Medhat

Software Requirements Specification

Version 2

26/12/2023

Mahmoud Ibrahim Barbary
Lead Software Engineer

Prepared for
Software Engineering
Instructor: Dr.Ahmed Salama
Fall 2023

Revision History

Date	Description	Author	Comments
28/11	<Version 1>	Menna	<First Revision>
26/12	<final version>	Menna	

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	<Your Name>	Lead Software Eng.	

Table of Contents

1. Introduction

This document covers the requirements specification for the software system Medhat, a medical chatbot application aimed to make healthcare more accessible, user-friendly, and convenient without sacrificing the precision that the medical field rigorously requires. The document contains detailed descriptions of the design, functionality, usage, benefits, cases, and expectations for this application and some models for clearer visualization.

1.1 Purpose

The purpose of this SRS is to clearly state the requirements of the Medhat personalized medical application.
The intended audience is the developers, testers, and managers of the project, and possibly the customers involved within the project [Stakeholders - Investors - Clients …]

1.2 Scope

Through this project, we plan on using machine learning to improve and fill a gap in the medical field. We plan on having a chatbot that has conversational capabilities to ease the user's experience, especially in cases where medical attention by professionals is hard or unavailable.

With this scope in mind, the application will diagnose patients using the chatbot, allowing users to have a personalized experience by allowing them to create accounts so that Medhat could consider information about them, and their medical history, and keep checking up on them. The application will provide a news section to raise medical awareness through the CDC and NHS websites.

In summary, Medhat goes beyond being a mere diagnostic tool; it aspires to be a pioneering force in the integration of machine learning within the medical domain, contributing to the ongoing evolution of healthcare technology and fostering a holistic approach to user well-being.

1.3 Objective

The objective of this project is to make the medical field as accessible as possible، and to raise awareness about the medical field and any potential dangers or advancements.
The goal is to enter this niche market with this application which provides users with an accessible way to check up on themselves and get medical information through its integrated database.

1.4 Definitions, Acronyms, and Abbreviations

- *CDC: Centers for Disease Control of the United States*
- *NHS: National Health Service*
- *DSDB: Disease-Symptom Database*

1.5 References

Chat Bot With PyTorch - NLP And Deep Learning - Python Tutorial (Part 1) - YouTube

spaCy · Industrial-strength Natural Language Processing in Python

NLTK :: Natural Language Toolkit

Microsoft Azure

The NHS website - NHS (www.nhs.uk)

1.6 Overview

- *Section 1 Illustrates a clear picture of the idea of the app, what it is supposed to do, and a perspective for its usage through its environment. It also acts as a guide through the rest of the SRS.*
- *Section 2 Gives a general overview of requirements and constraints for the project, it acts as an introduction to the general idea of what is required of the application and why.*
- *Section 3 Describes in full detail the requirements and constraints of the project, both functional and non-functional. It describes more closely the "how" of what is required from the project.*
- *Section 4 Provides descriptive models visualizing the project from different perspectives.*

2. General Description

2.1 Product Perspective

People after the COVID-19 pandemic have been naturally more paranoid and more careful about their medical well-being and more aware of the topic overall.
The application Medhat is meant to fit within these developments and close the gap between the average person wanting to check up on themselves and the complexities of the medical world. A person can have a lot of doubts about their health constantly, and it is not always feasible to go to a doctor for each and every one, due to financial or time limitations.

A clear example of this is when an individual has the symptoms of common cold or the flu and immediately panics thinking they have COVID. Or the more dangerous opposite, if they thought they had a common cold and actually had COVID. Both these situations are undesirable and can have costly consequences.
Thus the solution offered is that these individuals could first try to verify their concern whether it is valid or not, and if it is then Medhat would recommend them possible treatments, medications and the specialized doctor they should visit.Now the individual could pursue to resolve his issue after verifying its validity. This is the main purpose of the system.

Medhat is not a doctor and thus it does not aim to diagnose or prescribe to patients replacing the role of a professional doctor, it only tries to infer the potential issue and verify the user's worries, then suggests the best course of action.

Medhat aims to promote accessibility to users` healthcare not through those means only, it also acts as an effective, easier to use one-stop search engine for information about the medical field

and specifically about the traits associated with diseases from their symptoms to their potential causes and preventative measures. The user asks the chatbot the required information and the chatbot presents it, ensuring that this information is verified and up-to-date, further helping the idea of convenience instead of searching from multiple sources

Concerning related products or possible competition, there is a similar service provided by "Ada Health". While they offer a similar functionality, Medhat offers a more approachable and intuitive interface with a larger scope encompassing the many aspects of healthcare including awareness, check-up, educational and more.

And while an argument exists regarding the many existing large language model [LLM] chatbots -most notorious being ChatGPT- could serve the same function, It is of utmost importance to note that Medhat is specifically tailored to the field of healthcare, and has consistent, verified and updated data on the various aspects within that these LLMs do not. They are not a dependable source for such sensitive and critical data.

2.2 Product Functions

The application is required to have the following functionalities:

- *Chatbot that shall be capable of:*
  - *Understand user input sentences.*
  - *Respond accordingly to the user and hold conversation.*
  - *Store the conversations and context for the current chat.*
  - *Classify entities within the user input (i.e symptoms) and store them.*
  - *Give the user a basic diagnosis based on their given and inferred symptoms. It shall be able to recommend potential curative/preventive measures accordingly.*
  - *Access the medical database and the required information within.*
  - *Respond to users' questions involving said information accordingly.*
- *Medical database that shall:*
  - *Include extensive data about diseases, their symptoms, treatments, causes and other properties.*
  - *Have data verified by medical experts.*
  - *Be designed with optimal performance in mind.*
- *Features:*
  - *A UI that shall be user friendly and ease of use.*
  - *A user log-in screen with user account registration and characteristics (i.e age, weight) being stored.*
  - *A news section related to the medical field, providing the latest news, outbreaks and advancements within the field.*

2.3 User Characteristics

The Medhat application aims to be accessible to anyone who wants to check their health, this is why we aim for the application to be used by the general adult population aged 18 and above. Medhat's main demographic reaches almost everyone, but it is more specialized for users who have the following characteristics:

- *Age range: Over 18 years of age, Medhat does not cater enough to the diagnosis of children and specializes in adult conditions.*
- *Location: The main functionality is generalized to all regions, but for specialized news and a more personalized experience, it is limited to Egypt and later on The Middle East.*

- *Behavior:Users that are health-conscious and knowledge-seeking particularly within the medical field. Additionally,it caters to ones who are unable to verify the validity regarding their issue or are doubtful about their current health status.*
- *Educational Orientation: Medhat caters to users with an education background as it serves as an educational resource for users keen on expanding their understanding of medical conditions. It encourages continuous learning and awareness in the realm of healthcare.*
- *Accessibility: The application is only in English, with no capacity for other languages or accommodation for disabilities, but with the potential to be expanded later on.*
- *Technology Adoption: Medhat's target audience is expected to be in possession of smartphones and be comfortable using mobile applications as part of their daily lives ,preferring the convenience of accessing health information remotely.*
  Little to no technical expertise is needed to use the application.
- *Limitation Considerations:The application caters to individuals concerned about their health that want to visit a doctor but are limited either financially, cannot visit a doctor due to distance, geographical boundaries, limited time factor or many other possible reasons.*
- *Occupation: The application aims to provide busy, working individuals with jobs that could be detracting from focusing on their health with the effective means to do so.*
- *Emergencies: Users might turn to Medhat only in emergent situations, which Medhat supports by guiding them immediately to the best course of action.*

2.4 General Constraints

- *Hardware:The application is intended for mobile devices , being cross-platform on iOS and android devices. It can run on any smartphone device and will later be expanded to other platforms such as Windows and as a web application.*

- *Software: Compatibility for SQL [Postgresql] ,Flutter, Python and all associated libraries is required to run the app. The software system would be published only on Google Play and Apple Store.*

- *Network Access: Medhat needs access to the internet to update its current information Database, provide news present within its own section and store the users` conversation, diagnoses etc..*

- *Development Limitations: The current development team is small and inexperienced in various aspects going into this project such as Machine Learning which inhibited the capabilities of the resulting chatbot, lacking experience in UI and application development and more. A bigger, more diverse development team with a multitude of differing experiences is needed.*

- *Time Limitations: Development time is quite tight consisting of only 10 weeks, which further constrains the project, the possibility to learn the required knowledge mentioned beforehand and further improvements towards the projects` functionalities.*

- *Medical Team: There are currently no medical experts overseeing the project progress which puts further weight on the team to verify the data, its validity and their sources. It also detracts from the accuracy and potential usefulness of the project as a whole from its data to the diagnosis and check-up processes. Having access to such professionals would substantially reinforce the robustness, precision and powerful value of the software.*

- *Third-Party Integration: Integrating various external services or APIs together, such as the healthcare databases or a flutter framework, may present challenges such as compatibility issues and data format differences which could lead to potential changes in the services and could impact the seamless functioning of the software system.*
- *Data Collection & Availability: There exists no public and reliable dataset for the data required for the application, at least none that are easily gathered and standardized in a clean format and would require tremendous effort in cleaning.*

- *Hosted Database: The database used is currently locally hosted which introduces many issues such as when several developers need to access/update the database simultaneously they would be limited in doing so, and if the system was to be quickly accessed as is required, it would need to be server-hosted.*

- *Continuous Updates: Given the ever evolving nature of the medical field, the application, its database and even the chatbot methodologies are required to be constantly and regularly updated to keep up with those changes. This presents a constraint that a maintenance process has to be established else the system would lose its credibility and correctness.*
- *Accessibility: Given the goal of the app, users with special needs need to be accommodated for, these people are ones that would be significantly and positively affected by the application and thus it would need to comply with accessibility standards.*
- *Legal: The software must adhere to regulatory and legal standards set for such software systems of healthcare, such as Health Insurance Portability and Accountability Act (HIPAA), keeping in consideration that these regulations could differ from one region to another.*
- *Ethical: The software must adhere to ethical standards of data protection and mining, and follow established ethical guidelines which is crucial to gain users' trust. These include the ethical guidelines set for software engineering by the IEEE. It is also required for such healthcare systems to be certified.*
- *Rule-Based Chatbot: Since given the team and its resources, it could not be built on an existing ML model, which constrains its capabilities, especially in unscripted input scenarios. But to accommodate for scalability and the ability to be transformed into a ML model seamlessly, it is required to be fundamentally created on a scalable foundation.*

2.5 Assumptions and Dependencies

The Medhat application is heavily dependent on its sources, from the CDC and the NHS, and while the currently present data on diseases, symptoms, and their related information is currently static, the news page to be implemented depends on the CDC/NHS news page. An assumption is made that these pages will remain available.

The data gathered from the NHS is assumed to be correct and up to date.
The user is assumed to expect accurate results from the chatbot but should never rely only on them and must still seek the help of professionals as required.
The user is assumed to have basic knowledge of applications and necessary do's/don'ts like safeguarding their password, complying with the application's EULA and terms of services, etc.
The user's input is assumed to be in recognizable and grammatically correct English with little error, otherwise the chatbot loses its precision and capability for correctly addressing user input.

The user is expected to be of 18 years of age or older, which is verified at the account creation stage, and it is assumed that none would enter incorrect data about themselves or their age, since it produces inaccurate results for the user themselves which could do more harm than good. Thus the user is informed of this assumed to acknowledge it and avoid practicing this behavior.

The application is dependent on the PostgreSQL database management system, Python programming language, existing libraries like [spaCy, NLTK, NumPy], and the Flutter framework, and thus all of them are assumed to be continuously available.

2.6 Gap Analysis

Point of Comparison	Current Status	Gap Analysis	Target
Software Application Framework	- Choose tools/framework to use. - Experience in full stack development.	- Decide on a framework and learn all its aspects.	- A framework to handle the application housing, application development.
Database	- Experience in database design, SQL, and indexing. - Based on the CDC Database.	- Learn how to access the database using a host language. - Modify the CDC Database.	- Query the data using a host language. - A compatible database.
AI/Chatbot	- A functional version of the chatbot. - Background in natural language processing.	- Learn machine learning techniques.	- Use Python for the chatbot. - Use machine learning techniques for a data-driven chatbot.
Overall	- No experience in linking multiple components/tools. - Find a way to accommodate each user.	- Use the concepts learned from the 3 courses to facilitate the linking. - Divide our target market into segments.	- A software application featuring a well-built database and a helpful chatbot that suits every user's needs.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface for the mobile application shall be designed using Flutter. It shall offer users a simple interface that delivers all the essential information on each page within the application. The first page will allow users to create an account or log into their user account. Another page will be for the user profile, displaying their personal and medical information. The chatbot shall have a dedicated page that will provide users with a personalized communication channel.

Lastly, the application shall have a page that acts as a news section. Overall, the application shall adopt a unified design that lets users find familiarity and clarity within its layout. The goal is to provide users with a straightforward navigation experience, minimizing any potential confusion during application use.

3.1.2 Software Interfaces

The application shall be tailored for Android using the cross-platform framework, Flutter, which provides flexibility for potential adaptation to other platforms. It shall have an integrated part for the chatbot component, developed using Python and relevant libraries, that processes input from the user during the medical diagnosis process. Moreover, the relational database management system (DBMS) PostgreSQL is jointly used by the application and the chatbot to process queries, further nurturing a dynamic user experience through efficient data querying.

3.1.3 Communications Interfaces

User interaction within the application shall be through a dedicated page for the chatbot within the application, allowing communication between the user and the chatbot. Users shall input their symptoms throughout the medical diagnosis process, wherein the chatbot communicates with the database to process queries and deliver the relevant information. In addition, the chatbot shall give appropriate responses regarding specific inquiries about various diseases, enhancing overall user engagement.

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 Application Database

3.2.7.1 Introduction

This constitutes the backbone of the system, it is an extensive database containing various medical information, specifically about diseases and all their characteristics and traits. It would also include the users` credentials and past diagnoses.
This database needs to be standardized, verified, include security measures, be accessible and optimized.

3.2.7.2 Inputs

The database contains medical data stored in separate tables as different entities [disease, symptoms, treatments, medications …] with relationship tables to represent relations between these entities. And it includes all user accounts` data and credentials that are being constantly inserted as the system is used.

3.2.7.3 Processing

At runtime, the database gets queried and accessed according to the processes within the chatbot, whether it needs to retrieve the symptoms associated with a certain disease or wants to update the database with a new user diagnosis.It could also be updated through the application if a new user created an account.

3.2.7.4 Outputs

The database is queried at various stages and activities of the system, each time giving as output the requested information.

3.2.7.5 Error Handling

The Data Base Management System [DBMS] handles errors such as a system crash in the middle of updating the database by keeping track through a system-wide activity catalog that allows it to roll back the changes and re-execute them after restoring the database to the last stable version.

3.2.2 Application User Interface

3.2.7.1 Introduction

The Application User Interface (UI) serves as the gateway for users to interact with the Medhat system. It provides a user-friendly and intuitive experience along with a visually appealing layout, ensuring seamless navigation and access to its various features.
This is one of the biggest steps the application takes in order to achieve its goal which is to make healthcare and the medical field more accessible.

3.2.7.2 Inputs

User inputs within the UI include interactions with the chatbot, account management actions, and requests for specific medical information. These inputs are processed to trigger relevant functionalities -which are mentioned later- within the application.
Also due to the nature of the system being on mobile, the UI shall have to receive inputs from the touch screen correctly.

3.2.7.3 Processing

The UI processes user inputs by facilitating communication with the chatbot and the database, retrieving and displaying medical information, and managing user accounts securely.

3.2.7.4 Outputs

The UI generates outputs in the form of the information displayed including the different tabs present, the chatbot responses, and account management confirmations.

3.2.7.5 Error Handling

The UI incorporates error handling mechanisms to address issues such as incorrect inputs, communication failures with the chatbot, or account management errors. Clear error messages are provided to guide users in resolving issues effectively.

3.2.1 Registration

3.2.1.1 Introduction

This functionality aims to allow the user to create his account on the system to be used within the application. It acts as the starting point for the user experience.

3.2.1.2 Inputs

The user inputs information regarding both his account credentials, including a unique username and password, and his personal information which include their date of birth, sex, height, weight and any chronic illnesses they have.

3.2.1.3 Processing

This system is supposed to support secure password policies, including a minimum length and other requirements.
It would also not allow users below 18 years of age to create an account.
After successfully creating an account, it is stored to a secure database specialized for user credentials.

3.2.1.4 Outputs

The user account is created and verified within the system, giving them the ability to log in the system.

3.2.1.5 Error Handling

Invalid inputs such as dates in the future, or negative weight numbers, are viewed as invalid and block the user from finishing the account creation process until valid inputs are entered in the correct fields.

3.2.1 User Authentication

3.2.1.1 Introduction

The user authentication functionality is essential to ensure secure access to the mobile application. This feature aims to verify the identity of users before allowing them to interact with the chatbot and access sensitive medical information regarding each user's medical history.

3.2.1.2 Inputs

The input is the user's credentials, including a unique username and password, to log in to the mobile application. The user's credentials are stored in a database designed for storing users' information that is secure.

3.2.1.3 Processing

Upon receiving user credentials, the system should be able to authenticate the user by comparing the provided credentials with the user data stored in the user database.

3.2.1.4 Outputs

If the user credentials are valid, the system grants access to the chatbot's functionalities.
If the user credentials are invalid, the system provides appropriate error messages that indicate authentication failure.

3.2.1.5 Error Handling

In the case of invalid credentials, the system informs the user of the authentication failure and provides them with the ability to re-enter their correct credentials.

3.2.2 Processing User Input

3.2.2.1 Introduction

This feature involves interpreting and processing user input to precisely identify and understand the intent behind the user's input, allowing the chatbot to engage in a meaningful conversation while responding accordingly and correctly to it.

3.2.2.2 Inputs

User-entered messages as text, these would include regular everyday conversation, the user mentioning some issue they have or requests for medical advice. They could also ask for specific information about a certain disease.

3.2.2.3 Processing

The system analyzes the user's input using Natural Language Processing (NLP) techniques to identify the user's intent and extract relevant information required for the conversation.
This would include Tokenization, Stemming/Lemmatization, Spell Correction, Stop Words and Punctuation Removal and more.

3.2.2.4 Outputs

The system generates appropriate responses based on the recognized intent, providing relevant information or asking follow-up questions to proceed with the conversation effectively based on the context of it currently.

3.2.2.5 Error Handling

In case of ambiguous or unclear user input, the system should request clarification or ask targeted questions to obtain the necessary information for accurate processing.
In case of user mis-input such as forgetting a space or misspellings, the system tries to correct that input then starts processing it.

3.2.3 Symptom/Disease Detection

3.2.3.1 Introduction

This feature encompasses the identification and analysis of symptoms or diseases mentioned or provided by the user in order to determine potential medical conditions, recommend a course of action or just provide general information about that symptom/disease.

3.2.3.2 Inputs

Symptoms or diseases entered by the user through the chatbot interface and user context obtained during the conversation, this functionality only starts when the user intent detected has to do with a symptom or a disease entity for performance optimization as the detection process is expensive, and if the intent did not require them then it is unnecessary to search for them.

3.2.3.3 Processing

The system matches user-input symptoms/disease with their known counterparts in the medical database. Then it stores this given information about the user.
If the system could not match any symptom/disease within the user input exactly, it tries to find the closest matching entity and asks the user if this was what they meant.

3.2.3.4 Outputs

Given the user intent and the mentioned symptom/disease, the system uses it to infer what the next best response is. One case is if the user was asking about treatments of a disease, the chatbot retrieves the required information about the detected disease within the input.

3.2.3.5 Error Handling

If the system could not match any symptom/disease within the user input exactly, it tries to find the closest matching entity and asks the user if this was what they meant.
If it wasn't, the chatbot tells the user it could not recognize what he typed, and asks for the user to rephrase.

3.2.3 Diagnostic Feedback

3.2.3.1 Introduction

This feature encompasses the identification and analysis of symptoms provided by the user to determine potential medical conditions, followed by generating an overview of the users` potential diseases, giving information, feedback and recommendations for each.

3.2.3.2 Inputs

The user input would be requesting a diagnosis or a check up from the chatbot.

3.2.3.3 Processing

The system combines the symptom analysis process with medical knowledge to generate potential diagnoses, considering probabilities and relevance based on the symptoms provided.

3.2.3.4 Outputs

The system provides a list of potential medical conditions or diseases based on the symptoms entered by the user, highlighting the most probable matches. It presents potential diagnoses along with relevant information about each disease, including possible causes, treatments, and recommendations for the user's consideration.

3.2.3.5 Error Handling

If the system cannot confidently associate symptoms with specific diseases or provide accurate diagnoses, it informs the user about the uncertainty and advises seeking professional medical advice.

3.2.4 Medical Information Provision

3.2.4.1 Introduction

This feature involves presenting detailed information about specific diseases to the user upon request.

3.2.4.2 Inputs

User queries or requests for information about particular diseases.

3.2.4.3 Processing

The system retrieves relevant information from the database based on the user's inquiry about a specific disease.

3.2.4.4 Outputs

The system provides comprehensive information about the requested disease, including symptoms, causes, treatments, preventive measures, and related medical fields or specializations.

3.2.4.5 Error Handling

If the requested disease information is unavailable or unclear, the system informs the user about the unavailability and suggests alternative sources or topics for exploration.

3.2.5 Medical News Section

3.2.5.1 Introduction

This feature offers users a dedicated section within the application that provides current and relevant medical news sourced from reputable sources like CDC or NHS. It aims to raise medical awareness among users.

3.2.5.2 Inputs

The news section fetches data from the respective CDC or NHS sources via integrated APIs or direct web scraping mechanisms.

3.2.5.3 Processing

The system processes and organizes the fetched news data to present it in a user-friendly format within the application.

3.2.5.4 Outputs

Users can access the latest medical news relevant to their region, providing insights into health advancements, alerts, and potential health hazards.

3.2.5.5 Error Handling

In the case of unavailability or technical issues with fetching news data, the system should inform users about the temporary unavailability and attempt to resolve the issue promptly.

3.2.6 User Interaction History

3.2.6.1 Introduction

This feature keeps a log of user interactions with the chatbot, storing past conversations for reference and continuity.

3.2.6.2 Inputs

The system automatically stores conversation logs between users and the chatbot.

3.2.6.3 Processing

The system processes and organizes the conversation logs for easy retrieval and reference.

3.2.6.4 Outputs

Users can review their past conversations with the chatbot to track symptom assessments or reference previous diagnoses.

3.2.6.5 Error Handling

The system should handle storage and retrieval errors, ensuring the availability of interaction history for users without compromising data integrity or security.

3.2.7 Account Management

3.2.7.1 Introduction
This feature allows users to manage their accounts by modifying personal information or deleting their accounts entirely.

3.2.7.2 Inputs
Users input their credentials to access their account management options.

3.2.7.3 Processing
The system processes the user's request and provides appropriate functionalities to modify or delete user accounts securely.

3.2.7.4 Outputs
Successful account management requests result in updated user information or the deletion of the user account, respectively.

3.2.7.5 Error Handling
In the case of failed account management operations, the system should inform users about the issue and provide guidance or support for resolution.

3.3 Use Case Diagram

3.4 Class Diagram

3.5 Non-Functional Requirements

3.5.1 Performance
Application startup shall take less than 5 seconds.
Transitions between different tabs [screens] shall take by maximum 0.5 seconds.
Chatbot responses shall take on average 0.1 to 0.2 seconds, and on maximum in cases such as diagnostic feedback, shall take 2-3 seconds.
Any other functionality shall be done almost instantaneously.
The system shall additionally maintain consistent performance under varying levels of load and user activity. It should be able to handle a large number of concurrent users without significant degradation in response times or system reliability.

3.5.2 Reliability

The chatbot responses are expected to be accurate so as not to mislead the user by giving incorrect diagnoses or advice. This requirement is particularly important as decisions regarding personal health are quite serious. This shall be addressed through thorough research and verified medical sources, while also informing the user of the accuracy of the diagnosis.
The application shall not be liable to errors, bugs, glitches or crashes. If it were prone to one of them, it shall be swiftly updated or backed up to a stable state to ensure reliability for users.

3.5.3 Availability

The chatbot must be available for all users all around the world to use and download, being published on smartphone operating systems` stores.
It is mandatory that the application must be available at all times, and any system failures such as system crashes shall be handled imminently using strategies such as back-up servers.

3.5.4 Security

It is paramount that all application users are assured of the security, privacy and confidentiality of their data that the application acquires and stores, especially since it is sensitive health information. It needs robust security measures to protect against potential data breaches, unauthorized access, and other cybersecurity threats. The database shall have a strong secure basis with the app as a whole being free of any security holes that malicious users could use to find or infer data about our users or other private information, using for example SQL injection.

3.5.5 Maintainability

The application shall be constantly updated to keep up with new changes or advancements in the medical field and to ensure that the information it uses is valid, correct and up-to-date.
The application shall be built with mechanisms for updates and maintenance in place, be updated at set regular intervals and be assigned a dedicated team to oversee this process.

3.5.6 Portability

The Medhat application shall be able to migrate to other platforms easily, as it shall be developed using platform-independent frameworks such as Flutter, which will facilitate porting to other platforms/operating systems.

3.5.7 Scalability

As the user base expands, the application's infrastructure needs to scale accordingly to ensure optimal performance and responsiveness. Scalability challenges may arise, especially during peak usage periods and have to be planned for and accommodated accordingly.

3.5.8 Usability

The application's user interface shall adhere to usability standards, ensuring that users can easily navigate through the features without any confusion. It should include intuitive design elements, clear how-to instructions, and user-friendly interactions to enhance the overall user experience. It shall also accommodate for the varying levels of diversity within its user base.

3.5.9 Accessibility

Medhat aims to be accessible to users with disabilities by following accessibility guidelines. The application shall provide features such as text-to-speech functionality, adjustable font sizes, and other accessibility features to accommodate users with varying needs.

3.5.10 Compliance

The application shall comply with relevant legal and regulatory standards in the healthcare and technology sectors. This includes adherence to data protection laws, medical data handling regulations, and any other applicable standards to ensure ethical and lawful operation.

3.5.11 Crash Recovery

In the event of a system-wide failure or unexpected data loss, the application should have a robust failure recovery plan. This includes regular data backups and an off-site storage to minimize downtime and data loss.

3.5.12 User Feedback

The application should include a user feedback mechanism, allowing users to provide input, report issues, and suggest improvements. This feedback loop will be instrumental in continuous improvement, addressing user concerns, and refining the application over time.

3.6 Inverse Requirements

1. *The chatbot was enriched with a vast database of different disease so that the user would never receive the message "we couldn't find a diagnosis"when conversing with the chatbot , it will either find a diagnosis or keep on asking user whether they have experienced different symptoms with the user answering with a yes or no until a diagnosis is reached*

2. *User should not be allowed to create an account if they are under the age of 18 as Medhat has the ability to diagnose only adults*

3. *Medhat operates as a health information platform and should not do so as a substitute for professional medical practitioners. The system should refrain from acting in that manner, explicitly avoiding definitive diagnoses, prescription recommendations and such.*

3.7 Design Constraints

As with any application in the medical field, the Medhat application is required to operate under a set of requirements and constraints associated with the medical field and public health in general. This includes but is not limited to

- *Regulatory Compliance*
  The Medhat application must adhere to medical data privacy regulations (e.g., HIPAA in the United States). It must also be compliant with local and international healthcare standards.
- *Security Requirements*
  Implementation of robust authentication and authorization mechanisms to protect patient data while encrypting sensitive data in storage.
- *Data Privacy and Confidentiality*
  Strict adherence to patient confidentiality and privacy policies.
- *Data Availability*
  Our database design and entity-relationship diagram was constrained by the lack of available datasets, which prevented some entities from having more attributes, more relationships etc..

3.8 Logical Database Requirements

Regarding the DSDB, it is statically stored within the application, therefore, all logical requirements regarding databases in the application refer to the logical correctness of the queries used.

See Medhat Presentation for ERD and Class Diagram.

4. Analysis Models

4.1 Sequence Diagram

See Medhat Presentation.

4.2 State-Transition Diagram (STD)

See Medhat Presentation.

5. Change Management Process

The main change management process used for the software system is change avoidance which will be followed by preparing mockups of how the system should look like and implementing a prototype that facilitates discussion and unravels any potential issues that would have required accommodating change within development otherwise, and it acts as an exploratory endeavor for any potential changes that would otherwise be unrecognized before the implementation stage, and within the implementation stage, they serve as benchmarks that help guide development and measure its success.
This is already enacted initially by an already present prototype of the chatbot on OCaml, and throughout the specification and implementation phases, multiple prototypes were created, each done so as to improve upon the last in an incremental manner.

The incremental and iterative process of prototype and functionality development acts as a secondary change management process which is change tolerance, as any sort of needed modification doesn't set back development as a whole, but only the minimal part that needs that change.

This is reinforced by the parallel development of the distinct components within the software system, namely the medical database, the chatbot and the app UI. Each advances separately to ensure one does not impede the progress of another.

The development team is divided upon those components incentivizing the concepts of pair programming as these teams automatically integrate the process of reviewing within development, which reduces any issues that could later lead to changes to the system.
These teams communicate with each other regularly, presenting any changes implemented in one component which could drastically help to avoid changes made in other components, and more importantly allows the step of integrating these components -with all teams involved- to become as smooth as possible with minimal to no changes needed.

Complementing this dynamic process is documenting any changes or improvements made within the software, which serves as a comprehensive record capturing the evolution steps of the system. This aspect is critical for roll-back and back-up strategies in the event of contingencies, it helps the development team to seamlessly retrace their steps and restore the system to a previous stable state. Any change shall be documented after it is thoroughly tested, approved by management and discussed by the other teams.