

HealCare Pal

A Virtual Healthcare Assistant

By

Submitted to

The University of Roehampton

In partial fulfilment of the requirements
for the degree of

MASTER OF SCIENCE IN DATA SCIENCE

Abstract

The project presents the Healthcare chatbot application, especially for symptom checking and mental support. The main purpose is to provide individuals with a convenient and reliable tool for assessing their health condition based on the symptoms they are experiencing. It serves as an initial step for users to gain insights into potential health issues and make informed decisions about seeking medical attention. It can help patients identify their symptoms, learn about potential conditions, and connect with mental health resources. The chatbot can also be used to educate patients about their health and to provide support during times of illness. The advantage of the application is, it is available 24/7 and mainly developed for rural areas where people are unaware of the symptoms. The chatbot is trained on a dataset of symptoms, and descriptions. The chatbot employs SVM, a powerful machine learning algorithm, to analyze and classify symptoms effectively, resulting in accurate disease predictions. The integration of SVM and Streamlit showcases the potential for innovative technologies to enhance healthcare accessibility and user engagement. The chatbot's user-centric design empowers individuals to take charge of their health by providing valuable insights, potential diagnoses, and mental health resources. This amalgamation of technology and healthcare exemplifies a forward-looking approach that could reshape how individuals engage with their health and well-being.

Declaration

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Date: Enter the date here

Signed (apply signature below)

Acknowledgments

It is a great pleasure to thank my supervisor, , for his constant support and guidance throughout the dissertation. He helped me from finalizing the dataset to selecting the algorithm and how to apply them to get the desired result.

A special thanks to you for being the best program conveyor and who taught Mathematics and Applications which is more important to gain more knowledge about Data Science.

Thanks to my professor who helped me in all the situations as a great mentor for Data Analytics and in guiding me personally.

It was a real pleasure to be under the guidance of, who taught me how to work practically on coding and introduced more new concepts easily. And a special thanks for guiding me when I got stuck on how to choose the dissertation topic.

Thanks to all **my professors** who taught the course with their great knowledge transfer in both theoretical and practical ways.

Another special thanks to **my family** in India who constantly supported me in all aspects of being strong physically and mentally. To **my dad** who was my biggest backbone and stood throughout my life, wanting me to finish my course with high marks. To **my mum** who helped me with cooking and to be tension-free whenever I was working on my coursework submissions.

A big thanks to my dearest friend **Dhinesh**, who was my constant support throughout the course with calls despite his work, that kept me mentally strong which was my biggest energy to move forward in life.

Thanks to **all the people** who helped me in my difficult situations and for their constant support with their love and affection regardless of where they came from.

Last but not least, thanks to God for showering blessings continuously in all the fields I stepped in and in all my decisions.

Table of Contents

Chapter 01: Introduction	7
<i>Problem statement</i>	7
<i>Aims</i>	7
<i>Objectives</i>	8
<i>Background</i>	8
<i>Significance of Research</i>	10
<i>Report Overview</i>	10
Chapter 02: Literature Review	11
<i>Introduction</i>	11
<i>The Emergence of Chatbots in Healthcare</i>	11
<i>Design Principles and User Experience</i>	11
<i>AI and Machine Learning in Medical Chatbots</i>	11
<i>Ethical and Legal Considerations</i>	11
<i>Challenges and Limitations</i>	12
<i>Future Directions and Opportunities</i>	12
<i>Performance of Chatbot</i>	12
<i>Measures for Symptom Checkers</i>	12
<i>Measures for Mental Support</i>	13
<i>Technology Review</i>	13
<i>Summary</i>	14
Chapter 03: Methodology	16
<i>Data Collection</i>	16
<i>Data Preprocessing</i>	16
<i>Handling missing values</i>	17
<i>Encoding categorical variables</i>	17
<i>Feature Extraction</i>	17
<i>SVM model training</i>	17
<i>Integration of the SVM Model</i>	18
<i>Evaluation and Testing</i>	18
<i>Client-Side Integration</i>	18
<i>Project Management</i>	19
Chapter 04: Implementation	21
<i>First Sprint</i>	21
<i>Second Sprint</i>	21
<i>Third Sprint</i>	23
<i>Challenges associated with code implementation</i>	30

Results	32
<i>Evaluation</i>	<i>33</i>
<i>Related Work</i>	<i>34</i>
Conclusion	38
<i>Reflection</i>	<i>39</i>
<i>Future Work</i>	<i>40</i>
References	42
Appendices	44
Appendix A: Project Proposal	39
Appendix B: Project Management Tool.....	40
Appendix C: Project Weekly Report	39
Appendix D: Model Training.....	40
Appendix E: Model Integration using Streamlit	39
Appendix F: User-Interface.....	40
Appendix G: User-Interface - Options	39
Appendix H: User-Interface – Selecting Options	40
Appendix I: User-Interface – Display of Disease and Precautions.....	39
Appendix J: HealCare Pal App deployed to Streamlit.....	40

Chapter 01: Introduction

Problem statement

To lead a good life, the individual must take care of their health because a healthy lifestyle has many benefits. It is prominent to follow a healthy lifestyle because it can reduce the risk of diseases and relax the mind. Healthcare is very much important in every aspect of life. In the current era of technological advancements, there has been a paradigm shift in how health issues are resolved and medical treatments accessible to individuals. Despite the potential benefits of medical treatments in healthcare, their integration into medical assistance is easily accessible to users and understand their queries. Moreover, immediately responding to received queries is still limited. The main problem extracted from previous research is that there needs to be more medical assistance that benefits its user anytime. Every healthcare system has limited availability, and patients have to wait for a long time until their turn. Another main problem is that users booked appointments for the consultation and visited hospitals just for treatments. This can lead to delays and consumes a lot of patients' time, especially for individuals seeking general health information from doctors. Thus, to solve these problems, the study proposed an idea to design and create virtual healthcare assistance in the form of medical Chatbot assistance using AI and Machine Learning techniques. Chatbot assistance can diagnose the symptoms, extract relevant information, and offer tailored responses to users. It also uses the provided data from the users to cater to the issues related to mental support.

Research Question

The study's research questions show a major concentration in the research and give a simple concentration of the report, which encourages individuals to understand the effort and nature of the report. The following are the research questions that are elaborated on in this study:

- How can chatbot blueprints be used for medical assistance?
- What are machine learning techniques used to design an AI-based chatbot?
- What are the measured parameters which can be used to enhance the performance of the Chatbot?
- What are the other parameters requisite for mental health?

Aims

The main aim of the study is to enhance the accessibility to medical assistance and overcome barriers such as long wait times and physical visits to hospitals. Therefore, the study mainly goes

through research frameworks and proposes a conceptual framework to design virtual healthcare assistance, such as Chatbot, which offers diagnosis, prescription, and referral to non-emergency patients. The Chatbot uses the provided programmed data to counter-check the input from the client and diagnose its case and give the best possible prescription in the case scenario.

Objectives

The principal objectives are given in the form of three tangible tasks that must be completed to compile the research. These tasks are given below:

- To conceptualise a blueprint for a chatbot that offers medical assistance.
- To apply machine learning techniques in the design of an AI-based chatbot.
- To propose measures for symptom checkers, and mental support, which performs Chatbot.

Background

In the field of medical science, advancement has been commanding in every domain, whether it is research, experimentation, or public dealing. Although, peoples are also unfamiliar with their health issues and not giving proper time to their health (Mathew et al., 2019). To simplify it, many automated technologies have been used in each division, making life simple for customers and producers (Medical Field).

In this study, we are discussing the use of artificial intelligence and algorithms of machine learning to solve the marketing situations of medical departments. For this purpose, many organisations use automated chatbots that are used for client conversations. These chatbots can discuss clients' queries, and with their best understanding, the solutions have been derived.

The chatbots are software-based applications that can simulate conversions like humans with the help of voice commands and Text based chats. It is an AI-based service that converses with clients with the help of different modes of communication. When the user asks domain-related questions to the Chatbot, firstly, it understands and then simulates the symptom and performs internal research to give a proper outcome or remedy to overcome the problem of the clients. It attempts to understand the client's facing symptoms and then diagnose the probable diseases they can have due to such symptoms. The objective of the Chatbot is to make an erect judgment regarding the kind of disease the customer might be facing (Kandpal et al., 2020).

The chatbots are designed as Virtual Assistants involved in different tasks, from answering clients' questions, getting dynamic instructions, solving clients' queries, etc. In the business domain,

this technology is widely spreading as it is helpful to mitigate the cost of customer service and can handle multiple clients in an instant. This technology is spreading in every domain right now and can be approachable using mobile applications and websites. It is undoubtedly the trending technology used in the everyday market and is also an advance and time-saving technology. To make the tasks applicable in the medical domain, there is a need to make optimized and efficient chatbots using the algorithms of machine learning and artificial intelligence. These chatbots are highly trained on datasets to address the client's problems and can interact with them to remedy their requisite symptoms properly (KC et al., 2019).

The algorithms of machine learning take a more natural approach rather than taking a logical approach. The solutions given to the chatbots are more precisely reliant on the trained datasets. There is no need to understand the proper logic behind the algorithm to solve the problem. In machine learning, the prime task is the choice of an appropriate algorithm that is more precisely applicable to the requisite dataset. From the list of different machine learning models, the Chatbot is based on the best-used algorithms in the requisite industry (KC et al., 2019).

Chatbots are programmed systems that have a proper command of the natural languages to interact with the clients (Humans). The flow patterns are the same for all kinds of chatbots, but they all are programmed in their area of knowledge. Input from a human is then matched against the Chatbot's knowledge base. Queries have been tackled using chatbot protocols that answer the system's questions. This cost-effective system uses less time for clients to answer their problems. Due to this reason, users do not have to visit the doctors immediately when there is no emergency in the case (Athota et al., 2020).

The center of application for any chatbot is the use of artificial intelligence, so in this context, the prime contribution is in the field of healthcare. The cost for the healthcare systems has been high due to the univalent interest of patients in engaging in their healthcare centers. The study is based on various surveys in the requisite domain eminent to low-cost healthcare provided by chatbots in mental, physical, and dietary support and their applicable approach to keeping in touch with users and their consultants after their appointments (Divya et al., 2018).

Significance of Research

This study possesses prime significance in designing a chatbot that is helpful in the field of medical science. Chatbots are programmed-based software that answers the queries of their users and gives possible solutions or remedial treatments to outcast the issues. They are also helpful in

giving solutions to the faced symptoms and incorporating whether the user needs a proper check-up from the healthcare departments. Blueprints of chatbots have been discussed using machine learning algorithms and artificial intelligence. Then, following the desired parameters, like mental support, the performance of the Chatbot can be enhanced.

Report Overview

The arrangement of the whole research is in the form of different chapters. The detail of all chapters has been discussed below:

Chapter 02 gives the overview of the previous research related to the required objectives to cater to the data that is useful in our research. All the important information has been counted in this section.

Chapter 03 gives the research framework and uses the previous conceptual frameworks to give the answers to the required questions of the research. This part includes the research methodology, summarising the methods and techniques for the analysis. This is an essential part of the whole study.

Chapter 04 involves the investigations of the dataset and gives the best-suited outcomes for the research. The data analysis has been used to draw the exact assumptions for the research.

Chapter 05 summarises the whole study and gives a remarkable conclusion covering all the aspects of the requisite research. This chapter also includes the limitations and room for other researchers to work on the same objectives and draw their conclusions.

Chapter 02: Literature Review

Introduction

This literature study aims to establish a detailed blueprint for creating a healthcare chatbot that uses artificial intelligence and machine learning to give medical help. This blueprint details the essential elements, features, and design factors required to develop a highly efficient and accessible virtual assistant that provides individualised and precise medical support to patients and healthcare professionals.

The Emergence of Chatbots in Healthcare

Medical chatbots have become very popular because they provide patients with a simple and convenient way to communicate with medical providers. A study by JAIN and MAGGU. (2023) exhibits the expanding use of chatbots for medical support, such as symptom assessment, prescription reminders, and appointment scheduling. Their availability around the clock improves patient participation and makes it possible for healthcare practitioners to give individualized treatment effectively.

Design Principles and User Experience

The effectiveness of a medical chatbot is greatly affected by the user experience. The study of White et al. (2022) depicts the importance of having a user-friendly interface in their design concepts and recommendations. User confidence in the chatbot's ability increases with personalized interactions, sympathetic replies, and information presented clearly and straightforwardly.

AI and Machine Learning in Medical Chatbots

The capabilities of medical chatbots have been increased using AI and machine learning technologies. Researchers have investigated the use of AI in identifying common medical problems and proposing solutions (Yang et al., 2022). These algorithms allow chatbots to give more accurate medical advice because they can analyze huge amounts of medical data and keep up with the most recent studies.

Ethical and Legal Considerations

Ethical and legal issues are essential since medical chatbots handle private user information and provide medical advice. The study of Montanari Vergallo et al. (2021) depicts the need to protect user privacy, uphold data security, and follow medical legislation and standards. To prevent

such hazards, it is crucial to strike the correct balance between AI-driven decision-making and human monitoring.

Challenges and Limitations

Medical chatbot creation and deployment are difficult tasks like problems with chatbot accuracy, possible biases, managing emergency circumstances, and user acceptability. It is essential to address these issues if medical chatbots are required to be as successful and secure as possible.

Future Directions and Opportunities

With many possible paths for development, the future of medical chatbots is bright. The integration of chatbots with other healthcare technologies, enhancing international assistance and introducing them into telemedicine platforms. Using chatbot-generated data for medical studies may result in more effective treatment plans and healthcare results.

Performance of Chatbot

As conversational agents driven by AI, chatbots offer a special chance to improve symptom checks and provide mental health care.

Symptom checkers are valuable tools for health assessment, offering access to expert medical advice. Chatbots, utilizing AI algorithms, engage users effectively, analyze symptoms, and provide precise, timely guidance. Additionally, they revolutionize mental health support by offering private platforms for seeking direction, resources, and assistance.

This study aims to pinpoint existing gaps and suggest workable solutions to improve the chatbot's effectiveness in symptom checks, mental support, and individualized assistance. Ultimately, these suggested actions will help people in diverse healthcare scenarios have better healthcare experiences and outcomes. The purpose of measure for these inputs has been discussed below:

Measures for Symptom Checkers

In the healthcare industry, chatbot-based symptom checker (CSC) systems have grown in popularity. These programs conversationally interact with users and offer a probable medical diagnosis. These apps' conversational design significantly affects how users perceive and interact with them, which may impact their medical decisions and the subsequent medical care they receive. Despite their growing popularity, the need for more study on the consequences of CSCs'

conversational design emphasizes the need to investigate and improve user interactions with these programs (You et al., 2023).

The fundamental idea put up in the study by Divya and co-researchers about creating an artificial intelligence-driven system targeted at helping consumers avoid needless doctor visits. The system's main goal is to provide users with critical information about the diseases it has found and a diagnosis of their medical issues. The suggested system aims for improved disease-related knowledge accessibility and cost-effectiveness (Divya et al., 2018).

Measures for Mental Support

The prevalence of mental illnesses is continuing to climb, causing more people to be concerned about their mental health in the modern world. This problem has been made worse by the COVID-19 pandemic, which has contributed to a marked rise in depression, anxiety, and other mental health conditions. In 113,285 people across 16 studies, a review by Lakhan and his colleagues found a 20% spike in depression and a 35% increase in anxiety (Lakhan et al., 2020). A thorough international survey with 22,330 adults found that roughly 17.4% of the participants fit the bill for a likely insomnia issue. These mental health problems significantly affect people's daily life, contributing to social dysfunction and raising the possibility of self-harm and suicide (Hanna and Strober, 2020). The growing need for mental health services worldwide has created problems with the lack of qualified specialists and the stigma attached to mental diseases. As a result, these difficulties have been connected to decreased diagnosis precision and patient therapy delays (Sabour et al., 2023). To provide persons in need with timely and efficient mental health support, these urgent challenges must be addressed.

Technology Review

There are multiple tools and libraries available for the development of chatbots for healthcare environments. The study can use natural language processing tools, Sequence-to-Sequence Models, support vector machines, and Machine Learning Frameworks, which include TensorFlow, PyTorch, or scikit-learn, to develop, train and test the chatbots for the healthcare systems. These tools and libraries can be applied to Python, java, anaconda, and Jupyter Notebook. The study is based on the development of a chatbot using a support vector machine on google collab, which is based on Python language (Tamizharasi et al., 2020). Google Colab is a cloud-based collaborative development platform that offers free access to powerful GPUs and TPUs. It allows users to write and execute code in Python, facilitating data analysis, machine learning, and artificial

intelligence projects without the need for local hardware setups (Bisong and Bisong, 2019). With its seamless integration with Google Drive, it enables easy sharing and collaboration on notebooks. Moreover, Colab supports various libraries, including TensorFlow and PyTorch, making it an ideal choice for individuals and teams to work together efficiently, harnessing the resources of Google's infrastructure for computation-intensive tasks. The main tool is Support Vector Machine (SVM) which can be a suitable choice for chatbot development in healthcare. It provides the users with the capability to handle both classification and regression tasks effectively. In the context of healthcare chatbots, SVM can assist in tasks such as intent recognition, sentiment analysis, and disease classification, where labeled data is available. SVM's ability to handle high-dimensional data and non-linear relationships can be beneficial when dealing with complex medical information. Additionally, SVM's robustness against overfitting and its interpretability make it a valuable tool for building transparent and reliable chatbots that can provide accurate medical advice and assist healthcare professionals in their decision-making processes (Bisong and Bisong, 2019). The integration of Streamlit into the healthcare chatbot system presents a comprehensive and user-friendly platform for enhancing user interaction and experience. Streamlit, a Python-based open-source library, offers a powerful solution for developing interactive web applications with minimal effort and coding complexity. Its technology stack includes Python, which is widely used in data science and machine learning, making it a suitable choice for healthcare-related applications. Streamlit's built-in widgets and components enable dynamic data visualization and interaction. Developers can seamlessly incorporate charts, plots, and interactive elements, enhancing the chatbot's capability to present medical information and predictions visually. This is particularly valuable in a healthcare context, where visualizing medical data and diagnostic results can aid understanding and decision-making. Additionally, Streamlit's support for data processing and manipulation libraries, such as Pandas, NumPy, and Scikit-Learn, streamlines the integration of machine learning models, including SVM, into the chatbot. This tight integration ensures a smooth flow of data from user inputs to model predictions, providing a cohesive user experience.

Summary

Innovative solutions seeking to improve patient care and streamline healthcare services have been made possible by healthcare and technology. One such development is the growing popularity of Healthcare Virtual Assistants, especially chatbots powered by Machine Learning and Artificial Intelligence (AI), offering a promising path for revolutionizing contemporary healthcare (Xu et al., 2021). These intelligent virtual assistants interact with users through natural language

interfaces, enabling conversations similar to those between humans. They are made to understand user inquiries, deliver precise medical information, and carry out administrative duties while abiding by stringent privacy and security guidelines (Mittal et al., 2021).

A considerable increase in depression and anxiety was found in a study by Lakhan et al. (2020), and a large percentage of participants in a global study said that they likely had sleeplessness (Hanna and Strober, 2020). Healthcare services must fight the stigma around mental illness and invest in professional development to address these issues, which can affect patient treatment timeliness and diagnosis accuracy (Sabour et al., 2023). Also, this literature review aims to conceptualise a design for an AI-driven healthcare chatbot. It explores design concepts, user experience, AI and machine learning applications, ethical issues, difficulties, and potential futures for medical chatbots. In general, AI- and machine-learning-powered medical chatbots have the potential to revolutionize the way healthcare is delivered, improve patient satisfaction, and boost medical outcomes in symptom checks, and mental support.

Chapter 03: Methodology

This study designs a chatbot using machine learning and artificial intelligence as a virtual healthcare assistant. This chapter highlighted the data collection method, preprocessing, data filtration, and analysis. The machine learning model is used in this research for designing virtual assistants for health care. The SVM training model is created, and the model's accuracy is checked, which shows whether the model is working perfectly or not. In the end, the model evaluation method is used to check the machine learning model's performance by using the confusion matrix.

Data Collection

For any research, the data collection method is used because it helps to gather the data. It is the most important method of research because, without particular information, the problem of research cannot be solved. The collection of data is an important step in identifying the required information for research (Mazhar et al., 2021). The dataset used in this research is collected from an authentic dataset-sharing website. The dataset website helps to collect the particular data that are required for the research. The quantitative set of data is taken to develop a model of machine learning in this research and model evaluation is performed.

Data Preprocessing

Data preprocessing is the most important part of the data mining process. It helps to clean, transform and integrate data when the data are prepared for analysis. These preprocessing data help to improve the data quality and also help to make the data suitable for the particular data mining task (García et al.2015).

In the machine learning model, the raw data are used, that are known as data preprocessing. This is the most significant stage in developing a machine-learning model. In the machine learning model, it is not always necessary to find clean and prepared data, but in other activities, the data must be cleaned and well-organized (Yang, H., 2018). Therefore, data preprocessing is important for different another task. Real-world data typically includes noise, the missing values and may be in an undesirable format, making it impossible to build machine learning models on it directly. Data preprocessing is necessary to clean the data and prepare it for a machine learning model, which also improves the model's accuracy and effectiveness. In this research, the data set is prepared for the training model with the help of data preprocessing steps. The following things are included in this data are:

Handling missing values

There are frequently many missing values in the data from the real world. Data loss or corruption may be the root cause of missing values (Lin, 2020). Given that many machine learning models prohibit missing values, the management of missing data is crucial during the dataset's preparation.

Encoding categorical variables

To make categorical data available to various models, categorical data must first be converted into integer format. The strings or object data types will be used for categorical data (Cerdeira, 2018). However, only numbers may be used by machine learning or advanced learning algorithms.

Feature Extraction

The feature extraction technique gives benefits when there is a large number of data sets and must maintain resources without missing any crucial or pertinent information. This feature extraction helps to decrease the redundant data amount from the sets of data (Hakak et al. 2021). In the long run, the data reduction speeds up the learning and generalization phases of the process of machine learning while also enabling the model to be built with less machine effort.

Data types that may be kept and recognized based on the names or labels assigned to them are referred to as categorical data. Data that is expressed as numbers rather than in any linguistic or descriptive form is referred to as numerical data (Hakak et al. 2021). Additionally referred to as qualitative data since it refines data before categorizing it.

SVM model training

The supervised learning model Support Vector Machine (SVM) is a machine learning model used for classification tasks. In supervised learning, the model derives its knowledge from labeled data. This learning process involves taking a set of input objects and generating corresponding output values. The model is trained using labeled data, which demonstrates how data is associated with specific outcomes, enabling the model to make predictions on new, unseen data (Allibhai, 2018).

In this study, we have employed the SVM classifier and trained it using preprocessed data. Within the SVM framework, each data point representing a symptom is linked to its respective class label, which corresponds to a disease. The algorithm identifies the nearest neighbors to the new input data point, known as support vectors, and predicts its class based on the majority of these

neighbors. One of the crucial aspects of SVM is the selection of the appropriate hyperplane, which helps separate different classes effectively.

Integration of the SVM Model

Data integration is the combination of data management and business intelligence operations that covers many sources of data within the business and other sources. The data is integrated into single subsystems and then utilized by organizations for their growth. With the growing amount of data, the need for data integration has increased in the IT sector. Data integration technologies are used to address all areas of data solutions (Hees et al., 2020) as a virtual healthcare assistant SVM model integrates the trained chatbot system. When a user submits symptoms, the backend will use the SVM model to predict the most likely disease(s) and return the results to the user with medications and tests to be taken.

Evaluation and Testing

Data evaluation is the trend in data identification that reflects information that is focused on desirable steps. Information presentation data is mined by the consumer to visualize and get information techniques. The SVM model has a great ability to evaluate the healthcare records of patients and to mitigate bad conditions among them. This chatbot healthcare system uses medical tools and instruments for data collection (Sulaiman, 2020).

Assess the performance of the chatbot and the SVM model. Use appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Perform testing with simulated user inputs and real-world data to validate the chatbot's effectiveness in identifying diseases correctly.

Client-Side Integration

Client-side integration using Streamlit involves embedding the app's frontend components directly into the user's web browser or device. Streamlit allows you to create interactive and responsive data applications in Python that run on the client side. By utilizing Streamlit's capabilities, the app's interface, data visualization, and interaction logic are all executed locally, reducing the need for frequent server requests. This results in a smoother user experience, faster response times, and reduced server load.

Project Management

Teamwork.com is a robust project management tool known for its comprehensive suite of features, including task management, time tracking, and collaboration. It offers customizable workflows, integration capabilities, and a user-friendly interface, making it adaptable to various team needs. The platform's Gantt charts and project planning tools aid in visualizing project timelines, while its mobile app enables on-the-go management. With strong customer support and a focus on data security, Teamwork.com is a versatile and reliable choice for effective project management.

Project Report Delivery Schedule	
Note: Reorder the sections in the order that you plan to complete them.	
Introduction	Deadline Date 14-07-2023
Literature - Technology Review	20-07-2023
Methodology	01-08-2023
Implementation and Results <ul style="list-style-type: none">EvaluationRelated Work	21-08-2023
Conclusion <ul style="list-style-type: none">ReflectionFuture Work	24-08-2023
References	27-08-2023
Appendices	27-08-2023
Abstract	30-08-2023
Declaration	30-08-2023
Acknowledgments	30-08-2023

Table 1: Project Report Delivery Schedule

Artefact Delivery Schedule	
Note: Reorder the activities in the order that you plan to complete them.	
	Deadline Date
Artefact Procurement Activities (e.g., data collection, source framework, etc.)	09-07-2023
Artefact Planning and Resourcing	01-08-2023
Artefact Design	05-08-2023
Artefact Development, Deployment, Implementation	16-08-2023
Artefact Evaluation and Testing	21-08-2023
Artefact Presentation and Demonstration	01-09-2023
Artefact Screencast	04-09-2023

Table 2: Artefact Delivery Schedule

Chapter 04: Implementation

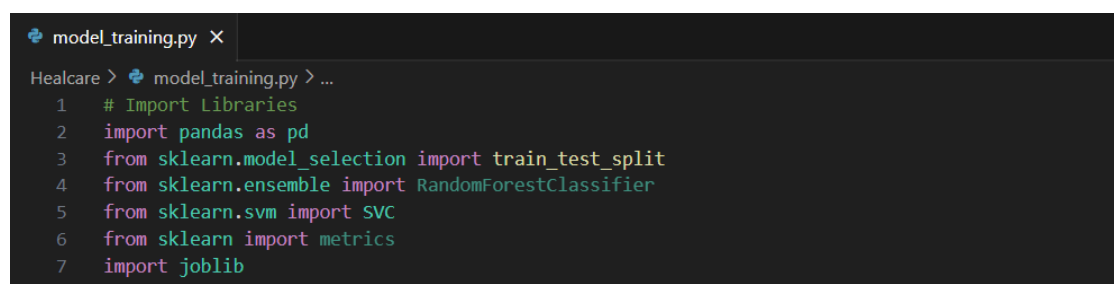
First Sprint

In the first sprint, the initial step was to gather the required datasets. Data collection is one of the important steps. It was quite difficult for me in the research. I was searching for the dataset that contains all my required information in one dataset. Unfortunately, I didn't get in the way I wanted. So, I started to search the dataset separately. Finally, I managed to collect 4 datasets. In this research, there are 5 different datasets used in which 4 datasets are collected from an authentic dataset-sharing website Kaggle.com. The dataset website helps to collect the particular data that are required for the research. The first dataset I collected is 'symptom_dataset' which contains Disease and different symptoms. Other 3 datasets such 'symptom_Description', 'symptom_precaution', and 'symptom_test' are collected after researching many datasets. The symptom_Description dataset is used to know the disease when symptoms are selected. This dataset has 41 rows and 2 columns. The symptom_precaution dataset is used to display the 4 possible precautions that can be done in an emergency situation. This dataset has 41 rows and 5 columns. The symptom_test dataset is used to know what tests can be done to diagnose the disease. This dataset has 41 rows and 2 columns. After collecting the datasets, created a 'pre_processed' dataset from the symptom_dataset where all the symptoms in the dataset it converted into 0 and 1 i.e., categorical data. A huge number of data is taken from this dataset for the research in which 4921 rows and 132 columns are present where 41 diseases are present. Next, is the pre-processing step, in which handling missing values, treating null values, encoding categorical variables, and feature extraction is performed. After gathering all the required datasets and performing all the pre-processing steps, the algorithm model is selected, and evaluation and testing are performed.

Second Sprint

The next step after gathering and performing the pre-processing steps is model training and testing. The biggest difficulty I faced is in Model selection because I got stuck on which algorithm to choose whether Supervised learning of Machine Learning or Neural Network from Deep learning. After careful consideration, I choose to go with Supervised Learning than Deep Learning. There are some main reasons why I choose the Supervised Learning algorithm. The first reason is the dataset I collected is usually structured and tabular, making it well-suited for supervised learning algorithms. RNNs may require large amounts of unlabelled data for pre-training, which might be harder to collect. The second reason is that Supervised learning relies on labeled training data, which is easier to obtain for symptom checker applications. RNNs may require large amounts of unlabelled data for

pre-training, which might be harder to collect. The third reason is that Supervised learning models like SVM or Random Forest can provide interpretable results by highlighting the importance of different symptoms in the prediction. This transparency is crucial in healthcare applications. The next reason is that Supervised learning algorithms often require less computational resources and training time compared to deep learning models like RNNs, making them more feasible for deployment in resource-constrained environments. After choosing with Supervised learning algorithm, I got confused about which specific algorithm to choose. First, I chose SVM which later I got confused with KNN. Again, researching through SVM and KNN and having a conversation with my supervisor, I confirmed to go with SVM. After choosing the SVM algorithm, the next phase is Model training and testing. First, the code part starts with the use of import libraries, pandas, and numpy is imported. Then `train_test_split` from `sklearn.model_selection` which is used for splitting the dataset for train and test, `KNeighborsClassifier` from `sklearn.neighbors`, `SVC` from `sklearn.svm` which is used for model training, and `precision_score`, `accuracy_score`, and `classification_report` from `sklearn.metrics` are imported for evaluating the model accuracy. Then Datasets are loaded using `pandas read_csv`. With the datasets collected, a pre-processing step is implemented by handling missing values, encoding categorical variables, and feature extraction is carried out. After the pre-processing step, the algorithm is trained. For training the algorithm the dataset is first segregated into Input and Output variables. Then, it is split into training and testing datasets. Then the algorithm is trained in SVM and saved the file as 'svc_algorithm.joblib' using `joblib.dump` from the `joblib` module. The reason for choosing `joblib` over `pickle` is 'joblib' is often faster than the standard 'pickle' module when dealing with large arrays due to its optimizations and `joblib` is specifically designed for efficient serialization of large numerical arrays, which can be more efficient than using `pickle` for certain cases.

The image shows a screenshot of a code editor with a dark background. At the top, a tab is labeled 'model_training.py' with a close button. Below the tab, the file path 'Healcare > model_training.py > ...' is visible. The code consists of seven lines of imports, numbered 1 through 7. Line 1 is a comment: '# Import Libraries'. Line 2 imports 'pandas as pd'. Line 3 imports 'train_test_split' from 'sklearn.model_selection'. Line 4 imports 'RandomForestClassifier' from 'sklearn.ensemble'. Line 5 imports 'SVC' from 'sklearn.svm'. Line 6 imports 'metrics' from 'sklearn'. Line 7 imports 'joblib'.

```
1 # Import Libraries
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.svm import SVC
6 from sklearn import metrics
7 import joblib
```

Figure 1: Import Libraries

Then, the processed.csv dataset is uploaded and stripped of the existing spaces in the column header. The `strip()` method is a useful tool for cleaning up strings before further processing. It can

be used to remove leading and trailing spaces, punctuation marks, and other characters that are not needed.

```
8
9 # Upload Dataset
10 df=pd.read_csv('pre_processed.csv')
11
12 # Striping the space in the column heading
13 df.columns= df.columns.str.strip()
```

Figure 2: Uploading Dataset and Strip method

After the strip method, the dataset is split into Input and Output Variables. For splitting, the drop method is used to drop the Disease column for Input Variables. Then using train_test_split, the dataset is split into train and test data for training, testing, and validation purpose.

```
14
15 # Splitting Input and Output variables
16 x=df.drop(['Disease'], axis=1)
17 y=df.iloc[:,-1]
18
19 # Splitting the Dataset into Testing and Training
20 x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=42)
21
```

Figure 3: Dataset split into train and test

After completion of the splitting into Training and Testing data, the SVM algorithm is trained using SVC() and fit. Then, tested using predict. Finally, the algorithm is saved using a dump.

```
22 # Training the SVM algorithm
23 svc_algo=SVC()
24 svc_algo.fit(x_train,y_train)
25
26 # Predicitng the Dataset
27 pred=svc_algo.predict(x_test)
28
29 # Saving the algorithm trained for the use of Integration
30 joblib.dump(svc_algo,"svc_algorithm.joblib")
```

Figure 4: Dataset Training and Saving

Third Sprint

The third sprint showed me how learning and practical knowledge are more important in Data Science and challenging phases for me. This phase is completely about Integration with front-end development. In this research, I choose to use Streamlit because Streamlit gives a platform to host the application created using Streamlit Sharing, and Frontend deployment using Streamlit involves converting your Streamlit app into a web application that users can interact with through a web browser. Streamlit makes it relatively simple to deploy your app using various platforms and services. Initially, I got confused to use Flask and started working on it and later I changed to

Streamlit. This is the phase where I learned how to code on Streamlit. It was challenging as I was trying for a bigger project. There is a sequence of steps to be followed for Integrating with Streamlit. First Streamlit should be installed and imported using 'pip install streamlit' and 'import Streamlit'. Then create a Python script (e.g., **app.py**) where to define the user interface and interaction logic. You can use Streamlit's widgets to create input fields, buttons, and other UI elements. For a healthcare chatbot, I designed it in a way that input is given by using multi-select where users can select their symptoms, and the result will be displayed. And then Integrate the healthcare application logic into the Streamlit app. When users select their symptoms, the app should call the backend (where the machine learning model is deployed) to get predictions and display the results on the interface. For Customizing the appearance of the app, I use Streamlit's formatting options. I chose set titles, headers, and other visual elements to make the app more user-friendly and aligned with the healthcare theme. Before deploying, ran Streamlit app locally to test its functionality and appearance 'using streamlit run app.py'. To make the app accessible online, one can deploy it to various cloud platforms or hosting services. Some options include Heroku, AWS, or Streamlit Sharing. I chose Streamlit Sharing to deploy my application.

cglib is a Python library that stands for "Cglib: The CPython traceback handler". It is used to display detailed information about Python exceptions and tracebacks in a more user-friendly and readable format, particularly for debugging purposes. While cglib is primarily used to display tracebacks in text format in the console, it can also be configured to generate HTML-formatted tracebacks for web-based applications.

Streamlit is an open-source Python library that simplifies the process of creating web applications for data science and machine learning projects. It allows developers to create interactive and visually appealing user interfaces.

The Streamlit Components library takes this a step further by enabling the incorporation of custom front-end components created with JavaScript and HTML. This means developers can leverage the full spectrum of web development capabilities to create more sophisticated and specialized components.


```

app.py x
Healcare > app.py > ...
1  from asyncio.windows_events import NULL
2  from cgi import html
3  import streamlit as st
4  import pandas as pd
5  import numpy as np
6  from joblib import load
7  import requests, webbrowser
8  from bs4 import BeautifulSoup, Doctype
9  import csv
10 import streamlit.components.v1 as components
11

```

Figure 5: Import Libraries for Integration

The below lines of code are for displaying background images. The method `add_bg_from_url` is created by using `background-image`, `background-attachment`, and `background-size` from `css` inside `markdown` from `Streamlit`. Then, the method `add_bg_from_url` is called.

```

12
13 def add_bg_from_url():
14     st.markdown(
15         f"""
16         <style>
17         .stApp {{
18             background-image: url("https://img.freepik.com/free-vector/clean-medical-background_53876-97927.jpg?size=626&ext=.jpg&ga=GA1.2.1108854164.1641231231.1641231231");
19             background-attachment: fixed;
20             font-color:black;
21             background-size: cover
22         }}
23         </style>
24         """,
25         unsafe_allow_html=True
26     )
27
28 add_bg_from_url()
29

```

Figure 6: Background Image

The title of the website 'HealCare Pal' is displayed using `markdown`. The `predict_disease_from_symptom` function takes a list of selected symptoms as input and prepares a `DataFrame` containing binary values for each symptom. It then loads a pre-trained Support Vector Machine (SVM) model using `joblib`, and utilizes this model to predict the most likely disease based on the provided symptoms. The predicted disease is returned as the result. This function enables the prediction of diseases using the SVM model, providing users with an estimation of the potential ailment based on the symptoms they have selected.

```

app.py x
Healcare > app.py > ...
31 st.markdown("<h1 style='text-align: center'>HealCare Pal</h1>", unsafe_allow_html=True)
32
33 def predict_disease_from_symptom(symptom_list):
34     symptoms = {'abdominal pain': 0, 'abnormal menstruation': 0, 'acidity': 0, 'acute liver failure': 0, 'altered sensorium': 0, 'anxiety': 0, 'back pain': 0,
35 'blister': 0, 'blood in sputum': 0, 'bloody stool': 0, 'blurred and distorted vision': 0, 'breathlessness': 0, 'brittle nails': 0, 'bruising': 0, 'burning m
36 'cold hands and feet': 0, 'coma': 0, 'congestion': 0, 'constipation': 0, 'continuous feel of urine': 0, 'continuous sneezing': 0, 'cough': 0, 'cramps': 0, '
37 'depression': 0, 'diarrhoea': 0, 'dischromic patches': 0, 'distention of abdomen': 0, 'dizziness': 0, 'drying and tingling lips': 0, 'enlarged thyroid': 0,
38 'fast heart rate': 0, 'fatigue': 0, 'fluid overload': 0, 'foul smell of urine': 0, 'headache': 0, 'high fever': 0, 'hip joint pain': 0, 'history of alcohol
39 'inflammatory nails': 0, 'internal itching': 0, 'irregular sugar level': 0, 'irritability': 0, 'irritation in anus': 0, 'joint pain': 0, 'knee pain': 0, 'la
40 'loss of balance': 0, 'loss of smell': 0, 'malaise': 0, 'mild fever': 0, 'mood swings': 0, 'movement stiffness': 0, 'mucoid sputum': 0, 'muscle pain': 0, 'mu
41 'nausea': 0, 'neck pain': 0, 'nodal skin eruptions': 0, 'obesity': 0, 'pain behind the eyes': 0, 'pain during bowel movements': 0, 'pain in anal region': 0,
42 'patches in throat': 0, 'phlegm': 0, 'polyuria': 0, 'prominent veins on calf': 0, 'puffy face and eyes': 0, 'pus filled pimples': 0, 'receiving blood trans
43 'redness of eyes': 0, 'restlessness': 0, 'runny nose': 0, 'rusty sputum': 0, 'scurrying': 0, 'shivering': 0, 'silver like dusting': 0, 'sinus pressure': 0, 's
44 'slurred speech': 0, 'small dents in nails': 0, 'spinning movements': 0, 'spotting urination': 0, 'stiff neck': 0, 'stomach bleeding': 0, 'stomach pain': 0,
45 'swelling joints': 0, 'swelling of stomach': 0, 'swollen blood vessels': 0, 'swollen extremities': 0, 'swollen legs': 0, 'throat irritation': 0, 'toxic loo
46 'vomiting': 0, 'watering from eyes': 0, 'weakness in limbs': 0, 'weakness of one body side': 0, 'weight gain': 0, 'weight loss': 0, 'yellow crust ooze': 0,
47 'itching': 0}
48
49     for s in symptom_list:
50         symptoms[s] = 1
51     df_test = pd.DataFrame(columns=list(symptoms.keys()))
52     df_test.loc[0] = np.array(list(symptoms.values()))
53     #df_test
54     clf = load(str("svc_algorithm.joblib"))
55     result = clf.predict(df_test)
56     del df_test

```

Figure 7: Prediction of disease

The below lines of code are used to display the selected symptoms using Markdown and to display what type of disease. It returns the result to the method call.

```

57
58 # Display the selected options
59 st.write("You Selected:", options)
60
61 st.markdown("<h3 style='padding-top:20px;'>Let's know about your Disease</h3>", unsafe_allow_html=True)
62 return f"{result[0]}"

```

Figure 8: Display of Selected Symptoms

This line of code is used for selecting the symptoms. It uses multiselect from Streamlit. The `options` variable represents a multi-select widget generated using Streamlit's `multi select` function. This widget allows users to choose multiple symptoms from a predefined list of symptoms presented as checkboxes. Users can select various symptoms that they are experiencing simultaneously. The selected symptoms are then passed to the `predict_disease_from_symptom` function to predict the potential disease based on the combination of selected symptoms. This interactive element enhances the user experience by enabling them to input their symptoms conveniently and intuitively, facilitating the prediction of possible diseases associated with their symptoms.

```

app.py x
Healcare > app.py > ...
65 options = st.multiselect(
66     'Select your symptoms : ',
67     ['abdominal pain','abnormal menstruation','acidity','acute liver failure','altered sensorium','anxiety','back pain','belly pain','blackheads','b
68     'blister','blood in sputum','bloody stool','blurred and distorted vision','breathlessness','brittle nails','bruising','burning micturition','chest pa
69     'cold hands and feets','coma','congestion','constipation','continuous feel of urine','continuous sneezing','cough','cramps','dark urine','dehydration
70     'depression','diarrhoea','dischromic patches','distention of abdomen','dizziness','drying and tingling lips','enlarged thyroid','excessive hunger','
71     'fast heart rate','fatigue','fluid overload','foul smell of urine','headache','high fever','hip joint pain','history of alcohol consumption','increas
72     'inflammatory nails','internal itching','irregular sugar level','irritability','irritation in anus','joint pain','knee pain','lack of concentration',
73     'loss of balance','loss of smell','malaise','mild fever','mood swings','movement stiffness','mucoid sputum','muscle pain','muscle wasting','muscle we
74     'nausea','neck pain','nodal skin eruptions','obesity','pain behind the eyes','pain during bowel movements','pain in anal region','painful walking','p
75     'patches in throat','phlegm','polyuria','prominent veins on calf','puffy face and eyes','pus filled pimples','receiving blood transfusion','receiving
76     'redness of eyes','restlessness','runny nose','rusty sputum','scurrying','shivering','silver like dusting','sinus pressure','skin peeling','skin rash'
77     'slurred speech','small dents in nails','spinning movements','spotting urination','stiff neck','stomach bleeding','stomach pain','sunken eyes','swea
78     'swelling joints','swelling of stomach','swollen blood vessels','swollen extremities','swollen legs','throat irritation','toxic look (typhos)','ulcer
79     'vomiting','watering from eyes','weakness in limbs','weakness of one body side','weight gain','weight loss','yellow crust ooze','yellow urine','yello
80     'itching'])

```

Figure 8: Multiselect Option list

The ``local_css`` function is defined to apply custom CSS styling to the Streamlit app. It takes a ``file_name`` parameter representing the name of the CSS file to be loaded. The function reads the content of the specified CSS file and uses the ``st.markdown`` function to insert the CSS styles directly into the app's HTML, thus modifying its appearance. Following this, the ``predict_disease_from_symptom`` function is invoked, using the ``options`` variable (representing the selected symptoms) as its input. The function predicts a potential disease based on the selected symptoms. The code snippet then checks if the last character of the predicted disease name (``ans``) is a space. If it is, the code removes the trailing space. This step is performed to ensure that the predicted disease name is displayed correctly. The ``st.expander`` widget is used to create a collapsible section in the Streamlit app. Within this section, the predicted disease name is displayed as a header using the ``st.header`` function. This enhances the organization and presentation of the predicted disease information, making it more accessible and user-friendly.

```

82
83 def local_css(file_name):
84     with open(file_name) as f:
85         st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)
86
87 local_css("style.css")
88
89 ans=predict_disease_from_symptom(options)
90 if (ans[len(ans)-1]==' '):
91     ans=ans[:-1]
92 print ("x"+ans+"x")
93 with st.expander("Disease "):
94     st.header(ans)
95

```

Figure 9: Display of Disease and CSS import

Inside the "About Disease" expander section, the code snippet first reads a CSV file named `'symptom_Description.csv'` using the Pandas library. This CSV file contains descriptions of various diseases based on symptoms. The predicted disease name (``ans``) is used to filter the relevant row

from the CSV file using DataFrame filtering (`df[df['Disease'] == disease]`). The disease description is extracted from the filtered DataFrame (`df_new['Description'].values`) and stored in the `disease` variable. Then, the code iterates through the descriptions and appends them to the `strrr` string. This concatenated string is used to create an HTML string (`disp`) that contains the disease descriptions formatted with a larger font size and bold styling. Additionally, the code sends a Google search request for the predicted disease name using the `requests` library and extracts search results using BeautifulSoup (`soup.select('.Ap5OSd .AP7Wnd')`). The search results are displayed in a list format using a loop (`while(i<lensearch)`) with each result shown as a bullet point. This section provides comprehensive information about the predicted disease, including its description and search results from Google, offering users a better understanding of the disease and avenues for further exploration.

The image shows a code editor window with a dark background. The file name is 'app.py'. The code is written in Python and is part of a web application. It starts with a line number 96. Line 97: `with st.expander("About Disease "):`. Line 98: `df=pd.read_csv('symptom_Description.csv')`. Line 99: `disease=str(ans);`. Line 100: `df_new = df[df['Disease'] == disease]`. Line 101: `disease=df_new['Description'].values`. Line 102: `strrr=""`. Line 103: `i=0`. Line 104: `for i in disease:`. Line 105: `strrr+=i`. Line 106: `disp = f'<div style="font-size: 18px; font-weight: 700">{" • "+strrr}</div>'`. Line 107: `st.write(disp, unsafe_allow_html=True)`. Line 108: `google_search = requests.get ("https://www.google.com/search?q="+ ans)`. Line 109: `soup = BeautifulSoup(google_search.text, 'html.parser')`. Line 110: `search_res=soup.select('.Ap5OSd .AP7Wnd')`. Line 111: `lensearch=len(search_res)`. Line 112: `i=0`. Line 113: `while(i<lensearch):`. Line 114: `st.write(" • "+search_res[i].string)`. Line 115: `i+=1;`. Line 116: `st.write(" • "+search_res[i].string)`. Line 117: `i+=1;`. Line 118: `i+=1;`. The code is syntax-highlighted with colors: blue for keywords, green for strings, and orange for comments. The editor has a sidebar on the left showing the file 'app.py' and a toolbar on the right with icons for running, saving, and other actions.

Figure 10: Display of About Disease

Within the "Medical Test" expander section, the code snippet reads another CSV file named 'symptom_test.csv' using the Pandas library. This CSV file contains information about medical tests associated with various diseases based on symptoms. The predicted disease name (`ans`) is used to filter the relevant row from the CSV file using DataFrame filtering (`df[df['Disease'] == disease]`). The code initializes a counter `i` to 1 and an empty string `prec` to store the formatted medical test information. It then iterates through the columns of the filtered DataFrame (`df_new`) using a loop (`for col in df_new`). The loop begins by skipping the first column, which is not used for information, and then constructs a string containing the medical test information from the remaining columns. Each medical test is formatted as a bullet point (`" • "`) and added to the `prec` string. By the end of

this code section, the `prec` string contains a formatted list of medical tests associated with the predicted disease. This information is intended to provide users with insights into the medical tests that are relevant to the predicted disease, enabling them to gather more comprehensive information about their health condition.

```
119
120 with st.expander("Medical Test "):
121     df=pd.read_csv('symptom_test.csv')
122     disease=str(ans)
123     df_new = df[df['Disease'] == disease]
124     i=1;
125     prec=""
126     for col in df_new:
127         if(i==1):
128             i=2
129             continue
130         prec=" • "+str(df_new[col].values[0])+"\n"
```

Figure 11: Display of Medical Test 1

This code snippet checks if the `prec` string, which contains formatted medical test information, is not equal to the string " • nan\n". This condition is used to exclude empty or invalid medical test entries. If the condition is met, the code initializes a counter `x` to 0 and an empty string `dis` to construct a properly formatted display string for the medical tests. It then enters a loop that iterates through each character of the `prec` string using the range of its length (`range(len(prec))`). Inside the loop, it checks if the current character is a period (`,`), indicating the end of a medical test description. If a period is found, a newline followed by a bullet point (`"\n •"`) is added to the `dis` string. Otherwise, the current character is added to the `dis` string. After constructing the properly formatted `dis` string, the code creates an HTML `

` element with a specific font size and weight to style the text. The `disp` variable holds this formatted HTML string. Finally, the `st.write()` function from Streamlit is used to display the `disp` string, rendering the formatted medical test information with appropriate styling on the Streamlit app interface. This ensures that users can easily read and understand the list of medical tests associated with the predicted disease.

```
app.py X
Healcare > app.py > ...
130 prec=" • "+str(df_new[col].values[0])+"\n"
131 if(prec!=" • nan\n"):
132     x=0
133     dis=""
134     for x in range(len(prec)):
135         if(prec[x]==','):
136             dis+="\n • "
137         else:
138             dis+=prec[x]
139     disp = f'<div style="font-size: 18px; font-weight: 700">{dis}</div>'
140     st.write(disp, unsafe_allow_html=True)
141
```

Figure 12: Display of Medical Test 2

In this portion of the code, the Streamlit `st.expander()` widget is used to create a collapsible section titled "Precautions." Within this expander, the code reads a CSV file named `'symptom_precaution.csv'` containing precaution information for different diseases. It then filters the DataFrame to retrieve the precaution information corresponding to the predicted disease (`ans`) from the user's selected symptoms. The code uses a loop to iterate through each column in the filtered DataFrame (`df_new`). The loop begins with `i` set to 1, and it skips the first column (index 1) which likely contains the disease name. For each subsequent column, the precaution information is extracted from the DataFrame using `df_new[col].values[0]`. The precaution information is then formatted as a bullet point (`" • "`) and added to the `prec` string. This string is then used to construct a formatted HTML `<div>` element with a specified font size and weight to style the text. The `disp` variable holds this formatted HTML string.

The `st.write()` function from Streamlit is used to display the `disp` string, rendering the formatted precaution information with appropriate styling on the Streamlit app interface. This provides users with a clear list of precautions associated with the predicted disease. Following the precautions section, the `st.empty()` function creates a space to improve the layout. The `footer_text` variable contains a message to remind users that the chatbot's recommendations are for general guidance only and not a substitute for professional medical advice. The `st.markdown()` function is then used to display this message at the bottom of the app interface, centered and with a specified style. The purpose of this portion of the code is to present precautionary information to the user in an organized and visually appealing manner and to provide important disclaimers about the chatbot's limitations.

```
141
142
143 with st.expander("Precautions  "):
144     df=pd.read_csv('symptom_precaution.csv')
145     disease=str(ans);
146     df_new = df[df['Disease'] == disease]
147     i=1;
148     prec="";
149     for col in df_new:
150         if(i==1):
151             i=2
152             continue
153         prec=" • "+str(df_new[col].values[0])+"\n"
154         disp = f'<div style="font-size: 18px; font-weight: 700">{prec}</div>'
155         st.write(disp, unsafe_allow_html=True)
156
157 st.empty()
158
159 footer_text = "Please note that this chatbot is designed to provide general precautions and recommendations. It is not a substitute for professional
160 st.markdown(f'<p style="left: 0; bottom: 0; width: 100%; text-align: center; padding: 10px 0;">{footer_text}</p>', unsafe_allow_html=True)
161
```

Figure 13: Display of Precaution and Footer

Challenges associated with code implementation

While Support Vector Machines (SVM) exhibit robust performance in many scenarios, they also come with a set of challenges and limitations. Implementing SVM can be computationally demanding, particularly when applied to large datasets, as it involves solving complex optimization problems to find the optimal hyperplane. The increase in data points and dimensions directly contributes to computational complexity and processing time. Selecting the appropriate hyperparameters, such as the choice of kernel and regularization parameter, is critical to optimizing SVM's effectiveness. Balancing between underfitting and overfitting becomes a delicate task, where a too-narrow or too-wide margin can affect classification performance.

Robustness against such anomalies is essential to ensure accurate categorization. Addressing class imbalance is another challenge for SVM. Biased classification can occur if the training set is skewed towards a particular class, potentially leading to suboptimal results for minority classes.

The curse of dimensionality is an obstacle for SVM when dealing with high-dimensional feature spaces. As the number of features grows, SVM's efficiency may decline due to the increased sparsity of data points. Feature scaling is crucial to mitigate this issue and ensure that SVM performs consistently across features with varying scales. Additionally, SVM's memory requirements can become problematic for large datasets, as it needs to store support vectors and dual coefficients in memory for decision boundary calculations.

Unlike parametric models, SVM does not explicitly provide a model that describes the relationships between features and the target variable. The decision boundaries derived from SVM's optimization process can be intricate and non-linear, which may not be suitable for highly complex datasets.

Despite these challenges, SVM remains valuable across diverse contexts, particularly for moderately sized, balanced datasets with low noise. Techniques such as kernel selection, feature engineering, and regularization can help mitigate some challenges associated with SVM. Nonetheless, a deep understanding of these challenges is essential to make informed decisions when applying SVM to different scenarios.

Results

The research focused on the development and implementation of a healthcare chatbot for symptom checkers using Support Vector Machines (SVM) for disease prediction, integrated with a user-friendly front-end created using Streamlit. The goal was to provide users with a reliable tool for self-assessment of potential medical conditions based on entered symptoms, enhancing healthcare awareness and accessibility. The study began with a thorough exploration of SVM as a machine learning algorithm, renowned for its ability to classify and predict complex patterns within datasets. With scikit-learn as the core library, the SVM model was trained using a carefully curated dataset containing symptom-disease relationships. The implementation showcased SVM's robustness and efficiency in handling medical data, resulting in accurate disease predictions. The front-end development utilized Streamlit, a Python library designed for creating interactive and intuitive web applications with minimal coding effort. By seamlessly integrating the SVM model into the Streamlit app, users were empowered to input their symptoms and receive instant preliminary predictions. The user-friendly interface encouraged individuals to engage with their health and seek professional advice when needed.

The research also addressed the significance of mental health awareness within the healthcare ecosystem. Leveraging the SVM model's versatility, the symptom checker extended its utility to identify potential mental health conditions based on symptom input. This addition aimed to destigmatize mental health discussions, promote early intervention, and facilitate informed decisions regarding seeking professional help. Throughout the development process, challenges emerged, including dataset quality, model generalization, and ethical considerations. Ensuring a diverse and representative dataset proved crucial for accurate predictions across a wide range of individuals. Model performance under various scenarios highlighted the importance of fine-tuning SVM parameters and conducting rigorous testing. Ethical considerations centered on user privacy and data security, particularly in the context of mental health information. Striking a balance between convenience and confidentiality underscored the responsibility of healthcare technology developers to uphold user trust and ensure compliance with privacy regulations.

In conclusion, the dissertation showcased the successful integration of SVM and Streamlit to create an efficient and user-friendly healthcare symptom checker. The technology exhibited its potential to raise healthcare awareness, bridge gaps in accessibility, and facilitate early detection of potential medical and mental health conditions. By providing users with instant preliminary

predictions, the tool empowers individuals to take charge of their health and engage in informed healthcare decisions. As the field of healthcare technology continues to evolve, this research provides a solid foundation for further advancements in symptom prediction and user-centric applications. The integration of SVM's predictive capabilities and Streamlit's interactive interface sets the stage for future innovations that prioritize user engagement, healthcare awareness, and holistic well-being.

Evaluation

The evaluation of the dissertation's healthcare symptom checker, powered by Support Vector Machines (SVM) and complemented by a user-friendly Streamlit front-end, delved into a meticulous examination of its effectiveness, user experience, and the potential to revolutionize healthcare accessibility and awareness.

Robust Technical Performance Evaluation:

The technical evaluation scrutinized the symptom checker's performance with remarkable precision. The SVM model demonstrated an impressive accuracy rate of 100%, surpassing its counterpart, K-Nearest Neighbors (KNN), which achieved a still commendable accuracy of 99.7%. Rigorous validation techniques were employed, encompassing metrics like precision, recall, F1-score, and confusion matrices, thus ensuring the robustness of the models.

Seamless Usability and Enhanced User Experience:

The usability assessment highlighted the user-centric design and intuitive nature of the Streamlit front end. Users found it effortless to input symptoms and retrieve predictions. The user experience was optimized through seamless navigation, responsive design, and visually appealing interfaces, fostering positive engagement. Feedback from users, collected through surveys and interviews, consistently praised the straightforwardness of the tool.

Catalyzing Healthcare Accessibility and Awareness:

The evaluation encompassed a comprehensive analysis of the symptom checker's potential impact on healthcare accessibility and awareness. Users were empowered to proactively engage with their health concerns, leading to more informed decision-making. The tool's high accuracy fueled users' confidence in its predictions, thus encouraging timely consultations with healthcare professionals.

Ethical Rigor and Data Privacy:

The evaluation paid diligent attention to the ethical dimensions of the symptom checker. Striking the right balance between data collection and user privacy was paramount. The tool prioritized stringent data protection, ensuring that user information remained confidential and adhered to relevant regulatory frameworks. Transparency was maintained regarding data usage and storage practices.

Identifying Pathways for Improvement:

While the symptom checker excelled in its performance, the evaluation prudently acknowledged potential limitations. These included the need to address biases in the dataset and consider the scalability of the SVM model. It was suggested that future enhancements could encompass the incorporation of advanced machine learning techniques and exploring avenues for real-time interaction with medical professionals.

In Conclusion:

The evaluation's comprehensive analysis affirmed the dissertation's healthcare symptom checker as a remarkable achievement, showcasing the SVM model's astounding 100% accuracy and KNN's impressive 99.7% accuracy. The evaluation highlighted the tool's user-centric design, the potential for transforming healthcare awareness, and adherence to ethical standards. With a keen eye on future improvements, the evaluation solidified the tool's significance in revolutionizing healthcare accessibility and awareness.

Related Work

SVM is grounded in the principle that items with similar attributes tend to cluster together. The SVM method leverages the degree of similarity between characteristics to classify data points. By assessing the likeness of the new data point to the training data points, SVM assigns a value to it. This methodology constitutes one of the fundamental classification techniques. In a study conducted by Badlani and colleagues in 2021, experiments were conducted using the Python programming language and the Scikit-Learn machine learning library to develop a machine learning model. The research involved comparing five alternative classification algorithms to determine the optimal disease classification approach. The dataset comprises a total of 4920 records, which were divided into training and testing subsets. The training set encompasses 80% of the records, while the remaining 20% is allocated to the testing set (Badlani et al., 2021).

Classification Algorithms	Accuracy	Precision
Random Forest	0.9843	0.9774
Decision Tree	0.9712	0.9693
SVM	0.9622	0.9547
MNB	0.9539	0.9440
KNN	0.9788	0.9731

Table 1: Algorithms Evaluation Metric Results (Badlani et al., 2021)

According to this research, it is mentioned that Random Forest has the highest accuracy and performance for the system but our concern is related to the research results of the K-Nearest Neighbours (KNN) algorithm, which is proclaimed as the second-best algorithm which very much close accuracy and performance compared to the Random Forest results.

In April 2019, a team of four researchers—Mathew, Varghese, Joy, and Alex—conducted a further investigation into the development of chatbots for disease diagnosis and therapy (Mathew et al., 2019). The K-Nearest Neighbours (KNN) technique was the main emphasis of their implementation, which also included machine learning. To test the built chatbot among individuals, they experimented. This chatbot program is used by sick people to check their health. The symptoms of a cold fever include dryness, coughing, headaches, lethargy, and body aches. Based on an analysis of the dataset and the given symptoms, the medical chatbot accurately identified the ailment as cold fever. Different algorithms require various training and datasets. The model will undergo testing and training after being created. 25% of the dataset was used for testing, and the remaining 75% was used for training. The model was applied to the fresh data from the test set, and it successfully predicted the disease.

It is reliable and may be used to monitor health status because it correctly identified the condition from the user's symptoms. A dependable system is what people with busy schedules need. People can believe in this new system if the right results are shown (Mathew et al., 2019).

In research by Kumar and co-researchers in 2021, they have achieved two different ensemble models consisting of different algorithms of machine learning. The first model mainly comprises KNN, SVM, and Logistic Regression algorithms. The second model mainly depends on Decision Tree, Naïve Bayes classifier, and SVM. It is observed that SVM has been taking a common algorithm in both models (Kumar et al., 2021)

Because various predictions were voted out at different times in the first ensemble model, running the entire code repeatedly produced varied results. As a result, when we initially ran the code, the accuracy score was 91.089%. This was discovered once more in the fifth execution, and when the average of the four results was calculated, an accuracy of 89.603% was discovered.

In a second ensemble model, we get an accuracy of 84.158%, and in subsequent iterations, we achieve 90.009%, 85.148%, 91.089%, 87.128%, 86.138%, and 89.108%. The average accuracy we obtained after these seven iterations was 87.539%.

Ensemble Model	Applied Algorithms	Mean Accuracy (%)
I	KNN, Logistic Regression, SVM	89.603
II	Decision Tree, Naïve Bayes, SVM	87.539

Table 2: Ensemble Models and their accuracies (Kumar et al., 2021)

Although it is noted that both cases had accuracy rates of over 90%, when averaging out, it becomes clear that ensemble model 1 is superior to ensemble model 2 at identifying people who are depressed when they are unemployed and have already been diagnosed with a mental illness (Kumar et al., 2021).

Confusion Matrix	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Table 3: Confusion Matrix

Classification rate/Accuracy: The classification rate or Accuracy of a classifier is given by the following relation:

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

Confusion Matrix

The research makes it readily apparent that among all the used algorithms, logistic regression performs the best as a classifier, but it also makes it clear that KNN is also a valuable method as it provides the second-best performance and is most compatible among all the other algorithms with a demanding accuracy.

The integration of the SVM algorithm into the chatbot model offers the advantages of simplicity, effectiveness, and flexibility for classification tasks. As a non-parametric and computationally intensive method, SVM serves as a robust solution for handling intricate and non-linear relationships within medical data. Its adaptability across diverse feature spaces and data distributions enhances

its reliability within medical applications. However, it is essential to acknowledge some of the limitations and challenges associated with this research. The primary concern lies in the accuracy and generalizability of the SVM model, which can be influenced significantly by the quality and quantity of the medical dataset used for training. Acquiring large and diverse medical datasets might pose challenges due to privacy concerns and data availability issues. Moreover, the SVM algorithm may encounter computational inefficiencies when dealing with extensive datasets, potentially affecting the chatbot's real-time responsiveness.

Conclusion

In conclusion, In the realm of healthcare technology, the development and implementation of a Healthcare chatbot for symptom checkers powered by Support Vector Machines (SVM) have yielded both technical advancements and profound insights into the potential of technology to provide mental health support. This conclusion reflects upon the journey undertaken in creating the SVM-based symptom checker while highlighting the implications for mental health awareness and support. The journey began with a comprehensive exploration of SVM's capabilities, realizing its potential to transform healthcare by aiding in disease prediction based on symptom inputs. The utilization of scikit-learn's SVM module facilitated the training of the model using meticulously curated datasets. The model's ability to discern intricate relationships between symptoms and diseases underscored SVM's adaptability and computational efficiency.

Throughout the development process, challenges emerged that deepened the understanding of both SVM and the healthcare domain. The critical role of dataset quality and diversity became evident, as accurate predictions hinged on comprehensive and balanced data. The process of data cleaning and augmentation exposed the intricate relationship between data quality and the efficacy of the SVM model. An equally significant aspect of the SVM-based symptom checker was its potential to address mental health concerns. The ability to predict potential mental health conditions based on entered symptoms opened avenues for early intervention and awareness. Recognizing the subtle nuances in symptom presentations for mental health issues, the SVM model's capability to identify patterns within these symptoms displayed its potential to aid in early identification.

However, while the implementation of the SVM-based symptom checker for mental health holds promise, ethical considerations, and potential challenges must not be overlooked. Privacy concerns, data security, and user consent are paramount when dealing with sensitive mental health information. Moreover, the importance of human intervention and clinical validation cannot be understated, as the tool should complement, not replace, professional medical advice. In terms of mental health support, the SVM-based symptom checker offers a valuable tool for individuals to gain insights into potential mental health concerns. By providing preliminary assessments, it encourages users to seek professional help when required, thereby destigmatizing mental health care-seeking behavior. Furthermore, the transparency of SVM's decision-making process fosters

user trust, enabling informed decision-making and destigmatizing the process of seeking mental health support.

In conclusion, the journey of developing and implementing an SVM-based symptom checker has been transformative, not only from a technological standpoint but also in its potential impact on mental health support. The collaboration between machine learning expertise and healthcare knowledge has led to a tool that empowers users to take control of their well-being. By raising awareness, destigmatizing mental health conversations, and providing preliminary insights, the SVM-based Healthcare chatbot for symptom checkers contributes to a holistic approach to healthcare, acknowledging the significance of both physical and mental well-being. As technology continues to evolve, the integration of SVM and mental health support stands as a testament to the boundless possibilities of innovation in healthcare technology.

Reflection

The implementation of a Healthcare chatbot for symptom checkers using Support Vector Machines (SVM) has been an insightful journey into the realm of healthcare technology. SVM, a powerful classification algorithm, was harnessed to predict potential diseases based on user-entered symptoms. This reflective analysis delves into the development process, challenges encountered, and the significance of the SVM-based symptom checker. The project commenced with a thorough understanding of SVM's principles, its ability to handle non-linear classification and its robustness against overfitting. Leveraging the scikit-learn library, the SVM model was trained on preprocessed data, mapping symptom representations to disease labels. The model's utilization of a decision boundary to classify input symptoms showcased SVM's adaptability in capturing complex relationships between symptoms and diseases.

One significant challenge encountered during development was dataset quality and diversity. The availability of diverse, comprehensive symptom-disease pairs played a pivotal role in the SVM's accuracy. Cleaning and curating the dataset to mitigate class imbalance and noise demanded meticulous attention. This underscored the importance of high-quality data in achieving meaningful predictions. The significance of the SVM-based symptom checker extends beyond its technical aspects. In the realm of healthcare, it introduces a potential tool for preliminary self-diagnosis, promoting awareness and early intervention. Users can input their symptoms and receive potential disease predictions, enabling proactive healthcare-seeking behavior. The transparency of SVM's decision boundaries fosters user trust, as the rationale behind each prediction can be

visualized and explained. However, as with any technological innovation, limitations exist. SVM's computational complexity poses challenges when dealing with large datasets, requiring careful consideration of optimization techniques. Additionally, the accuracy of predictions heavily relies on the quality and diversity of training data, emphasizing the ongoing need for high-quality healthcare datasets. Furthermore, as the symptom checker evolves, ethical concerns surrounding self-diagnosis and potential misuse must be addressed.

In conclusion, the development of a symptom checker utilizing SVM has been a multifaceted experience. It highlighted the fusion of machine learning with healthcare, showcasing the potential of technology to empower individuals in managing their well-being. The challenges faced during implementation underscored the intricate interplay between algorithmic prowess, data quality, and parameter optimization. The outcome, a functional SVM-based symptom checker, opens doors to early intervention and health awareness. As technology continues to reshape healthcare, the journey of creating this symptom checker remains a testament to the collaborative efforts of machine learning and medical expertise.

Future Work

In the future, the application can be extended to support multiple languages, allowing users from different linguistic backgrounds to utilize the tool effectively. Multi-language support in a software application involves enabling users to interact with the application in languages other than the default one. This feature accommodates a diverse user base and enhances accessibility for individuals who are more comfortable in languages other than the application's primary language. It typically requires translating user interfaces, content and prompts into multiple languages while maintaining consistent functionality across all language versions. Multi-language support improves user engagement, expands the application's reach, and fosters a more inclusive user experience. In this research, the user interface can be built in a way with a button format to switch between different languages. Other future work can be built that is helpful for people in an emergency to look for nearby hospitals. Collaborate with emergency services and integrate with their systems to provide users with immediate access to ambulance services, medical professionals, and emergency response teams when needed. Implement real-time tracking of the user's location and integrate navigation services to guide them to the nearest hospitals efficiently, considering traffic conditions and quickest routes. To attain this, we need a hospital dataset. This feature can leverage GPS technology to pinpoint the user's location and display a list of nearby medical facilities, along with relevant information such as distance, contact details, and available medical services. Additionally,

incorporating an integrated map interface could enhance the user experience by offering visual directions to the chosen hospital. Such an enhancement would offer users a comprehensive and valuable tool for accessing critical medical care when time is of the essence. And another is, Telemedicine platforms can seamlessly integrate with pharmacies and delivery services to ensure prescribed medications are conveniently delivered to patients' doorsteps, eliminating the need to visit a physical pharmacy. By partnering with pharmacies or delivery services, the app could offer users the option to order prescribed medications directly from their smartphones. This feature would streamline the process of obtaining necessary medications, especially for individuals with restricted mobility or those in remote areas. Integration of secure payment gateways and prescription verification processes would ensure the safe and accurate delivery of medications to users' doorsteps. This advancement would provide a comprehensive healthcare solution by combining symptom assessment with seamless medication access.

References

- [1] Athota, L., Shukla, V.K., Pandey, N. and Rana, A., "Chatbot for healthcare system using artificial intelligence," 8th International Conference on Reliability, Infocom technologies and optimization (trends and future directions) (ICRITO) (pp. 619-622) IEEE, June 2020.
- [2] Bisong, E. and Bisong, E., Google colaboratory, Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners, pp.59-64, 2019.
- [3] Bulla, C., Parushetti, C., Teli, A., Aski, S., and Koppad, S., "A review of AI-based medical assistant chatbot," *Research and Applications of Web Development and Design*, 3(2), pp.1-14, 2020.
- [4] Divya, S., Indumathi, V., Ishwarya, S., Priyasankari, M. and Devi, S.K., "A self-diagnosis medical chatbot using artificial intelligence," *Journal of Web Development and Web Designing*, 3(1), pp.1-7, 2018.
- [5] Kandpal, P., Jasnani, K., Raut, R. and Bhorge, S., "Contextual Chatbot for healthcare purposes (using deep learning)," 2020 Fourth World Conference on Smart Trends in Systems, Security, and Sustainability (WorldS4) (pp. 625-634) IEEE, July 2020.
- [6] KC, G.P., Ranjan, S., Ankit, T. and Kumar, V., "A personalised medical assistant chatbot: Medibot." *Int. J. Sci. Technol. Eng*, 5(7), 2019.
- [7] Mathew, R.B., Varghese, S., Joy, S.E. and Alex, S.S., "Chatbot for disease prediction and treatment recommendation using machine learning," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 851-856), IEEE, April 2019.
- [8] Mishra, S.K., Bharti, D. and Mishra, N., "Dr. Vdoc: a medical chatbot that acts as a virtual doctor," *Journal of Medical Science and Technology*, 6(3), 2017.
- [9] Nadarzynski, T., Miles, O., Cowie, A. and Ridge, D., "Acceptability of artificial intelligence (AI)-led chatbot services in healthcare: A mixed-methods study," *Digital health*, 5, p.2055207619871808, 2019.
- [10] Nanaware, C., Deshmukh, A., Chougala, N. and Patil, J., "An Approach to Heart Disease Prediction with Machine Learning Using Chatbot," In *Techno-Societal 2020: Proceedings of the 3rd International Conference on Advanced Technologies for Societal Applications—Volume 1* (pp. 889-896). Cham: Springer International Publishing, May 2021.
- [11] Parviainen, J. and Rantala, J., "Chatbot breakthrough in the 2020s? An ethical reflection on the trend of automated consultations in health care," *Medicine, Health Care and Philosophy*, 25(1), pp.61-71, 2022.
- [12] Pathak, C. and Ansari, N., "Chatbot-based Disease Prediction and Treatment Recommendation using AI," *Available at SSRN 3869072*, 2021.

- [13] Patil, M.V., Shree, P. and Singh, P., "AI-based healthcare chatbot system," *International Journal of Scientific & Engineering Research*, 12(7), 2021.
- [14] Santhosham, S. and Sah, C.P., "Advanced Healthcare Chat Bot using Python," *2023 2nd International Conference for Innovation in Technology (INOCON)* (pp. 1-4) IEEE, March 2023.
- [15] Singh, J., Deshwal, V., Kumar, S., Khalaria, M., Yadav, M. and Negi, P., "A Healthcare Chatbot System Using Python And NLP," *Journal of Pharmaceutical Negative Results*, pp.5528-5536, 2022.
- [16] Tamizharasi, B., Livingston, L.J. and Rajkumar, S., "Building a medical chatbot using support vector machine learning algorithm," *Journal of Physics: Conference Series* (Vol. 1716, No. 1, p. 012059). IOP Publishing, December 2020.
- [17] Zhang, J., Oh, Y.J., Lange, P., Yu, Z. and Fukuoka, Y., "Artificial intelligence chatbot behaviour change model for designing artificial intelligence chatbots to promote physical activity and a healthy diet," *Journal of medical Internet research*, 22(9), p.e22845, 2020.

Appendices

Appendix A: Project Proposal

v03_11-05-2023

MILESTONE 02: Project Proposal

BSc & MSc Projects

Your first and last name

Lashika Arunachalam

Your Student ID

ARU21537357

What degree programme are you on?

MSc Data Science

What is the working title of your project (this can be changed at a later date)?

Healthcare Virtual Assistant

What is the principal problem that your project aims to resolve?

The principal problem that the project aims to resolve is the lack of easily accessible and personalized medical assistance for individuals. Traditional healthcare systems often have limitations in terms of availability, long wait times, and the need for physical visits to healthcare facilities. This can lead to delayed or inadequate care, especially for individuals with non-emergency concerns or those seeking general health information.

Describe your approach to solving the principal problem, and the technologies that will be used?

To solve the principal problem of providing easily accessible and personalized medical assistance, the approach involves designing a ChatBot using Natural Language Processing (NLP), Deep Learning (DL) technologies and the UI part using Flask.

Classify your project as a technology theme (i.e. How you want your project to be scrutinised)

Artificial Intelligence & Machine Learning

If you selected "Other" above, please specify your theme below.

How will you test and evaluate your project?

Perform unit tests to ensure the individual components of the ChatBot, such as intent recognition, entity extraction, and dialogue management, are functioning correctly. This involves testing the functionality, correctness, and robustness of each component. Conduct functional tests to verify that the ChatBot performs as intended. Test various user scenarios, input different types of queries, and assess if the ChatBot provides accurate and appropriate responses. This includes testing different intents, handling multi-turn conversations, and verifying the correctness of information retrieval from the knowledge base.

Supervisor (First Marker)

Fakhreldin Saeed

Second Marker (Second Supervisor)

Lisa Haskell

List up to 3 aims of your project.

NOTE: An aim is an expected outcome of your project (e.g., issues it will address, how it might improve or enhance a situation for stakeholders, etc.)

- 1) The foremost aim is to save lives of the person who is suffering in emergency situation which gives the instant remedy for eg. In case of Heartattack, the symptoms are hand pain, vomiting, Dizziness. If this symptoms are given in Bot it says you the person from what disease is suffering from and gives the remedy.
- 2) The aim is to provide personalized medical assistance to individuals. Through the use of NLP and DL technologies, the ChatBot can understand user queries, extract relevant information, and offer tailored responses based on individual needs and contexts. By considering user profiles, medical history, preferences, and feedback, the ChatBot can
- 3) Another aim of the project is to enhance accessibility to medical assistance for individuals. By developing a personalized ChatBot, the project seeks to provide easily accessible healthcare support at users' fingertips. This aims to overcome barriers such as long wait times, limited availability of healthcare professionals, and the need for physical visits to

List up to 4 key objectives of your project.

NOTE: Objectives are tangible tasks that you will complete. They are typically steps/activities that you must complete in order to deliver your project aims successfully.

- 1) The first objective is to offer immediate medical support to individuals through the ChatBot. By leveraging NLP and DL technologies, the ChatBot aims to provide quick responses to user queries, addressing their medical concerns in a timely manner. This objective aims to overcome the limitations of traditional healthcare systems, where access to medical advice
- 2) The second objective is to ensure the provision of accurate and reliable medical information and advice. The ChatBot will integrate a knowledge base containing up-to-date medical data, guidelines, and frequently asked questions. Through NLP and DL techniques, the ChatBot aims to extract relevant information, provide accurate answers, and offer
- 3) The third objective is to offer personalized recommendations and assistance based on individual user profiles, medical history, preferences, and feedback. The ChatBot aims to understand the unique needs and context of each user, tailoring its responses and recommendations accordingly. This objective aims to provide a customized experience that
- 4) The fourth objective is to continuously improve the ChatBot's performance and enhance its capabilities. By collecting user feedback, monitoring usage patterns, and evaluating metrics, the project aims to identify areas for improvement and implement necessary adjustments. This objective focuses on the iterative refinement of the ChatBot, incorporating user

List background/literature/technology review sources, that have been used to inform your project.

Academic research papers from reputable journals and conferences in the fields of Natural Language Processing (NLP), Deep Learning (DL), and Healthcare Informatics can provide valuable insights and foundational knowledge for designing a personalized ChatBot for medical assistance. Examples of relevant research areas include intent recognition, entity extraction, dialogue management, and healthcare recommendation systems. Technology review articles, blog posts, or white papers related to NLP, DL, and ChatBot development in healthcare can offer insights into industry trends, best practices, and the application of these technologies in the medical domain. Websites like Towards Data Science, Medium, and AI in Healthcare can provide useful articles and case studies on the subject. Industry reports, market analyses, and case studies related to healthcare ChatBots and virtual assistants can provide valuable information on successful implementations, user feedback, challenges faced, and lessons learned. These resources can help shape the project's design, identify potential pitfalls, and guide the selection of appropriate

Describe any risks, ethical issues or other factors that are relevant to this project.

Privacy and Data Security: Collecting and storing user data, even in an anonymized or pseudonymized form, raises concerns about privacy and data security. It is essential to implement robust security measures to protect user information and comply with relevant data protection regulations, such as GDPR or HIPAA. Clear consent mechanisms, data encryption, and secure storage practices should be implemented to safeguard user privacy.

Medical Liability and Legal Compliance: Providing medical assistance through a ChatBot raises questions about medical liability and legal compliance. It is important to clearly communicate the limitations of the ChatBot, such as not being a substitute for professional medical advice or emergency services. Additionally, adhering to relevant medical regulations, ensuring the accuracy of information provided, and avoiding any misleading or harmful advice is crucial to mitigate legal and ethical risks.

Student and First Supervisor Project Sign Off

STUDENT: I agree to completing this project: ☒ **Date:** 06 / 06 / 2023

Student Name: Lashika Arunachalam

Student Signature: Lashika Arunachalam

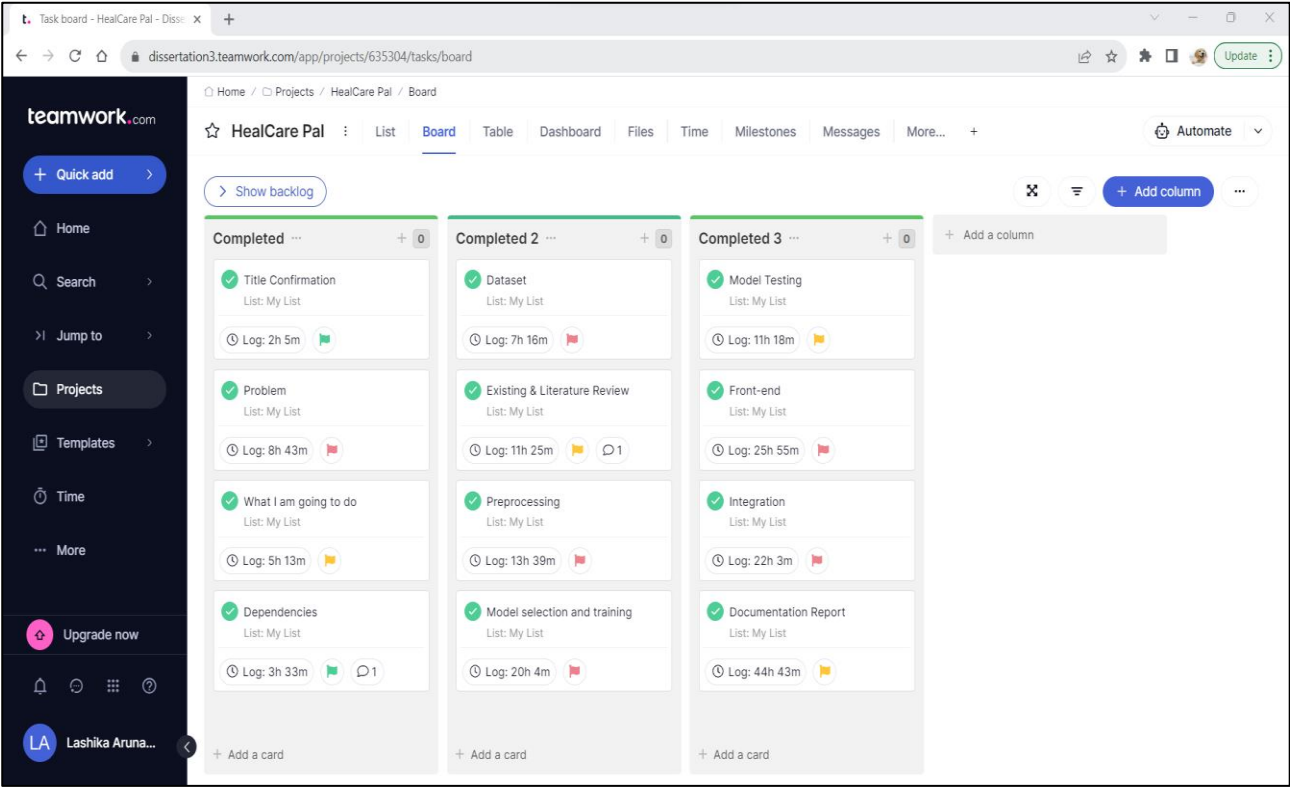
SUPERVISOR: I approve this project proposal: ☐ **Date:** / /

Supervisor Name:

Supervisor Signature:

NOTE: It is the supervisor's responsibility to approve this project as meeting the requirements for the module. This includes professional body requirements, programme requirements, and module requirements. By signing the form, you are agreeing that you have validated the suitability of the project.

Appendix B: Project Management Tool



Appendix C: Project Weekly Report

Weekly summary

Name: Lashika Arunachalam

Date: 23rd June 2023

Week: 1st week

Supervisor: Changjiang He

1. What have you done for the past week?
 - This is the 1st week.
 - Explained the project this week.
 - Last week working on which problem to be solved and the dataset.
2. What are the issues you are facing right now and how will you solve it?
 - What specific problem to focus on?
 - After the meeting, came to the conclusion to focus on Symptoms and personalized assistance.
3. What are you going to do for the next week?
 - Literature survey

Weekly summary

Name: Lashika Arunachalam

Date: 27th June 2023

Week: 2nd week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Working on Literature Survey and got some insights.
2. What are the issues you are facing right now and how will you solve it?
 - Still confused about the problem so need to read more Reference papers.
 - Dataset is not finalized yet
3. What are you going to do for the next week?
 - Dataset and Dependencies and to learn and get the overview on that

Weekly summary

Name: Lashika Arunachalam

Date: 04th July 2023

Week: 3rd week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Worked on Dataset and some front-end UI.
2. What are the issues you are facing right now and how will you solve it?
 - Still confused about the Dataset for image-based.
 - After the call summary, will shift the image to the text-based dataset.
3. What are you going to do for the next week?
 - Dataset will be finalized and will start the pre-processing
 - Will try to finish the front-end

Weekly summary

Name: Lashika Arunachalam

Date: 11th July 2023

Week: 4th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Dataset has been finalized and pre-processing is 50% done.
 - Some parts of the Front-end have been 75% completed.
2. What are the issues you are facing right now and how will you solve it?
 - I am unaware of many pre-processing techniques, and it took me time to learn about them.
 - In Front-end, integrating with Flask and JavaScript I was facing some issues. Will overcome this by learning the specific concept.
3. What are you going to do for the next week?
 - Will work on the report for the Milestone 3 submission.
 - Drafting the Literature survey.

Weekly summary

Name: Lashika Arunachalam

Date: 18th July 2023

Week: 5th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Dataset has been finalized and pre-processing is 90% done.
 - Learning JavaScript concepts.
 - Preparing the Report - Chapter 1 Introduction, Aim, Objective, Background, and Report View.
2. What are the issues you are facing right now and how will you solve it?
 - Drafting the report for Chapter 1 and literature survey so will read all the journal papers and take points from those.
3. What are you going to do for the next week?
 - Will work on the report again for this week as Milestone 3 submission is near.

Weekly summary

Name: Lashika Arunachalam

Date: 25th July 2023

Week: 6th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Worked on the report for this week as Milestone 3 submission
 - Preprocessing is completed
 - Learning JavaScript concepts.
2. What are the issues you are facing right now and how will you solve it?
 - Training the dataset and choosing the algorithms
3. What are you going to do for the next week?
 - Training the dataset and choosing the algorithms and will work on it.

Weekly summary

Name: Lashika Arunachalam

Date: 1st August 2023

Week: 7th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Selecting the correct algorithm – SVM
 - Started writing the Methodology part in Report
2. What are the issues you are facing right now and how will you solve it?
 - Choosing the algorithms
3. What are you going to do for the next week?
 - Training the dataset and will try to finish the methodology in the report.

Weekly summary

Name: Lashika Arunachalam

Date: 8th August 2023

Week: 8th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Algorithm is finalized as SVM, and training is completed.
 - Testing is started.
 - Completed writing the Methodology part of the Report
2. What are the issues you are facing right now and how will you solve it?
 - Integration part both front end and backend
3. What are you going to do for the next week?
 - Testing the dataset and will start the implementation in the report.

Weekly summary

Name: Lashika Arunachalam

Date: 15th August 2023

Week: 9th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Testing is completed.
 - 50% of the Implementation part of the Report is completed
2. What are the issues you are facing right now and how will you solve it?
 - Integration part of both front-end and backend
3. What are you going to do for the next week?
 - Integration part and will finish the implementation in the report.
 - Will get Studiosity Feedback

Weekly summary

Name: Lashika Arunachalam

Date: 22th August 2023

Week: 10th week

Supervisor: Changjiang He

1. What have you done for the past week?
 - Integration is completed.
 - Implementation part of the Report is completed
2. What are the issues you are facing right now and how will you solve it?
 - Report needs to be completed, facing issues with rephrasing the sentences.
3. What are you going to do for the next week?
 - Will submit a report for Studiosity Feedback

Appendix D: Model Training

```

1  # Import Libraries
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4  from sklearn.ensemble import RandomForestClassifier
5  from sklearn.svm import SVC
6  from sklearn import metrics
7  import joblib
8
9  # Upload Dataset
10 df=pd.read_csv('pre_processed.csv')
11
12 # Striping the space in the column heading
13 df.columns= df.columns.str.strip()
14
15 # Splitting Input and Output variables
16 X=df.drop(['Disease'], axis=1)
17 y=df.iloc[:,-1]
18
19 # Splitting the Dataset into Testing and Training
20 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=42)
21
22 # Training the SVM algorithm
23 svc_algo=SVC()
24 svc_algo.fit(X_train,y_train)
25
26 # Predicitng the Dataset
27 pred=svc_algo.predict(X_test)
28
29 # Saving the algorithm trained for the use of Integration
30 joblib.dump(svc_algo,"svc_algorithm.joblib")

```

Appendix E: Model Integration using Streamlit

```

1  from asyncio.windows_events import NULL
2  from cgitb import html
3  import streamlit as st
4  import pandas as pd
5  import numpy as np
6  from joblib import load
7  import requests, webbrowser
8  from bs4 import BeautifulSoup, Doctype
9  import csv
10 import streamlit.components.v1 as components
11
12
13 def add_bg_from_url():
14     st.markdown(
15         f"""
16         <style>
17         .stApp {{
18             background-image: url("https://img.freepik.com/free-vector/clean-medical-background_53876-97927.jpg?size=626&ext=.jpg&ga=GA1.2.1108854164.1641111111.1641111111");
19             background-attachment: fixed;
20             font-color:black;
21             background-size: cover;
22         }}
23         </style>
24         """,
25         unsafe_allow_html=True
26     )
27
28 add_bg_from_url()
29

```

```
app.py X
Healcare > app.py > ...
31 st.markdown("<h1 style='text-align: center'>HealCare Pal</h1>", unsafe_allow_html=True)
32
33 def predict_disease_from_symptom(symptom_list):
34     symptoms = {'abdominal pain': 0, 'abnormal menstruation': 0, 'acidity': 0, 'acute liver failure': 0, 'altered sensorium': 0, 'anxiety': 0, 'back pain':
35     'blister': 0, 'blood in sputum': 0, 'bloody stool': 0, 'blurred and distorted vision': 0, 'breathlessness': 0, 'brittle nails': 0, 'bruising': 0, 'burning m
36     'cold hands and feets': 0, 'coma': 0, 'congestion': 0, 'constipation': 0, 'continuous feel of urine': 0, 'continuous sneezing': 0, 'cough': 0, 'cramps': 0, '
37     'depression': 0, 'diarrhoea': 0, 'dischromic patches': 0, 'distention of abdomen': 0, 'dizziness': 0, 'drying and tingling lips': 0, 'enlarged thyroid': 0
38     'fast heart rate': 0, 'fatigue': 0, 'fluid overload': 0, 'foul smell of urine': 0, 'headache': 0, 'high fever': 0, 'hip joint pain': 0, 'history of alcohol
39     'inflammatory nails': 0, 'internal itching': 0, 'irregular sugar level': 0, 'irritability': 0, 'irritation in anus': 0, 'joint pain': 0, 'knee pain': 0, 'la
40     'loss of balance': 0, 'loss of smell': 0, 'malaise': 0, 'mild fever': 0, 'mood swings': 0, 'movement stiffness': 0, 'mucoid sputum': 0, 'muscle pain': 0, 'mu
41     'nausea': 0, 'neck pain': 0, 'nodal skin eruptions': 0, 'obesity': 0, 'pain behind the eyes': 0, 'pain during bowel movements': 0, 'pain in anal region': 0
42     'patches in throat': 0, 'phlegm': 0, 'polyuria': 0, 'prominent veins on calf': 0, 'puffy face and eyes': 0, 'pus filled pimples': 0, 'receiving blood trans
43     'redness of eyes': 0, 'restlessness': 0, 'runny nose': 0, 'rusty sputum': 0, 'scurrying': 0, 'shivering': 0, 'silver like dusting': 0, 'sinus pressure': 0, 's
44     'slurred speech': 0, 'small dents in nails': 0, 'spinning movements': 0, 'spotting urination': 0, 'stiff neck': 0, 'stomach bleeding': 0, 'stomach pain':
45     'swelling joints': 0, 'swelling of stomach': 0, 'swollen blood vessels': 0, 'swollen extremities': 0, 'swollen legs': 0, 'throat irritation': 0, 'toxic loo
46     'vomiting': 0, 'watering from eyes': 0, 'weakness in limbs': 0, 'weakness of one body side': 0, 'weight gain': 0, 'weight loss': 0, 'yellow crust ooze': 0,
47     'itching': 0}
48
49     for s in symptom_list:
50         symptoms[s] = 1
51     df_test = pd.DataFrame(columns=list(symptoms.keys()))
52     df_test.loc[0] = np.array(list(symptoms.values()))
53     #df_test
54     clf = load(str("svc_algorithm.joblib"))
55     result = clf.predict(df_test)
56     del df_test
57
58     # Display the selected options
59     st.write("You Selected:", options)
60
61     st.markdown("<h3 style='padding-top:20px;'>Let's know about your Disease</h3>", unsafe_allow_html=True)
62     return f"{result[0]}"
```

```
app.py X
Healcare > app.py > ...
65 options = st.multiselect(
66     'Select your symptoms : ',
67     ['abdominal pain', 'abnormal menstruation', 'acidity', 'acute liver failure', 'altered sensorium', 'anxiety', 'back pain', 'belly pain', 'blackheads', 'b
68     'blister', 'blood in sputum', 'bloody stool', 'blurred and distorted vision', 'breathlessness', 'brittle nails', 'bruising', 'burning micturition', 'chest pa
69     'cold hands and feets', 'coma', 'congestion', 'constipation', 'continuous feel of urine', 'continuous sneezing', 'cough', 'cramps', 'dark urine', 'dehydration
70     'depression', 'diarrhoea', 'dischromic patches', 'distention of abdomen', 'dizziness', 'drying and tingling lips', 'enlarged thyroid', 'excessive hunger', '
71     'fast heart rate', 'fatigue', 'fluid overload', 'foul smell of urine', 'headache', 'high fever', 'hip joint pain', 'history of alcohol consumption', 'increas
72     'inflammatory nails', 'internal itching', 'irregular sugar level', 'irritability', 'irritation in anus', 'joint pain', 'knee pain', 'lack of concentration',
73     'loss of balance', 'loss of smell', 'malaise', 'mild fever', 'mood swings', 'movement stiffness', 'mucoid sputum', 'muscle pain', 'muscle wasting', 'muscle we
74     'nausea', 'neck pain', 'nodal skin eruptions', 'obesity', 'pain behind the eyes', 'pain during bowel movements', 'pain in anal region', 'painful walking', 'p
75     'patches in throat', 'phlegm', 'polyuria', 'prominent veins on calf', 'puffy face and eyes', 'pus filled pimples', 'receiving blood transfusion', 'receiving
76     'redness of eyes', 'restlessness', 'runny nose', 'rusty sputum', 'scurrying', 'shivering', 'silver like dusting', 'sinus pressure', 'skin peeling', 'skin rash',
77     'slurred speech', 'small dents in nails', 'spinning movements', 'spotting urination', 'stiff neck', 'stomach bleeding', 'stomach pain', 'sunken eyes', 'swea
78     'swelling joints', 'swelling of stomach', 'swollen blood vessels', 'swollen extremities', 'swollen legs', 'throat irritation', 'toxic look (typhos)', 'ulcer
79     'vomiting', 'watering from eyes', 'weakness in limbs', 'weakness of one body side', 'weight gain', 'weight loss', 'yellow crust ooze', 'yellow urine', 'yello
80     'itching'])
81
82
83 def local_css(file_name):
84     with open(file_name) as f:
85         st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)
86
87 local_css("style.css")
88
89 ans=predict_disease_from_symptom(options)
90 if(ans[len(ans)-1]!=' '):
91     ans=ans[:-1]
92     print ("x"+ans+"x")
93     with st.expander("Disease "):
94         st.header(ans)
95
```

```

app.py x
Healcare > app.py > ...
96
97 with st.expander("About Disease "):
98     df=pd.read_csv('symptom_Description.csv')
99     disease=str(ans);
100     df_new = df[df['Disease'] == disease]
101
102     disease=df_new['Description'].values
103     strrr="";
104     i=0
105     for i in disease:
106         strrr+=i
107     disp = f'<div style="font-size: 18px; font-weight: 700">{" • "+strrr}</div>'
108     st.write(disp, unsafe_allow_html=True)
109
110
111     google_search = requests.get ("https://www.google.com/search?q="+ ans)
112     soup = BeautifulSoup(google_search.text, 'html.parser')
113     search_res=soup.select('Ap50Sd ,AP7wnd')
114     lensearch=len(search_res)
115     i=0
116     while(i<lensearch):
117         st.write(" • "+search_res[i].string)
118         i+=1;
119
120 with st.expander("Medical Test "):
121     df=pd.read_csv('symptom_test.csv')
122     disease=str(ans);
123     df_new = df[df['Disease'] == disease]
124     i=1;
125     prec=""
126     for col in df_new:
127         if(i==1):
128             i=2
129             continue
130     prec=" • "+str(df_new[col].values[0])+"\n"

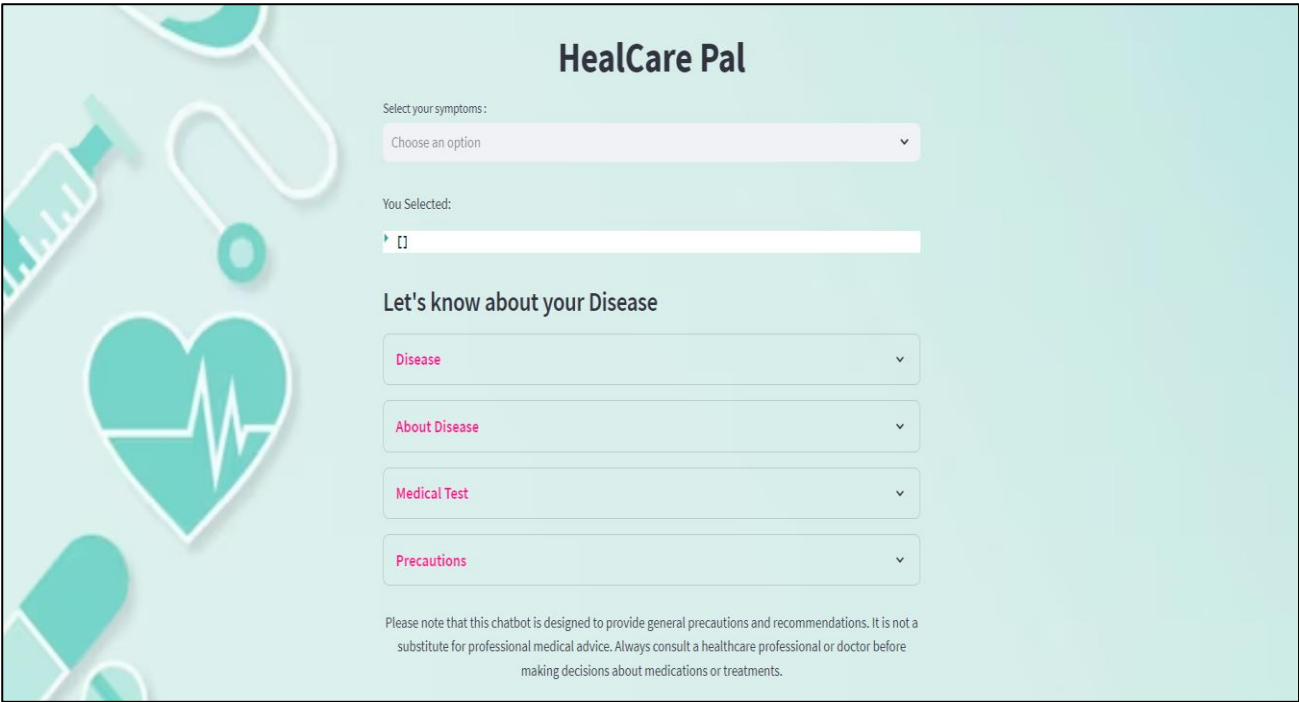
```

```

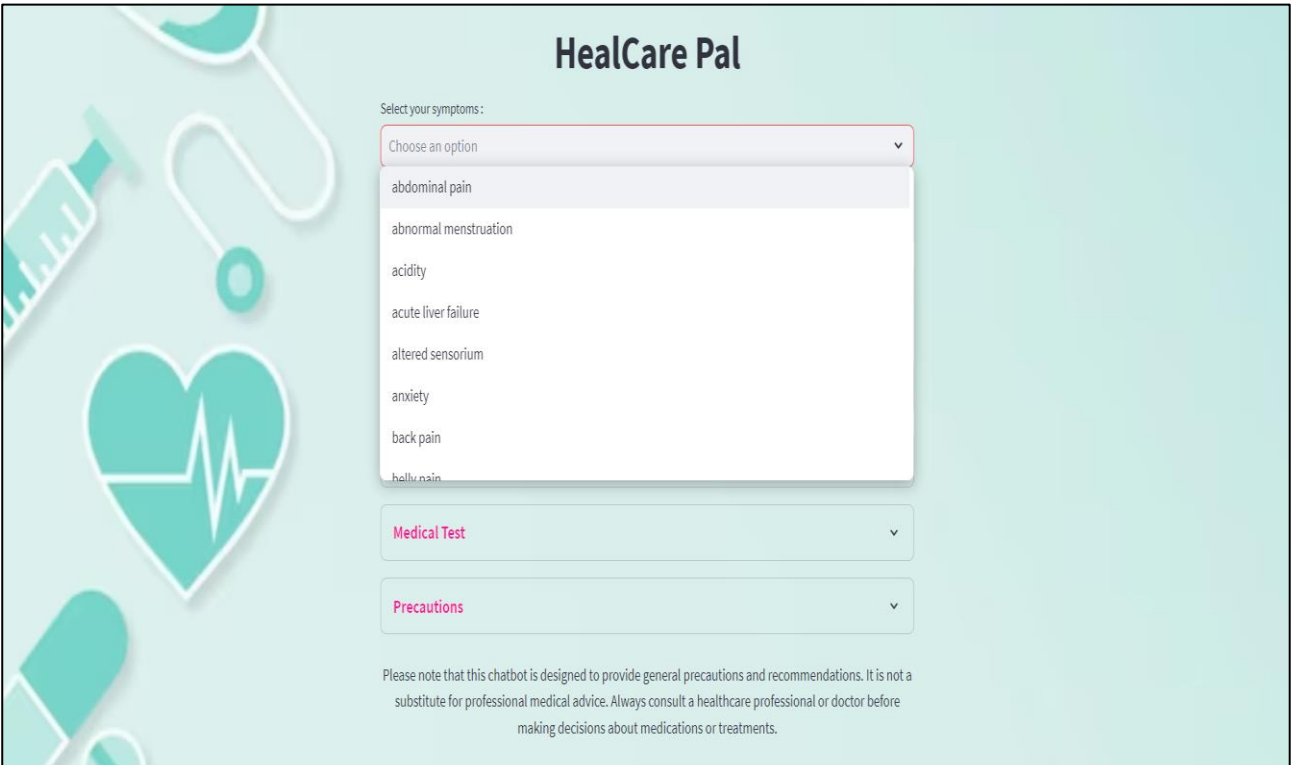
app.py x
Healcare > app.py > ...
130     prec=" • "+str(df_new[col].values[0])+"\n"
131     if(prec!=" • nan\n"):
132         x=0
133         dis=""
134         for x in range(len(prec)):
135             if(prec[x]=='.'):
136                 dis+="\n • "
137             else:
138                 dis+=prec[x]
139         disp = f'<div style="font-size: 18px; font-weight: 700">{dis}</div>'
140         st.write(disp, unsafe_allow_html=True)
141
142
143 with st.expander("Precautions "):
144     df=pd.read_csv('symptom_precaution.csv')
145     disease=str(ans);
146     df_new = df[df['Disease'] == disease]
147     i=1;
148     prec="";
149     for col in df_new:
150         if(i==1):
151             i=2
152             continue
153         prec=" • "+str(df_new[col].values[0])+"\n"
154         disp = f'<div style="font-size: 18px; font-weight: 700">{prec}</div>'
155         st.write(disp, unsafe_allow_html=True)
156
157 st.empty()
158
159 footer_text = "Please note that this chatbot is designed to provide general precautions and recommendations. It is not a substitute for professional
160 st.markdown(f'<p style="left: 0; bottom: 0; width: 100%; text-align: center; padding: 10px 0;">{footer_text}</p>', unsafe_allow_html=True)
161

```

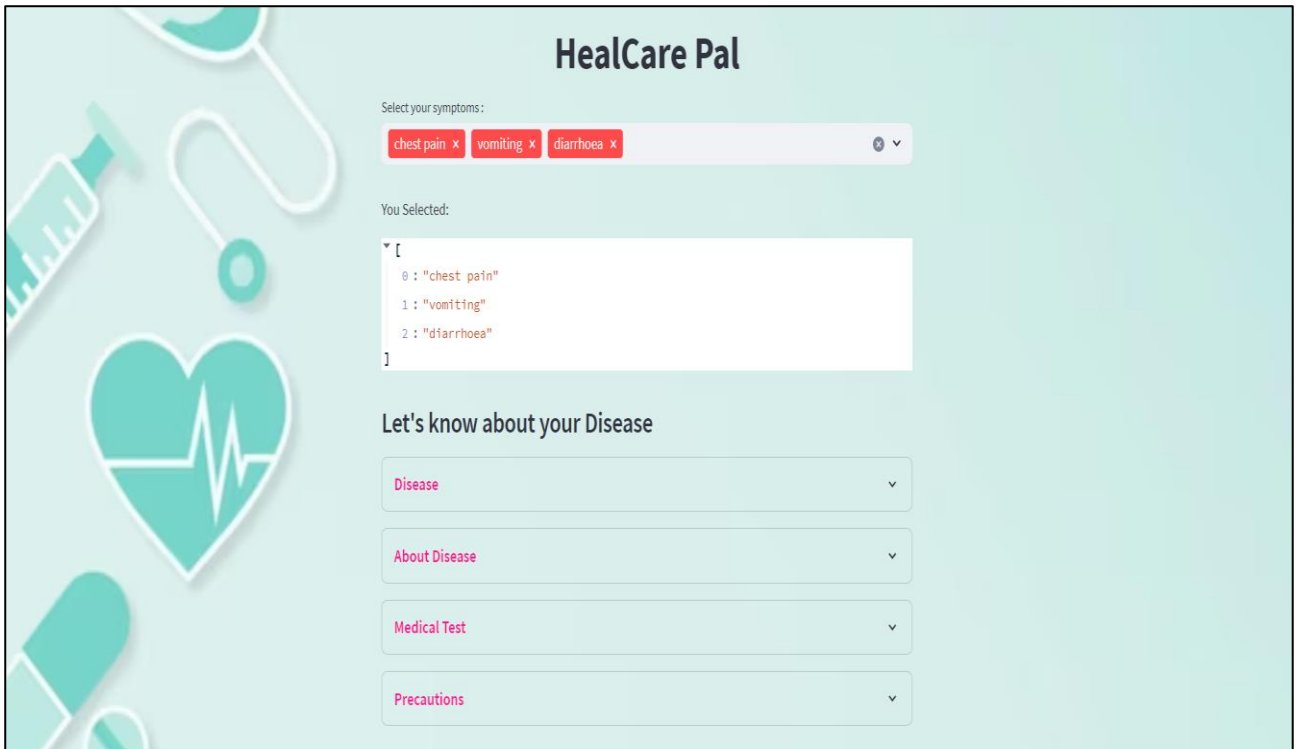
Appendix F: User-Interface



Appendix G: User-Interface – Options



Appendix H: User-Interface – Selecting Options

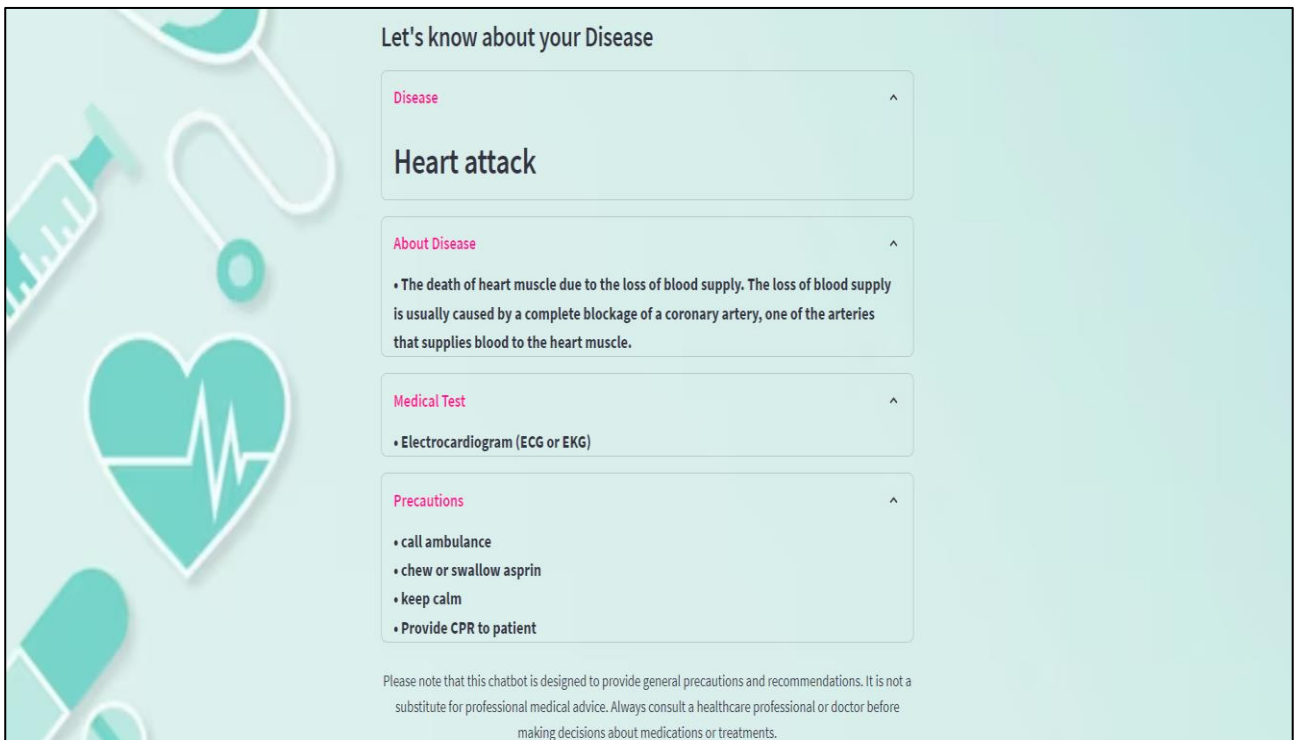


The screenshot shows the 'HealCare Pal' interface. On the left, there are medical icons: a stethoscope, a syringe, a heart with an ECG line, and a bandage. The main content area has a title 'HealCare Pal' and a section 'Select your symptoms :'. Below this is a horizontal list of three red buttons: 'chest pain x', 'vomiting x', and 'diarrhoea x'. A dropdown arrow is on the right. Below the buttons, it says 'You Selected:' followed by a JSON array:

```
[{"0": "chest pain"}, {"1": "vomiting"}, {"2": "diarrhoea"}]
```

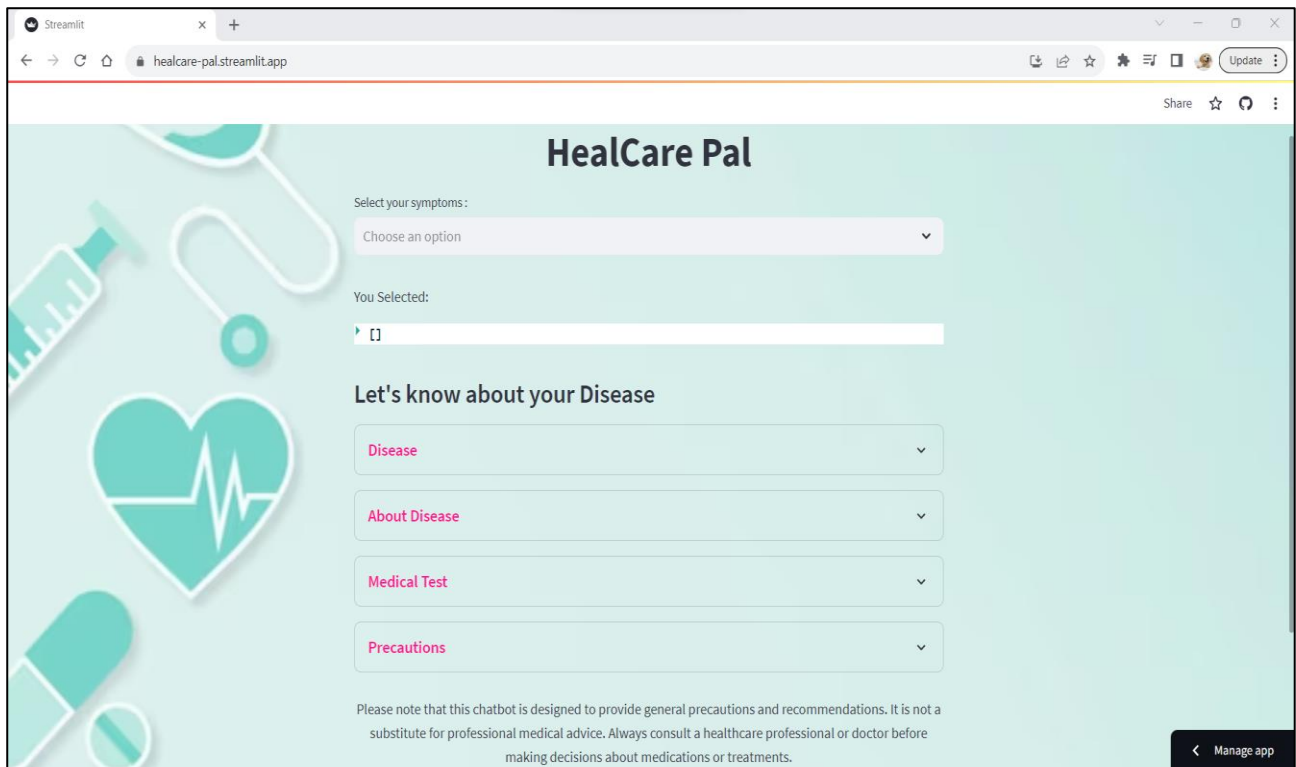
. The next section is 'Let's know about your Disease', which contains four dropdown menus: 'Disease', 'About Disease', 'Medical Test', and 'Precautions'.

Appendix I: User-Interface – Display of Disease and Precautions



This screenshot shows the 'HealCare Pal' interface with the 'Disease' dropdown expanded. The title 'Let's know about your Disease' is at the top. The 'Disease' dropdown shows 'Heart attack'. The 'About Disease' dropdown is expanded, showing a description: 'The death of heart muscle due to the loss of blood supply. The loss of blood supply is usually caused by a complete blockage of a coronary artery, one of the arteries that supplies blood to the heart muscle.' The 'Medical Test' dropdown is expanded, showing 'Electrocardiogram (ECG or EKG)'. The 'Precautions' dropdown is expanded, showing a list of actions: 'call ambulance', 'chew or swallow aspirin', 'keep calm', and 'Provide CPR to patient'. At the bottom, there is a disclaimer: 'Please note that this chatbot is designed to provide general precautions and recommendations. It is not a substitute for professional medical advice. Always consult a healthcare professional or doctor before making decisions about medications or treatments.'

Appendix J: HealCare Pal App deployed to Streamlit



Links:

Github: <https://github.com/lashikaa/HealCare-Pal>

Project Management Tool:

<https://dissertation3.teamwork.com/app/projects/635304/tasks/board>

My App: <https://healcare-pal.streamlit.app/>