fact that the weight and penalty values of all ***hasSymptom*** relationships in the Knowledge Graph are set to a default value of 1 and -0.2 respectively as there is currently insufficient data on the probabilistic weight or likelihood of occurrence of the symptoms for a given disease. By setting the likelihood of occurrence of all symptoms to be equal for a given disease, this limits the reasoning model's ability and performance in narrowing down the list of possible diseases efficiently as the reasoning model would need to iterate through several symptoms. In reality, there might be a case in which one of the symptoms identified is a sufficiently strong indicator of a particular disease, thus, checking the other symptoms is not needed.

By capturing user input data when the user is using the chatbot, the user generated responses and the suspected disease diagnosis will be stored as case data. Such data will persist as a "suspected instance of a disease" in the database. This is to simulate the real-life scenario in which a hospital or doctor keeps a record of each patient and these records are used by the doctors to aid them in providing more precise and accurate diagnosis.

When a substantial number of case instances is generated over a period of time, the case dataset can be aggregated to learn the probabilistic weight between all the diseases and the associated symptoms. Such weightage will be used for:

A. Directing the chat bot to ask questions related to the heavier weighted symptoms.
B. Influencing the confidence of an attempt at disease diagnosis

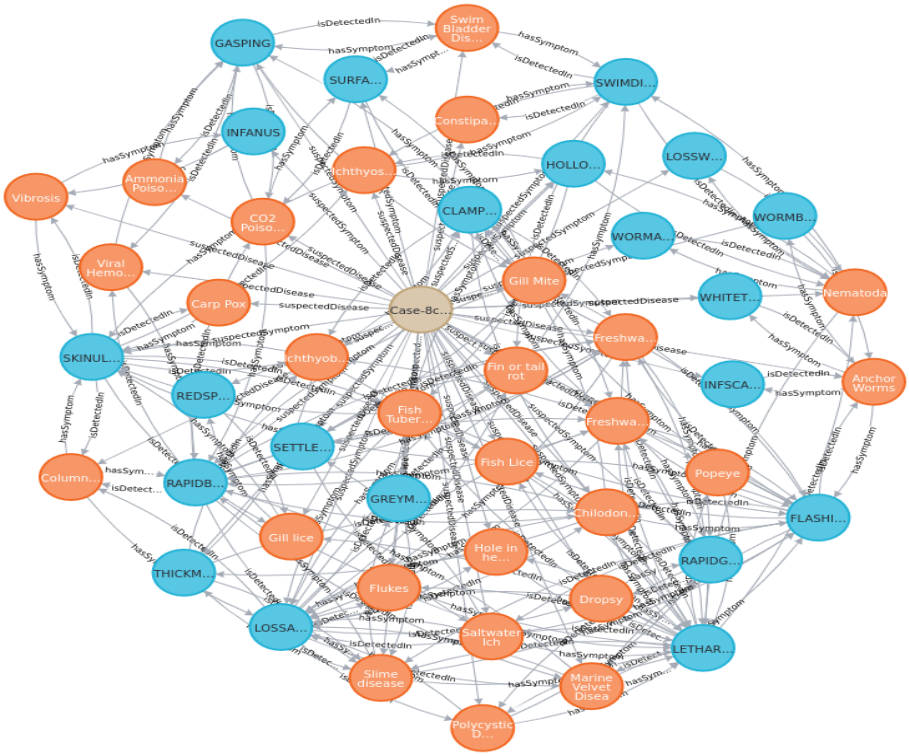


Figure 3.4.4-1. Diagram of the full Knowledge Graph in the Neo4j graph data of OhMyFish

Figure 3.4.4-1 depicts an example of a logged case after a user has completed a session. When a user confirms the presence of a symptom, a *suspectedSymptom* relationship between the *Case* node and the *Symptom* node is created and the suspectedLevel attribute of the relationship is set to 1. Upon user's confirmation of the presence of a symptom, the algorithm (which will be further explained in section 3.6.2) running in the background will create the corresponding *suspectedDisease* relationship between the *Case* node and *Disease* node, calculate the confidence level based on the symptoms confirmed thus far and assign this value to the confidence attribute of the *suspectedDisease* relationship. All of the created *suspectedSymptom* and *suspectedDisease* relationships as well as the suspectedLevel and confidence attribute values are logged in this way for future expansion of the knowledge base.

3.5. Flow Diagram

OhMyFish has two forms of self-service diagnosis inquiry, namely the Guided Approach for novices and Unguided Approach for experts. The Guided Approach as it is called, will guide the hobbyist by recommending symptoms of suspected diseases with the highest confidence scores at each iteration of the diagnosis inquiry whereas the Unguided Approach will allow the hobbyist to deviate from the recommended symptoms partially or completely by recommending symptoms of suspected diseases while also defining additional symptoms that may not be associated with any of the suspected diseases during the diagnosis inquiry.

The diagram in Figure 3.5-1 below illustrates the process flow of OhMyFish, depicting the variation between Guided and Unguided Approaches.

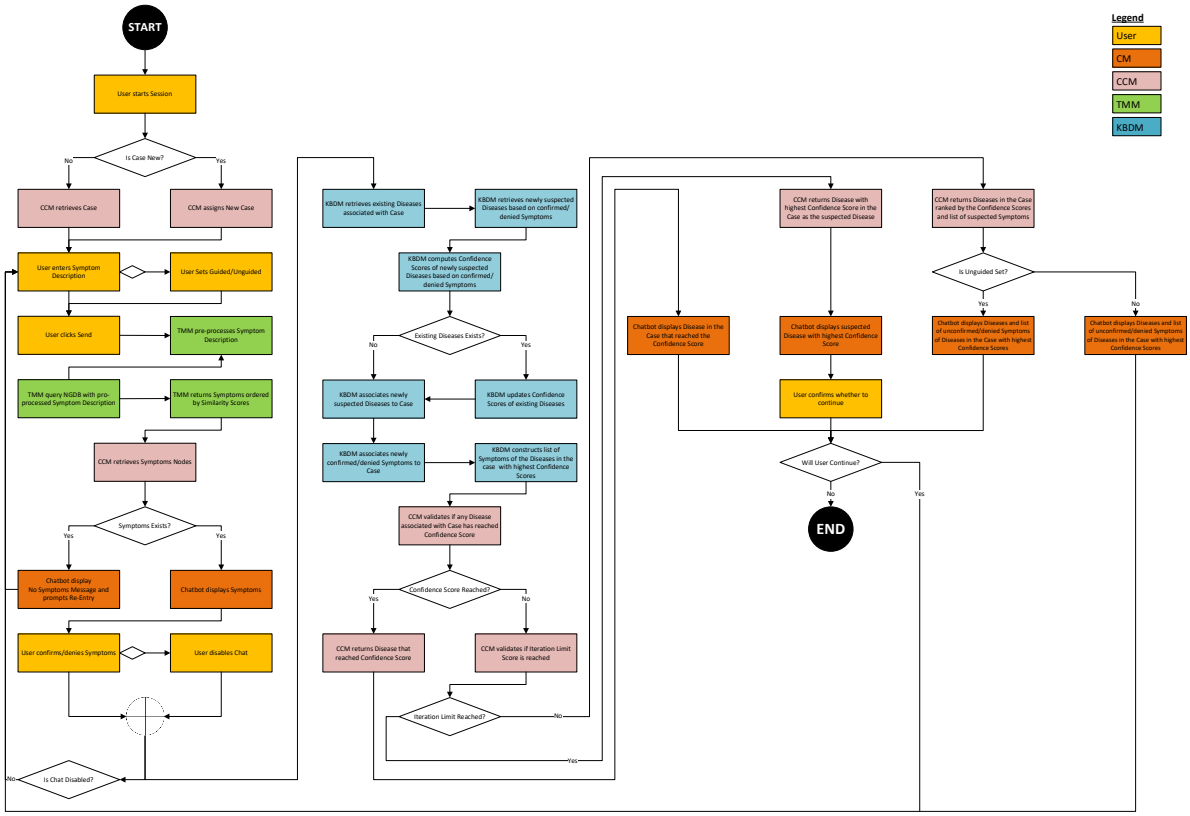


Figure 3.5-1. Flow diagram demonstrating the application flow of OhMyFish

3.6. Sequence Diagrams

3.6.1. Guided Approach

Figure 3.6.1-1 depicts the Guided Approach where it guides the hobbyist by recommending symptoms of suspected diseases with the highest confidence scores at each iteration of the diagnosis inquiry.

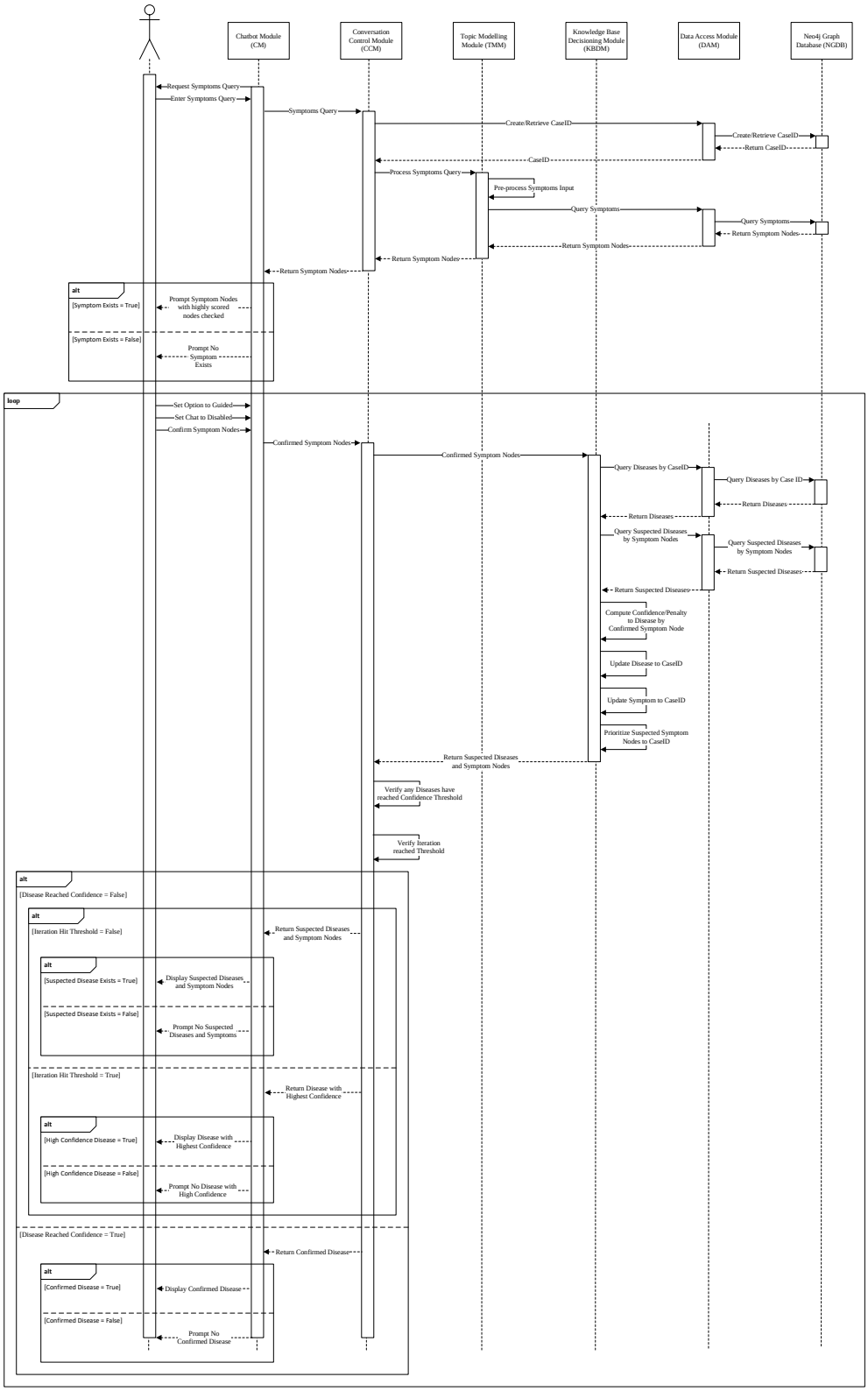


Figure 3.6.1-1. Sequence diagram demonstrating the object interactions of the Guided Approach

3.6.2. Unguided Approach – Expert

Figure 3.6.2-1 depicts the Unguided Approach where it allows the hobbyist to deviate from the recommended symptoms partially or completely by recommending symptoms of suspected diseases while also defining additional symptoms that may not be associated with any of the suspected diseases during the diagnosis inquiry.

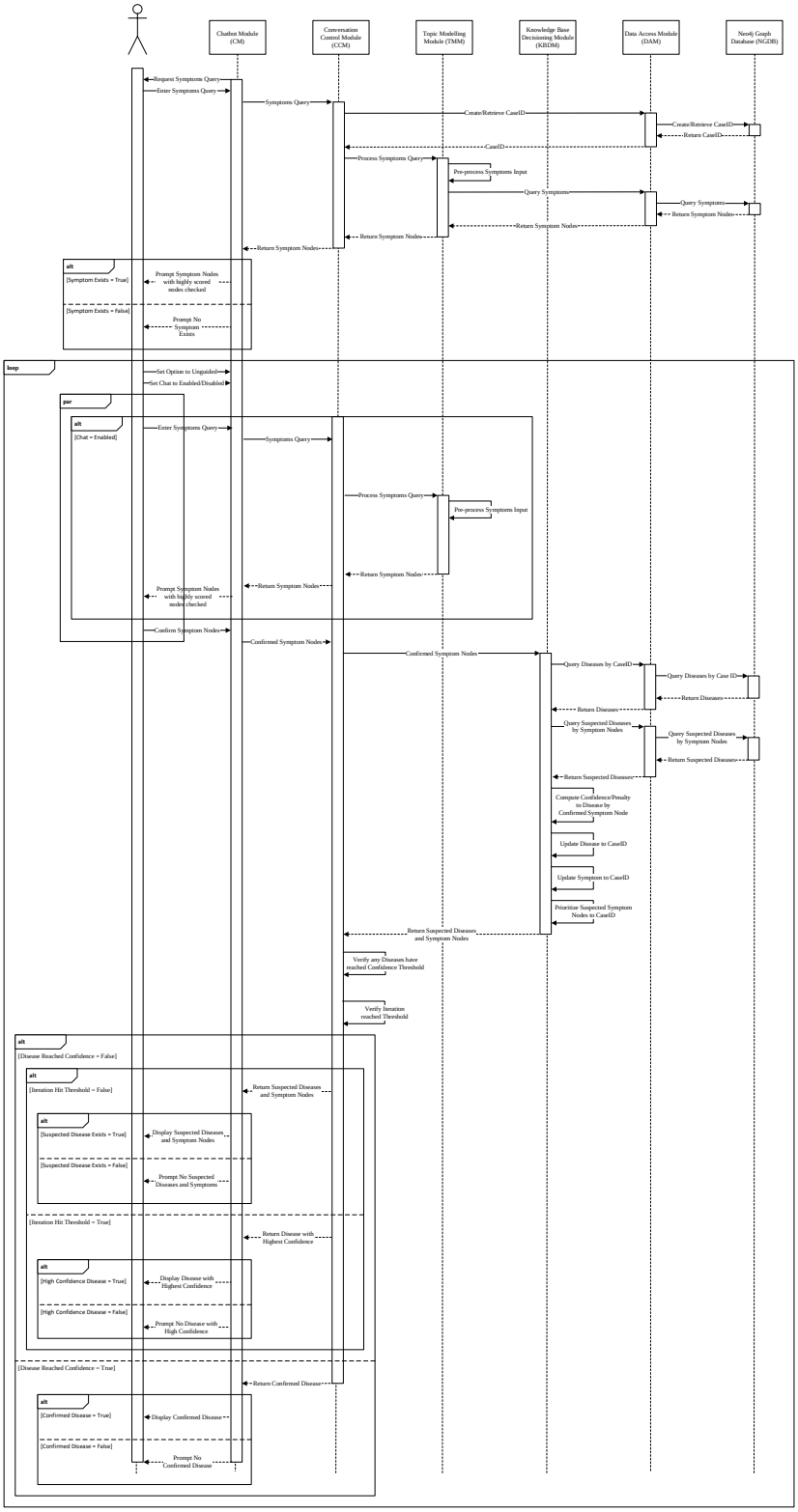


Figure 3.6.2-1. Sequence diagram demonstrating the object interactions of the Unguided Approach

3.7. Core Reasoning Intelligence

This section explains the approaches that we adopted to implement the core reasoning intelligence of **OhMyFish**. It details the process and explorations that were taken to determine the best technique that we can use to produce results that have a level of confidence that is highly reliable.

3.7.1. Topic Modelling Module (TMM)

3.7.1.1. Structure of Graph Database for Fish Diseases

Before we explain about the technique that we used to extract the most likely disease(s) from the user input/query, it is important to understand how the structure of our graph database for Fish Diseases helps to achieve this objective.

In the most basic form of the database, we have linked one or more symptoms to a particular disease through the ***hasSymptom*** and ***isDetectedIn*** relationships as illustrated in Figure 3.7.1.1-1 below.

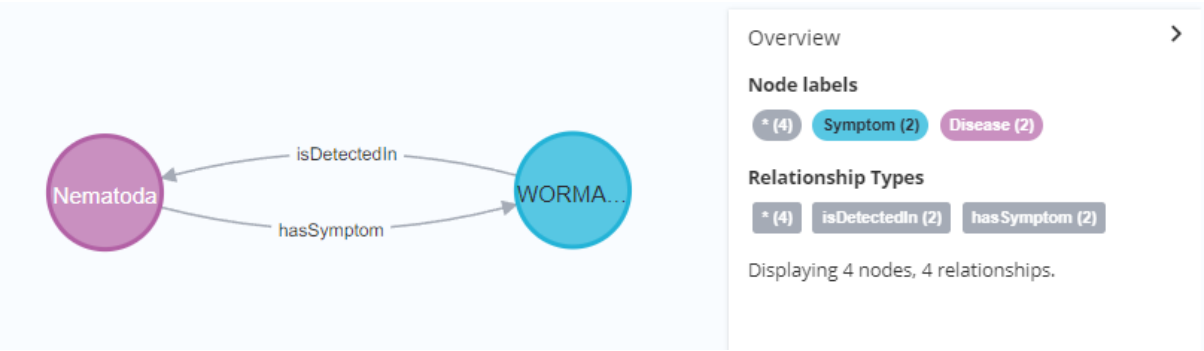


Figure 3.7.1.1-1. Basic structure of the graph database for Fish Diseases

In the above figure, the blue circle node denotes a **Symptom**, and the purple circle node denotes a **Disease**. The text appearing in the circle are the short name of the symptom and the name of the disease respectively. Each node contains a set of properties which are respectively listed in Figure 3.7.1.1-2 and Figure 3.7.1.1-3 below.

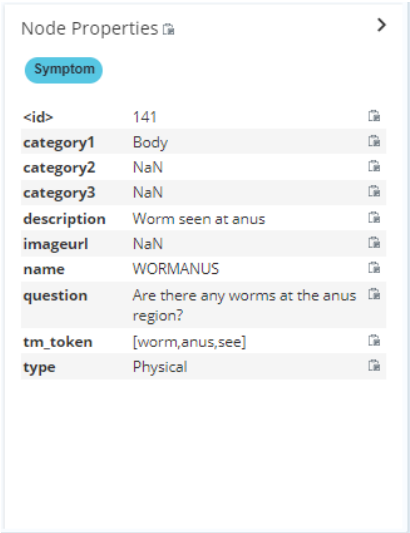


Figure 3.7.1.1-2. Properties of a Symptom

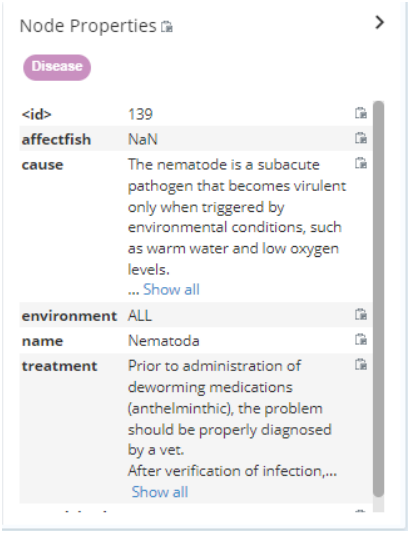


Figure 3.7.1.1-3. Properties of a Disease

The main property in the Symptom node that is involved in getting the most likely disease(s) is the **description**. The **description** of all symptoms is retrieved to compare with the user input to determine the highest similarity, and the linked disease(s) are subsequently obtained through the relationship with the symptom(s). Likewise, the medication that is associated with the disease is also retrieved through the ***knownTreatment*** and ***useMedication*** relationships with each other.

Figure 3.7.1.1-4 below illustrates a fuller example of how symptom(s), disease and medication are linked with each other. The medication is denoted by a red circle node.

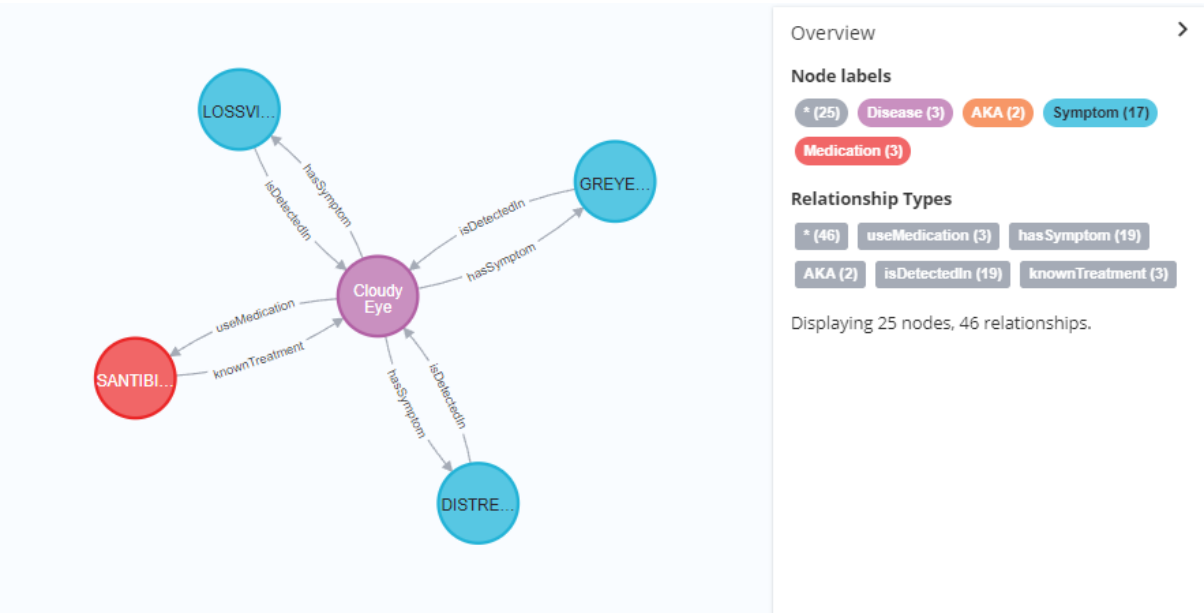


Figure 3.7.1.1-4. Figure illustrating a complete example of relationship between nodes

3.7.1.2. Finding the Best Technique

A few techniques have been explored to determine the optimum result that can be attained in getting the most relevant disease(s) (in relation to the symptom(s)) with the highest confidence level. This is done by exploring different ways to compare the similarity between the sentence(s) input by the user and the description of the symptoms that we have in the graph database.

Natural Language Processing relies largely on similarity in highly dimensional spaces where some text will be processed to create a big vector or array followed by several transformations being performed.

A. Sentence Transformer with Bert and Cosine Similarity

The first technique that we explored was using BERT (Bidirectional Encoder Representation from Transformers). BERT is a transformer-based machine learning technique for natural language processing pre-training developed by Google.

BERT can embed the meaning of words into densely packed vectors. It makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google [13].

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word) [13].

The process of using BERT can be summarised as follow:

a. Take the sentence from the user input and convert it into a vector
b. Retrieve the description of all symptom nodes from our graph database, and convert each of them into vectors
c. Compare the user sentence with each of the description of the symptoms and find the smallest distance (Euclidean) or smallest angle (Cosine Similarity) between them
d. Obtain the result of the measure of semantic similarity between sentences from above

To implement this, we used the **sentence-transformers** Python library which wraps the above process into a few lines of codes. This library uses the HuggingFaces's transformers [14] which provides pre-trained models that can be used for sentence/text embedding generation. We tested with two pre-trained models. The first pre-trained model that we tested with is the **bert-base-nil-mean-tokens** model.

An example of the implementation is shown in Figure 3.7.1.2-1.

```
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity

model = SentenceTransformer('bert-base-nli-mean-tokens')
sentenceFromUser = "My fish has white string from its anus"
sentences = [sentenceFromUser]

dbcon = DataAccessLayer().CreateDBConnection
allSymptomNodes = dbcon.GetAllNodeListOfType('Symptom')

# allSymptomNodes contains all symptom nodes retrieved from the graph database
for symptomNode in allSymptomNodes:
    # add description from each symptom node into the sentences list
    sentences.append(symptomNode.description)

# Convert the sentences to vectors by calling model.encode()
sentence_embeddings = model.encode(sentences)

# Call cosine_similarity to compare user sentence with every symptom description to get similarity score
similarity = cosine_similarity([sentence_embeddings[0]], sentence_embeddings[1:])
```

Figure 3.7.1.2-1. Code snippet showing the implementation with BERT and Cosine Similarity

We used the following sentence as a user input to simulate how a user describes a worm at the fish anus:

"*My fish has string from its anus*"

In the above code, the **encode** function is called to compute the sentence embeddings and obtain the **Numpy** vectors. Since the first sentence is the input from user, we compare it with the rest of the sentences in the list by calling the **cosine_similarity** function from the **scikit-learn** library.

The cosine similarity calculates similarity by measuring the cosine of angle between two vectors. The formula of this calculation is shown in Equation 3.7.1.2-1 below where variables A and B denote the vector of two sentences respectively:

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

Equation 3.7.1.2-1. Mathematical equation of Cosine Similarity

Cosine Similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity [14].

The result of the execution of the above codes is shown in **Error! Reference source not found.**.

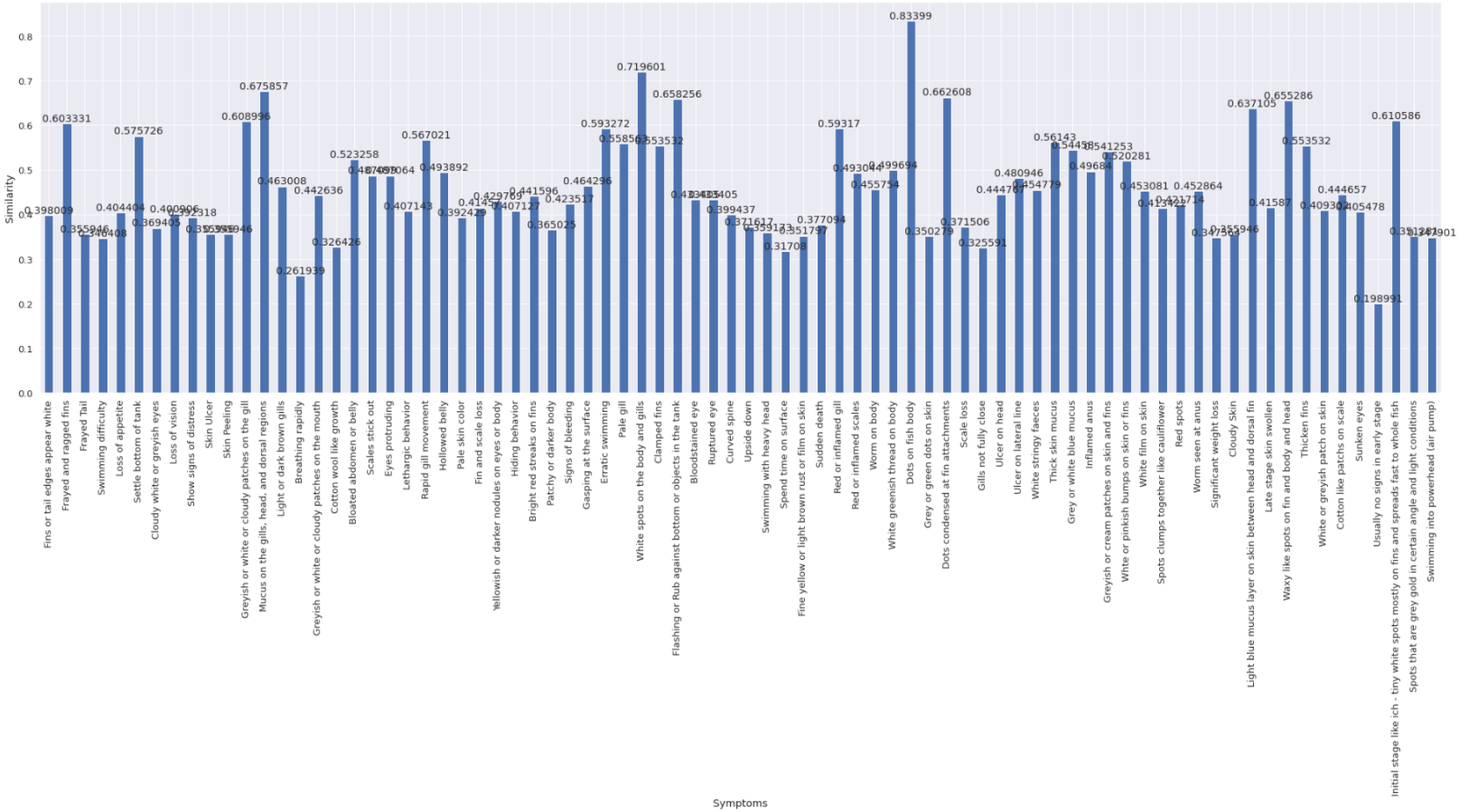


Figure 3.7.1.2-2. Bar chart showing the results using model "bert-base-nli-mean-tokens"

The above bar chart shows the highest similarity score to be 0.83399 with the symptom description: "*Dots on fish body*". The second highest score was 0.719601 with the symptom description: "*White spots on the body and gills*". What should be the most similar description is supposed to be "*Worm seen at anus*" but it only has a low score of 0.452864. None of the first three results has anything to do with worm or anus. On further investigation, it was found that the pre-trained model, **bert-base-nli-mean-tokens** is deprecated as it produces sentence embeddings of low quality [15].

A different pre-trained model was then tested using **LaBSE**. **LaBSE** is a language-agnostic BERT sentence embedding model that supports 109 languages. The result is shown on below:

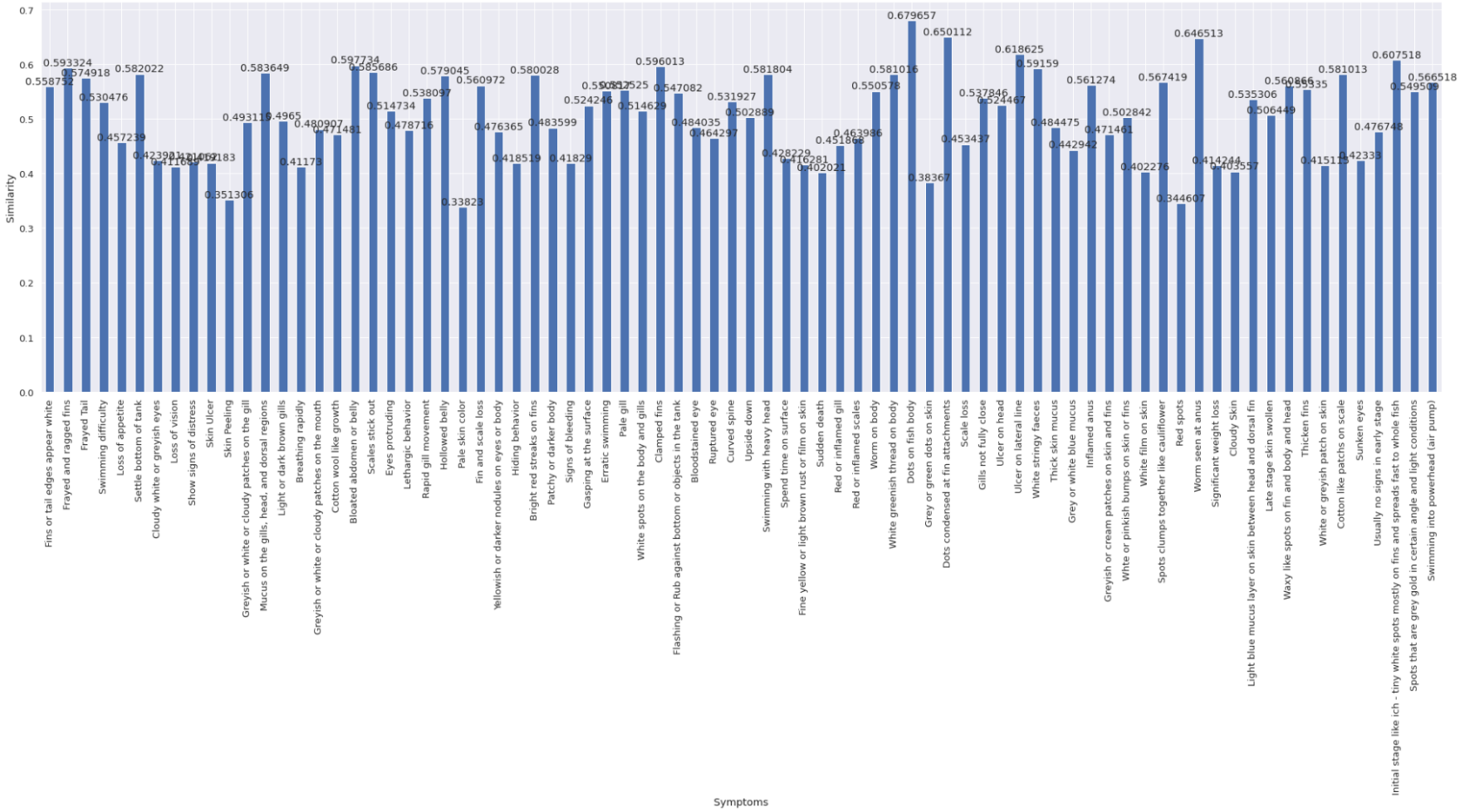


Figure 3.7.1.2-3. Bar chart showing the results using model "LaBSE"

The above results shows that the top score is still the symptom description: “Dots on fish body” at 0.679657. At second place is the description: “Dots condensed at fin attachments” at 0.650112. However, the symptom description: “Worm seen at anus” has a higher score of 0.646513. A good improvement from using the previous pre-trained model. Yet, none of the first two results has anything to do with something white or worm or anus.

Despite having a better result with “Worm seen at anus”, we are still not satisfied because this will greatly affect the confidence level of the solution. We then move on to the next exploration using **Sentence Transformer with Bert, LDA and Cosine Similarity**.

B. Sentence Transformer with Bert, LDA and Cosine Similarity

In the previous exploration, we understand that we were using the raw sentences to convert into vectors using the sentence transformer. We then thought of the idea to extract the topic from the sentences before plugging it into the transformer in the hope of getting better results. Such an approach might be able to narrow down the meaning of the sentences and improve the similarities. We then proceed to experiment with topic modelling.

There are existing algorithms available for us to implement topic modelling such as **Latent Semantic Analysis** (LSA/LSI), **Probabilistic Latent Semantic Analysis** (pLSA), and **Latent Dirichlet Allocation** (LDA). Since LDA was taught in this course, we decided to use this.

LDA is a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities [16].

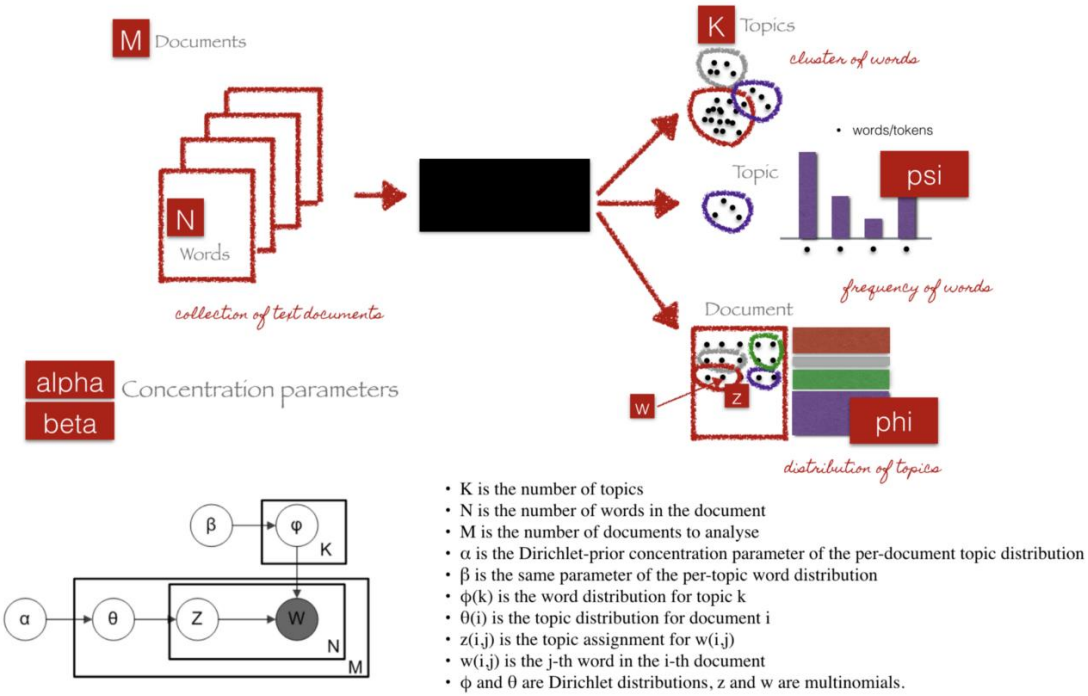


Figure 3.7.1.2-4 Latent Dirichlet Allocation (LDA) [17]

We can describe the generative process of LDA as, given the *M* number of documents, *N* number of words, and prior *K* number of topics, the model trains to output [16]:

psi, the distribution of words for each topic *K*

phi, the distribution of topics for each document *i*

Our objective was to extract the topic(s) of two sentences and compare their similarity. We used the ***LatentDirichletAllocation*** algorithm from the **scikit-learn** library. The process of implementation was as follow:

a. Load the sentence
b. Perform text pre-processing on the sentence
c. Train the LDA model
d. Get the top 10 keywords and join them into a sentence.
e. Do this for the user input sentence and every symptom description
f. Load the sentence transformer model (in this case it will be **LaBSE**)
g. Use the **cosine_similarity** function from **scikit-learn** to compare the newly generated user sentence with each of the newly generated symptom description
h. We also timed the whole process to see how fast (or slow) it will take since we are now adding additional steps.

There are 3 main parameters to set when using the LDA technique [18]:

- *n_components* – defines the number of topics. Since we foresee that the sentence will not be long, we set this to 1. This is also considering that every symptom description is based on one particular topic.
- *max_iter* - The maximum number of passes over the training data (aka epochs). Since the sentences would not be long, we set this to 50.
- *learning_method* - Method used to update _component. Options include:
    - 'batch': Batch variational Bayes method. Use all training data in each EM update. Old `components_` will be overwritten in each iteration.
    - 'online': Online variational Bayes method. In each EM update, use mini-batch of training data to update the ``components_`` variable incrementally. The learning rate is controlled by the ``learning_decay`` and the ``learning_offset`` parameters.

The code snippet is shown in the Figure 3.7.1.2-4 below.

```
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
from DataAccessLayer import DataAccessLayer
import matplotlib.pyplot as plt
import pandas as pd
import time as time

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from TextPreprocessing import text_preprocessing

# Function to implement topic modelling on a sentence
def build_topic_model(text, delimiter=" "):
        text = text_preprocessing(text)
        sparse_vectorizer = CountVectorizer(strip_accents = 'unicode')
        sparse_vectors = sparse_vectorizer.fit_transform(text)

        # Set number of topics
        n_topics = 1

        # Run LDA to generate topics/clusters
        lda = LatentDirichletAllocation(n_components=n_topics, max_iter=50,
                                        learning_method='online',
                                        random_state=0)

        lda.fit(sparse_vectors)

        # Get the first n_top_words key words
        n_top_words = 10
        feature_names = sparse_vectorizer.get_feature_names()

        t = None
        for i, topic in enumerate(lda.components_):
            t = delimiter.join([feature_names[i] for i in topic.argsort()[:-n_top_words - 1:-1]])

        return t

start = time.time()
model = SentenceTransformer('LaBSE')
sentenceFromUser = "My fish has white string from its anus"

# run the user input sentence through the topic modelling function
# before adding to the list called sentences
sentences = [build_topic_model(sentenceFromUser)]

dbcon = DataAccessLayer().CreateDBConnection
allSymptomNodes = dbcon.GetAllNodeListOfType('Symptom')

# allSymptomNodes contains all symptom nodes retrieved from the graph database
for symptomNode in allSymptomNodes:
    # add description from each symptom node into the sentences list
    # Before adding, run the sentence through the topic modelling function
    sentences.append(build_topic_model(symptomNode.description))

# Convert the sentences to vectors by calling model.encode()
sentence_embeddings = model.encode(sentences)

# Call cosine_similarity to compare user sentence with every symptom description to get similarity score
similarity = cosine_similarity([sentence_embeddings[0]], sentence_embeddings[1:])

end = time.time()

print("duration = ", end - start)
```

```
duration =  6.230898141860962
```

Figure 3.7.1.2-4. Code snippet showing the implementation with LDA

The execution of the above code took about 6.23 seconds and produced the following results:

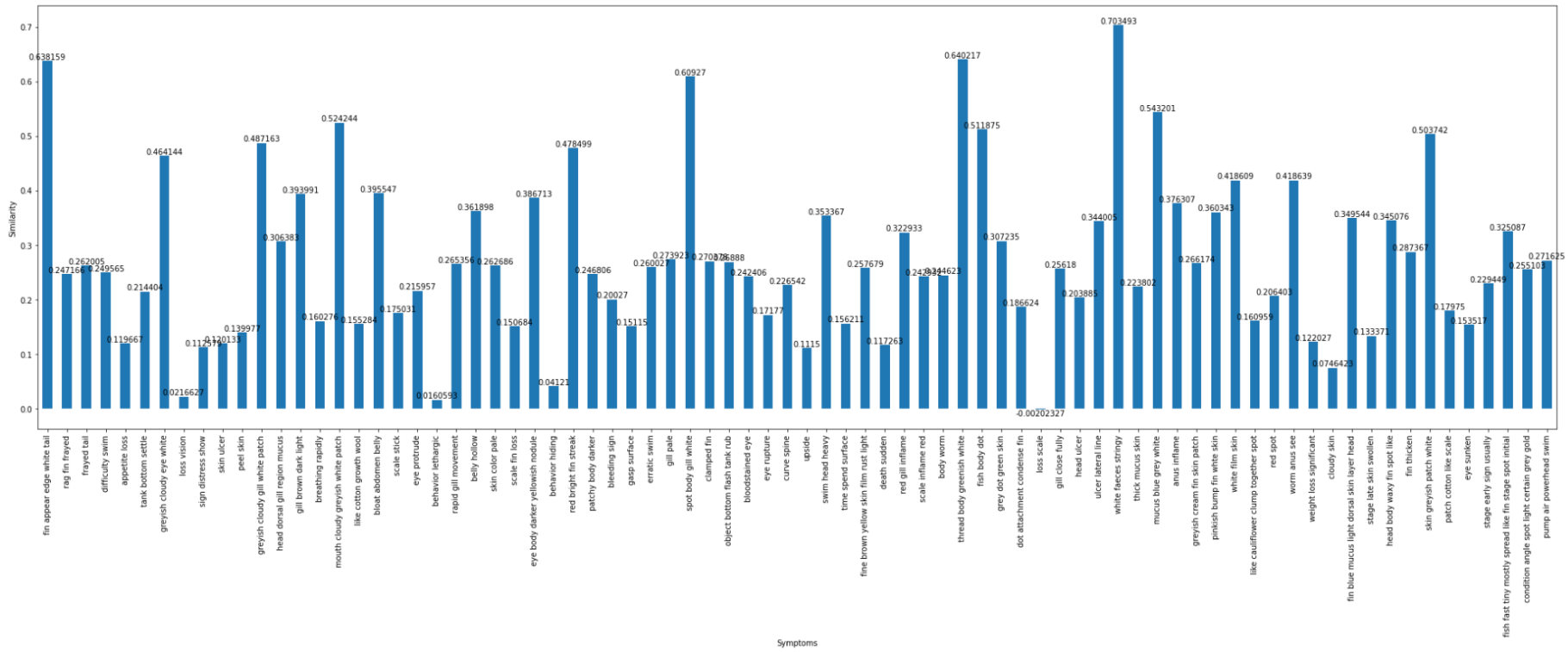


Figure 3.7.1.2-5 Results of using LDA

In the above results, we can see that the top 5 similarities are:

S/No.	Sentence	Topics	Similarity Score
1	White stringy faeces	white faeces stringy	0.703493
2	White greenish thread on body	Thread body greenish white	0.640217
3	Fins or tail edges appear white	Fin appear edge white tail	0.638159
4	White spots on the body and gills	Spot body gill white	0.60927
5	Grey or white blue mucus	Mucus blue grey white	0.543201

The topic for symptom description: "*Worm seen at anus*" is "worm anus see", and its score is 0.418609, which is rather low. Although "*White stringy faeces*" and "*White greenish thread on body*" could be good possibilities to match the user input sentence, the rest of the sentences in the top 5 are quite far from what the user sentence means. This execution was longer than without using LDA, which was about 4.9 seconds.

We then tried to add another parameter to the ***LatentDirichletAllocation*** algorithm - *learning_decay* - and set it to 0.5. This parameter controls the learning rate in the online learning method. Unfortunately, we got the same results as before and with slightly longer execution of about 6.61 seconds. Even when we changed this value to 0.8 or to the default value of 0.7, we still got the same results and similar duration.

We felt that this technique is not able to provide a good coverage of similarities between the user sentence and the description of the symptoms. Hence, we decided to explore using the **Universal Sentence Encoder**.

C. Universal Sentence Encoder

The Universal Sentence Encoder [19] is a language model that encodes text into fixed-length embeddings. It aims to convert sentences into semantically meaningful dense real-valued vectors. [20]

It encodes text into high dimensional vectors that can be used for text classification, semantic similarity, clustering, and other natural language tasks. The pre-trained sentence encoders have been trained on a variety of supervised and unsupervised tasks to capture as much universal semantic information as possible.

The model is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable length English text, and the output is a 512-dimensional vector. The sentence embeddings can then be trivially used to compute sentence level meaning similarity as well as to enable better performance on downstream classification tasks using less supervised training data [21].

Semantic similarity is a measure of the degree to which two pieces of text carry the same meaning. This is broadly useful in obtaining good coverage over the numerous ways that a thought can be expressed using language without needing to manually enumerate them [21].

Figure 3.7.1.2-5 below shows the illustration of the universal sentence encoder.

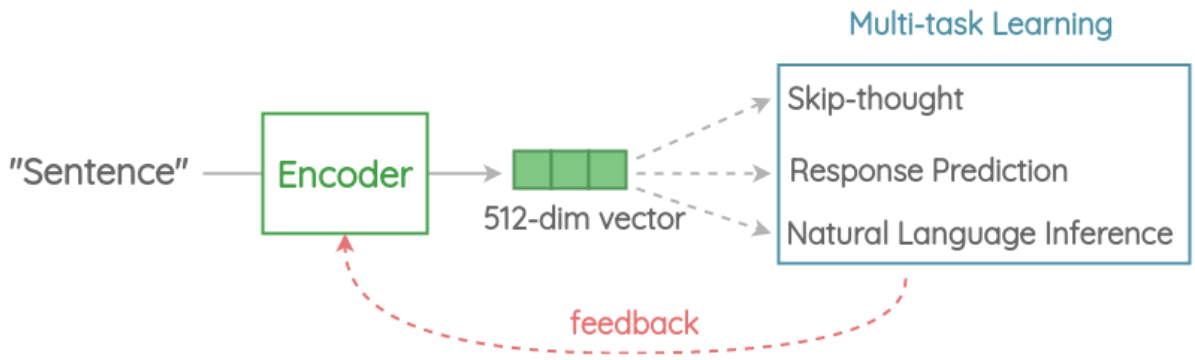


Figure 3.7.1.2-5. Diagram illustrating the Universal Sentence Encoder

We used version 4 of the universal-sentence-encoder module in our test. The module does not require pre-processing the data before applying the module, it performs best effort text input pre-processing inside the graph. The module is loaded through the **tensorflow_hub** library. Again, we employed

the **cosine_similarity** function to calculate the similarities between the user input and each of the symptom description.

The code snippet that we used to test with our text comparisons is shown below:

```
import tensorflow as tf
import tensorflow_hub as hub
from sklearn.metrics.pairwise import cosine_similarity

module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
model = hub.load(module_url)

sentenceFromUser = "My fish has white string at its anus"
sentences = [sentenceFromUser]

dbcon = DataAccessLayer().CreateDBConnection
allSymptomNodes = dbcon.GetAllNodeListOfType('Symptom')

# allSymptomNodes contains all symptom nodes retrieved from the graph database
for symptomNode in allSymptomNodes:
    # add description from each symptom node into the sentences list
    sentences.append(symptomNode.description)

# Convert the sentences to vectors by calling model.encode()
sentence_embeddings = model(sentences)

# Call cosine_similarity to compare user sentence with every symptom description to get similarity score
similarity = cosine_similarity([sentence_embeddings[0]], sentence_embeddings[1:])
```

Figure 3.7.1.2-6. Code snippet showing the implementation with Universal Sentence Encoder

The results from executing the above codes show in Figure 3.7.1.2-7.

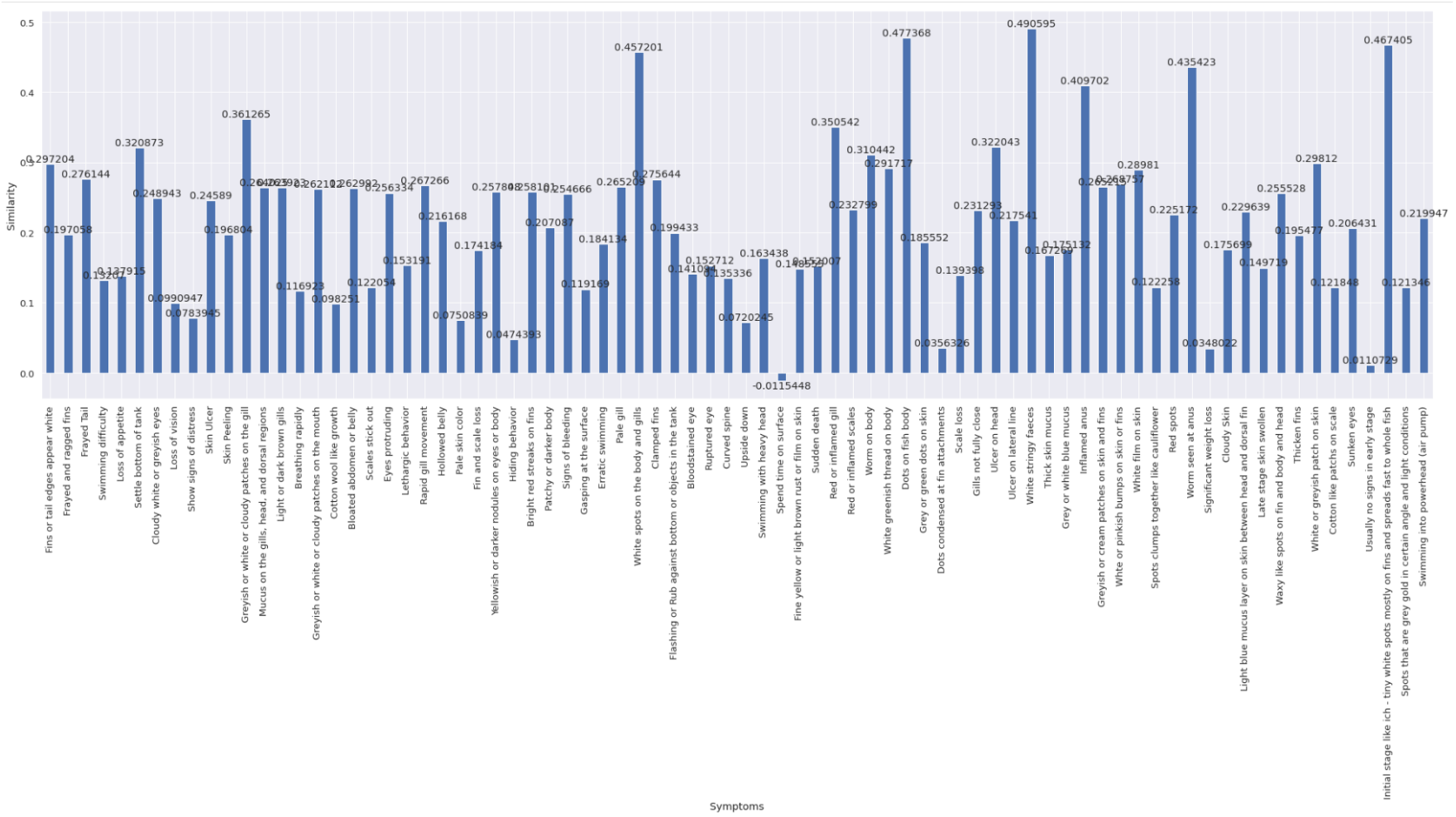


Figure 3.7.1.2-7. Bar chart showing the results using model "Universal Sentence Encoder"

In general, the results do not reflect a high score for all the comparisons, ranging between -0.0115448 to 0.490595. However, the similarities that were derived from universal-sentence-encoder was quite good.

We can see this in Table 3.7.1.2-1 below, that the highest similarity is with the description "*White stringy faeces*". However, the description that should be closest to the user's intended input, "*Worm seen at anus*" was only at the fifth place, and the descriptions between the second and fourth places do not reflect the similarities that well.

S/No.	Symptom Description	Similarity Score
1	White stringy faeces	0.490595
2	Dots on fish body	0.477368
3	Initial stage like ich – tiny white spots mostly on fins and spreads fast to whole fish	0.467405
4	White spots on the body and gills	0.457201
5	Worm seen at anus	0.435423

Table 3.7.1.2-1. Summarization of the results using "Universal Sentence Encoder"

We feel that the Universal Sentence Encoder technique may not be idea to be used in our context. Hence, we moved on to explore another technique using **spaCy**.

D. spaCy with Text Pre-Processing augmented with Custom Stopwords

NLTK and **spaCy** are two most popular NLP tools available in Python. NLTK provides many algorithms to build something from scratch, while spaCy provides a large set of functionalities and larger word vectors to quickly build an NLP module for an application.

NLTK provides a set of string processing library, where each function takes strings as input and returns a processed string. spaCy does it differently in that it takes an object-oriented approach where each function returns objects instead of strings or arrays. This allows for faster and simpler way to explore the tool to be used for our application. Hence, in view of the short time frame, spaCy is chosen for the next tool to be explored and tested.

spaCy provides a mapping from a vocabulary of common words to vectors. These vectors, sometimes called "word embeddings," are designed using the GloVe (Global Vectors for Word Representation) algorithm to map semantic meaning into numeric proximity.

GloVe is a model used for the representation of the distributed words. This model represents words in the form of vectors using an unsupervised learning algorithm. This unsupervised learning algorithm maps the words into space where the semantic similarity between the words is observed by the distance between the words. These algorithms perform the training of a corpus consisting of the aggregated global word-word co-occurrence statistics, and the result of the training usually represents the subspace of the words in which our interest lies [22].

The GloVe model uses the matrix factorization technique for word embedding on the word-context matrix. It starts working by building a large matrix which consists of the words co-occurrence information, basically, the idea behind this matrix is to derive the relationship between the words from statistics. The co-occurrence matrix tells us the information about the occurrence of the words in different pairs [22].

Figure 3.7.1.2-8 below gives an illustration of the above explanation:

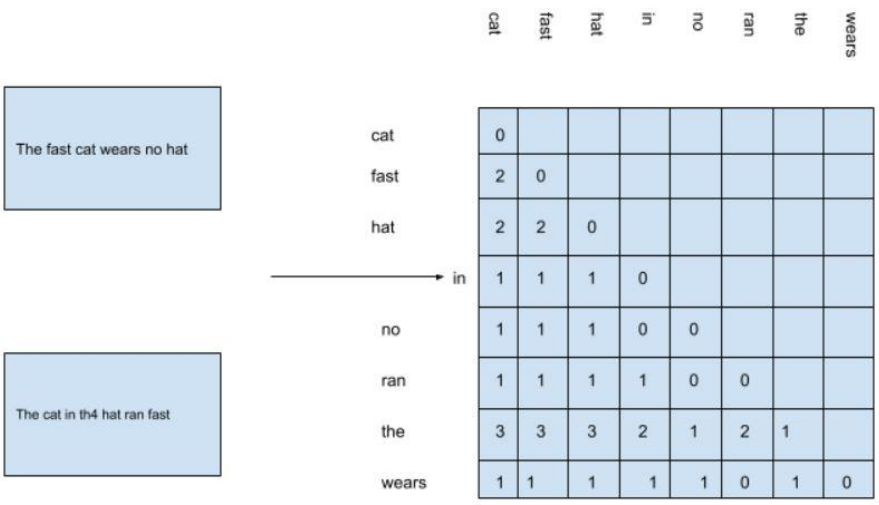


Figure 3.7.1.2-8. Figure illustrating the GloVe Model [22]

In the above figure, we have a matrix which is formed by considering the unique words from two sentences given on the left side of the image and in this we can see that the word "the" has accrued 3 times with cat fast and hat so the value in the matrix for the combination is 3 every time [22].

Now using the matrix, we can compute the ratio of probabilities between any two pairs of words. For example, let's say probabilities of (cat/fast) = 1 and (cat/the) = 0.5; the ratio of these probabilities will be 2 and by this ratio, we can infer that '*fast*' is more relevant than '*the*' based on probabilities. The GloVe method can be represented mathematically as shown in Equation 3.7.1.2-2 [22]:

$$F\left(w_i, w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

Equation 3.7.1.2-2. Mathematical equation of the GloVe method

It can thus be seen that GloVe is a "count-based" model. The default model from spaCy does not provide word vector data. Instead, it provides "good guess" vectors that are of lower quality. [23] Hence, we must use a larger language model to receive vector data: **en_core_web_lg**.

The code snippet for the testing is shown in Figure 3.7.1.2-9 below:

```
import spacy

nlp = spacy.load('en_core_web_lg')

sentenceFromUser = "My fish has white string from its anus"
threshold = 0.65

dbcon = DataAccessLayer().CreateDBConnection
allSymptomNodes = dbcon.GetAllNodeListOfType('Symptom')

symptoms = {}
sentence1 = nlp(sentenceFromUser)

# allSymptomNodes contains all symptom nodes retrieved from the graph database
for symptomNode in allSymptomNodes:
    sentence2 = nlp(symptomNode.description)

    # calculate similarity between two sentences
    similarity = sentence1.similarity(sentence2)

    # take only similarities that are above the threshold
    if similarity > threshold:
        symptoms[symptomNode.description] = similarity

symptoms = dict(sorted(symptoms.items(), key=lambda item: item[1], reverse=True))
```

Figure 3.7.1.2-9. Code snipping showing the implementation using model "en_core_web_lg"

A few tests were conducted to explore the performance and results of using spaCy. In the first test, all sentences were not pre-processed and a threshold value of 0.65 was set to filter out the sentences with better scores. The results were sorted in descending order and are shown in Figure 3.7.1.2-10 below.

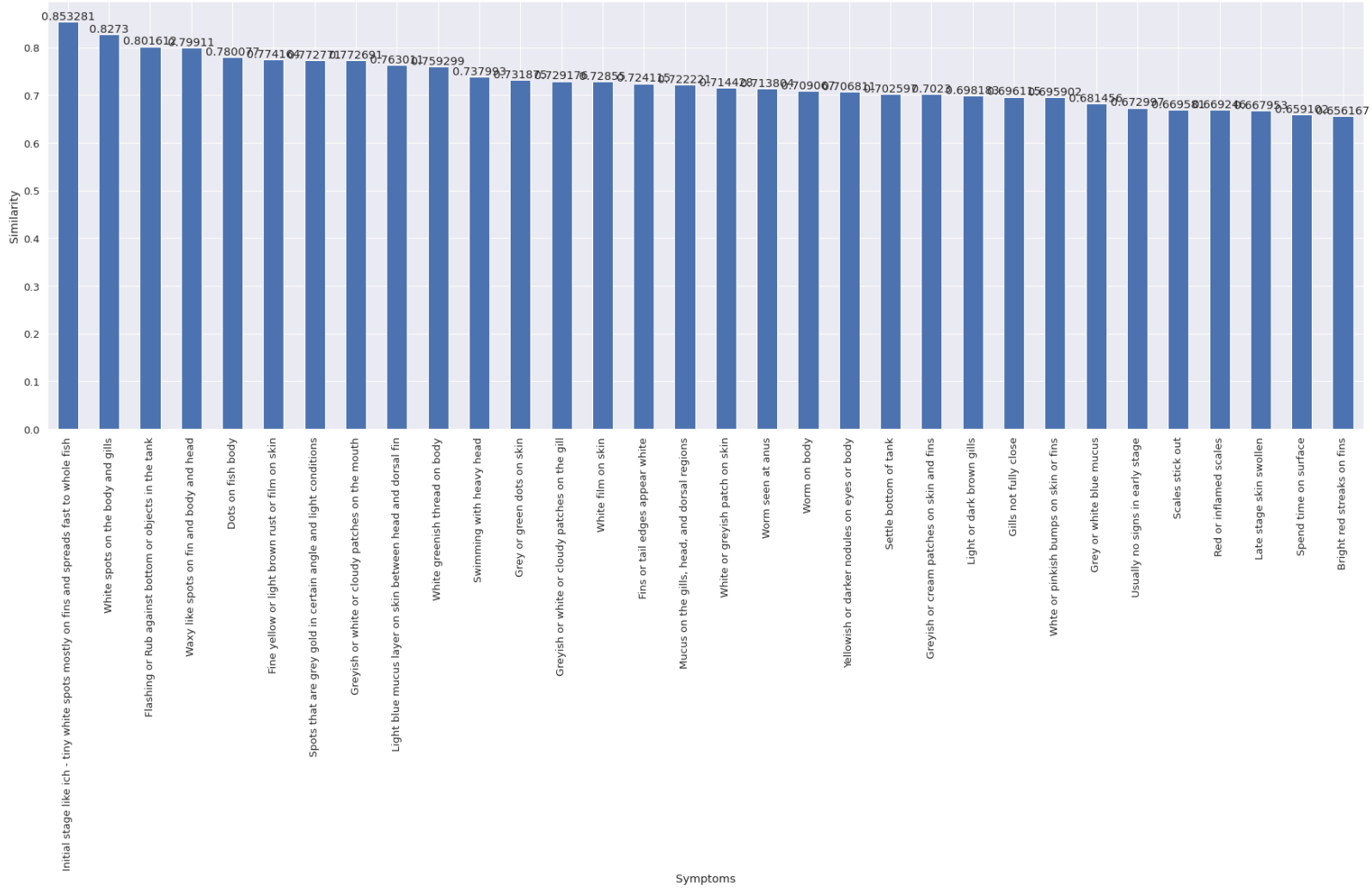


Figure 3.7.1.2-10. Bar chart showing the results from using spaCy without text proprocessing

In the above Figure 3.7.1.2-10, we have 32 sentences that scored above the threshold level. The top three sentences have nothing to do with anus or worm but the second one did have something related to white. The sentence "*Worm seen at anus*" is at the 18th position. This is a rather bad result.

We then proceeded to make some fine tunings to see how the results can be improved. The following are what was performed:

a. Change the sentences to lower-case
b. Apply the following text pre-processing to each sentence:
    - Remove punctuations
    - Remove quotes
    - Remove digits
    - Remove default spaCy stopwords
    - Lemmatization

Figure 3.7.1.2-11 below shows the code snippet with the text pre-processing step.

```
import spacy

nlp = spacy.load('en_core_web_lg')

sentenceFromUser = "My fish has white string from its anus"
threshold = 0.65

dbcon = DataAccessLayer().CreateDBConnection
allSymptomNodes = dbcon.GetAllNodeListOfType('Symptom')

symptoms = {}
# Apply text pre-processing to the sentence before converting the sentence into vectors
sentence1 = nlp(text_preprocessing(sentenceFromUser, nlp))

# allSymptomNodes contains all symptom nodes retrieved from the graph database
for symptomNode in allSymptomNodes:
    # Apply text pre-processing to the sentence before converting the sentence into vectors
    sentence2 = nlp(text_preprocessing(symptomNode.description, nlp))

    # calculate similarity between two sentences
    similarity = sentence1.similarity(sentence2)

    # take only similarities that are above the threshold
    if similarity > threshold:
        symptoms[symptomNode.description] = similarity

symptoms = dict(sorted(symptoms.items(), key=lambda item: item[1], reverse=True))
```

Figure 3.7.1.2-11. Code snipping showing the implementation using model "en_core_web_lg" with Text Pre-procesing

After executing the above codes, we get the following results shown in Figure 3.7.1.2-12.

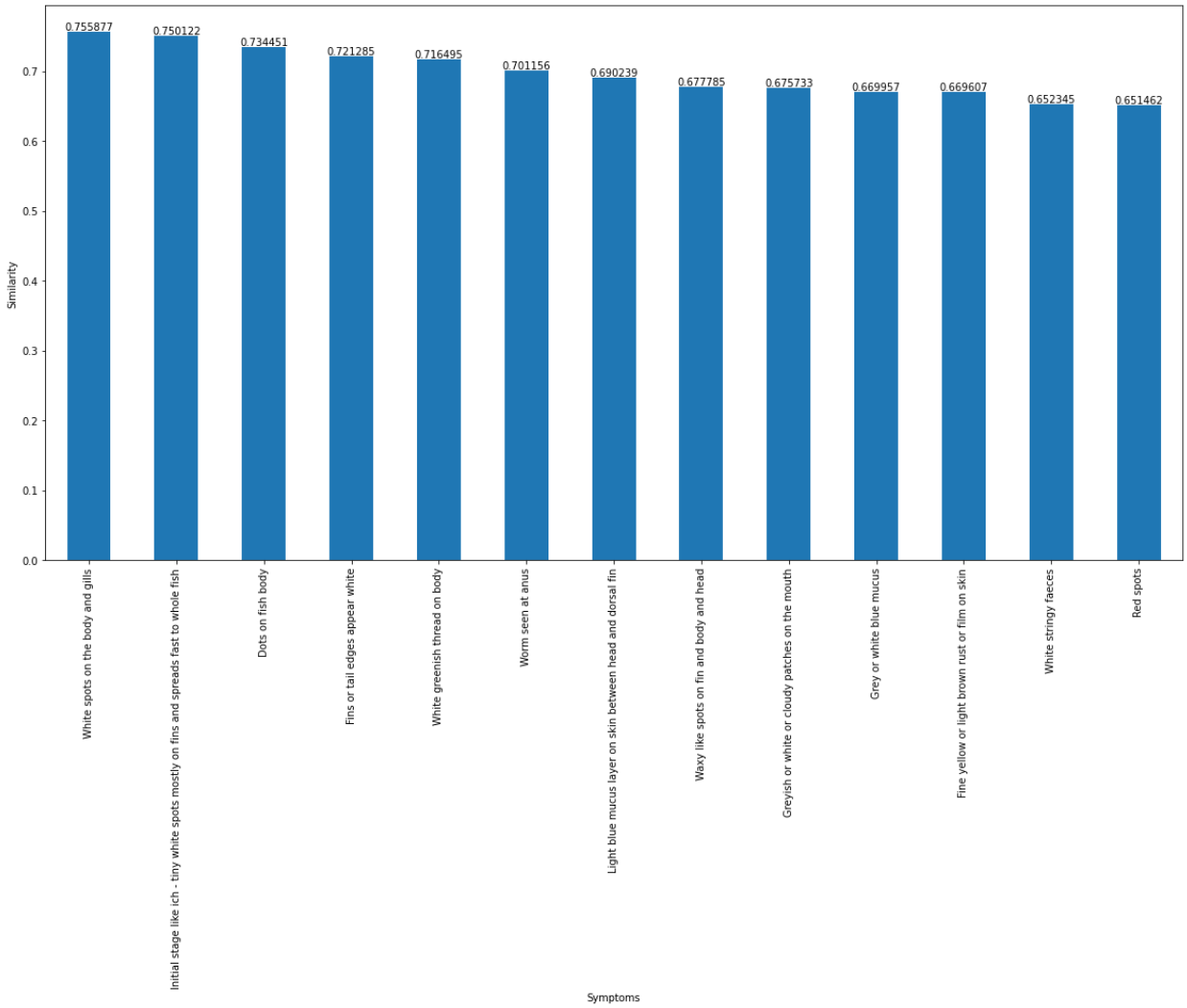


Figure 3.7.1.2-12. Bar chart showing the results from using spaCy with Text-Pre-processing

The results shown above are not as exciting as we hoped to be. To analyse the tokenized words in each sentence after text pre-processing, we listed them down in Table 3.7.1.2-2 below:

S/No.	Sentence	Tokens
1	My fish has white string from its anus	fish, white, string, anus
2	White spots on the body and gills	white, spot, body, gill
3	Initial stage like ich - tiny white spots mostly on fins and spreads fast to whole fish	initial, stage, like, ich, tiny, white, spot, fin, spread, fast, fish
4	Dots on fish body	dot, fish, body
5	Fins or tail edges appear white	fin, tail, edge, appear, white
6	White greenish thread on body	white, greenish, thread, body
7	Worm seen at anus	worm, anus
8	Light blue mucus layer on skin between head and dorsal fin	light, blue, mucus, layer, skin, head, dorsal, fin
9	Waxy like spots on fin and body and head	waxy, like, spot, fin, body, head
10	Greyish or white or cloudy patches on the mouth	greyish, white, cloudy, patch, mouth
11	Grey or white blue mucus	grey, white, blue, mucus
12	Fine yellow or light brown rust or film on skin	fine, yellow, light, brown, rust, film, skin
13	White stringy faeces	white, stringy, faece
14	Red spots	red, spot

Table 3.7.1.2-2. Analysis of the tokenized words after Text Pre-processing

We see that the tokens from the user sentence include "*fish*", "*white*", "*string*", "*anus*". These words do appear in some of the description of the symptoms but some symptoms in this list do not make sense such as number 2, 3, 4, 8, 9, 12 and 14.

Looking at the tokens, non-related sentences could be matched due to the nature of the GloVe algorithm, which is essentially a "count-based" model based on the word occurrences. One possible way we could try is to remove "noises" from each of the sentences before comparing the similarities.

A separate list of **stopwords** were created to list down words that could contribute as "noises" in a sentence. One consideration on the choice of **stopwords** is the context of our application. Since our application is related to fish symptoms and diseases, it is not helpful to have words such as "*fish*" or "*it*" in the sentence because the subject of all symptoms is already the fish! Words that do not appear in the description of symptom can also contribute negatively to the results. Another point to consider is that the calculation of similarity does not consider whether two different words have the same meaning. The created list of **stopwords** is then added to the default **stopwords** in spaCy.

The updated code snippet is shown on the next page.

```
from DataAccessLayer import DataAccessLayer
import spacy

nlp = spacy.load('en_core_web_lg')

def add_stopwords(nlp):
    stopwords = None
    with open('./stopwords.txt') as file:
        stopwords = [stopword.replace('\n', '') for stopword in file.readlines()]

    for stopword in stopwords:
        nlp.Defaults.stop_words.add(stopword)

    return nlp

def lower_casing(sentence):
    new_sentence = ''.join([(chr(ord(char) + 32) if ord(char) > 64 and ord(char) < 91 else chr(ord(char))) for char in sentence])
    return new_sentence

def text_preprocessing(raw_sentence, nlp_tool):
    token_sentence = nlp_tool(lower_casing(raw_sentence))
    preprocessed_sentence = None
    # Add stopwords from external file to the default spaCy stopwords
    nlp_tool = add_stopwords(nlp_tool)
    stopword = nlp_tool.Defaults.stop_words
    preprocessed_sentence = [token.lemma_ for token in token_sentence if token.text not in stopword and not token.pos_ == 'X'
                             and not token.is_punct and not token.is_digit and not token.is_quote]
    preprocessed_sentence = " ".join(preprocessed_sentence)

    return preprocessed_sentence


sentenceFromUser = "My fish has white string from its anus"
threshold = 0.65

dbcon = DataAccessLayer().CreateDBConnection
allSymptomNodes = dbcon.GetAllNodeListOfType('Symptom')

symptoms = {}

# Apply text pre-processing to the sentence before converting the sentence into vectors
sentence1 = nlp(text_preprocessing(sentenceFromUser, nlp))

# allSymptomNodes contains all symptom nodes retrieved from the graph database
for symptomNode in allSymptomNodes:
    # Apply text pre-processing to the sentence before converting the sentence into vectors
    sentence2 = nlp(text_preprocessing(symptomNode.description, nlp))

    # calculate similarity between two sentences
    similarity = sentence1.similarity(sentence2)

    # take only similarities that are above the threshold
    if similarity > threshold:
        symptoms[symptomNode.description] = similarity

symptoms = dict(sorted(symptoms.items(), key=lambda item: item[1], reverse=True))
```

Figure 3.7.1.2-13. Code snippet showing the implementation using model "en_core_web_lg" with Text Pre-processing and addition of the customized stopwords list

After running the above codes, the result produced is shown in Figure 3.7.1.2-14 below.

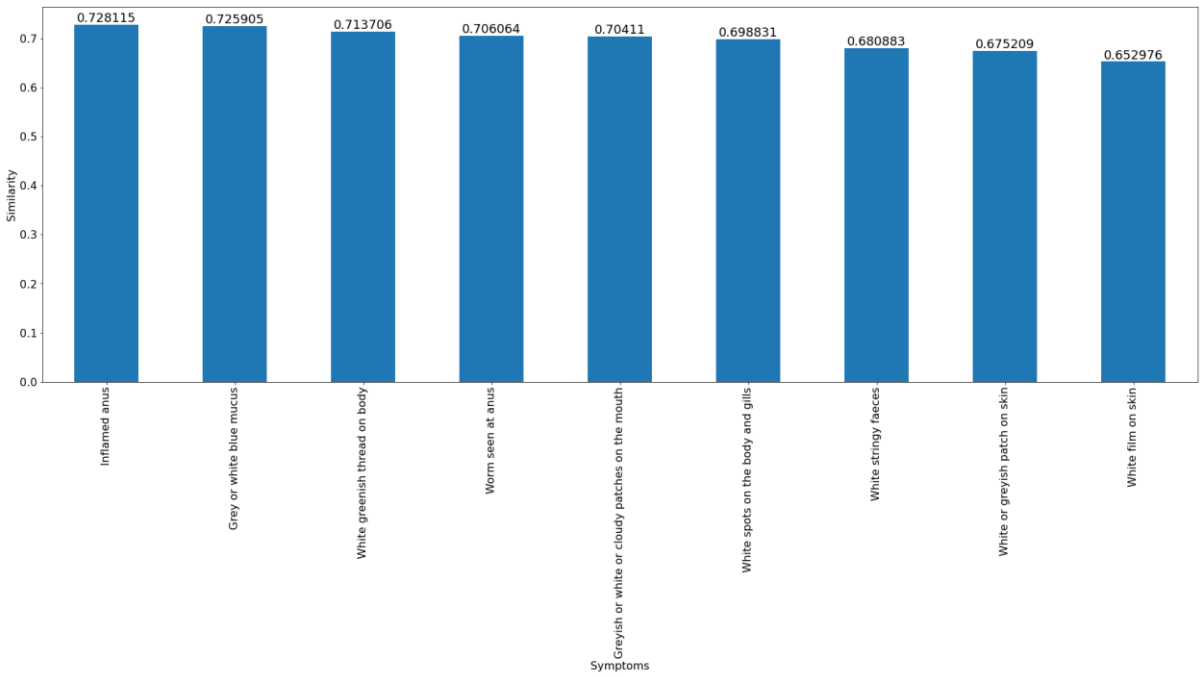


Figure 3.7.1.2-14. Bar chart showing the results from using spaCy with Text Pre-processing and addition of the customized stopwords list

We can see that there are improvements where all the above top nine results in Figure 3.7.1.2-14 contain the key words: "white" and "anus", which are very much related to the user sentence. In a real-life scenario, our application will not be able to know the actual intention of what the user is trying to find out. However, it is important that our results match as close as possible to the symptoms that the user's fish has from his/her description.

To see how our core reasoning intelligence perform with a good user description, we tested using this implementation with the following user input:

"My fish has worm around its anus"

And we achieved the results shown in Figure 3.7.1.2-15 below.

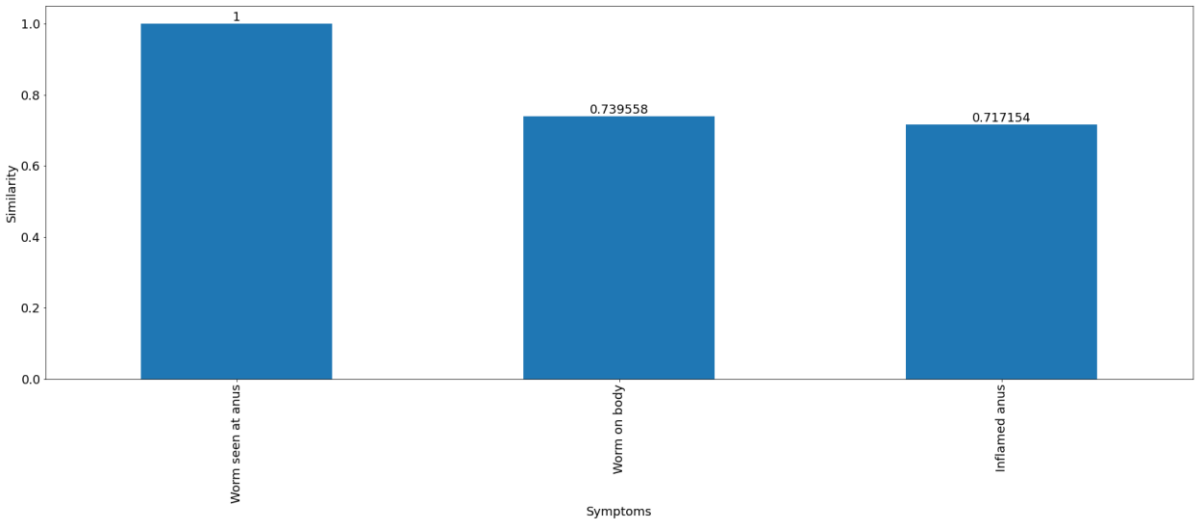


Figure 3.7.1.2-15. Bar chart showing the results with good user descriptions

3.7.1.3. Verdict

After all the explorations and investigations, we finally arrived at using **spaCy** with **Text Pre-Processing** augmented with custom **Stopwords**. This technique produces acceptable results for users to query about the possible disease(s) based on the free text input.

One major shortcoming of our implementation is that the comparison of the sentences does not take into account of the meaning of the words in the sentences. It is not able to know the synonyms of a word in the user sentence to be able to match another word of the same meaning in the symptom description,

We do not have a large data set of fish diseases but given more time, we would like to explore other ways of implementation such as training a custom model using **Word2Vec**; and work with synonyms using **sense2vec** and **WordNet**.

Natural Language Processing is a fascinating area of artificial intelligence with so much room for exploration. We are glad to have applied some of what we have learnt from the series of lectures in this course to accomplish this project.

3.7.2. Knowledge Base Decisioning Module (KBDM)

3.7.2.1. Developing a Custom Heuristic Model

The Knowledge Base Decisioning Module is the model to determine the confidence level of the suspected diseases based on confirmed/denied symptoms. As mentioned above in Section 3.4.4, each symptom has a weight and penalty in its relationship with the disease. When the user confirms or denies a symptom for the case, the confidence level for all the diseases related to the symptom will be recomputed real-time based on the formula in Equation 3.7.2.1-1 below and the slate of suspected diseases are refreshed based on the updated computation.

$$C_d = \frac{\Sigma w_{s'} + \Sigma p_{s''}}{\Sigma w_s}$$

Equation 3.7.2.1-1. Mathematical equation adopted by KBDM to compute the confidence level of diseases

C_d: confidence level of the disease
w : weight of a symptom for the disease
p : penalty (negative number) of a symptom for the disease
s : symptom of the disease
s' : confirmed symptom for the disease in the case
s'' : denied symptom for the disease in the case

3.7.2.2. Verdict

Using the mathematical equation in Equation 3.7.2.1-1, the following methods were evaluated.

A. Return only one symptom associated to each of the suspected diseases
B. Return all symptoms associated to the suspected disease with the highest confidence
C. Return symptoms associated to the suspected diseases with the top three confidence

Option C was eventually selected due to its balanced approach where suspected diseases with higher confidence scores will be accorded higher weightages while not causing the diagnostic inquiry to become overly complex or dragged out. The model will sort the suspected diseases by the confidence in descending order and will return three unconfirmed symptoms from the disease with highest

confidence, two unconfirmed symptoms from the disease with the second highest confidence and one unconfirmed symptom from the disease with the three highest confidence. This approach is to guide him/her to quickly reach the disease with high confidence.

3.7.3. Summary of Hyperparameters

The following table summarizes details of all hyperparameters utilized by OhMyFish for the core intelligence reasoning along with their respective controls within the core intelligence reasoning.

Hyperparameter	**Purpose**	**Module**
confidenceScoreHP	Utilized to set the confidence of a confirmed disease.	CCM
similarityThreshold	Utilized to set the similarity thresholds of textual input from an user after topic modelling against the symptoms nodes within knowledge base.	TMM
numberDiseaseToSuspect	Utilized to set the number of suspected diseases ordered by the confidence in descending order during each iteration.	KBDM
numberSymptomToReturn	Utilized to set the number of symptoms associated to the number of suspected diseases ordered by the confidence in descending order during each iteration.	KDBM

4. Technology Stack

Table 3.7.3-1 below describes all technologies leverages for the development of OhMyFish.

Technology	Usage
ReactJS	An open-source JavaScript library used for building the user interface of OhMyFish.
Flask	A micro web framework used for the development of application built using Python.
Python	An open-source programming language used for building all backend modules of OhMyFish.
Neo4J	An open-source Graph-based database system used as the storage medium for the knowledge base of Fish Diseases that OhMyFish is built upon.

Table 3.7.3-1. Table listing all technologies leveraged in the development of OhMyFish

5. Test Results and Evaluation

As with any systems and more so with an intelligent reasoning system, testing is an essential phase of the development lifecycle to enable the results to be evaluated for opportunities of optimization on the intelligent techniques integrated within OhMyFish. This section consolidates all results of tests executed independently by each team member and observations for optimization summarized.

5.1. Test Results

Test Case ID:	Open-1		**Test By:**	Jason Lim	
Test Title:	Expert user guided 1 Disease		**Test Date:**	19 Apr 2022	
Pre-Conditions:	Expert hobbyist using guided Flow. Fish has 1 disease (Cloudy eye), and user is fully aware of ALL related symptoms.				
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Comments**
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 of the auto check not true for test case
2	Select 2 predefined symptoms	Selected “Does your fish shows signs of distress?” & “Is your fish not reacting to visual stimulus?”	A diagnosis or further sensible questioning	Suspected disease is highlighted to user	
3	Step 2 is performed and send button is pressed	As above step 2	A diagnosis or further sensible questioning	More symptoms given but not related to this test case	
Result (Pass/Fail):	Pass				
Post-Conditions:	User is happy to see a diagnosis and do not attempt to give unsure symptoms				

Test Case ID:	Open-2		**Test By:**	Jason Lim	
Test Title:	Beginner user guided 1 Disease		**Test Date:**	19 Apr 2022	
Pre-Conditions:	Beginner hobbyist using guided Flow. Fish has 1 disease (Cloudy eye), and user is only aware of physical symptoms, and unaware of behavioural symptoms				
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Comments**
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 of the auto check not true for test case
2	User clicks and enlarge the picture and make judgement if this photo shows similarities to cloudy eye	Symptom Photo	Photo showing cloudy eye	Photo reasonably show cloudy eyes	
3	User looks at the right-side diagnosis to look at disease with whitish eye symptom		Cloudy Eye should be within top 3 suspected disease	Cloudy Eye is number 1	
Result (Pass/Fail):	Pass				
Post-Conditions:	User is happy to exercise some exploration and use human judgement to find the information he/she desires				

Test Case ID:	Open-3		**Test By:**	Jason Lim	
Test Title:	Expert user guided 3 Disease at first entry		**Test Date:**	19 Apr 2022	
Pre-Conditions:	Expert hobbyist using guided Flow. Fish has 3 diseases (Cloudy eye, roundworm and ich). User is fully aware of the symptoms.				
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Comments**
1	Enter test data into user input box and send	eye looks whitish, Worm at anus, White spot on body	Some response from system	System responded with 25 symptom questions with 19 checked	17/19 of the auto check not true for test case
2	User will uncheck all irrelevant symptoms questions and checks all relevant ones	Check only “Are there white spots on the body and gills?”,” Does your fish's eyes look white or greyish?”,” Is your fish rubbing against the tank's bottom and/or objects in the tanks?”,” Does your fish shows signs of distress?”,” Is your fish not reacting to visual stimulus?”	Some changes in suspected diseases	Suspected disease Cloudy eye improves in confidence to 100%, Ich is at number 2	
3	User clicks send		Some response from system	System responded with 6 symptom questions	
4	User will uncheck all irrelevant symptoms questions and checks all relevant ones				
5	Repeat 3 to 4 until the 3 test diseases is at 100%				This actually took about 4 iterations of step 5

Result (Pass/Fail):	Pass
Post-Conditions:	User has patience to go through the loop to get that final diagnosis

Test Case ID:	Open-4	Test By:	Jason Lim
Test Title:	Beginner user guided 3 Disease at first entry	Test Date:	19 Apr 2022

Pre-Conditions:	Beginner hobbyist using guided Flow. Fish has 3 diseases (Cloudy eye, ich and roundworm). User is only aware physical symptoms and is unaware of behavioural symptoms.

Step	Test Step	Test Data	Expected Result	Actual Result	Comments
1	Enter test data into user input box and send	eye looks whitish, Worm at anus, White spot on body	Some response from system	System responded with 25 symptom questions with 19 checked	17/19 of the auto check not true for test case
2	User will uncheck all irrelevant symptoms questions and checks all relevant ones	Check only “Are there white spots on the body and gills?”,” Does your fish's eyes look white or greyish?”	Some changes in suspected diseases	Suspected disease Cloudy eye improves in confidence, Ich is at number 2	
3	User clicks send		Some response from system	System responded with 6 symptom questions	
4	User will uncheck all irrelevant and checks all relevant ones (only physical)				
5	Repeat 3 to 4 4 times		All possible physical symptoms are checked	Two suspected diseases are capture, round worm and cloudy eye. Ich is not diagnosed	

Result (Pass/Fail):	Fail
Post-Conditions:	N/A

Test Case ID:	Open-5	**Test By:**	Jason Lim
Test Title:	Beginner user guided 3 Disease at first entry (lengthy description)	**Test Date:**	19 Apr 2022

Pre-Conditions:	Beginner hobbyist using guided Flow. Fish has 3 diseases (Cloudy eye, ich and roundworm). User is only aware physical symptoms and is unaware of behavioural symptoms.

Step	Test Step	Test Data	Expected Result	Actual Result	Comments
1	Enter test data into user input box and send	Fish eye looks whitish at the outer ring, There seem to be a worm sticking at the fish anus, Several white spots are seen on fish body	Some response from system	System responded with 16 symptom questions with 10 checked	9/10 of the auto check not true for test case
2	User will uncheck all irrelevant symptoms questions and checks all relevant ones	Check only “Are there white spots on the body and gills?	Some changes in suspected diseases	Suspected disease Ich is at number 1	
3	User clicks send		Some response from system	System responded with 6 symptom questions	
4	User will uncheck all irrelevant and checks all relevant ones (only physical)				

5	Repeat 3 to 4 4 times		All possible physical symptoms are checked	Two suspected diseases are capture, round worm and cloudy eye. Ich is not diagnosed	
Result (Pass/Fail):	Fail				
Post-Conditions:	N/A				

Test Case ID:	Open-1	**Test By:**	Jason Lim
Test Title:	Expert user Open-end 1 Disease 2 questions	**Test Date:**	19 Apr 2022
Pre-Conditions:	Expert hobbyist using open Flow. Fish has 2 diseases (Cloudy eye and roundworm). User is fully aware of the symptoms.		

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Comments**
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 of the auto check not true for test case
2	User checks the other detected symptoms (if any)	Check “Does your fish shows signs of distress?”, “Is your fish not reacting to visual stimulus?” , “Is your fish swimming letharically?”	Some changes in suspected diseases	Suspected disease Cloudy eye improves in confidence to 100%	
3	User uncheck the irrelevant AI checked symptoms and enable chat	Check only “Does your fish's eyes look white or greyish?”			
3	Enter test data into user input box and send	Worm at bottom of fish			
4	User checks the other detected symptoms (if any)	Check “Is your fish having difficulty swimming?”,” Did your fish recently lose significant amount of weight?”, “Does your fish's belly look hollowed?”	Some changes in suspected diseases	Suspected disease Roundworm improves in confidence to 100%	

Result (Pass/Fail):	Pass
Post-Conditions:	User is happy to see a diagnosis and do not attempt to give unsure symptoms

Test Case ID:	Open-2	**Test By:**	Jason Lim
Test Title:	Expert user Open-end 1 Disease 3 questions	**Test Date:**	19 Apr 2022
Pre-Conditions:	Expert hobbyist using open Flow. Fish has 3 diseases (Cloudy eye, roundworm and ich). User is fully aware of the symptoms.		

Step	Test Step	Test Data	Expected Result	Actual Result	Comments
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 out of 3 auto check not true for test case
2	User checks the other detected symptoms (if any)	Check “Does your fish shows signs of distress?”, “Is your fish not reacting to visual stimulus?”, “Is your fish swimming letharically?”	Some changes in suspected diseases	Suspected disease Cloudy eye improves in confidence to 100%	
3	User uncheck the irrelevant AI checked symptoms and enable chat	Check only “Does your fish's eyes look white or greyish?”			
3	Enter test data into user input box and send	Worm at bottom of fish			
4	User checks the other detected symptoms (if any)	Check “Is your fish having difficulty swimming?”,” Did your fish recently lose significant amount of weight?”, “Does your fish's belly look hollowed?”	Some changes in suspected diseases	Suspected disease Roundworm improves in confidence to 100%	
5	Enter test data into user input box and send	White spot on body			

6	User checks the other detected symptoms (if any), and uncheck all irrelevant AI checked symptoms	Check “Does your fish settle at the bottom of the tank?”, “Is your fish rubbing against the tank's bottom and/or objects in the tanks?”,” Does your fish exhibit a loss of appetite?”	Some changes in suspected diseases	Suspected disease Ich improves in confidence to 72%	

Result (Pass/Fail):	Pass
Post-Conditions:	User is happy to see a diagnosis and do not attempt to give unsure symptoms

Test Case ID:	Open-3	Test By:	Jason Lim
Test Title:	Beginner Open-end 1 Disease 2 questions	Test Date:	19 Apr 2022

Pre-Conditions:	Beginner hobbyist using open Flow. Fish has 2 diseases (Cloudy eye and roundworm). User is only aware physical symptoms and is unaware of behavioural symptoms.

Step	Test Step	Test Data	Expected Result	Actual Result	Comments
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 out of 3 auto check not true for test case
2	User uncheck the irrelevant AI checked symptoms and enable chat	Check only “Does your fish's eyes look white or greyish?”			
3	Enter test data into user input box and send	Worm at bottom of fish			
4	User uncheck the irrelevant AI checked symptoms	Check only “Are there any worms at the anus region?”	Cloudy Eye and roundworms should be within top 3 suspected disease	Cloudy Eye is number 1 and Roundworms is number 2 suspected disease	

Result (Pass/Fail):	Pass
Post-Conditions:	User is happy to accept the only diagnosis (although a low confidence one)

Test Case ID:	Open-4	**Test By:**	Jason Lim
Test Title:	Beginner Open-end 1 Disease 3 questions	**Test Date:**	19 Apr 2022

Pre-Conditions:	Beginner hobbyist using open Flow. Fish has 3 diseases (Cloudy eye, ich and roundworm). User is only aware physical symptoms and is unaware of behavioural symptoms.

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Comments**
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 out of 3 auto check not true for test case
2	User uncheck the irrelevant AI checked symptoms and enable chat	Check only “Does your fish's eyes look white or greyish?”			
3	Enter test data into user input box and send	Worm at bottom of fish	Cloudy Eye and roundworms should be within top 3 suspected disease	Cloudy Eye is number 1 and Roundworms is number 2 suspected disease	
4	Repeat step 2 above	Check only “Are there any worms at the anus region?”			
5	Enter test data into user input box and send	White spot on body	Cloudy Eye, Roundworm and white spot should be within top 3 suspected disease	Cloudy Eye is number 1 and Saltwater and freshwater White spot at 2 and 3 respectively.	Only 2 out of 3 of the disease is detected

Result (Pass/Fail):	Fail
Post-Conditions:	N/A

Test Case ID:	Open-5	**Test By:**	Jason Lim
Test Title:	Beginner Open-end 1 Disease multiple variant questions	**Test Date:**	19 Apr 2022

Pre-Conditions:	Beginner hobbyist using open Flow. Fish has 1 disease (Cloudy eye), and user is only aware of physical symptoms, and unaware of behavioural symptoms. User will keep mentioning whitish or greyish eyes in different variant

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Comments**
1	Enter test data into user input box and send	eye looks whitish	Some response from system	System responded with 9 symptom questions with 3 checked	2 out of 3 auto check not true for test case
2	User uncheck the irrelevant AI checked symptoms and enable chat	Check only “Does your fish's eyes look white or greyish?”			
3	Enter test data into user input box and send	Eyes looks very white	Cloudy Eye should be within top 3 suspected disease	Cloudy Eye is number 1 suspected disease	
4	Repeat step 2 above				
5	Enter test data into user input box and send	Eyes look greyish	Cloudy Eye should be within top 3 suspected disease	Cloudy Eye is number 1 suspected disease	When this is repeated enough, Cloudy eyes has a low confidence diagnosis while other disease has ZERO confidence

Result (Pass/Fail):	Pass

Post-Conditions:	User is happy to accept the only diagnosis (although a low confidence one)

Test Case ID:	Open-6	**Test By:**	Jason Lim
Test Title:	Beginner Open-end 3 Disease 1 descriptive entry 2 Follow up Questions	**Test Date:**	19 Apr 2022

Pre-Conditions:	Beginner hobbyist using open Flow. Fish has 3 diseases (Cloudy eye, ich and roundworm), and user is only aware of physical symptoms, and unaware of behavioural symptoms. User will keep mentioning fish symptoms as long as it is not selected

Step	Test Step	Test Data	Expected Result	Actual Result	Comments
1	Enter test data into user input box and send	Fish eye looks whitish at the outer ring, There seem to be a worm sticking at the fish anus, Several white spots are seen on fish body	Some response from system	System responded with 16 symptom questions with 10 checked	9 out of 10 auto check not true for test case
2	User uncheck the irrelevant AI checked symptoms	Only check “Are there white spots on the body and gills?”			
3	Enable chat & Enter test data into user input box and send	Fish eye looks whitish at the outer ring, There seem to be a worm sticking at the fish anus	Cloudy eye, ich and roundworm should be within top 3 suspected disease	Ich is number 1 suspected disease	
4	Repeat step 2 above	Only check “Are there any worms at the anus region?”, “Does your fish's eyes look white or greyish?”	Cloudy eye, ich and roundworm should be within top 3 suspected disease	Cloudy eye, ich and roundworm are the top 3 suspected diseases	3 out of 4 auto check not true for test case

Result (Pass/Fail):	Pass
Post-Conditions:	User is happy to accept the a low confidence diagnosis and has to repeat themselves

5.2. Evaluation

All tests performed encompass a substantial portion of happy path tests cases with results indicating an 8 out of 11 or 72% passes based on the assumption that all test cases were performed with reasonable user behaviour. The application accuracy can certainly be further optimized but given this is an MVP, it is concluded that OhMyFish have fulfilled its intended purpose of being the solution for the problem statement that is to provide users with guided and unguided self-service diagnostic inquiry for their pet fishes.

6. Known Limitations and Opportunities for Future Development

A. Key Performance Indicators

The MVP version of OhMyFish is incapable of evaluating the effective of each diagnostic inquiry so an area of future development is to introducing chatbot performance Key Performance Indicators (KPIs) after deployment of application. KPIs can be split into 2 categories – quantitative and qualitative. There are many key indicators for evaluation of chatbot performance. [24] However, the best indicators to track vary depending on the project's goals. Indicators which are relevant to the project's goals are selected to track and evaluate OhMyFish chatbot performance after deployment:

a. Quantitative KPIs:

- Bounce rate - Percentage of user sessions that fail to result in the intended "specialized" use of your chatbot. An elevated rate indicates that the chatbot is not being consulted on subjects that are relevant to its area of competence. If bounce rate is high, then we would potentially need to update its knowledge base to expand its area of competence, but this also depends on the project's long-term goals and objectives.
- Questions per conversation - the number of questions our chatbot needs to be asked before it can provide necessary information to the users. A high number would indicate that the search algorithm for disease diagnosis is not optimised
- Non-response rate - Measures number of times that the chatbot fails to respond to user's input. This would reveal the limitations of the knowledge graph and/or topic modelling algorithm, thereby identifying potential areas of improvement.
- Most frequently asked questions – In OhMyFish case, this is analogous to the most frequently searched symptoms. By aggregating the statistics on this metric, we should be able to improve the accuracy and speed of diagnosis by learning new probabilistic weights between symptoms and diseases. Currently, OhMyFish has a feature which logs the user input data as a case node in neo4j database which is designed to serve this specific function. This would potentially drastically improve the current chatbot's performance as data is insufficient and thus, the probabilistic weights between the symptoms and diseases are set to be equal as mentioned in Section 3.4.4.

b. Qualitative KPIs

- User feedback – there are two steps of gathering user feedback after the chatbot session has concluded, which in this case, happens when diagnosis of disease is confirmed. The first step is to simply ask the users to reply "yes" or "no" or a 5-star rating to the question "Were you satisfied?". The second step would be a follow-up questionnaire which asks the users more in-depth questions to obtain more specific information. Examples of some follow-up questions would be "Is the information clear?" or "Do you have any suggestions for improving our chatbot?" By aggregating the responses, the satisfaction rate (average score of user evaluations) and evaluation rate (the percentage of sessions in which the user has evaluated chatbot's evaluation at least once). By keeping track of these KPIs over time, potential areas of improvement can be easily identified and worked on in the futures to deliver better user experience.

B. Bidirectional Speech Recognition

Supporting only textual input is a major limitation in OhMyFish because it makes the diagnostic inquiry extremely laborious for the hobbyist and extremely so if it becomes extended. Enabling bidirectional speech recognition to convert speech-to-text and text-to-speech will greatly enhance the usability of OhMyFish.

C. Computer Vision

One pain point of any pet fish hobbyist and especially novices is the inability to astutely recognize symptoms that can be articulated and that inhibits the effectiveness of OhMyFish. Enabling computer vision in OhMyFish to support any hobbyist uploading either images or videos of their pet fishes for the identification of imaging symptoms will elevate the effectiveness of OhMyFish tremendously. AI in medical imaging is gaining popularity and it is certain that this area of development will make OhMyFish a game changer in this domain.

D. Sentiment Analysis

Enabling sentiment analysis to the diagnostic inquiry of the hobbyist will add a second dimension to evaluate the effectiveness of the inquiry based on the emotional tone of the inputs.

E. Interoperability

OhMyFish is currently supported on a browser that limits its user base and enabling interoperability to extend onto other platforms such as Telegram, Whatsapp will extend the reach of OhMyFish and increase the potential user base.

F. Plugins to External Knowledge Bases

The current limited knowledge base that OhMyFish is built upon can be greatly extended by enabling data plugins to external knowledge bases such as Quora that has a community for pet fishkeeping. Quora's structured data of questions and answers will not only enable OhMyFish to expand its knowledge base but will also enable reinforced learning based on the responses to symptoms and effectiveness of associated treatments.

7. Conclusion

Covid-19 has brought about permanent changes that have directly and will continue to elevate pet fishkeeping as a hobby so hobbyists will need an equally permanent solution to address the problems of professional veterinary services. And to that end, OhMyFish is an Minimum Viable Product (MVP) version of an intelligent reasoning veterinary chatbot that we (***Group 8***) have designed and developed as the first iteration to that permanent solution.

Incorporating knowledge from major Machine Learning domains comprising Natural Language Processing, Natural Language Understanding, Deductive Reasoning, Logic-Based Inference, Reinforced Learning and practical techniques such as Topic Modelling, Graph Database, spaCy, Pandas, Hyperparameters that we have acquired throughout the Intelligent Reasoning System (IRS) Graduate Certificate, OhMyFish has been designed to be easily extensible and scalable. In addition, OhMyFish is built with an intuitive user interface to facilitate ease of use.

And based on the testing conducted to evaluate the accuracies of various diagnostic inquries, it is found that OhMyFish has a 72% pass rate that we concluded to have met its objectives as an MVP version built on a limited knowledge base and will deliver good value in the domain of intelligent pet fishkeeping veterinary services.

While acknowledging the value and viability, we are also cognizant there are still inherent limitations such support for only textual data, inability to understand if the hobbyist is beginning to be disengaged by an ineffectual diagnostic inquiry, lack of interoperability and a limited knowledge base. These limitations can open up promising areas for future development that includes Speech Recognition (Speech-to-Text and Text-to-Speech) for a genuine conversational capability, Computer Vision to enable the usage of images and videos of pet fishes, Sentiment Analysis to evaluate the usefulness of the diagnostic inquiry to the hobbyist, 3rd Party Integrations to applications like Telegram and Whatsapp, Plugins to External Knowledge Bases such as Quora and KPI Measurements to measure performances.

In wrapping of this rewarding learning journey, we (Group 8) are thankful for being able to apply what we have learned from the Intelligent Reasoning System (IRS) Graduate Certificate and what we have discussed, debated, learned, designed, and developed. We are also extremely thankful to our advisor for his constant guidance, and we conclude this project more knowledgeable about intelligent reasoning system than before we had started.

8. Appendix

8.1. References

[1] X. Lu, S. Leslie, L. Kay and J. C. L. Chow, “Chatbot for Health Care and Oncology Applications Using Artificial Intelligence and Machine Learning: Systematic Review,” National Library of Medicine, 29 November 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8669585/.

[2] D. A. Negro and D. V. Kus, “Bring Order to Chaos: A Graph-Based Journey from Textual Data to Wisdom,” neo4j, 19 September 2018. [Online]. Available: https://neo4j.com/blog/bring-order-to-chaos-graph-based-journey-textual-data-to-wisdom/.

[3] H. Khoo, “More people in Singapore interested in adopting or fostering pets during Covid-19 pandemic,” 29 August 2020. [Online]. Available: https://www.straitstimes.com/lifestyle/home-design/more-interested-in-adopting-or-fostering-pets-during-covid-19-pandemic-as-they.

[4] N. Awang, “Unable to travel, some young people turn to aquascaping in fish tanks as a different way to escape,” 31 October 2020. [Online]. Available: https://www.todayonline.com/singapore/unable-travel-abroad-young-people-are-turning-aquascaping-fish-tanks-different-way-escape.

[5] R. Hirschmann, “Share of respondents who were pet owners in Singapore as of October 2018, by pet type,” 19 June 2019. [Online]. Available: https://www.statista.com/statistics/1001965/singapore-pet-ownership-rate-by-pet-type/.

[6] F. Nazren, “Circuit breaker: Elective vet services, including husbandry & sterilisation, not provided at vet clinics,” 8 April 2020. [Online]. Available: https://mothership.sg/2020/04/vet-covid-19-sterilisation/?msclkid=12744cbca79611eca72833c30b6bf3a2.

[7] K. Lui, “More people in S'pore are getting pets despite higher prices during Covid-19,” 22 January 2021. [Online]. Available: https://mothership.sg/2021/01/singapore-pet-ownership-increase-covid/?msclkid=42d15109a5b411ecacd3f7f092e0f54a.

[8] T. Economy, “Live ornamental fish | Imports and Exports | 2020,” 24 November 2021. [Online]. Available: https://trendeconomy.com/data/commodity_h2/030110.

[9] Q. Ang and W. K. Ng, “More pet owners, community feeders struggle with upkeep of animals amid Covid-19 pandemic,” 24 January 2021. [Online]. Available: https://www.straitstimes.com/singapore/more-pet-owners-and-community-feeders-struggle-with-upkeep-of-the-animals-amid-pandemic?msclkid=d0ef4482a5af11ec875f6aae688a3736.

[10] N. Rayda, “Catfish, betta and flower horn: How COVID-19 spawned interest in fish keeping among Indonesians,” 11 July 2020. [Online]. Available: https://www.channelnewsasia.com/asia/indonesia-covid-19-fish-keeping-catfish-betta-and-flower-horn-662436.

[11] C. Adams, “ModestFish,” 8 July 2021. [Online]. Available: https://modestfish.com/fish-disease-guide/. [Accessed 20 February 2022].

[12] "the sprucePets," [Online]. Available: https://www.thesprucepets.com/. [Accessed 20 February 2022].

[13] R. Horev, "BERT Explained: State of the art language model for NLP," 11 November 2018. [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[14] A. Sieg, "Text Similarities : Estimate the degree of similarity between two texts," 5 July 2018. [Online]. Available: https://medium.com/@adriensieg/text-similarities-da019229c894.

[15] Hugging Face, "sentence-transformers/bert-base-nli-mean-tokens," 5 August 2021. [Online]. Available: https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens.

[16] S. Kapadia, "Topic Modeling in Python: Latent Dirichlet Allocation (LDA)," 15 April 2019. [Online]. Available: https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0.

[17] C. Doig, "Introduction to Topic Modeling in Python," 2015. [Online]. Available: https://chdoig.github.io/pygotham-topic-modeling/images/lda-4.png.

[18] P. Fabian, V. Gael, G. Alexandre, M. Vincent, T. Bertrand, G. Oliver, B. Mathieu, P. Peter, W. Ron, D. Vincent, V. Jake, P. Alexandre, B. Matthieu, P. Matthieu and D. Edouard, "sklearn.decomposition.LatentDirichletAllocation," scikit-learn, 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html.

[19] D. Cer, Y. Yang, S.-y. Kong, H. Nan, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope and R. Kurzweil, "Universal Sentence Encoder," Cornell University, 12 April 2018. [Online]. Available: https://arxiv.org/pdf/1803.11175.pdf.

[20] H. (. K., "Sentence Similarity with TensorFlow.js Sentence Encoder," [Online]. Available: https://jinglescode.github.io/textual-similarity-universal-sentence-encoder/.

[21] M. e. a. Abadi, "Universal Sentence Encoder," 6 January 2021. [Online]. Available: https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder.

[22] Y. Verma, "Word2Vec vs GloVe – A Comparative Guide to Word Embedding Techniques," October 2019. [Online]. Available: https://analyticsindiamag.com/word2vec-vs-glove-a-comparative-guide-to-word-embedding-techniques/.

[23] Unbox Research, "Word vectors with spaCy," [Online]. Available: http://mlreference.com/word-vectors-spacy.

[24] "visiativ," [Online]. Available: https://www.visiativ.com/en/actualites/news/measuring-chatbot-effectiveness/. [Accessed 16 February 2022].

8.2. Image References

ImageURL	Source
bloat	https://bullet1227.rssing.com/chan-53097784/all_p23.html
bloodeye	https://www.reddit.com/r/Goldfish/comments/52y5dq/blood_in_telescopeeye/
noimage	https://wdrfree.com/stock-vector/no-image-available
redfin	https://www.myaquariumclub.com/why-are-red-streaks-on-fantail-goldfish-veil-two-goldfish-as-shown-in-phot...-720777.html
clampfin	https://forum.aquariumcoop.com/topic/1246-clamped-fins-on-betta/
cloudskin	https://snapirimfarm.com/product/%D7%AA%D7%A8%D7%95%D7%A4%D7%94-%D7%9C%D7%98%D7%99%D7%A4%D7%95%D7%9C-%D7%91%D7%9E%D7%97%D7%9C%D7%95%D7%AA-%D7%91%D7%A7%D7%98%D7%A8%D7%99%D7%90%D7%9C%D7%99%D7%95%D7%AA-ektol-bac-plus-250-jbl/
greyeye	https://watcherfish.com/online-hospital-https:/watcherfish.com/online-hospital-aquarium-fish/diagnosis-based-on-picture/visible-signs-on-the-fish-body/bacterial-infections-fish/
cottonscale	http://www.londonaquariasociety.com/wp-content/uploads/2017/10/October-2017-newsletter.pdf
cottongrowth	https://www.pinterest.com/pin/double-tail-bettas--99642210494424534/
cspine	https://helpusfish.com/1/10/guppy-bent-spine-causes-treatments.html
anglespots	https://www.aqua-tipps.de/samtkrankheit/2F&psig=AOvVaw0xWvRdLf_KtfaCuQ8ztZ2Q&ust=1649767186969000&source=images&cd=vfe&ved=0CAoQjRxqFwoTCMi42uyDjPcCFQAAAAAdAAAAABAD
dotbody	https://www.tetra-fish.com/learning-center/troubleshooting/fish-illnesses-how-to-spot-them.aspx2Ftroubleshooting%2Ffish-illnesses-how-to-spot-them.aspx&psig=AOvVaw2iGbIH0No_5HVy12TyeaMp&ust=1649767209460000&source=images&cd=vfe&ved=0CAoQjRxqFwoTCJiA1_WDjPcCFQAAAAAdAAAAABAI
dotfin	https://www.facebook.com/aquafarmsantigua/posts/look-out-for-lice-on-your-fish-symptoms-fish-are-aggravated-and-restless-usually/272790566559481/
noimage	https://wdrfree.com/stock-vector/no-image-available
eyeprod	http://wallpaperswide.com/telescope_goldfish_aquarium-wallpapers.html

finscaleloss	https://www.pencilperceptions.com/learn-to-draw-with-these-exercises/
brownrust	https://gdmorganic.com/mata-ikan-cupang-bengkak/
whiteedge	http://turbonfree.chat.ru/fishdoc/disease/finrot.htm
flashing	https://aquariumsathome.com/why-do-fish-rub-themselves-against-rocks/
frayfin	https://www.sobatakuatik.com/ikan-cupang-panduan-lengkap-cara-merawat-jenis-makanan-dll/
fraytail	https://bettablog.me/allaboutbettafish
gasping	https://ajkreview58.com/why-are-my-fish-breathing-heavy/
gillopen	https://www.tropicalfishsite.com/gill-mites-fish-disease/
greydot	https://www.tetra-fish.com/learning-center/troubleshooting/fish-illnesses-how-to-spot-them.aspx
greymucus	https://aquariumscience.org/index.php/11-13-slime-coat/
greyskinfin	https://dorothyddawson.blogspot.com/2019/11/goldfish-lymphocystis.html
greygill	https://veterinaryresearch.biomedcentral.com/articles/10.1186/1297-9716-44-27/figures/2
greymouth	https://fishlab.com/columnaris/
hiding	https://www.thesprucepets.com/missing-aquarium-fish-1381216
hollowbelly	https://aquariumscience.org/index.php/11-5-hollow-belly/
infanus	https://www.researchgate.net/figure/Common-carp-swollen-anus-and-skin-ulcers_fig3_311867630
ichlike	https://agrozine.id/cara-menanggulangi-penyakit-white-spot-pada-ikan/
swollenskin	https://watcherfish.com/aquarium-online-lab/microscope-pictures/brooklynella/
lethargic	https://japanesefightingfish.org/common-betta-fish-disease-symptoms/
browngills	https://www.researchgate.net/figure/F-columnare-elicited-gill-lesions-in-rainbow-trout-Oncorhynchus-mykiss-fry-operculum_fig1_236336784

bluemucus	https://aquaclubdesign.com/%CF%86%CE%B1%CF%81%CE%BC%CE%B1%CE%BA%CE%BF-%CF%88%CE%B1%CF%81%CE%B9%CF%89%CE%BD-%CE%B5%CE%BD%CF%85%CE%B4%CF%81%CE%B5%CE%B9%CE%BF%CF%85-%CE%B2%CE%B1%CE%BA%CF%84%CE%B7%CF%81%CE%B9%CE%B1%CE%BA%CF%89%CE%BD-%CE%BA%CE%B1%CE%B9-%CF%80%CF%81%CF%89%CF%84%CE%BF%CE%B6%CF%89%CF%89%CE%BD-%CE%BB%CE%BF%CE%B9%CE%BC%CF%89%CE%BE%CE%B5%CF%89%CE%BD-%CE%B4%CF%85%CF%83%CE%BA%CE%BF%CE%BB%CE%B7%CF%83-%CE%BC%CE%BF%CF%81%CF%86%CE%B7%CF%83-hs-aqua-ectocell-forte-20-ml-for-800-l.html
noimage	https://wdrfree.com/stock-vector/no-image-available
noimage	https://wdrfree.com/stock-vector/no-image-available
mucushead	https://japanesefightingfish.org/columnaris-betta/
palegill	https://outflourish-unproclaimed-lactifuge.xyz/bd476u1y?key=0f22c1fd609f13cb7947c8cabfe1a90d&submetric=14920667
paleskin	https://www.sciencedirect.com/science/article/abs/pii/S0001706X16300055
patchbody	https://www.serendipitywave.com/why-does-a-goldfish-turn-black/
noimage	https://wdrfree.com/stock-vector/no-image-available
infgill	http://joaquincarranza1.blogspot.com/2018/12/non-mammalian-disease.html
infscale	https://tropical-fish-keeping.com/tag/sepsis
redspot	http://www.a2systems.com.br/notes.asp?iid=153355386-red+gill+fish&cid=6
ruptureeye	https://www.tankarium.com/types-of-goldfish/
lossscale	https://www.petmetwice.com/fish-losing-scales/
scaleout	https://fishkeepingadvice.com/dropsy/
settlebottom	https://www.myaquariumclub.com/my-fish-is-lying-at-the-bottom-of-the-tanks-gasping-736495.html

distress	https://www.allpondsolutions.co.uk/fishkeeping-advice/fish-stress.html
lossweight	https://www.myaquariumclub.com/my-goldfish-has-lost-a-lot-of-weight-and-is-extremely-thin-1002072.html
bleeding	https://en.wikipedia.org/wiki/Viral_hemorrhagic_septicemia
skinpeel	https://cabettadep.com/benh-thuong-gap-o-ca-betta/
skinulcer	https://www.thesprucepets.com/treating-pseudomoniasis-in-fish-5094501
surfacedwell	https://petfishonline.com/betta-fish-top-tank/
cauliflower	https://dorothyddawson.blogspot.com/2019/11/goldfish-lymphocystis.html
noimage	https://wdrfree.com/stock-vector/no-image-available
sunkeneye	https://carpfarmingmyerscoughcollege.weebly.com/disease.html
noimage	https://wdrfree.com/stock-vector/no-image-available
noimage	https://wdrfree.com/stock-vector/no-image-available
swimdownward	https://www.theveterinarynurse.com/review/article/how-can-you-tell-when-a-fish-is-sick
thickmucus	https://21discos.com.br/doenca-do-algodao-sintomas-causas-e-tratamento/
thickfin	https://canhquansanvuonxanh.com/ca-koi-bi-nam-mang-dau-hieu-mac-benh-khong-the-bo-qua-khi-nuoi-ca-koi.html
headulcer	https://wittily-monopteron-preerection.xyz/bd476u1y?key=0f22c1fd609f13cb7947c8cabfe1a90d&submetric=14920667
noimage	https://wdrfree.com/stock-vector/no-image-available
upsidedown	https://www.myaquariumclub.com/my-goldfish-is-bloated-and-swimming-upside-down.-1051149.html
noimage	https://wdrfree.com/stock-vector/no-image-available
waxyspot	https://www.thekoiclinic.co.uk/white-blobs-on-koi-carp-pox/
whitefilm	https://discusrescue.com/lymphocystis-in-discus-discus-cauliflower-disease-treatment/

whitethread	http://www.thepetscentral.com/fish/fish-diseases/symptoms-of-aquarium-fish-diseases/
whitepatch	https://myanimals.com/fishes/
whitespot	https://cacanhthaihoa.com/kien-thuc/chua-benh-cho-ca-canh-bien.html
whitefaeces	https://www.ultimatebettas.com/betta-fish-white-stringy-poop/
whitebump	https://www.myaquariumclub.com/hello-my-goldfish-is-really-sick.-he-has-a-large-bump-on-his-side-and-blood...-484756.html
wormbody	http://www.thepetscentral.com/fish/fish-diseases/symptoms-of-aquarium-fish-diseases/
wormanus	https://wdrfree.com/stock-vector/no-image-available
yellownodule	https://www.tankarium.com/aquarium-fish-diseases/

8.3. Project Proposal

Date of proposal:

5th March 2022

Project Title:

ISS Project – OhMyFish, Your Friendly Pet Fish Veterinarian

Sponsor/Client: *(Name, Address, Telephone No. and Contact Name)*

Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore

NATIONAL UNIVERSITY OF SINGAPORE (NUS)

Contact: Mr. GU ZHAN / Lecturer & Consultant

Telephone No.: 65-6516 8021

Email: zhan.gu@nus.edu.sg

Background/Aims/Objectives:

Pet fish keeping is a popular hobbit in Singapore where hobbyists make significant financial investments to own expensive breeds such as Flowerhorn Cichlid (known colloquially as luo han yu or 罗汉鱼 in Mandarin), Asian Arowana (known colloquially as golden dragon fish or 金龙鱼 in Mandarin) and to continually maintain a conducive living environment for their pet fishes.

Despite the popularity and pricy nature of pet fish keeping, research has shown a general lack of veterinary services for pet fishes suffering from diseases and limited to a small select of pet fish retailers. This lack is now exacerbated by the Covid-19 pandemic and pushing such services to the blink of oblivion, leaving pet fish hobbyists frustrated when their beloved pet fishes fall sick leading to a complete loss from their financial investments.

And so with the intent of providing hobbyists an intelligent avenue to seek help for their diseased pet fishes, we are proposing OhMyFish which is a pet fish veterinary chatbot for hobbyists to conduct self-service diagnostic inquiries guided by an intelligent reasoning system to triangulate on the most possible disease enabling them to strategize on the appropriate treatment.

The scope of OhMyFish in this project will involve the development of a chatbot that will interact with hobbyists on the symptoms of their pet fishes through conversational English. The keywords of the symptoms will be extracted through preprocessing then queried against a disease knowledge graph with topic Modeling in an iterative yet intelligent approach with the chatbot progressively probing the hobbyist for the presence of other associated symptoms until a match is identified.

Requirements Overview:

Research ability

- Conduct research to obtain information on the popular breeds of pet fishes and associated diseases
- Conduct research to identify language models for the chatbot

Programming ability

- Data Preparation Component
    - Ingest articles on pet fish diseases to index all textual data

    - Preprocess textual data
    - Perform Topic Modeling to build a knowledge graph
- Data Access Component
- Knowledge Base Query Component
- Decision Modeling Component
- Conversational Control Component
    - Preprocesses input from the hobbyist
    - Handles the iterative probing of the hobbyist
- User Interface Component
    - Accepts inputs from the hobbyist

System integration ability

- Integrates all components mentioned
- Conduct comprehensive integration and end-to-end testing

Resource Requirements (please list Hardware, Software and any other resources)

Hardware/System proposed for consideration:

- Ubuntu 64-BIT Version 20.04 VDI

Software proposed for consideration:

- Flask (Pythonic Web Framework)
- React (Frontend)
- Python (Backend)
- Neo4J (Database)
- Chatbot (TBD)
- Learning Models

Number of Learner Interns required: (Please specify their tasks if possible)

a team of four to six project members (or individual work upon lecturer approval)

1. Lim Eng Hwee Jason (A0180557Y)
2. Tan Sio Poh (SXXXX916J)
3. Teo Wei Ming (A0249278M)
4. Teoh Jeng Wei (A0093899B)
5. Wang Song (A0026411X)

Methods and Standards:

Procedures	**Objective**	**Key Activities**
Requirement Gathering and Analysis	The team should meet with ISS to scope the details of project and ensure the achievement of business objectives.	1. Gather & Analyze Requirements 2. Define internal and External Design 3. Prioritize & Consolidate Requirements 4. Establish Functional Baseline

Technical Construction	· To develop the source code in accordance to the design. · To perform unit testing to ensure the quality before the components are integrated as a whole project	1. Setup Development Environment 2. Understand the System Context, Design 3. Perform Coding 4. Conduct Unit Testing
Integration Testing and acceptance testing	To ensure interface compatibility and confirm that the integrated system hardware and system software meets requirements and is ready for acceptance testing.	1. Prepare System Test Specifications 2. Prepare for Test Execution 3. Conduct System Integration Testing 4. Evaluate Testing 5. Establish Product Baseline
Acceptance Testing	To obtain ISS user acceptance that the system meets the requirements.	1. Plan for Acceptance Testing 2. Conduct Training for Acceptance Testing 3. Prepare for Acceptance Test Execution 4. ISS Evaluate Testing 5. Obtain Customer Acceptance Sign-off
Delivery	To deploy the system into production (ISS standalone server) environment.	1. Software must be packed by following ISS's standard 2. Deployment guideline must be provided in ISS production (ISS standalone server) format 3. Production (ISS standalone server) support and troubleshooting process must be defined.

Team Formation & Registration

Team Name: OhMyFIsh
Project Title (repeated): OhMyFish – Your Friendly Pet Fish Veterinarian
System Name (if decided): OhMyFish
Team Member 1 Name: Lim Eng Hwee Jason
Team Member 1 Matriculation Number: A0180557Y
Team Member 1 Contact (Mobile/Email): e0284043@u.nus.edu

Team Member 2 Name: Tan Sio Poh
Team Member 2 Matriculation Number: SXXXX916J
Team Member 2 Contact (Mobile/Email): siopoh@gmail.com
Team Member 3 Name: Teo Wei Ming
Team Member 3 Matriculation Number: A0249278M
Team Member 3 Contact (Mobile/Email): weimingteo@u.nus.edu
Team Member 4 Name: Teoh Jeng Wei
Team Member 4 Matriculation Number: A0093899B
Team Member 4 Contact (Mobile/Email): e0938628@u.nus.edu
Team Member 5 Name: Wang Song
Team Member 5 Matriculation Number: A0026411X
Team Member 5 Contact (Mobile/Email): wangsong@u.nus.edu

For ISS Use Only		
Programme Name:	**Project No:**	**Learner Batch:**
Accepted/Rejected/KIV:		

Learners Assigned:

Advisor Assigned:

Contact: Mr. GU ZHAN / Lecturer & Consultant

Telephone No.: 65-6516 8021

Email: zhan.gu@nus.edu.sg

8.4. Mapping of System Functionalities and Modular Courses

Courses	**Knowledge and Techniques**
Machine Reasoning (MR)	Deductive Reasoning Logic-Based Inference Reinforced Learning Text Pre-Processing Topic Modelling Knowledge Modules
Reasoning Systems (RS)	Knowledge Graph Cosine Similarilty Implicit Feedback
Cognitive Systems (CGS)	Natural Language Processing Natural Language Understanding *Text Pre-Processing*

8.5. Installation Guide

8.5.1. Linux

The following provides the instructions for installation on Linux.

A. Install Neo4j

a. Run the below command to install prerequisite
sudo apt install apt-transport-https ca-certificates curl software-properties-common

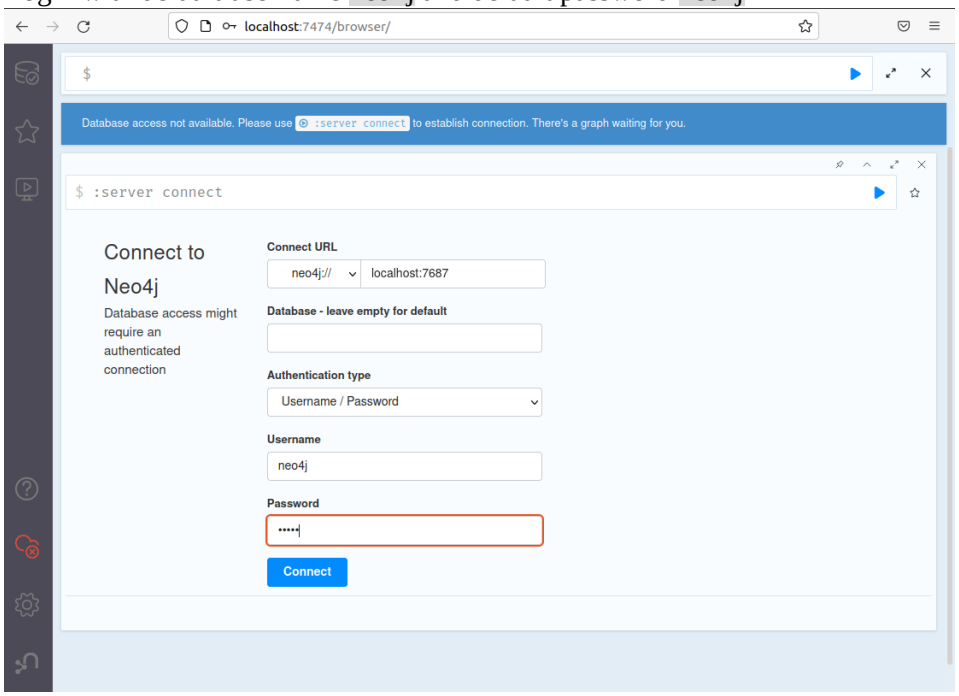
b. Add the GPG key for the official Neo4j package repository to your system
curl -fsSL https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -

c. Add the Neo4j 4.4 repository to your APT
sudo add-apt-repository "deb https://debian.neo4j.com stable 4.4"

d. Install Neo4j database
sudo apt install neo4j

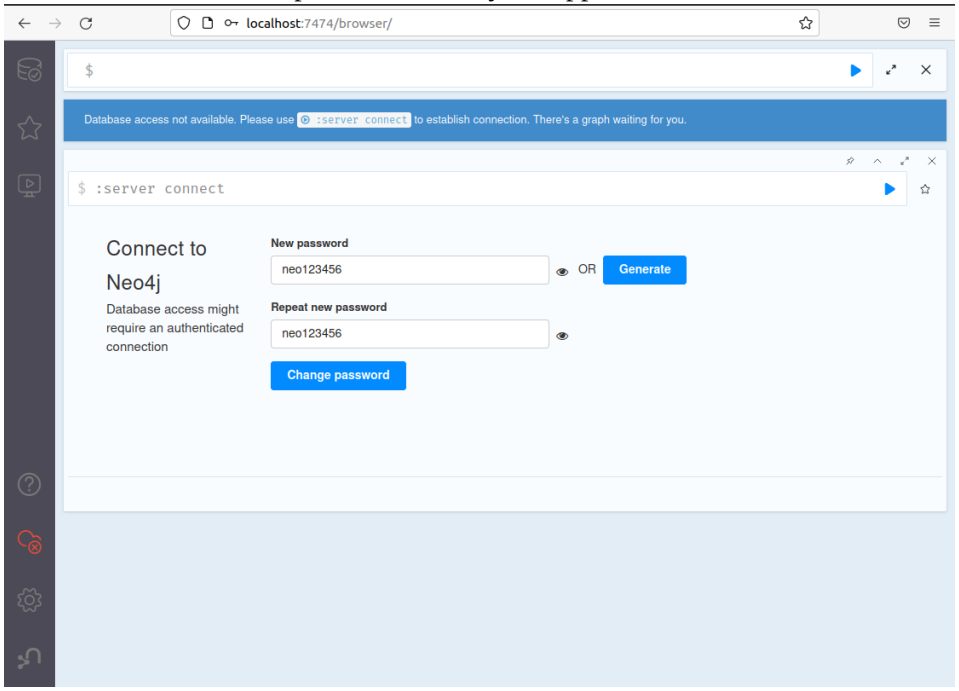
e. Start Neo4j service
sudo service neo4j start

f. Update password
Open Neo4j browser http://localhost:7474/browser
Login with default username neo4j and default password neo4j



Update neo123456 as new password

Note: This is the default password used by the application.



B. Install Anaconda (Optional)

You can activate it and skip this step if you have Python 3.9 environment

a. Run the below command to download the Anaconda installer
wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh

b. Install Anaconda
bash Anaconda3-2021.11-Linux-x86_64.sh
Follow the guide to complete the installation.

c. Create a new python 3.9.12 environment
conda create --name py39 python=3.9.12

d. Activate the environment
conda activate py39

C. Setup the application

a. Download the code repository from github and save the code under ~/MRRSProject
https://github.com/TeamEightIS04/MRRSProject

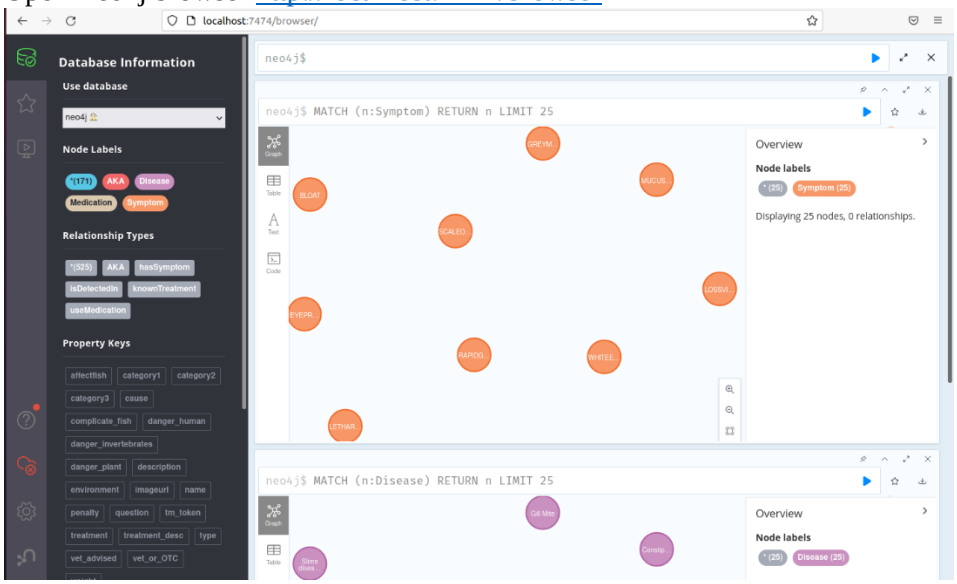
b. Go to the SystemCode folder
cd ~/MRRSProject/SystemCode

c. Install the dependencies (make sure the py39 environment is activated)
pip install sklearn spacy neo4j neomodel nltk pandas Flask flask_cors

d. Setup the data in the DB
python dataSetup.py

e. Verify the data in Neo4j

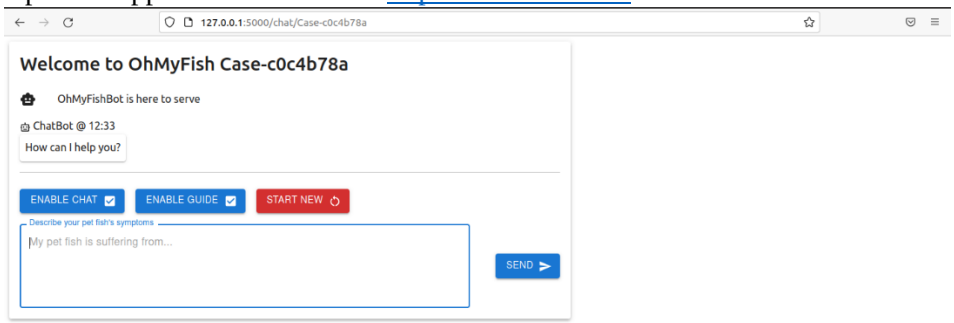
Open Neo4j browser http://localhost:7474/browser



f. Start the application

flask run

g. Open the application in browser http://localhost:5000



8.6.1 Windows

The following provides the instructions for installation on Linux.

A. Neo4J
   a. Download Neo4j desktop from https://neo4j.com/download-center/#desktop

      IMPORTANT
      After filling up the registration and while awaiting download, copy and paste the Neo4j Desktop Activation Key into a notepad to be used for the final installation step

   b. Rename the download as NEO4J_Setup.exe
   c. Note the folder which you have download NEO4J_Setup.exe. eg. c:\downloads
   d. Open up a command prompt with ADMIN rights (elevated)
   e. Copy and paste the entire code block below in the prompt (remember to change the c:\downloads to the correct folder if you have downloaded NEO4J to another folder)

      cd c:\downloads
      NEO4J_Setup.exe /allusers /S

   f. Once you boot up NEO4J, you will be asked for the activation key (which you should have it copied previously in step 1), click the register later button to complete the installation.

B. Python & Anaconda
   a. This project mainly written in Python (backend) and uses Python libraries and a compiled React frontend. Python installation is essential. Python 3.8 & above is recommended.
      https://www.python.org/downloads/windows/

   b. As we have provided an environment.yml file for sharing our package configuration, Anaconda installation is also recommended.
      https://docs.anaconda.com/anaconda/install/windows/

   c. After the above 2 application are installed, run the following in command prompt in the path where the environment.yml file is residing

      conda env create --file environment.yml

C. Download (or Git pull) OhMyFish
   a. Download OhMyFish from https://github.com/TeamEightIS04/MRRSProject or alternatively Git pull

   b. Open up a command prompt and run the following commands

   c. cd to <ohmyfish_directory_which_you_install_to>\MRRSProject\SystemCode

      $env:FLASK_APP = "app.py"
      python -m flask run

      OhMyFish will be available by default at http://localhost:5000

8.7 User Guide

The following provides the instructions for using OhMyFish.

A. Overview

a. Figure 8.7-1 below shows OhMyFish's landing page that the user will see after launching the application.

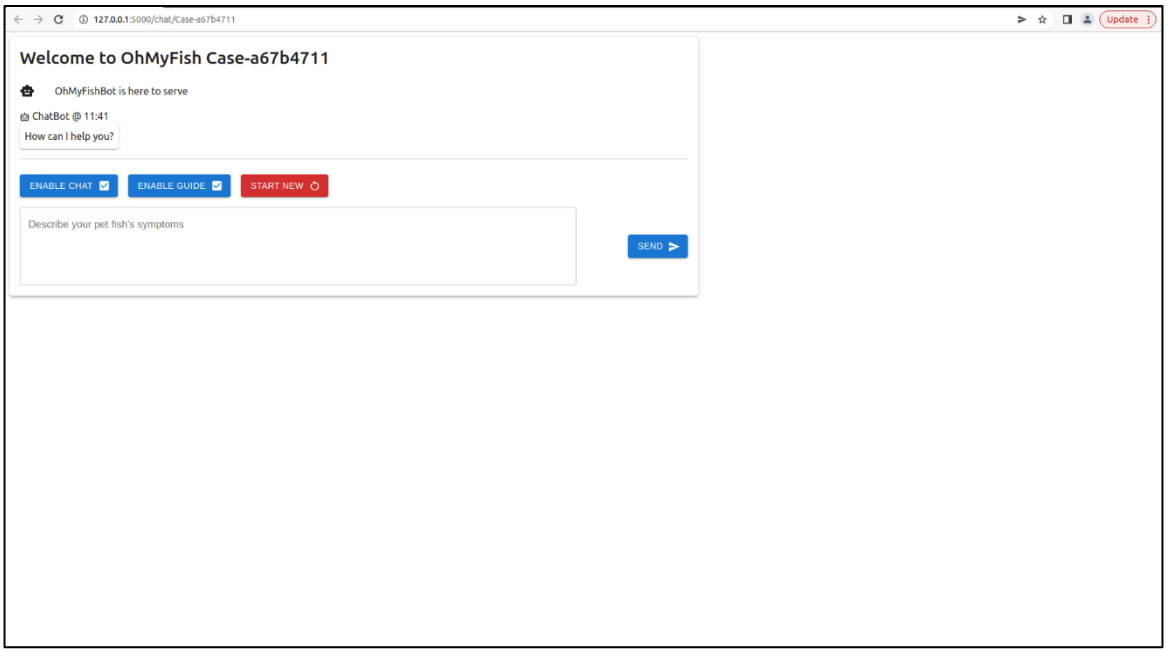


Figure 8.7-1. Landing Page of OhMyFish

b. Figure 8.7-2 below shows the 'Enable Chat' functionality of OhMyFish which enables or disables the text box. Note: Besides the enabling or disabling of the text box, there is no functionality impacts.

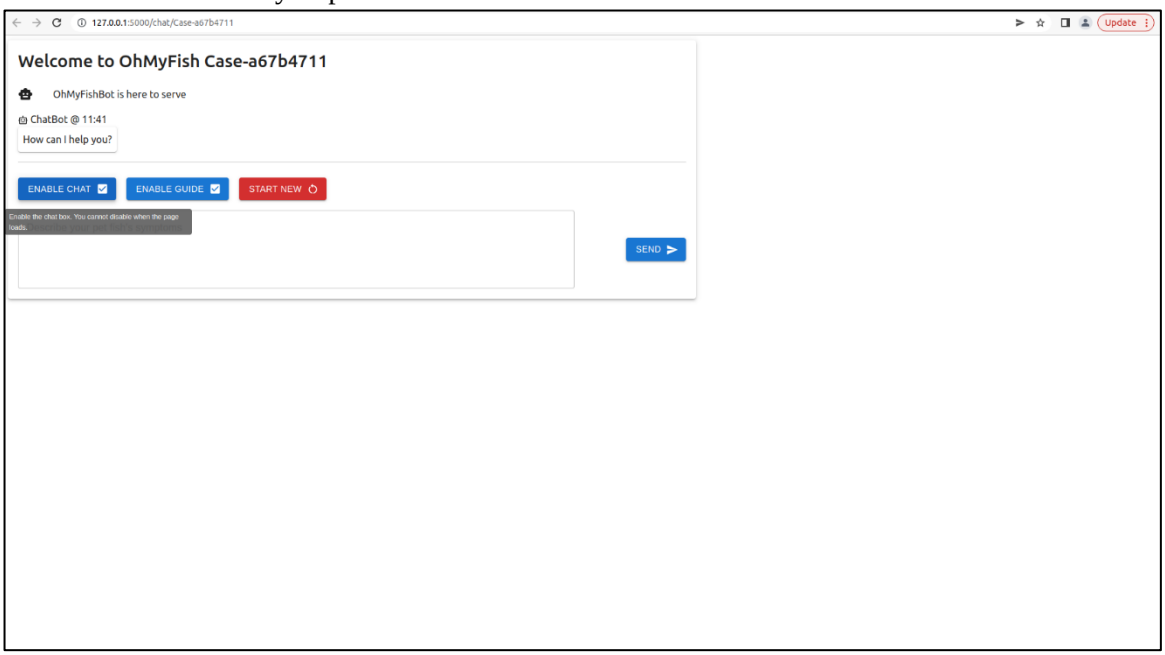


Figure 8.7-2. 'Enable Chat' functionality of OhMyFish

c. Figure 8.7-3 below shows the ‘Enable Guide’ functionality of OhMyFish which activates the Guided Approach when checked or Unguided Approach when unchecked. Guided approach is meant for novice hobbyists with OhMyFish taking charge of the inquiry from symptom recommendations to disease triangulation while Unguided approach is meant for expert hobbyists to discover the symptoms leaving only the disease triangulation to OhMyFish.

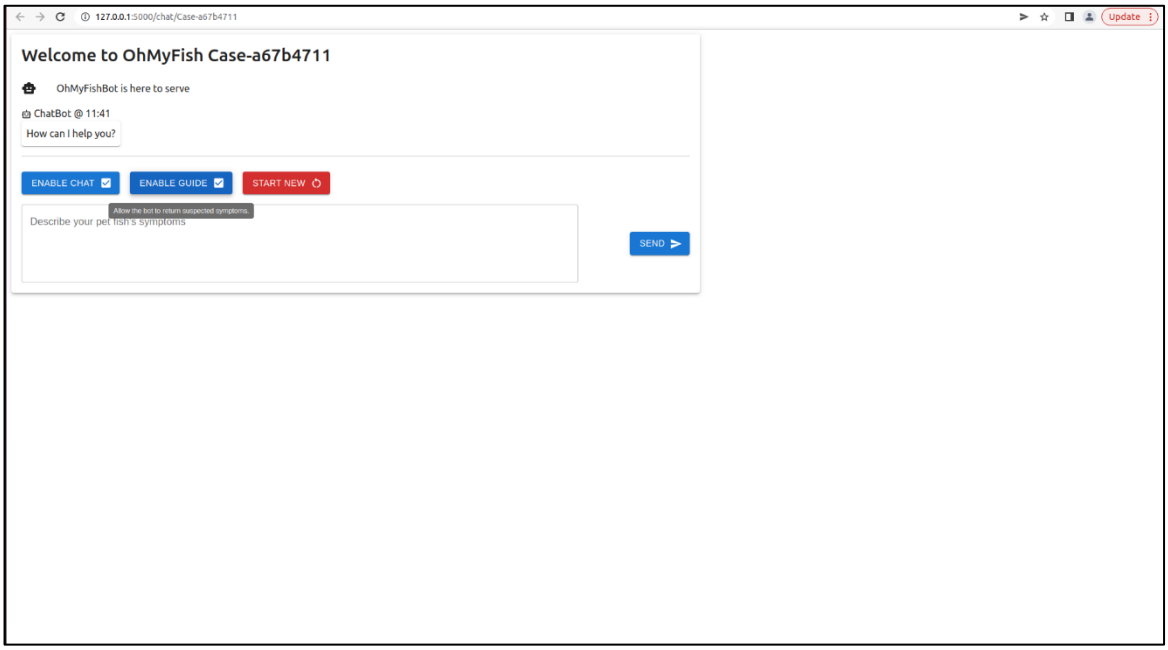


Figure 8.7-3. ‘Enable Guide’ functionality of OhMyFish

B. Guided Approach (GA)

a. Input description of visible symptoms in the chat box and click “SEND” as show in Figure 8.7-4 below.

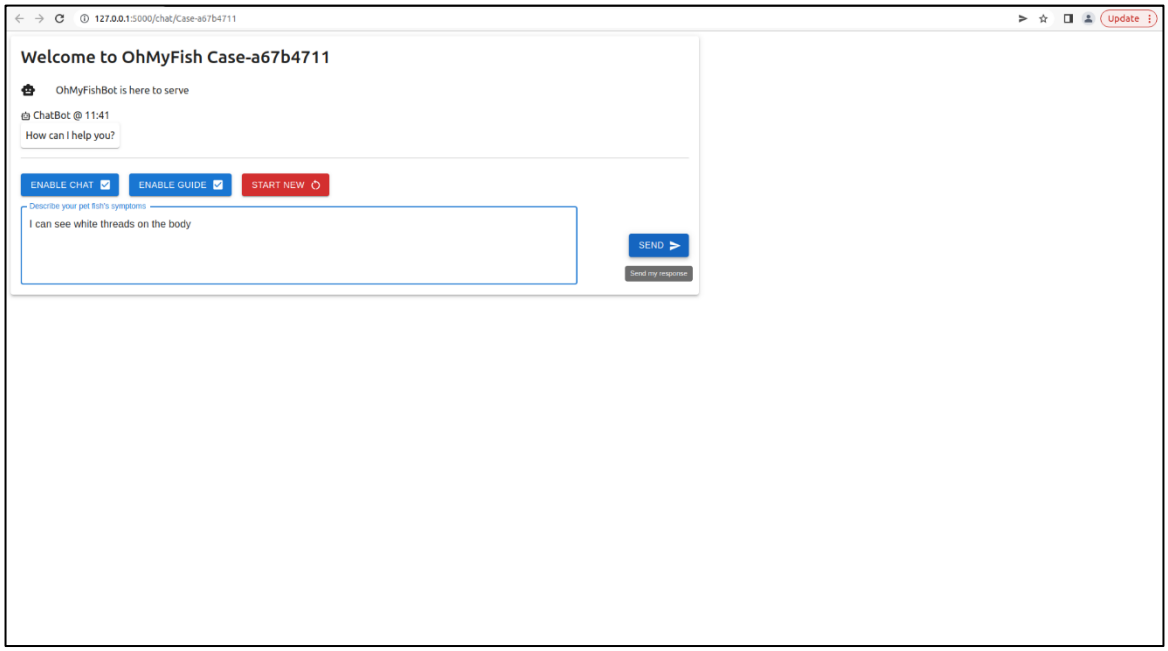


Figure 8.7-4. Using Guided Approach, user enters description of observable symptoms

b. As shown in Figure 8.7-5 below, examine the list of related symptoms then select/deselect the relevant symptoms by toggling the checkboxes. Note: Some symptoms are automatically checked as they have a high similarity score when matched against the user symptom description.

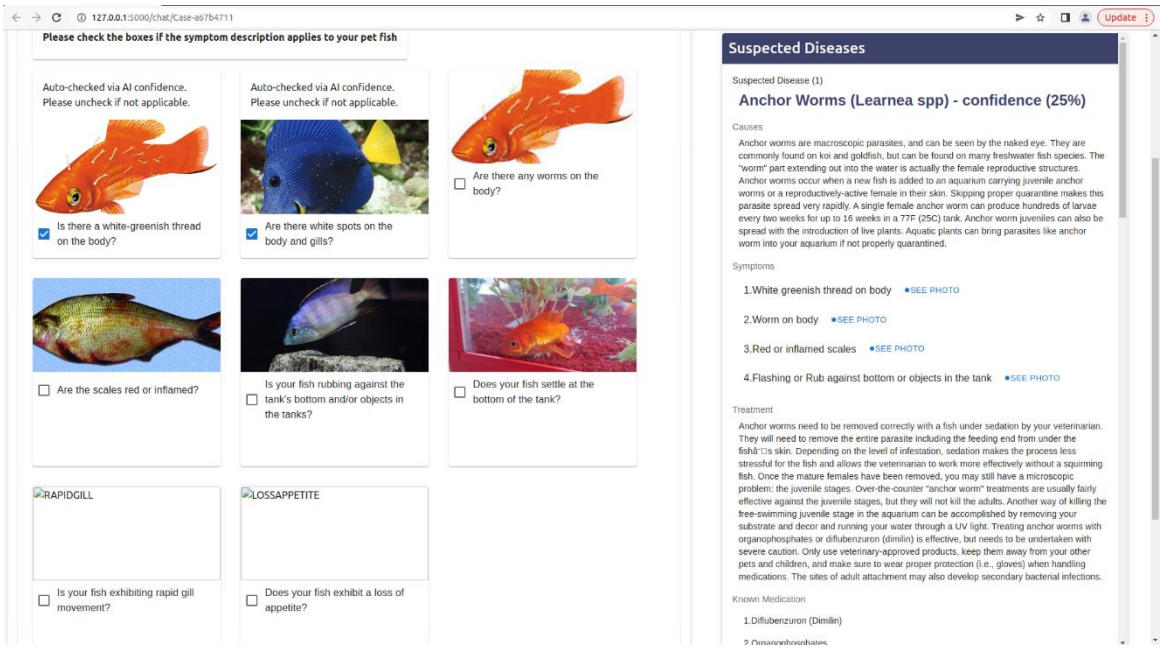


Figure 8.7-5. Using Guided Approach, OhMyFish returns the list of high-similarity symptoms

c. As shown in Figure 8.7-6 below, while selecting/deselecting the relevant symptoms that are visible to the user, the list of suspected diseases and its associated confidence level is updated on the right-hand panel. User may refer to the right-hand panel for more information of the suspected diseases such as causes, other symptoms and known treatments.

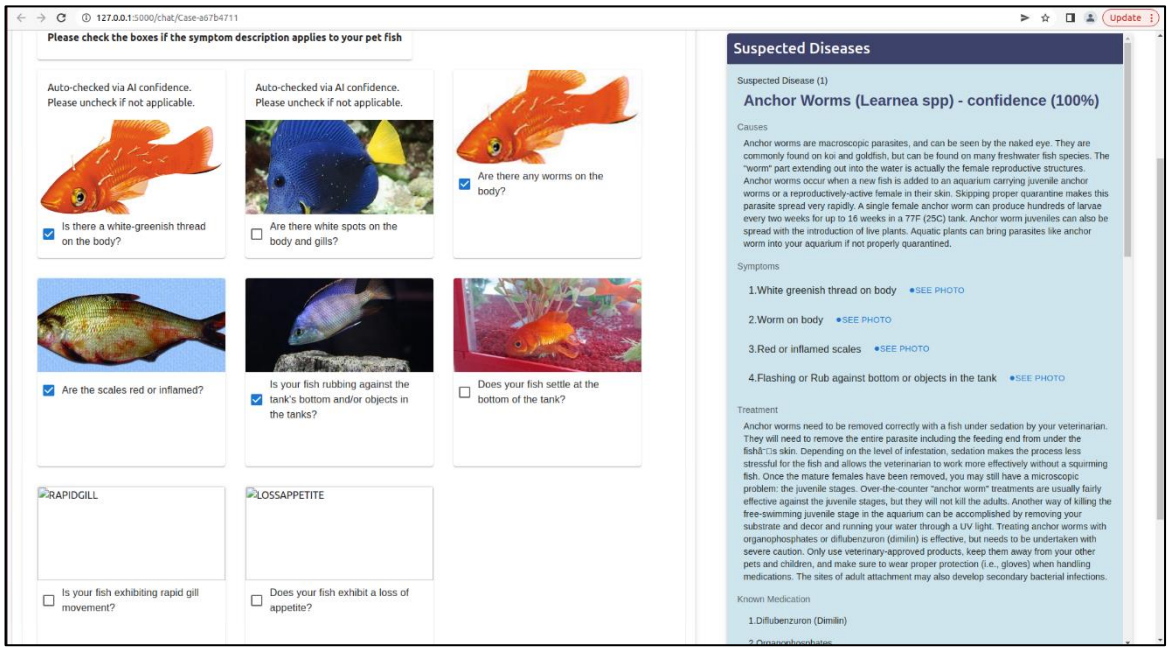


Figure 8.7-6. Using Guided Approach, user selects the relevant symptoms

d. To further refine the results as shown in Figure 8.7-7 below, the user can choose to select “SEND” to send in the list of confirmed symptoms to the chatbot for another round of search.

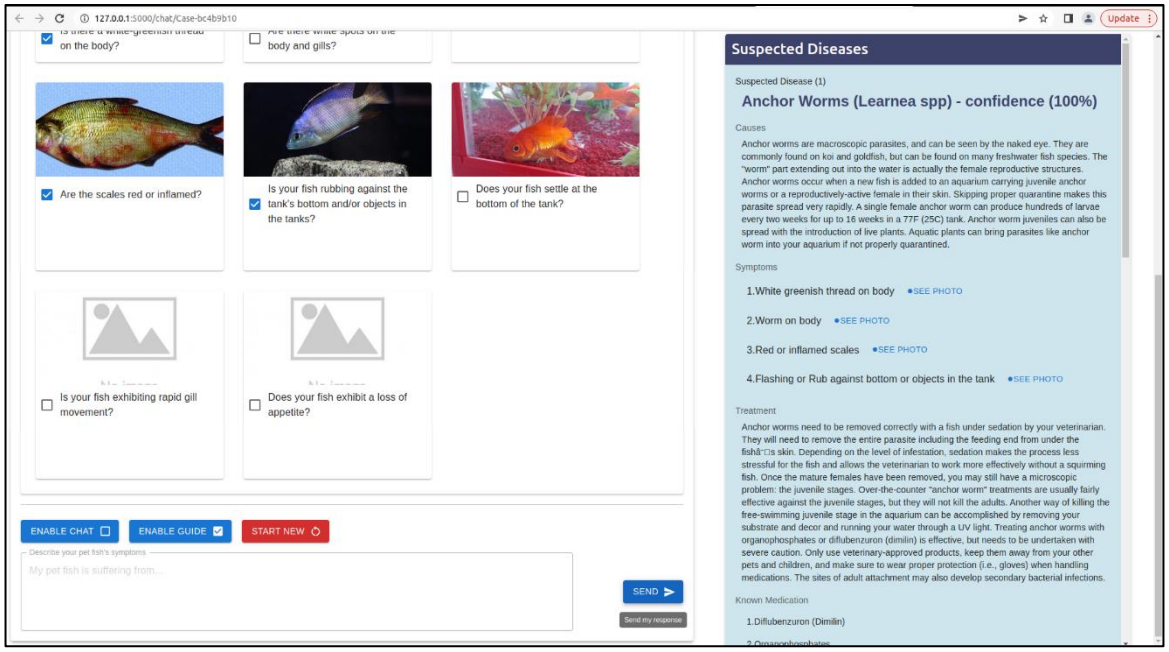


Figure 8.7-7. Using Guided Approach, user confirms the selected symptoms by clicking SEND

e. Repeat GM Steps 3 and 4 to further refine search. The list of suspected diseases and confidence level will be updated as user continues to iterate the flow until all the visible symptoms are selected as shown in Figure 8.7-8 below.

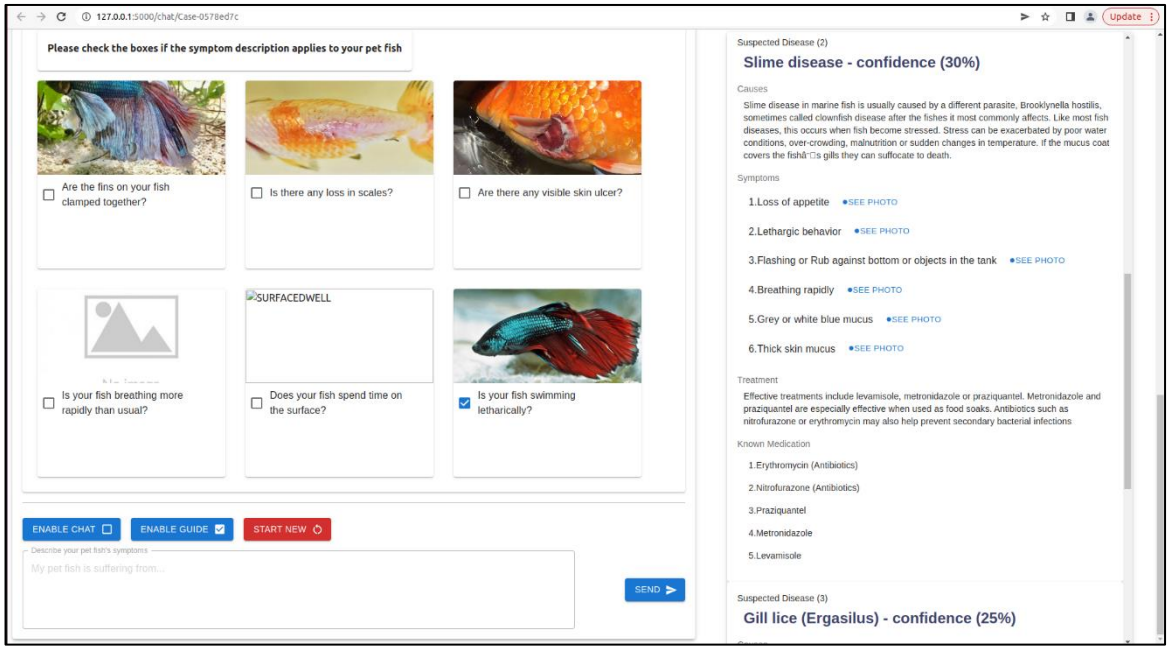


Figure 8.7-8. Using Guided Approach, user continues the selection and confirmation in the next iteration of symptoms

f. User may choose to terminate the session at any point in time by closing the browser or clicking on “START NEW” button to initialize a new session as shown in Figure 8.7-9 below.

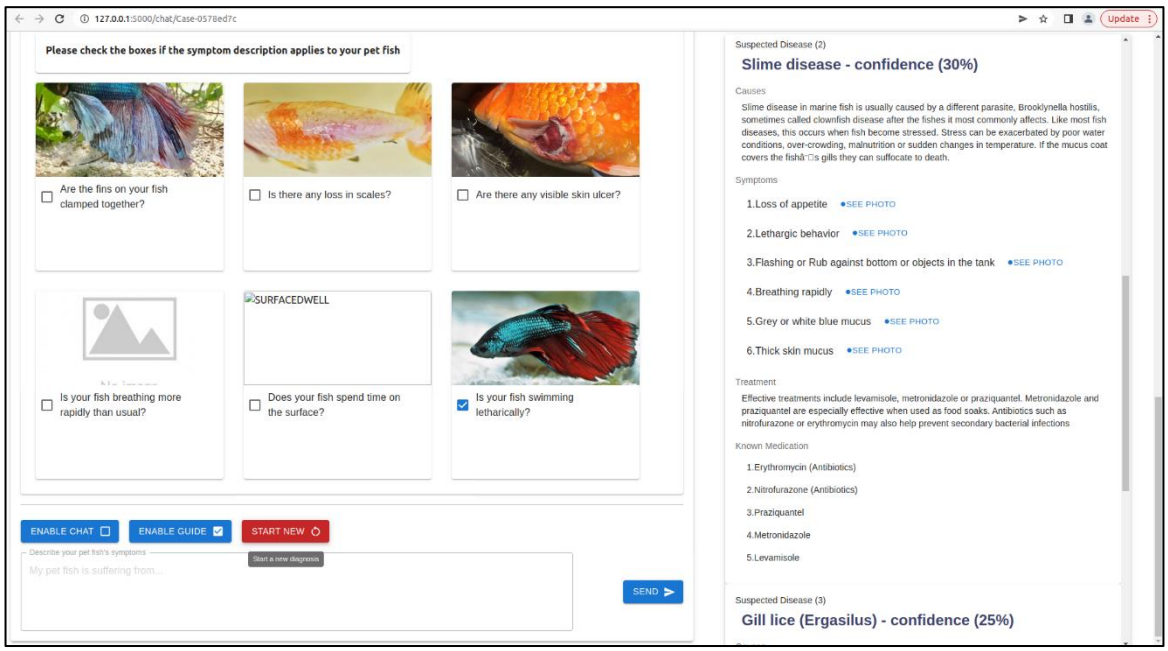


Figure 8.7-9. Using Guided Approach, user starts a new diagnostic inquiry

C. Unguided Approach (GA)

a. Deselect the “ENABLE GUIDE” button as shown in Figure 8.7-10 below.

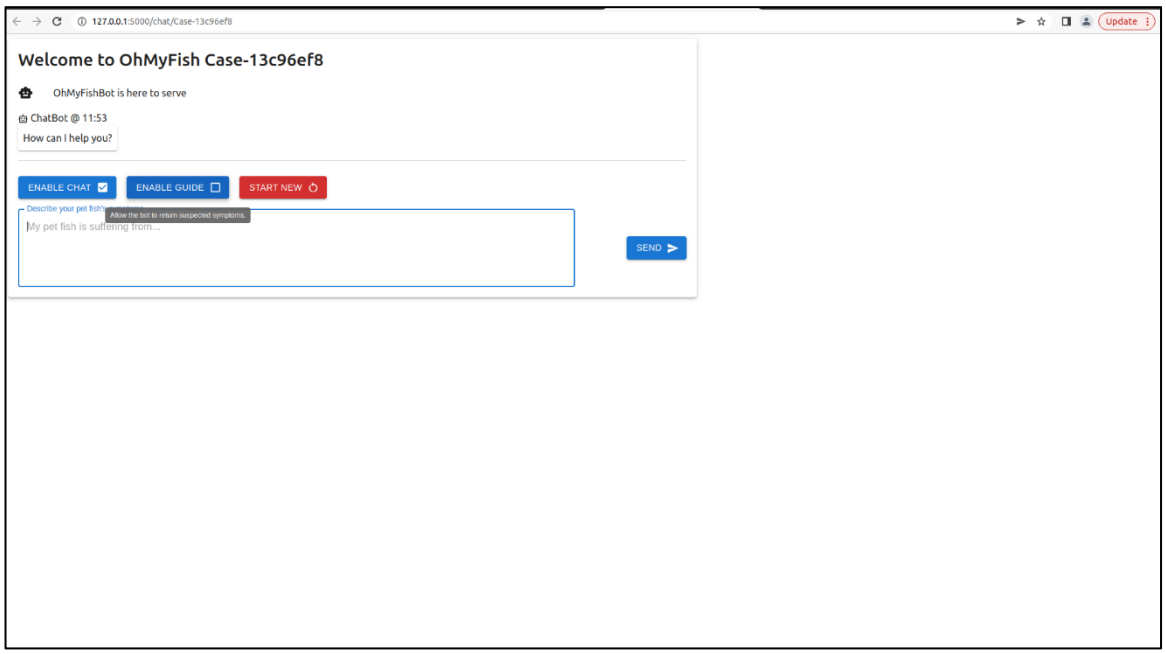


Figure 8.7-10. Using Unguided Approach, user enables the Unguided Approach by unchecking ‘Enable Guide’

b. Input description of visible symptoms in the chat box and click “SEND” as shown in Figure 8.7-11 below.

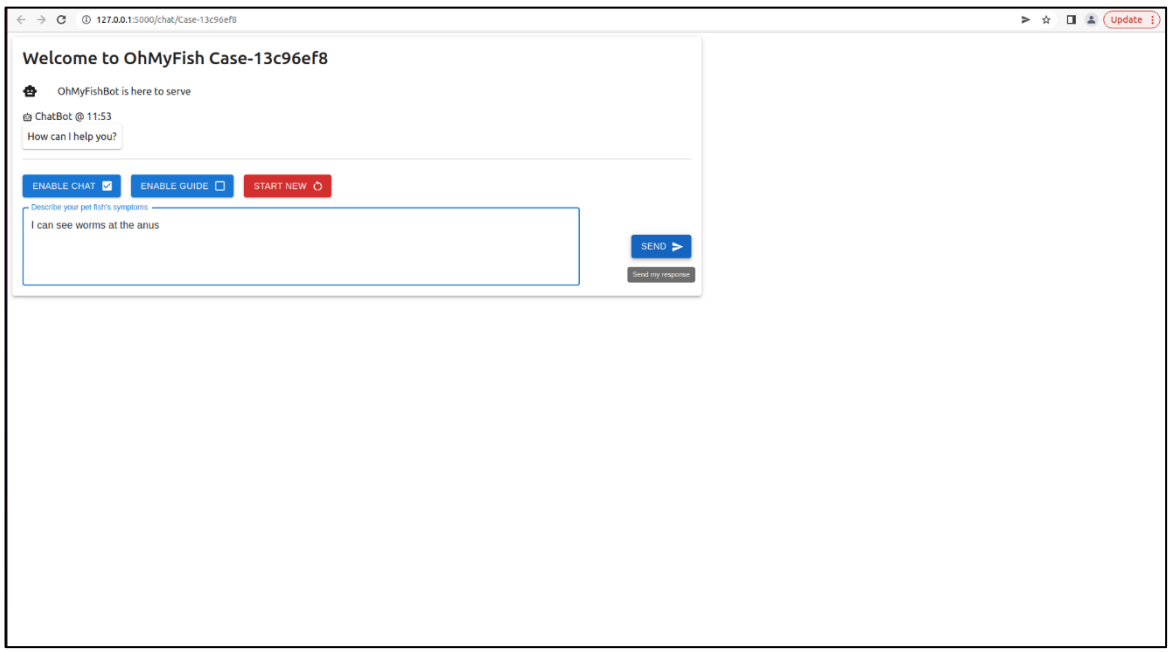


Figure 8.7-11. Using Unguided Approach, user enters description of observable symptoms

c. Chatbot returns a list of symptom(s) that are highly similar to the user's input (similarity scores above a certain threshold parameter). User can choose to deselect the symptom if it does not match what he/she is seeing as shown in Figure 8.7-12 below.

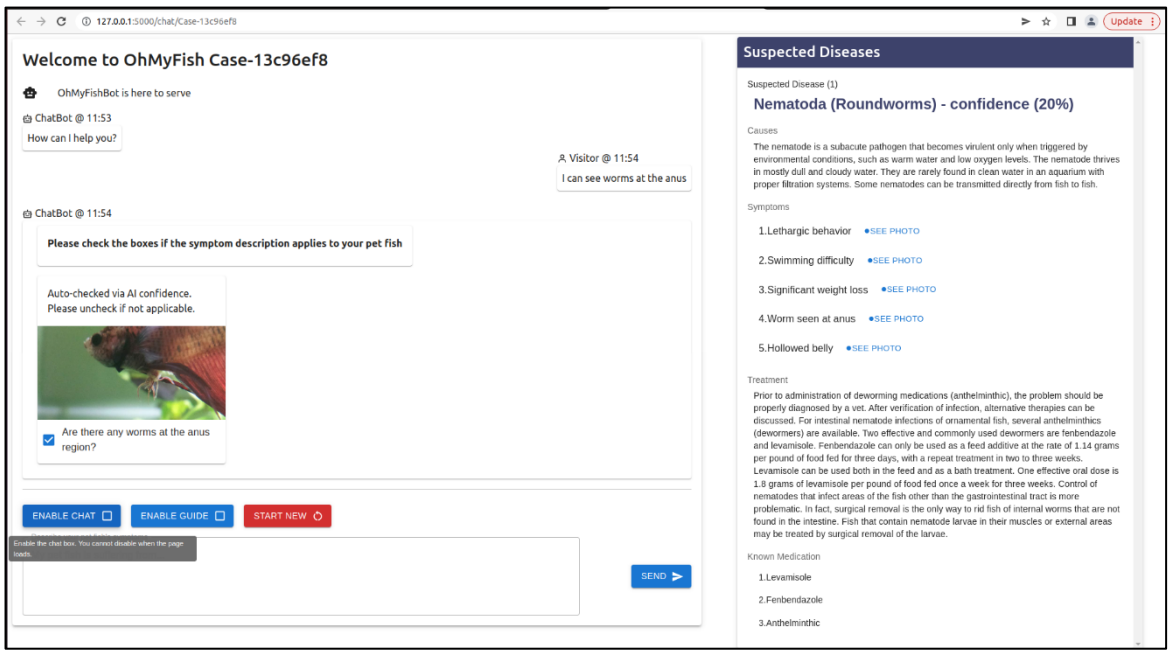


Figure 8.7-12. Using Unguided Approach, OhMyFish returns only list of high-similarity symptoms without any further symptoms of suspected diseases

d. If user would like to provide more description on visible symptoms, select the "ENABLE CHAT" button and input description in the chat box. Then click "SEND" as shown in Figure 8.7-13 below.

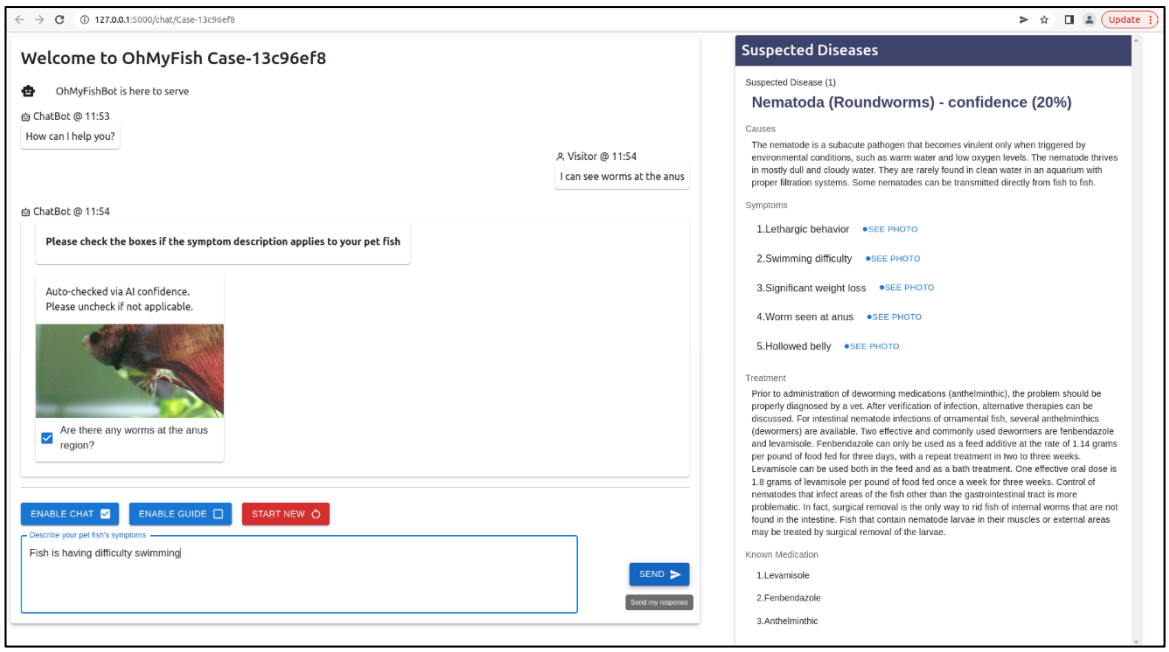


Figure 8.7-13. Using Unguided Approach, user enters description of observable symptoms

e. Repeat UM Step 3 as shown in Figure 8.7-14 below.

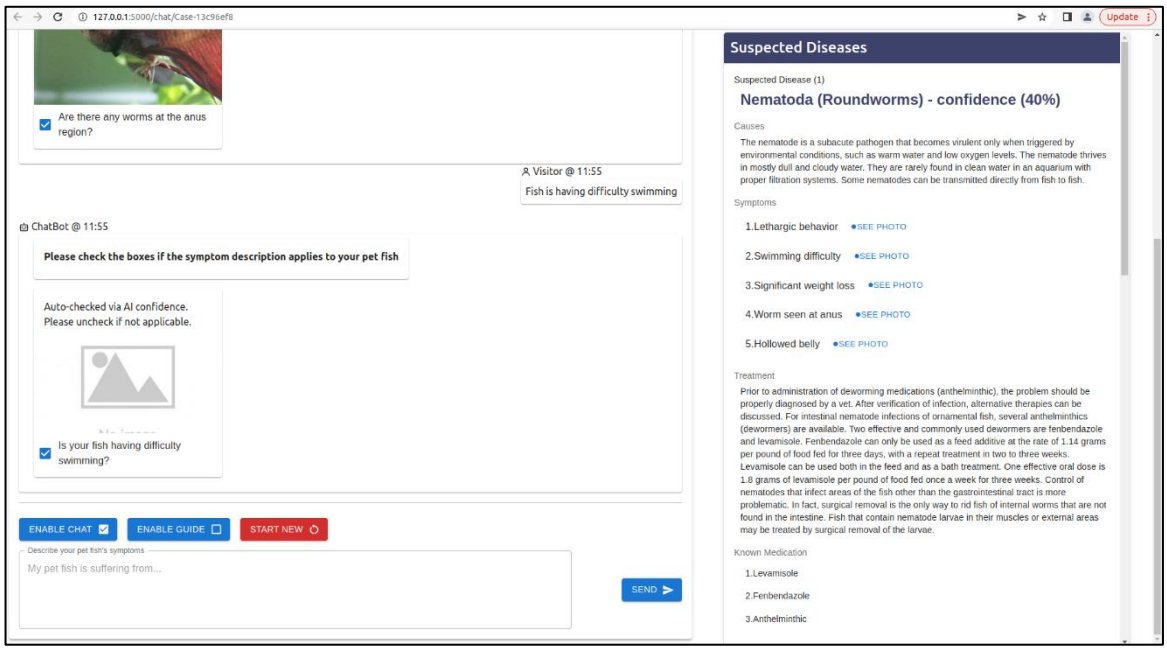


Figure 8.7-14. Using Unguided Approach, user user continues the selection and confirmation in the next iteration of symptoms

f. User may choose to terminate the session at any point in time by closing the browser or clicking on "START NEW" button to initialize a new session as shown in Figure 8.7-15 below.

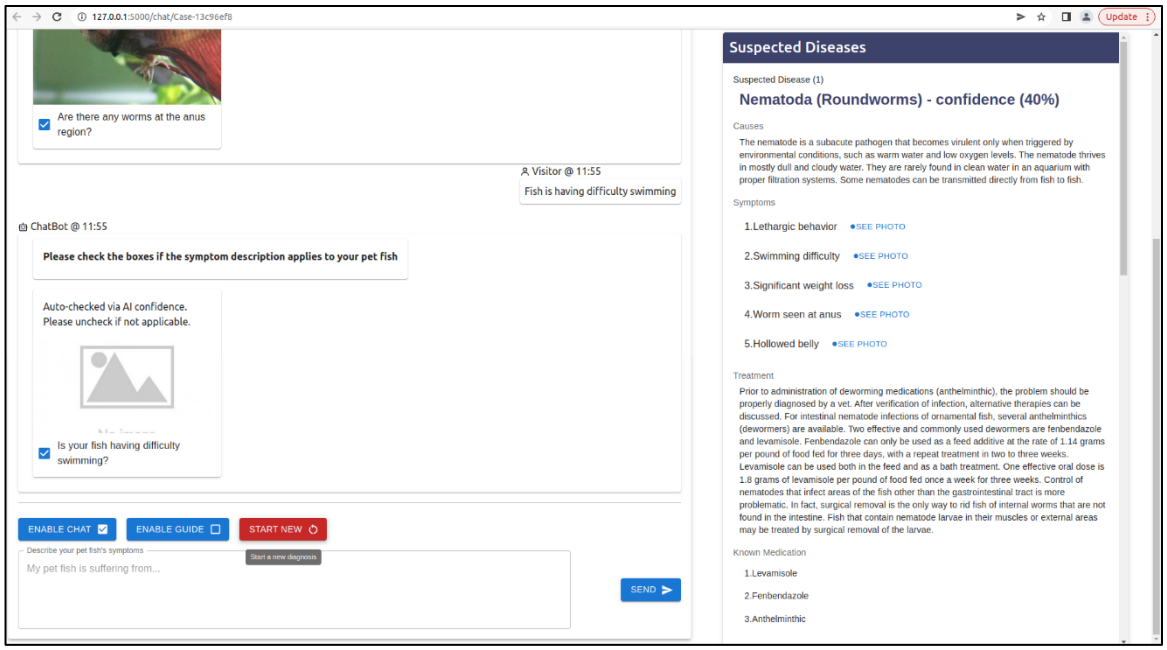


Figure 8.7-15. Using Unguided Approach, Using Guided Approach, user starts a new diagnostic inquiry

8.8 Team Member Reports

8.8.1 Jason Lim Eng Hwee

Name:	Jason Lim Eng Hwee	**ID:**	A0180557Y

Personal Contribution:

Designing the data preparation module, Neomodel layer abstraction and data access layer.

Filling up data fields related to disease treatments and causes.

Cleaning up CSV and labelling of diseases and symptoms.

Front-end UI enhancements and revamp of desired cosmetic (user) behaviours.

Testing of front and backend integration.

Test cases write up and test evaluation.

Learning Journey and Outcome:

Learned how to prepare data, and the real messy work of cleaning the data to "fit" the application.

Learn about how easy python libraries have made the basic analysis and data manipulation work - to understand why becomes more important than how, and the whole group learnt that inserting data features halfway through the project can be painful.

Hyper-parameters tuning is really a time consuming process, and results are really local maximum cases most of the time.

Finally, there is quite a mixed feeling about the final result.

From a software engineering perspective, the "acceptance level" is really subjective, and this made us realise that pre-written and agreed acceptance tests / continuous testing is very important because the results can go another way even though the software engineering aspects are right.

Application of Knowledge:

The necessity of data science inclusion in modern applications has made data science overhyped. Like the Chinese proverb of killing a chicken with an oxen knife, many of the neural networks and its variants are over engineered solutions. In our project, we made use of the basic understanding and building blocks of data science and engineered the right size solution. This helps to improve the understanding of concepts like supervised vs unsupervised, hyper-tuning topic modelling, and also give us a glimpse of what one can achieve without necessarily deep diving into complex data science topics and its application. To apply what one learn is more important than learning what to apply, and this project help us achieved both.

8.8.2 Tan Sio Poh

Name:	Tan Sio Poh	**ID:**	SXXXX916J

Personal Contribution:

My contribution includes the research, exploration and development of the Core Reasoning Intelligence or Topic Modelling Module. I have documented my journey in the whole of Section 3.7 of this report.

Within the limited time frame, my research covered a few ways to extract topical keywords from user sentence and the description of fish symptoms, and then perform sentence similarity to identify the possible symptoms and diseases.

I have also contributed to the creation of the Topic Modelling portion of the video presentation.
As a team, each of us has also contributed to the idea and design of the whole application including the database, the chatbot, etc.

Learning Journey and Outcome:

The most useful thing that I have learnt from this Graduate Certificate is a better understanding of what is artificial intelligence (A.I.) and how it can be implemented. I have learnt that there are various ways of implementing A.I., not just machine learning and deep learning, such as using decision trees, generic algorithms, fuzzy logic, text processing, etc. This gives me a deeper understanding of the evolution and definition of A.I.

The concept of Reasoning Systems is an important part that helps me to understand how to break down or construct an intelligent system. By differentiating the types of reasoning systems such as retrieval or generative or hybrid, I have learnt to identify the different techniques that can be used in different scenarios.

The Machine Reasoning and Cognitive modules provide me with a fascinating introduction to Natural Language Processing. This is also a very useful skill that I can use in my work.

Application of Knowledge:

There are a few areas where I will be applying my knowledge and skills in my workplace. I am currently working in Temasek Polytechnic and I am planning to introduce some machine learning subjects such as Natural Language Processing. I do have some ideas in mind on how to design the subject to fit the level for Polytechnic students. The important items to cover are definitely the concepts and skills in text pre-processing and introducing libraries such as NLTK and spaCy. Some other items could be using transformers, etc which I will consider in my plan.

Other areas of applications could be in developing chatbot for frequently asked question on courses/assessments/school, developing recommendation system to recommend suitable courses for students based on their ‘O’ Level/ITE results, passion, etc.

The above are just some of the applications of what I learnt that can be implemented. I will be able to work on developing suitable solutions for problems that my school has been facing.

8.8.3 Teo Wei Ming

Name:	Teo Wei Ming	**ID:**	A0249278M

Personal Contribution:

My personal contributions to this project can summarized as follows:

1. Contributed to the initial ideation and selection of the idea
2. Established the project proposal
3. Defined the problem statement, business opportunity, solution with the value proposition
4. Project managed the scope and delivery schedule
5. Participated and contributed to the system architecture and module designs
6. Researched on the pet fishkeeping and market value in Singapore
7. Established the executive summary and conclusion
8. Amalgamated the project report
9. Developed the Conversational Control Module (CCM)

Learning Journey and Outcome:

My learning journey throughout the duration of this project had led to several outcomes.

1. Understanding of a Machine Learning or Intelligent Reasoning System

   I came to understand that unlike a traditional software engineering project on a financial system or a transactional consumer-facing portal where results often have a clear definition of valid and invalid, it isn't the case for a machine learning or intelligent reasoning system where the definition for results are much broader than just being valid or invalid. And in fact, the results learned must be analysed and relearned iteratively towards producing the best optimum results. Being formerly trained as a conventional software engineer, this understanding on machine learning or intelligent reasoning system has created a new dimension in my thought process in this domain. Case in point is how our project group had to remind ourselves that there isn't a 'perfect' diagnosis but only one that best aligns with the described symptoms.

2. Knowledge and Techniques

   I have picked up knowledge in the areas of Machine Learning, Conversational AI, Deductive Reasoning, Logic-Based Inference and Reinforced Learning that I am now able to articulate upon. During the implementation of OhMyFish, I have picked up techniques on Chatbot such as Natural Language Processing, Topic Modelling and Text Pre-Processing which I'm able to apply in actual practical avenues. And for the research conducted for this project, I have read extensively on to expand my general knowledge in this domain and topics includes Conversational AI in Medical Diagnostics, Topic Modelling, Cosine Similarity, etc.

3. Tools

   After being a software engineering manager that focuses on planning and execution for many years, this project has allowed me to get reacquainted with hands-on development and specifically with Python. Throughout my development cycles, I have gained adequate development knowledge with Python and also discovered the extensive libraries within Python that includes Pandas, spaCy, Scikit-Learn, etc. Finally, I have also gained practical knowledge with a NoSQL database in the Neo4J.

Application of Knowledge:

Being an engineering manager leading the primary B2B customer-facing transactional portal with more than 500K registered users, I have an avenue to chart the roadmap for the design and implementation of an intelligent reasoning help system with a conversational chatbot using all the knowledge and techniques I have acquired. This chatbot would not only be utilized as an intelligent help system but also as a transactional recommender based on the past transactions made by the user.

8.8.4 Teoh Jeng Wei

Name:	Teoh Jeng Wei	**ID:**	A0093899B

Personal Contribution:

I worked on the Data Acquisition and Knowledge Discovery aspect of the project where I did web scrapping to extract and structure information on fish diseases, symptoms, images of symptoms, causes and recommended treatment. Next, I worked together with Jason to load the structured dataset into neo4j graph DB to create a Knowledge Graph which serves as the Knowledge Base of the OhMyFish application.

I was also in-charge of the Marketing and System Architecture videos. This involves doing up the storyboarding, animation, scripting, AI audio creation and video-audio synchronization.

I also actively participated in the discussion on system architecture design and application logic flow design during my team's weekly meetings.

Learning Journey and Outcome:

Through the discussions we had as a team, I have learnt a lot regarding software engineering – system architecture, version control, collaborative coding and documentation. I have learnt that in collaborative coding, requirements in terms of inputs and outputs must be clearly communicated within the team in order to design code that can be integrated easily together. I have also learnt to communicate my thoughts in a clear and concise manner. This was something that I had not experienced before as my projects up to this point were all solo coding projects and I now understand the importance of effective communication in a collaborative software development project. I have also learnt how to use github effectively as a means to do version control, collaborative programming and documentation.

I also had the opportunity to reinforce my understanding of the topics on machine reasoning, reasoning systems and cognitive systems taught in the lessons, through practicing and discussing about core concepts with my team mates during the course of the project work. One key concept that we really deep dived into as a team was Natural Language Processing and how to tune the model to optimise the chatbot's performance.

Application of Knowledge:

Equipped with the knowledge that I have acquired through the lessons and the group project, I will be able to better scope out potential intelligent system projects using the framework taught in the earlier classes, create a more robust and scalable system architecture and implement good coding practices. All of these should translate into higher quality applications which will eventually be deployed in the production environment within my company.

As my company is an aerospace manufacturing and MRO company, there are many types of data and knowledge (structured and unstructured). The knowledge that I have learnt about different ways of knowledge representation and knowledge discovery will be useful when I work on analytics project which uses a wide range of datasets (sensor data, quality data, machine metadata, operator worksheet, etc).

Furthermore, with the practical hands-on experience on collaborative programming, I am now more comfortable and competent in taking up or initiating projects which are bigger in scope, within my department.

8.8.5 Wang Song

Name:	Wang Song	**ID:**	A0026411X

Personal Contribution:

I worked on the overall application architecture design and the integration interfaces for all the components. From coding perspective, my main contributions are design and code the knowledge base decisioning model, design and code the backend APIs and frontend logic to integrate with the backend APIs.

I also contributed to the report for installation guide, class diagram and knowledge base model.

I also actively involved through out project to provide ideas feedbacks.

Learning Journey and Outcome:

My learnings:

1. Understanding of machine learning

   As the traditional software engineering, the business logic I developed is highly deterministic. Where for machine learning models that is not the case. For models like decision tree, the goal is not to have a perfect model that works for every case. In fact, overfitting the training data would result bad performance in real use case. Another learning is the natural language processing. I have learnt the libraries like spacy and build the project on top of them. This journey helps me understand how the machine is processing the words and there are many tools like cosine similarity, adding stop words to improve the accuracy.

2. Technologies

   I have been a backend Java developer for many years. In this project, I have learned how to use React to build interactive frontend and how to build backend application with Python Flask. It is also my first-time using NoSQL graph database Neo4J. All the technologies I have learned through the project are helpful for me to success in my role.

Application of Knowledge:

As an engineering manager leading a team for payment solutions, with the machine learning knowledge acquired through the project, we can build some model by studying the transaction histories to find out what are the areas that people would pay with installments. Also, we can build a tool using NLP to analyze consumer feedbacks about our product.