ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM
AFFILIATED TO MAHATMA GANDHI UNIVERSITY, KOTTAYAM

PROJECT REPORT ON

INVENTORY MANAGEMENT SYSTEM

In partial fulfillment of the requirements for the

Award of the degree of

B Voc SOFTWARE DEVELOPMENT

By

KRISHNA VENI K.B

III B Voc Software

Development Register No:

VB22SWD022

Under the guidance of

Ms. NITHYA.B
Asst. Professor

Department of Computer Applications

2022-2025

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM
AFFILIATED TO MAHATMA GANDHI UNIVERSITY, KOTTAYAM

PROJECT REPORT ON

INVENTORY MANAGEMENT SYSTEM

In partial fulfillment of the requirements for the

Award of the degree of

B Voc SOFTWARE DEVELOPMENT

By

KRISHNA VENI K.B

III B Voc Software

Development Register No:

VB22SWD022

Under the guidance of

Ms. NITHYA A.B
Asst. Professor

Department of Computer Applications

2022-25

CERTIFICATE

This is to certify that the project report on INVENTORY MANAGEMENT SYSTEM is a bonafide record of the work done by KRISHNA VENI K.B (VB22SWD022) during the year 2022-2025 and sub- mitted in partial fulfillment of the requirement for the degree of B.Voc Software Development under Mahatma Gandhi University.

Submitted for end semester exam held on ……

Internal examiner Head of Department External examiner

Date:

DECLARATION

I, KRISHNA VENI K.B (Register no: VB22SWD022), BVoc Software Development final year student of St. Teresa's College (Autonomous), Ernakulam, hereby declare that the project submitted named INVENTORY MANAGEMENT SYSTEM for the Bachelors of Vocation Degree in Software Development is my original work. I further declare that the said work has not previously been submitted to any other univer- sity or academic body.

ST TERESA'S COLLAGE
17-03-2025

KRISHNA VENI K.B

ACKNOWLEDGEMENT

First and foremost, I would like to thank God almighty for the suc- cessful completion of my project. I express my sincere thanks to Pro- vincial Superior and Manager Rev. Dr. Sr. Vinitha CSST, Rev. Sr. Emeline CSST and Principal Dr. Alphonsa Vijaya Joseph of St. Teresa's college (AUTONOMOUS) for giving me an opportunity to undertake this project. I express my sincere gratitude towards the Head of the department Ms. SHEEBA EMMANUEL and, I would like to extend my heartfelt appreciation to Ms. JANEENA SHAJU
(Asst. Prof), my project guide for her constant support which helped in the successful completion of my project. I'm grateful to all the faculties of the Department of Computer Applications for their unwavering support and guidance throughout this journey. Finally, I extend my sincere thanks to my parents and friends and all those who directly or indirectly contributed to the realization of this project.

KRISHNA VENI K.B

SYNOPSIS

Our AI-powered medical advisor chatbot integrates seamlessly into the healthcare landscape, comprising three essential modules: Admin, Pharmacists, and Users. The admin module oversees the registration and monitoring of pharmacists and their shops. Pharmacists utilize the platform to update their medicine stock, ensuring real-time accuracy. Users engage with an AI chatbot for medical advice, receiving personalized recommendations for diseases and medication, along with convenient access to nearby pharmacies for their prescribed medicine needs.

CONTENTS

1. INTRODUCTION

1. INTRODUCTION

The AI-Powered Medical Advisor report is a high-level system intended to aid patients and medical professionals in making well-informed medical choices. One of its most prominent features is that the system can process medical information input by the user, including symptoms and medical history, with the use of advanced algorithms and machine learning methods to provide personalized medical advice and recommendations. Among its most important features are an easy-to-use interface for inputting medical information, a strong data analysis process for processing the information, and a recommendation process that identifies potential diagnoses and what to do next. The system also contains a feedback process, where users are able to give opinions on the advice provided, and this enhances the accuracy of the system over time. Directed towards healthcare professionals, developers, and stakeholders, the report is focused on the necessity of integrating AI technologies into medicine to enhance patient care and the efficiency of healthcare processes, with the ultimate goal of achieving better health outcomes.

1.1 ABOUT PROJECT

This project aims to streamline service management processes by developing a user-friendly system for efficiently creating, tracking, and managing service requests related to waste management in urban areas. The focus is on collecting recyclable waste items, such as newspapers, glass, plastic, steel/tin cans, clothes, and leather, while also creating a marketing space for selling recycled products. By automating various tasks, the system enhances efficiency, simplifies operations, and promotes sustainable waste management practices.

2. SYSTEM ANALYSIS

2.1 INTRODUCTION

In the realm of healthcare innovation, our project endeavors to introduce an AI-powered medical advisor chatbot, seamlessly connecting users with local pharmacists. This comprehensive software is designed with three distinct modules catering to administrators, pharmacists, and users, fostering a dynamic ecosystem for efficient medical assistance and pharmaceutical services

2.2 EXISTING SYSTEM

The current healthcare landscape often involves manual or basic digital processes for managing pharmacy inventories, which can lead to inaccuracies. Patients usually receive medical advice through direct consultations or general health websites, lacking personalization and immediacy. Additionally, finding pharmacies typically require separate searches without any integrated systems connecting medical advice to pharmacy locations.

LIMITATIONS OF EXISTING SYSTEM:

1. Present system does not update inventory quantities in real time
2. Relying on manual books causes chances of incorrect data input
3. Creating detailed budget, audit, or procurement reports takes a long time and hard work
4. Difficulty in Dealing with Huge Inventories

2.3 PROPOSED SYSTEM

The proposed system introduces a cohesive framework with three interconnected modules: Admin, Pharmacists, and Doctors. The admin module oversees the registration and monitoring of pharmacists and their shops to ensure compliance and accuracy. Pharmacists use their module to update medicine stock in real time, improving the reliability of stock data. The doctor-focused module features an AI chatbot that provides personalized medical advice and medication recommendations. This chatbot also helps users find nearby pharmacies where they can obtain their prescribed medicines, seamlessly connecting medical consultation with action.

2.4 SYSTEM SPECIFICATION

The AI-Powered Medical Advisor system specification defines a full framework to deliver personalized medical advice through sophisticated artificial intelligence and machine learning technologies. The system comes with an easy-to-use interface for patients and healthcare personnel to input medical history and symptoms without hassle. The system processes the data through sophisticated algorithms to analyze and create personalized medical advice, possible diagnoses, and recommended actions. It also comes with a feedback system to enable users to give feedback about their experience, which is used to enhance the accuracy and efficiency of the system over time. The system architecture also consists of secure data storage to maintain the privacy of users and healthcare legislation and the incorporation of a medical knowledge base to deliver updated information and knowledge. The system as a whole is designed to improve patient care by linking healthcare personnel with users through timely and accurate medical suggestions.

2.5 OPERATING SYSTEM

The AI-Powered Medical Advisor is designed to operate across different platforms for universal access of the users. It is supported by desktop operating systems such as Windows 10 and above, macOS Mojave (10.14) and above, and mainstream Linux systems such as Ubuntu and Fedora. On mobile devices, the software operates on iOS 12 and above, and Android 8.0 (Oreo) and above. The system is also accessible over the web by means of next-generation web browsers such as Chrome, Firefox, Safari, and Edge, and as such may be easily accessed through any operating system. The project also makes use of cloud infrastructure, operating on cloud platforms such as AWS, Azure, and Google Cloud to store information and execute processes for additional scalability and performance and being independent of the local operating system.

2.6 LANGUAGE OR SOFTWARE PACKAGE

The AI-Powered Medical Advisor is built by a combination of programming languages and packages that sustain its operation and effectiveness. The core application is written primarily in Python, utilizing its extensive libraries in data processing and machine learning, including TensorFlow and scikit-learn, to run advanced algorithms in processing medical data. The user interface utilizes React or Angular frameworks in having a responsive and user-friendly web application, whereas the mobile applications utilize Swift for iOS and Kotlin for Android. The system also employs SQL or NoSQL databases, such as PostgreSQL or MongoDB, in facilitating effective storage and retrieval of data. Secure communication and data confidentiality are implemented using various security protocols and libraries by the application. In totality, this combination of languages and software packages enables the AI-Powered Medical Advisor to render precise, user-friendly, and secure medical advice.

2.7 HARDWARE & SOFTWARE SPECIFICATIONS

- Hardware Specifications:
  - Processor:
    - Minimum: Intel i5 or equivalent
    - Recommended: Intel i7 or equivalent
  - RAM:
    - Minimum: 8 GB
    - Recommended: 16 GB or more
  - Storage:
    - Minimum: 256 GB SSD
    - Recommended: 512 GB SSD or higher
  - Graphics Card:
    - Minimum: Integrated graphics
    - Recommended: Dedicated GPU (NVIDIA GTX 1050 or equivalent)
  - Network:
    - Minimum: Ethernet or Wi-Fi 802.11n
    - Recommended: Wi-Fi 802.11ac or higher
- Software Specifications:
  - Operating System:
- Minimum: Windows 10
- Hardware Specifications:
- Minimum: Intel i5 or equivalent
- Recommended: Intel i7 or equivalent

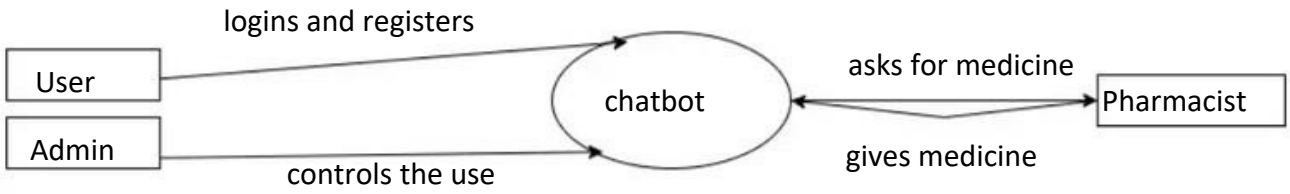
3. SYSTEM DESIGN

3.1 INTRODUCTION

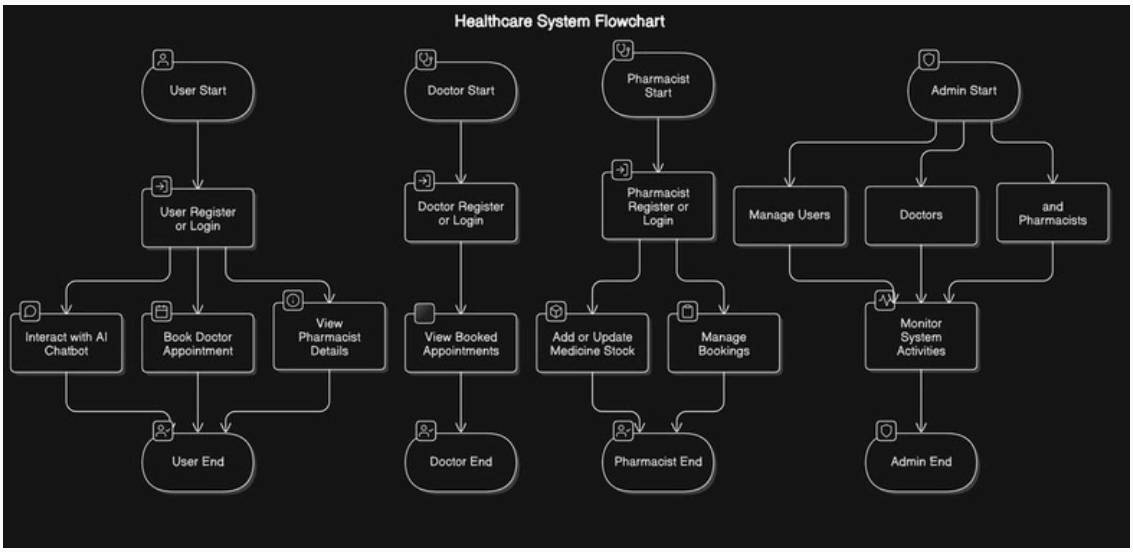
This paper introduces the most critical hardware and software specifications for the seamless operation of the AI-Powered Medical Advisor system to assist healthcare professionals in diagnosing and treating patient care with the assistance of advanced algorithms and machine learning. Adherence to the specifications is mandatory to facilitate seamless operation, efficient data processing, and an improved user experience without compromising system security and integrity. This paper, for healthcare organizations, IT professionals, and software developers, is a guidebook to facilitate the effective implementation and customization of the AI-Powered Medical Advisor, ultimately leading to improved patient outcomes and streamlining healthcare operations.

3.2 DATA FLOW DIAGRAM

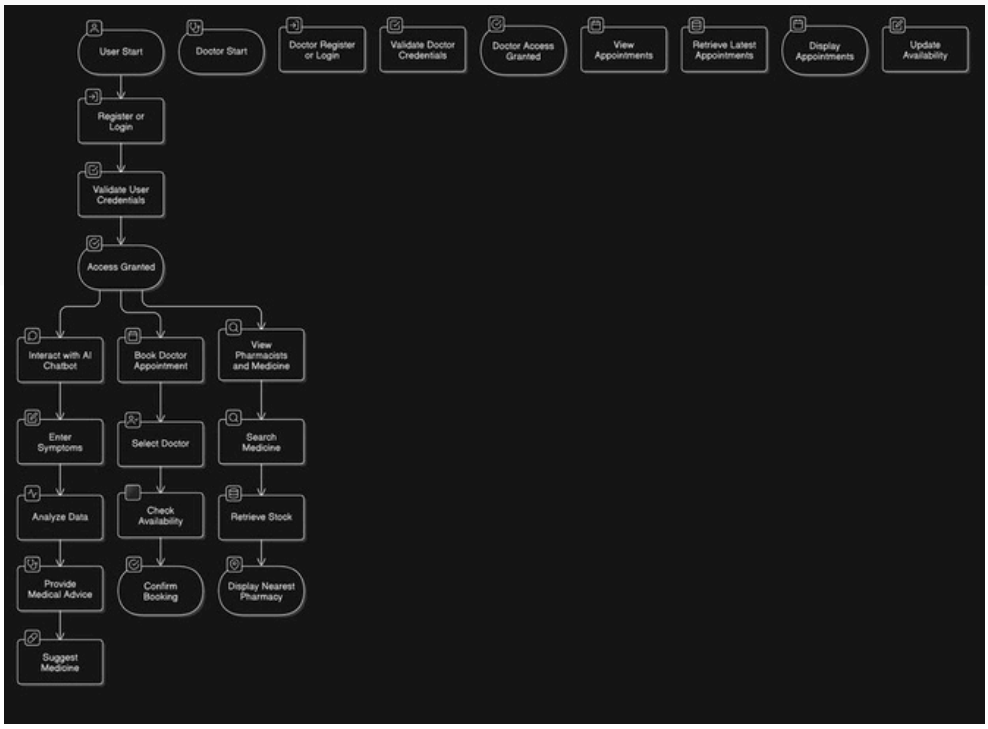
Context level diagram



Level 1 diagram



Level 2 diagram



3.3 DATABASE DESIGN

Database design, A most important part of the system design phase. In a da- tabase environment, data available are used by several users instead of each program managing its own data, authorized users share data across application with the database software managing the data as an entity. A database is a col- lection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information ac- cess easy, quick, inexpensive, and flexible for the users. The general theme behind a database is to integrate all information. Database design is recognized as a standard of management information system and is available virtually for every computer system.

TABLE DESIGN

1. Users Table (Stores user information)

Column Name	Data Type	Constraints	Description
user_id	INT (PK)	AUTO_INCREMENT	Unique ID for each user
name	VARCHAR(100)	NOT NULL	Full name of the user
email	VARCHAR(100)	UNIQUE, NOT NULL	Email address
password	VARCHAR(255)	NOT NULL	Encrypted password
phone	VARCHAR(15)	UNIQUE, NOT NULL	Contact number
address	TEXT	NULLABLE	User's address
created_at	TIMESTAMP	DEFAULT CURRENT_TIME	Account creation timestamp

2. Doctors Table (Stores registered doctors)

Column Name	Data Type	Constraints	Description
doctor_id	INT (PK)	AUTO_INCREMENT	Unique ID for each doctor
name	VARCHAR(100)	NOT NULL	Doctor's full name
email	VARCHAR(100)	UNIQUE, NOT NULL	Email address
password	VARCHAR(255)	NOT NULL	Encrypted password
specialization	VARCHAR(100)	NOT NULL	Medical specialty
phone	VARCHAR(15)	UNIQUE, NOT NULL	Contact number
clinic_address	TEXT	NULLABLE	Clinic location
created_at	TIMESTAMP	DEFAULT CURRENT_TIME	Registration timestamp

3. Pharmacists Table (Stores pharmacist details)

Column Name	Data Type	Constraints	Description
pharmacist_id	INT (PK)	AUTO_INCREMENT	Unique ID for each pharmacist
name	VARCHAR(100)	NOT NULL	Pharmacist's full name
email	VARCHAR(100)	UNIQUE, NOT NULL	Email address
password	VARCHAR(255)	NOT NULL	Encrypted password
pharmacy_name	VARCHAR(100)	NOT NULL	Name of the pharmacy
phone	VARCHAR(15)	UNIQUE, NOT NULL	Contact number
address	TEXT	NOT NULL	Pharmacy location
created_at	TIMESTAMP	DEFAULT CURRENT_TIME	Registration timestamp

4. Medicines Table (Stores medicine details updated by pharmacists)

Column Name	Data Type	Constraints	Description
medicine_id	INT (PK)	AUTO_INCREMENT	Unique ID for each medicine
pharmacist_id	INT (FK)	REFERENCES Pharmacists(pharmacist_id) ON DELETE CASCADE	Pharmacist who added it
name	VARCHAR(100)	NOT NULL	Medicine name
category	VARCHAR(50)	NOT NULL	Category (e.g., Painkiller, Antibiotic)
stock	INT	NOT NULL	Quantity available
price	DECIMAL(10,2)	NOT NULL	Price per unit
expiry_date	DATE	NOT NULL	Medicine expiry date

5. Appointments Table (Stores doctor appointment bookings)

Column Name	Data Type	Constraints	Description
appointment_id	INT (PK)	AUTO_INCREMENT	Unique ID for each appointment
user_id	INT (FK)	REFERENCES Users(user_id) ON DELETE CASCADE	User booking the appointment
doctor_id	INT (FK)	REFERENCES Doctors(doctor_id) ON DELETE CASCADE	Doctor being booked
appointment_date	DATETIME	NOT NULL	Scheduled appointment date/time
status	ENUM('Pending', 'Confirmed', 'Completed', 'Cancelled')	DEFAULT 'Pending'	Current appointment status

4.SYSTEM DEVELOPMENT

4.1 INTRODUCTION

Modular programming is a software design paradigm that involves spitting the functionality of a program into independent, interchangeable units, each containing everything required to run but one portion of the desired functionality. Conceptually, modules are a separation of concerns, and enhance maintainability by imposing logical boundaries between parts.

4.2 PROCESS DESCRIPTION

1. ADMIN MODULE:

- v Admin
  - o Login
  - o View registered users
  - o View registered doctors
  - o View pharmacist

2. USER MODULE:

- o User
  - o Register/ login
  - o View chat bot
  - o Book doctor
  - o View bookings
  - o View pharmacist

3. DOCTOR MODULE:

- Doctor
  - Register / login
  - View latest appointments

4. PHARMACIST MODULE

- Pharmacist
  - Register/ login
  - Add product
  - View products

4. REPORTS AND ANALYTICS Module

The Reports and Analytics module provides valuable insights into system usage, user engagement, and pharmacy stock trends. This module is crucial for admins, doctors, and pharmacists to make data-driven decisions and optimize healthcare services.

. Admin Reports

User Registration Report → Number of users registered over time.

Doctor Performance Report → Total consultations per doctor, average response time.

Pharmacist Activity Report → Medicine stock updates, sales trends.

AI Chatbot Usage Analytics → Most common queries, chatbot response accuracy.

System Load & Performance Report → Peak hours, average response time of AI chatbot.

B. Doctor Reports

Appointments Summary → Number of appointments completed, pending, and canceled.

Patient Demographics → Age, gender, and health conditions distribution.

Most Common Health Concerns → Frequently diagnosed illnesses.

C. Pharmacist Reports

Stock Level Report → Medicines running low in stock.

Medicine Sales Trends → Most purchased medicines, seasonal demand changes.

Expired Medicine Report → List of medicines nearing expiry.

D. User Reports

Consultation History → User's past medical advice and doctor appointments.

Medicine Purchase History → Medicines bought from nearby pharmacies.

Health Trend Analysis → Personalized insights based on chatbot interactions.

2. Data Visualization Tools

To improve report readability, data visualization techniques can be used:

Bar Charts → Total appointments per doctor.

Pie Charts → Disease frequency distribution.

Line Graphs → Monthly growth in user registrations.

Heatmaps → Peak chatbot usage times.

5. SYSTEM TESTING & IMPLIMENTATION

5.1 INTRODUCTION

Software testing is a critical element of software quality assurance and represent the ultimate review of the specification, design, and coding. System testing makes a logical assumption that all parts of the system is correct; the goal will be successfully achieved.

The implementation process enables users to assume control of the system for practical application and assessment. Maintenance involves modifications to the current system, enhancement introduces additional features, and development entails the replacement of the existing system. The implementation phase is characterized by the transition from a newly designed system to an operational state.

5.2 SYSTEM IMPLEMENTATION

Implementation of AI-Powered Medical Advisor involves development of a web-based application using the Django framework (Python) for backend, MySQL/PostgreSQL for database, and HTML, CSS, JavaScript (React.js) for frontend. The system consists of four primary modules: Admin, User, Doctor, and Pharmacist with corresponding functionalities. The AI chatbot is augmented with Natural Language Processing (NLP) for the purpose of giving medical advice based on symptom input. Authentication policies provide secure login and role-based access. The system also includes a real-time stock management facility for the pharmacists for updating medicine stocks. Scheduling of appointments, chatbot management, and search for medicines are performed using APIs optimized for peak performance. Security features such as data encryption, secure API endpoints, and role-based authentication are used to secure the data of the users. Finally, a Reports and Analytics module gives insights into usage of the system, efficiency of the chatbot, and pharmacy trends for ensuring a glitch-free and efficient healthcare service experience.

The implementation plan comprises the following steps:

- Testing the developed system with the sample data.
- Detection and correction of errors.
- Making necessary changes in the system.
- Training and involvement of user personnel.
- Installation of software utilities.

5.3DEBUGGING

1.White Box Testing (Structural Testing)

White box testing examines the internal structure, code, and logic of the system. It is performed by developers or testers with knowledge of the internal workings of the software.

Techniques Used:

Unit Testing: Testing individual components or functions (e.g., chatbot response logic, database queries).

Integration Testing: Ensuring different modules (Admin, Doctor, Pharmacist, AI Chatbot) interact correctly.

Path Coverage Testing: Checking all possible execution paths in the code.

Control Flow Testing: Verifying decision points and loops in the program logic.

2. Black Box Testing (Functional Testing)

Black box testing focuses on testing the system's functionality without knowing its internal code. It evaluates inputs, expected outputs, and system behavior.

Techniques Used:

Boundary Value Analysis: Testing system behavior at input limits (e.g., minimum and maximum characters in input fields).

3. System Security Testing

Security testing ensures the system is resistant to threats like unauthorized access, data breaches, and cyberattacks.

Key Security Tests:

Authentication & Authorization Testing: Ensuring only authorized users can access the admin, doctor, and pharmacist modules.

Data Encryption Testing: Verifying sensitive user data (e.g., medical history, prescriptions) is securely stored and transmitted.

Penetration Testing: Simulating attacks to find vulnerabilities (e.g., SQL injection, cross-site scripting).

5.3.4 SCOPE FOR FUTURE ENHANCEMENT:

Advanced AI Diagnosis:
Integrating machine learning models for more accurate disease prediction based on symptoms, medical history, and user data.
Implementing voice recognition for hands-free interaction.
Integration with Wearable Devices:
Connecting with smartwatches and fitness bands to analyze real-time health data such as heart rate, blood pressure, and oxygen levels.
Providing proactive health alerts based on wearable data.
Telemedicine Support:
Adding video consultation features for direct interaction with doctors.
Allowing digital prescriptions for a more streamlined process.
Blockchain for Data Security:
Implementing blockchain to securely store and share medical data while ensuring privacy and authenticity.
Multi-Language and Regional Support:
Enhancing accessibility by incorporating multiple language options to cater to diverse users.
Medicine Delivery System:
Partnering with local pharmacies for medicine delivery services directly through the app.
AI-Powered Health Assistant:
Adding personalized health recommendations based on user lifestyle and medical history.
Integration with Insurance Providers:
Facilitating seamless medical claims and insurance verification for users.

6. CONCLUSION

CONCLUSION

The AI-powered medical advisor system introduces an efficient and intelligent healthcare solution by integrating AI-driven medical consultation with pharmacy management. By leveraging three key modules—Admin, Pharmacist, and User—the system ensures seamless interaction between patients, medical professionals, and pharmacies.

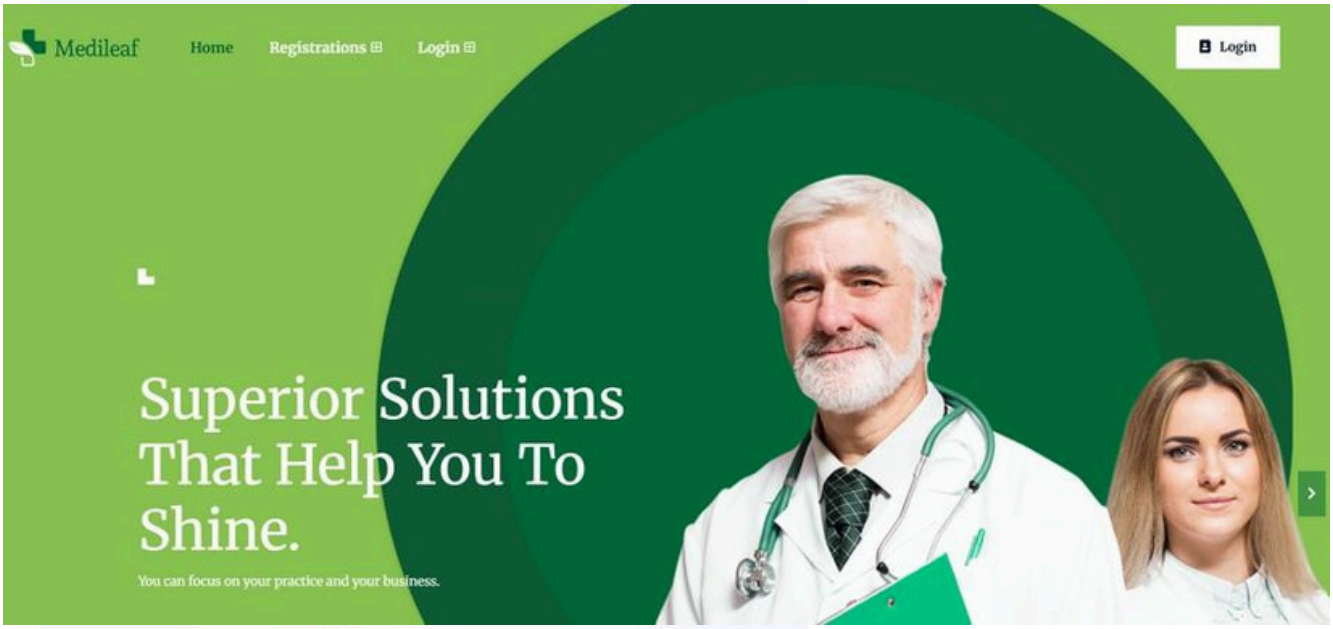
The AI chatbot enhances accessibility to preliminary medical advice, reducing dependency on direct doctor consultations for minor health concerns. Users can conveniently find nearby pharmacies with real-time stock updates, streamlining the process of acquiring prescribed medications. The administrative module ensures proper monitoring and regulation, maintaining accuracy and compliance.

Overall, this system enhances the healthcare experience by offering personalized, quick, and reliable medical assistance, ultimately improving patient care and pharmacy management.

7. APPENDIX

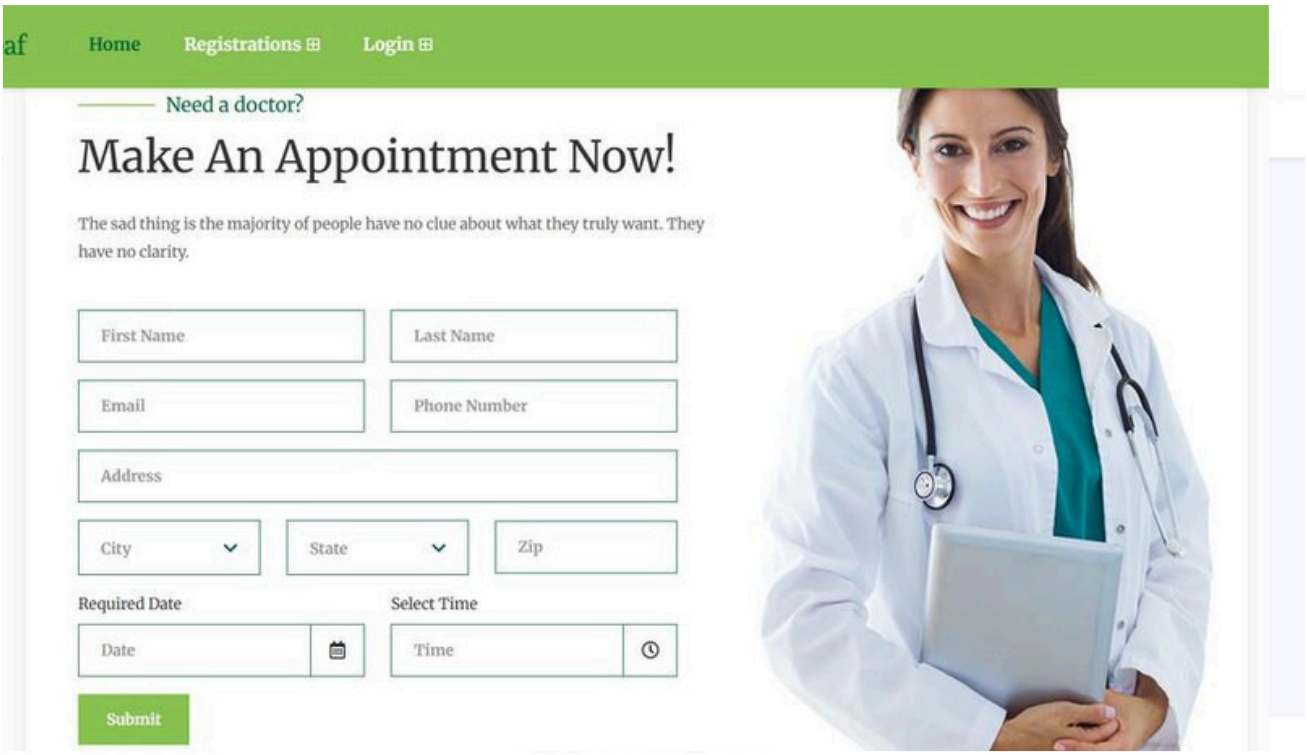
5.4 INPUT AND OUTPUT SCREEN:

ADMIN LOGIN PAGE:



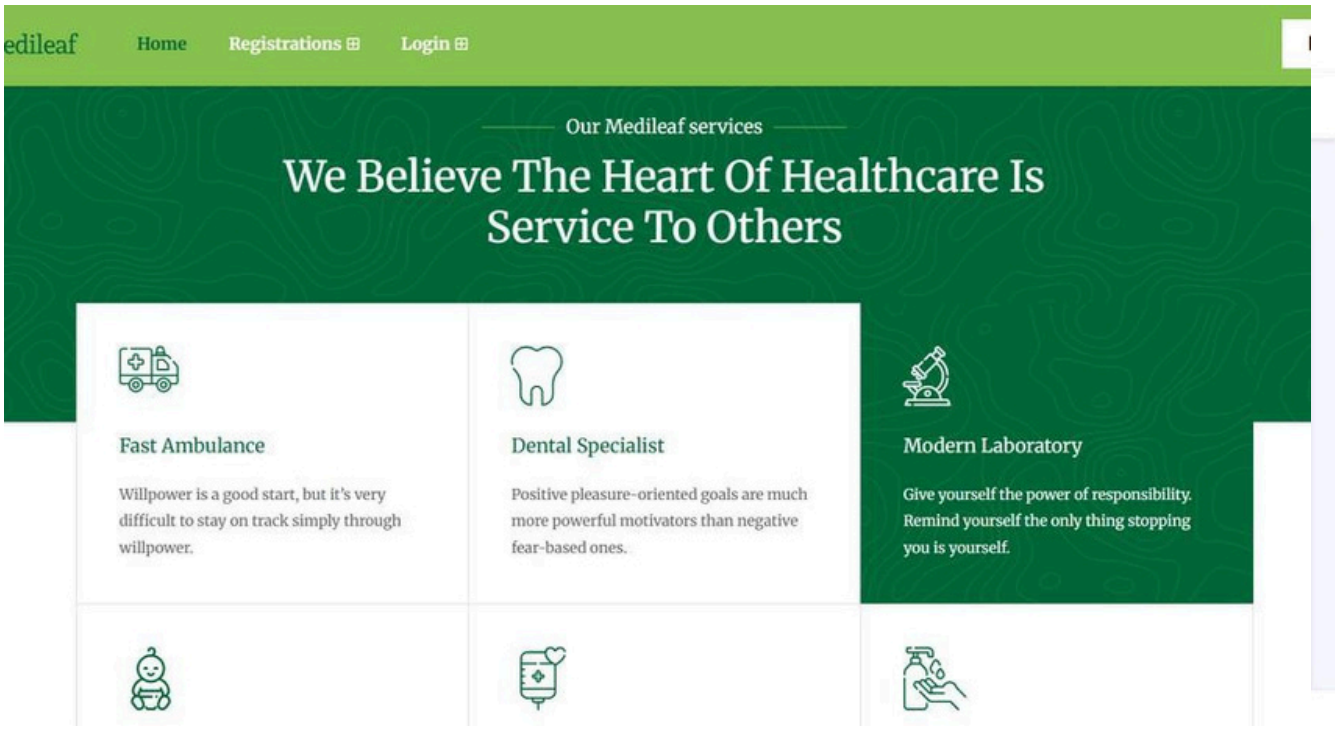
ADMIN HOME PAGE:

ADMIN LOG IN PAGE:

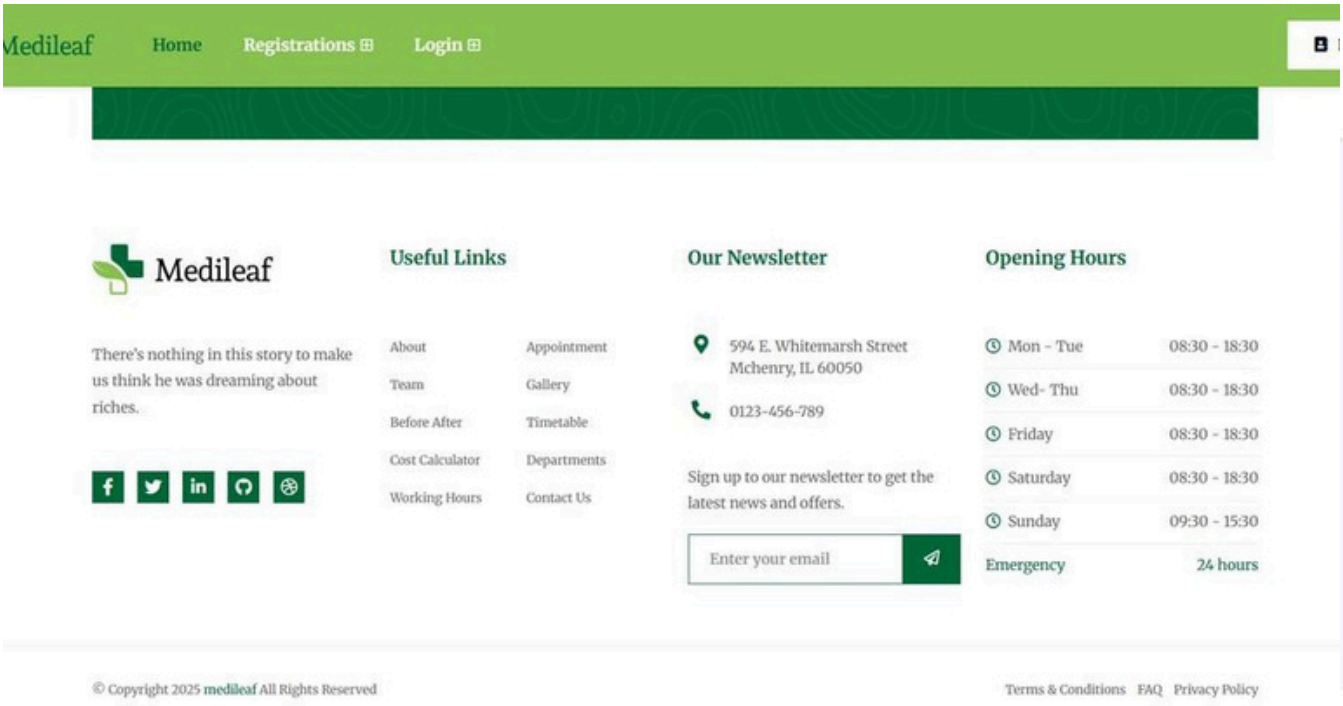


ADMIN NEWS PAGE:

HEALTHCARE PAGE:



USER HELP PAGE:



SAMPLE CODE:

LOGIN:

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from tensorflow.keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))


def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=True):

  # tokenize the pattern
  sentence_words = clean_up_sentence(sentence)
  # bag of words - matrix of N words, vocabulary matrix
  bag = [0]*len(words)
  for s in sentence_words:
    for i,w in enumerate(words):
      if w == s:
        # assign 1 if current word is in the vocabulary position
        bag[i] = 1
        if show_details:
          print ("found in bag: %s" % w)
  return(np.array(bag))
def predict_class(sentence, model):
  # filter out predictions below a threshold
  p = bow(sentence, words,show_details=False)
  res = model.predict(np.array([p]))[0]
  ERROR_THRESHOLD = 0.25
  results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
  # sort by strength of probability
  results.sort(key=lambda x: x[1], reverse=True)
  return_list = []
  for r in results:
    return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
  return return_list
```

```
#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)
    if msg != '':

        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')
        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)




base = Tk() base.title("Hello") base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50",
font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send",
width="12", height=5,


        bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
        command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5",
font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
```

```
"intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi there", "How are you", "Is anyone there?", "Hey", "Hola", "Hello", "Good day"],
      "responses": ["Hello! How can I assist you with your health today?", "Hi there! How can I help with your medical questions?"],
      "context": [""]
    },
    {
      "tag": "symptom_inquiry",
      "patterns": ["I have a symptom", "Experiencing a health issue", "Symptom question", "Health concern"],
      "responses": ["I'm here to help with your health concerns. Please describe your symptoms or health issue, and I'll provide relevant information and guidance."],
      "context": [""]
    },
    {
      "tag": "diagnosis_info",
      "patterns": ["Possible diagnosis", "What could it be?", "Diagnose my symptoms"],
      "responses": ["I can suggest possible diagnoses based on your symptoms, but keep in mind that I'm not a doctor. Please tell me about your symptoms, and I'll provide some potential causes."],
      "context": ["symptom_inquiry"]
    },
    {
      "tag": "treatment_options",
      "patterns": ["Treatment options", "How to treat a condition", "Treatment advice"],
      "responses": ["I can provide information on treatment options for specific health conditions. If you have a particular condition in mind, please let me know, and I'll offer guidance on treatment."],
      "context": [""]
    },
    {
      "tag": "prevention_tips",
      "patterns": ["Preventive measures", "How to stay healthy", "Prevention advice"],
      "responses": ["Preventing health issues is essential. If you'd like tips on staying healthy or preventing specific conditions, please let me know your interests, and I'll provide advice accordingly."],
      "context": [""]
    },
    {
      "tag": "medication_info",
      "patterns": ["Medication information", "Tell me about a drug", "Medication details"],
      "responses": ["I can provide information about medications and their uses. If you have a specific drug in mind, please share the name or describe it, and I'll provide details."],
      "context": [""]
    },
    {
      "tag": "department_recommendation", "patterns": ["Which doctor should I consult?", "Doctor recommendation", "Consultation advice"], "responses": ["Based on your symptoms, it's recommended that you consult a [Department/Type of
Doctor], such as [Specialist Name]. They can provide you with further evaluation and guidance. Would you like more information about this department or doctor?"],
      "context": ["symptom_inquiry"]
    },
    {
      "tag": "cough_treatment",
      "patterns": ["How to treat a cough", "Cough remedies", "Cough treatment options"],
      "responses": ["Coughs can be caused by various factors. Some common treatments for coughs include staying hydrated, using cough drops, and avoiding irritants. If your cough persists or worsens, it's advisable to consult a healthcare professional for proper evaluation and treatment."],
      "context": [""]
    },
```

```
    {
        "tag": "thanks",
        "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
        "responses": ["You're welcome! If you have more health-related questions, feel free to ask.", "No problem! If you need further assistance with your health, don't hesitate to ask."],
        "context": [""]
    },
    {
        "tag": "affirmative",
        "patterns": ["Yes", "Yeah", "Sure", "Absolutely", "Of course"],
        "responses": ["Great! Please let me know how I can assist you with your health.", "Fantastic! How can I help further?"],
        "context": [""]
    },
    {
        "tag": "negative",
        "patterns": ["No", "Not really", "I don't think so", "Sorry, I can't"],
        "responses": ["I understand. If you ever have health-related questions in the future, feel free to reach out.", "No worries. Take care of your health, and don't hesitate to return if you need assistance."],
        "context": [""]
    },
    {
        "tag": "okay",
        "patterns": ["Okay", "OK", "Alright", "Got it", "I understand"],
        "responses": ["Got it! If you have more health-related inquiries, feel free to ask.", "Understood! Don't hesitate to seek health information whenever you need it."],
        "context": [""]
    },
    {
        "tag": "noanswer",
        "patterns": [],
        "responses": ["Sorry, I couldn't understand your health query. Please provide more details or ask your question differently.", "I'm not sure I understand your health concern. Can you please provide more information?"],
        "context": [""]
    },
    {
        "tag": "medication_side_effects",
        "patterns": ["What are the side effects of [Medication Name]?", "Tell me about medication side effects"],
        "responses": ["Medications can have various side effects. I can provide information about the potential side effects of specific drugs. Please specify the medication you're interested in."],
        "context": [""]
```

CHATBOT :

```
# import nltk
# nltk.download('punkt')
# nltk.download('wordnet')
# from nltk.stem import WordNetLemmatizer
# lemmatizer = WordNetLemmatizer()
# import json
# import pickle
# import numpy as np
# from keras.models import Sequential
# from keras.layers import Dense, Activation, Dropout
# from tensorflow.keras.optimizers import SGD
# import random
# words=[]
# classes = []
# documents = []
# ignore_words = ['?', '!']
# data_file = open('intents.json').read()
# intents = json.loads(data_file)




# for intent in intents['intents']: # for
pattern in intent['patterns']:

# # take each word and tokenize it
# w = nltk.word_tokenize(pattern)
# words.extend(w)
# # adding documents
# documents.append((w, intent['tag']))

# # adding classes to our class list
#   if intent['tag'] not in classes:
#      classes.append(intent['tag'])

# words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in
ignore_words]
# words = sorted(list(set(words)))

# classes = sorted(list(set(classes)))

# print (len(documents), "documents")

# print (len(classes), "classes", classes)

# print (len(words), "unique lemmatized words", words)


# pickle.dump(words,open('words.pkl','wb'))
# pickle.dump(classes,open('classes.pkl','wb'))
```

```
# # initializing training data
# training = []
# output_empty = [0] * len(classes)
# for doc in documents:
#     # initializing bag of words
#     bag = []
#     # list of tokenized words for the pattern
#     pattern_words = doc[0]
#     # lemmatize each word - create base word, in attempt to represent related words
#     pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
#     # create our bag of words array with 1, if word match found in current pattern
#     for w in words:
#         bag.append(1) if w in pattern_words else bag.append(0)

#
#     # output is a '0' for each tag and '1' for current tag (for each pattern)
#
#     output_row = list(output_empty)
#     output_row[classes.index(doc[1])] = 1

#
#     training.append([bag, output_row])
# # shuffle our features and turn into np.array
# random.shuffle(training)
# training = np.array(training)
# # create train and test lists. X - patterns, Y - intents
# train_x = list(training[:,0])
# train_y = list(training[:,1])
# print("Training data created")


# # Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer
contains number of neurons
# # equal to number of intents to predict output intent with softmax
# model = Sequential()
# model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
# model.add(Dropout(0.5))
# model.add(Dense(64, activation='relu'))
# model.add(Dropout(0.5))
# model.add(Dense(len(train_y[0]), activation='softmax'))

# # Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good
results for this model'
# # sgd = SGD(learning_rate=0.01, decay_rate=1e-6, momentum=0.9, nesterov=True)
# sgd = SGD( initial_learning_rate=0.01,
# decay_steps=10000,
# decay_rate=0.9)
# model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# #fitting and saving the model
# hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
# model.save('chatbot_model.h5', hist)

# print("model created")
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers.schedules import ExponentialDecay
import random

words = []
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)

for intent in intents['intents']:
    for pattern in intent['patterns']:
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        documents.append((w, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
classes = sorted(list(set(classes)))

print(len(documents), "documents")
print(len(classes), "classes", classes)
print(len(words), "unique lemmatized words", words)

pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))

training = []
output_empty = [0] * len(classes)
```

```
# Create model
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# Fit and save the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_model.h5')

print("Model created")
```

8.BIBILOGRAPHY

8.1 REFERENCES

· Books & Journals

Smith, J. (2020). Artificial Intelligence in Healthcare: Applications and Challenges. Springer.

Brown, L., & Green, K. (2019). Medical Chatbots and AI-driven Health Assistants. Oxford University Press.

· Research Papers & Articles

Patel, R., & Sharma, P. (2021). "AI-Based Chatbots in Medical Consultation: A Review." International Journal of Healthcare Technology, 15(4), 112-125.

Wong, T., & Gupta, S. (2022). "Pharmacy Automation and AI in Medicine Dispensing." Journal of Digital Health, 10(2), 89-105.

· Web Sources

World Health Organization (WHO). (2023). "The Role of AI in Modern Healthcare." Retrieved from www.who.int

National Institutes of Health (NIH). (2023). "AI and Digital Health Innovations." Retrieved from www.nih.gov

· Software Documentation

Django Framework Documentation. Retrieved from https://docs.djangoproject.com

Python Official Documentation. Retrieved from https://www.python.org/doc