# TriageBot Project Overview

The TriageBot is intended to be a first line of diagnosis for how a User feels, using data chosen with a subset of illnesses and diseases it was to provide a base upon which a User’s symptoms could be triaged. Any secondary steps would be to provide some suggestions for treatment, either simple remedies like rest, Over the Counter (OTC) medicines, or see a doctor. With that in mind the databases were structured with a SQL and NoSQL versions so that any work would have options, I stuck with the Command Line Interface (CLI) and MySQL database to complete this phase of the project.

## **Project Planning:**

For Week 6 the intent was to do the following:

* **Environment Setup:** Be sure to update the database you developed in ITEC5020 with the necessary tools and libraries including Python, TensorFlow, and MySQL.
* **Python Review:** Revisit Python basics, focusing on essential concepts like variables, data types, and control structures.
* **Database Fundamentals:** Brush up on the fundamentals of databases, including tables, records, and SQL queries. Familiarize yourself with how databases store and retrieve data.
* **Project Planning:** Begin outlining your chatbot project. What type of chatbot do you want to create, and what capabilities will it have? Think about potential use cases and interactions.

As the project continued, I made the Environment Setup with a local Python environment, a virtual environment was created in my project directory and all the necessary libraries and compatible versions were installed. As TensorFlow and Keras had specific compatible versions some adjustments needed to be made, finding that Tokenize in Keras was no longer directly available with 3.x using NTLK and was necessary and any adjustments needed for any work I integrated later needed to be accounted for. Continuing to use MySQL required installing the MySQL libraries and adding try/except for the connections to ensure that the connection was created.

When coding I had found with the MySQL there were issues with the connection and data retrieval and focused at first on just using the JSON files as data sources, then reviewing MySQL connectors and cursors rebuilt my functions so that I was clearing the cursor with each connection which cleared up the data retrieval and responses. Adding many debugging statements to the code to ensure I was on the right path, and that I was retrieving the right information was necessary as I learned going along, uncommenting them as I moved along until I was sure I did not need them and removed them entirely. Adding comments to code, and to the functions I broke the code out into, provided a good structure and with the comments ensured I knew what work was being done.

As my database was normalized and well-structured it was easy to pull out just the information I needed for training, unsure how the chatbot would utilize and need information I could have simplified the structure more effectively early on. Although I was still able to pull out and store information, as well as provide the ability to add new diseases as necessary, which could be expanded to symptoms as well, I had the ability to read information and update it as needed.

I had intended more features for the chatbot, with more Natural Language Processing, and the use of BioBERT to help build the models using the disease information I had obtained. With the time to learn and implement I was not able to take that next step and implement BioBERT at this time but am situated to do so with any additional work I may do on this project in the future. I also intended to collect symptoms, check against existing diseases, provide a diagnosis and steps of treatment, but was able to provide an interface to prompt users for information but not perform the diagnosis yet, more future work I can perform.

## **Integration:**

Building on the chatbot project to avoid recreating the wheel, and to utilize a proven mechanism to import the intents.json and later connect to the database. Using tokenization on the intents file built out the patterns and responses necessary to start handling user inputs and questions in my domain for healthcare and illness triage. Also adding in a context history to maintain a listing of previous entries from the User so that they could be used for any further interactions, my intended design was not to include that as data to be saved but it is something I could do and improve upon.

Disease and Symptom data is stored in a MySQL database and is pulled into the chatbot so that a User can ask for information about specific Diseases or Symptoms. The initial data is from Kaggle.com - <https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset> where it is an older curated set that provides a subset of Disease and Symptom information. For improvements I added the capability to add new Diseases so that the underlying data could be improved on a case-by-case basis provides a mechanism from the chatbot to save and update the base data. This could be extended for Symptoms as well easily by copying the existing functions or extending them to support either data type.

Building out the chatbot into functions allows organizing the code so that it’s easy to follow with similar functions grouped together or put in order so that the steps can be followed by reading the code. The functions also have descriptions of what they are doing, for the most part, and there are comments to note what steps are doing if the steps are complex or are hard to follow. The complexity of much of the code could be simplified some with further work but for the length of the chatbot it was fine and didn’t go on too long, other than the main chat function which could be broken out in the future.

## **User Interactions:**

Breaking out the patterns and responses into some general subject areas for diagnosis worked to be able to provide the right interactions for the intent of the chatbot. As I have come to understand more about how the chatbot interacts with Users and how the patterns and responses are used there are more I can add. Being able to respond properly to queries about how one is feeling will need some improvement but starting from a point of triage the initial responses feel like a good start. Being able to have different methods, so the chatbot doesn’t get confused, on how to provide information about Diseases and Symptoms so that it can return information from the database was an improvement to make things easier for a User.

## **Testing and Validation:**

Continually testing the chatbot was necessary to ensure that updates and changes did not affect the underlying code. The way I did this was with the following:

* Use each pattern to ensure that it is responding with the proper response.
* Do an introduction and ask about the chatbot, and its name, so that those responses are checked.
* Mention being ill or needing a doctor, to ensure that those responses are being returned to the User.
* Add nonsense to the input to see the response returned is an I don’t understand.
* Ask about Diseases and Symptoms to see the information being returned.

While unit tests would be useful to have here, due to timing, and just starting to use AI functionality, there are none implemented yet. Having those would make it simpler and easier to check that no underlying functionality has changed and would be the next step to implement.

## **Documentation:**

Using the ReadMe to provide information on what the project is, what libraries it uses, and some setup information will provide Users the ability to understand the project and what it can do. Keeping this with the project contains everything and stores it all in one place.