

ADVANCED HEALTHCARE CHATBOT FOR DISEASE DIAGNOSIS AND TREATMENT RECOMMENDATION

**A Project report submitted in partial fulfilment of the requirements
For the award of the degree of**

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING - (DATA SCIENCE)

Submitted by

LODAGALA TEJA - 20Q71A4417

RATNALA MANASA - 20Q71A4427

VANAPALLI PRIYANKA - 20Q71A4449

KONNA RAVI TEJA – 21Q75A4403

Under the esteemed Guidance of

Dr. A CHANDRASEKHAR

Director - HR, Professor



AVANTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – (DATA SCIENCE)

(Approved by A.I.C.T.E., New Delhi, & Permanently Affiliated to JNTU-GV, Vizianagaram

Accredited by NAAC with “A+” Grade)

Cherukupalli (Village), Near Tagarapuvalasa Bridge, Bhogapuram (Mandal), Vizianagaram (Dist)

ADVANCED HEALTHCARE CHATBOT FOR DISEASE DIAGNOSIS AND TREATMENT RECOMMENDATION

**A Project report submitted in partial fulfilment of the requirements
For the award of the degree of**

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING - (DATA SCIENCE)

Submitted by

LODAGALA TEJA - 20Q71A4417

RATNALA MANASA - 20Q71A4427

VANAPALLI PRIYANKA - 20Q71A4449

KONNA RAVI TEJA – 21Q75A4403

Under the esteemed Guidance of

Dr. A CHANDRASEKHAR

Director - HR, Professor



AVANTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – (DATA SCIENCE)

(Approved by A.I.C.T.E., New Delhi, & Permanently Affiliated to JNTU-GV, Vizianagaram

Accredited by NAAC with “A+” Grade)

Cherukupalli (Village), Near Tagarapuvalasa Bridge, Bhogapuram (Mandal), Vizianagaram (Dist)

AVANTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – (DATA SCIENCE)

(Approved by A.I.C.T.E., New Delhi, & Permanently Affiliated to JNTU-GV, Vizianagaram

Accredited by NAAC with “A+” Grade)

Cherukupalli (Village), Near Tagarapuvalasa Bridge, Bhogapuram (Mandal), Vizianagaram (Dist)



CERTIFICATE

This is to certify that the project report titled **ADVANCED HEALTHCARE CHATBOT FOR DISEASE DIAGNOSIS AND TREATMENT RECOMMENDATIONS** is being submitted by **L. TEJA** bearing **20Q71A4417**, **R. MANASA** bearing **20Q71A4427**, **V. PRIYANKA** bearing **20Q71A4449**, **K. RAVI TEJA** bearing **21Q75A4403** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in CSE- (Data Science) of Avanthi Institute of Engineering and Technology, Vizianagaram is a record of Bonafede work carried out under my guidance and supervision.

Internal Guide

Dr. A Chandrasekhar.

Director - HR, Professor

CSE Dept.

Head of the Department

Mrs. P Srilakshmi,

M. Tech, (Ph.d)

CSE (DS, AI & ML) Dept.

External Examiner:

ACKNOWLEDGEMENT

This report would be incomplete without mentioning certain individuals whose guidance and encouragement have been of immense help to complete this Thesis. It is difficult to acknowledge all of them. we can only mention some of them here.

We acknowledge our deep sense of gratitude to our beloved Chairman **Mr. Muttamsetti Srinivasa Rao, AVANTHI EDUCATIONAL INSTITUTIONS** for giving us a chance to do B.Tech.

I am highly indebted to express our grateful thanks to **Dr. B. Murali Krishna, Principal of Avanthi Institute of Engineering and Technology**, for the overall valuable support he provided during the project work and also during our course study.

We take the privilege to express our profound sense of gratitude to **Mrs. P Srilakshmi, Head of the Department, Computer Science and Engineering with Specialization Data Science, Artificial Intelligence & Machine Learning of Avanthi Institute of Engineering and Technology**.

We extend a heartfelt thanks to **Dr. A Chandrasekhar, Director - HR & Professor, Computer Science and Engineering Department of Avanthi Institute of Engineering and Technology** for his cooperation and for the guidance throughout the Project.

We sincerely thank all the Staff Members of the Department for giving us their support for the completion of this project.

We would like to express our sincere thanks to our parents who encouraged us throughout our educational endeavours and to do this project work.

Finally, we would like to express our thanks to the library staff, who have supported us in the accomplishment of the project.

DECLARATION

We hereby declare that the project work entitled **ADVANCED HEALTHCARE CHATBOT FOR DISEASE DIAGNOSIS AND TREATMENT RECOMMENDATIONS** submitted to the JNTU-GV is a record of an original work done by **L. TEJA (20Q71A4417), R. MANASA (20Q71A4427), V. PRIYANKA (20Q71A4449), K. RAVI TEJA (21Q75A4403)** under the esteemed guidance of **Dr. A Chandrasekhar, Director - HR & Professor, Computer Science and Engineering Department of Avanthi Institute of Engineering and Technology**. This project work is submitted in the partial fulfilment of the requirements for the award of the degree Bachelor of Technology in Computer Science & Engineering with Specialization Data Science. This entire project is done with the best of our knowledge and is not submitted to any university for the award of degree/diploma.

LODAGALA TEJA
20Q71A4417

RATNALA MANASA
20Q71A4427

VANAPALLI PRIYANKA
20Q71A4449

KONNA RAVI TEJA
21Q75A4403

Institute Vision and Mission

Vision and Mission of the Institute

Vision:

To develop highly skilled professionals with ethics and human values.

Mission:

1. To impart quality education with industrial exposure and professional training.
2. To produce competent and highly knowledgeable engineers with positive approach.
3. To self confidence among students which is an imperative prerequisite to face the challenges of life.

COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Vision and Mission of the Department

Vision:

To emerge as recognized center through strong research and teaching environment to address technological needs of the stakeholders.

Mission:

M1: Provide quality education in both the theoretical and applications of computer science and apply education to solve real-world problems.

M2: Conduct research to advance the state of the art in computer science and integrate research results and innovations into other scientific disciplines.

M3: Graduates with ability to solve complex problems that address societal needs and develop appropriate computer Programs with latest tools and technologies.

M4: Graduates with ability to pursue advanced education, research, other creative and innovative efforts in computer science & engineering for successful career.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Graduates are prepared to apply analysis, predictions, optimization, decision making and develop skills in order to formulate and solve complex problems in the area of Data Science.

PEO2: Graduates are prepared to take up higher studies, research & development and other creative efforts in the area of Data Science which drives scientific and societal advancement through technological innovation and entrepreneurship.

PEO3: Graduates are prepared to use their skills and abilities in an ethical & professional manner.

PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1: Apply data science concepts and methods to solve real world problems.

PSO2: Design and develop statistical models for data preparation, exploration, visualization, and governance.

PSO3: Adapt to a rapidly changing environment by learning and employing emerging software tools and technologies in the area of Data Science.

PROGRAM OUTCOMES (POS)

PO1: Engineering Knowledge: Apply the knowledge of Mathematics, Science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of Mathematics, Natural Sciences, and Engineering Sciences.

PO3: Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long Learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

Health care has grown to be a vital component of our living in the modern world. To see a doctor for advice on every health issue, meanwhile, has grown to be exceedingly challenging. The goal of this project is to create a medical chatbot using artificial intelligence to help with problem diagnosis. The project's specific objective is to create a medical chatbot powered by artificial intelligence that can identify health issues, provide an overview of the ailment we have, and recommend us to a doctor for additional treatment. Since there will be no longer be a need to visit a doctor for minor health issues, this project will lead to lower medical costs. The bot will also function more effectively because it makes use of Natural Language Processing (NLP) to communicate with users and preserve their data. The chatbot can enhance itself using background information to respond to queries more quickly in the future. Depending on your needs, you may install the chatbot anywhere, whether on a site or in an app. The effectiveness of medical chatbots rest on Natural language processing technique that helps users post their concerns about any disease and also their health. The user can ask any private question related to health care via chatbot without being actually present at the clinic or hospital. This will help reduce the cost of health care and improve access to medical information through medical chat-bot. The program development plan is to analyse customer feelings.

- **Keywords – Artificial Intelligence, NLP, Dialogflow, Chatbot, Symptoms, Healthcare.**

LIST OF CONTENTS

S.NO	TITLE	PAGE NO
1	INTRODUCTION 1.1 Introduction 1.2 Project Overview 1.3 Project Objective 1.4 Project Scope	
2	LITERATURE SURVEY	
3	PROBLEM ANALYSIS 3.1 Existing System 3.1.1 Limitations 3.2 Proposed System 3.2.1 Advantages	
4	SYSTEM ANALYSIS 4.1 Software Requirements Specification 4.1.1 Functional Requirements 4.1.2 Non - Functional Requirements 4.2 Feasibility Study 4.3 Use Case Scenario 4.4 System Requirements 4.4.1 Software Requirements 4.4.2 Hardware Requirements	
5	SYSTEM DESIGN 5.1 Introduction 5.1.1 Sequence Diagram 5.1.2 Activity Diagram 5.2 System Architecture 5.3 Dialogflow Description	
6	IMPLEMENTATION 6.1 Technology Description 6.2 About the Python Programming 6.2.1 Features of Python 6.3 Libraries Used	

	6.4 Dialogflow Console 6.5 Source Code	
7	TESTING 7.1 Introduction 7.2 Test Cases	
8	RESULTS	
9	CONCLUSION	
10	BIBLIOGRAPHY	

CHAPTER - 1

INTRODUCTION

INTRODUCTION

An artificial intelligence (AI) chatbot is a computer programme that mimics human communication. It is a piece of software that communicates with humans using written language. It is frequently embedded in web pages or other digital applications to answer customer inquiries without the need for human agents, resulting in low-cost and hassle-free customer service. Chatbots based on machine learning produce an AI chatbot that is very capable of having an organic conversation with the user and answering their queries. Chatbots use the data that is provided to them to answer queries as accurately as possible using various training algorithms. In our proposed system we create a conversational chatbot that is integrated into a hospital website.

It is trained with machine learning algorithms and serves as an extremely efficient interface between the user and the application. There is no predefined format in which users can ask their questions. The chatbot responds to the query in the best way possible. Users have the option of submitting a query in both text and speech format. Users can use this chatbot to access hospital information, doctor availability, diagnostics, and other related data. They are navigated to different pages according to their requests, which makes it easier and faster for them to explore. They can schedule appointments and identify the problem by describing the symptoms in order to be prepared. This allows them to take any necessary precautions and schedule an appointment with the doctor as soon as possible.

Health care is very significant for you to live a well and healthy life.

Often people do not know all the treatments or symptoms associated with a particular disease. For young people, users with the problem go to the hospital in person for a laborious test and it is very problematic to get a doctor's appointment for any health problems, and managing phone calls.

These types of problems can be solved by using Healthcare/Medicinal Chatbot by providing appropriate guidance regarding healthy living. The notion is to create a medical chatbot using

Neural Networks that can provide with the info and diagnose the disease and deliver basic information about the disease and when and where to consult a doctor.

The effectiveness of medical chatbots rest on Natural language processing technique that helps users post their concerns about any disease and also their health. The user can ask any private question related to health care via chatbot without being actually present at the clinic or hospital. This will help reduce the cost of health care and improve access to medical information through medical chat-bot. The program development plan is to analyze customer feelings.

In order to create chatbots, which replicate natural language discussions with users through messaging applications, websites, mobile apps, and phones, AI software is used. This application is essential since knowledge is prestored, allowing for self-learning and knowledge restoration via human or online resources. As the system application takes the shape of a chatbot, it responds to user requests using the question-and-answer protocol. Since consumers cannot quickly visit a doctor or other specialist, this system was created to reduce healthcare expenses and time for users. The chatbot may analyse the user's questions while they are being asked in order to give the most accurate response. I also have a way to keep track of former users' information so I can ask them for comments when they come back. We require input (Optional) from the user on whether the provided previous answer is correct in order for the bot to learn from the old cases so it can get better for the new ones. How can a chatbot system be made intelligent enough?

There are two ways of doing so:

1. Rule-based chatbots:

In this case, a chatbot system operates in accordance with predetermined rules. However, this Chatbot system is ineffective to provide a solution when the input pattern does not comply with any specified rule. AIML (Artificial Intelligence Markup Language) is used by developers to design the rules for chatbot systems. A language based on XML is AIML. Writing rules for different situations is a very laborious task, and it is challenging to develop rules for every conceivable circumstance. These rule-based Chatbot systems can handle simple inquiries, but it is essential to handling complex inquiries. The majority of social media chatbot platforms rely on rules.

2. Chatbots that can teach themselves new things:

These Chatbots employ machine learning techniques to learn new things. These robots come in two varieties: Using Retrieval Based Models: These bots have been taught to respond to a variety of questions. The bot may select the most significant responses for each question from the collection of all possible responses. The language and phrase structure are also fine because the acceptable responses are predetermined and can never go wrong in a sentence structure sense. When using generative models, keep in mind that they never select the same response from a pool of options. They respond appropriately by quoting the question word for word. Since these models are tolerant of spelling and grammar mistakes, preparation should be more precise.

3. Self-learning chatbots are still in their infancy. These chatbots frequently give irrelevant answers to user questions, which might irritate corporate clients. Because of this, organizations and sectors now favor rule-based chatbot systems. This article focuses on a number of issues relating to the creation, development, and use of rule-based chatbot systems.

The following are the key contributions of this document.

- The various existing rules-based chatbot systems are critically analysed in this research.
- The paper addresses challenges associated with developing chatbot systems and directs businesses toward selecting the best chatbot platform and framework.
- The two most well-known rules-based chatbot systems, Google Dialog flow and IBM Watson, are the subjects of case studies in this paper. Based on a number of criteria, the document contrasts the two frameworks.

The format of this document is as follows:

The history of chatbot development is covered in Section II.

The implementation platforms and development frameworks for chatbots are discussed and contrasted in Section III.

The metrics used to gauge the chatbot system's performance are described in section IV.

Section V discusses the specifics of Google Dialog flow and the IBM Watson Chatbot system's designs and constraints.

Project Overview:

The project aims to develop an advanced healthcare chatbot for disease diagnosis and treatment recommendations. Leveraging artificial intelligence, natural language processing, and medical knowledge databases, the chatbot will provide personalized and accurate guidance to users regarding their health concerns.

Key Components and Features:

- 1. Symptom Assessment:** The chatbot will engage users in natural language conversations to gather information about their symptoms, using advanced NLP algorithms to interpret and analyse their input.
- 2. Diagnostic Capability:** Based on the symptoms reported by the user, the chatbot will employ a diagnostic algorithm to generate potential diagnoses or narrow down possible medical conditions.
- 3. Treatment Recommendations:** Drawing from a vast database of medical knowledge and guidelines, the chatbot will offer evidence-based treatment recommendations tailored to the user's individual needs, medical history, and preferences.
- 4. Interactive Communication:** The chatbot will engage users in interactive dialogue, guiding them through the symptom assessment process, providing explanations, and clarifying any ambiguities.
- 5. Integration with Healthcare Systems:** The chatbot will be designed to seamlessly integrate with existing healthcare systems, electronic health records (EHRs), and telemedicine platforms, facilitating communication and data exchange with healthcare providers.
- 6. User-Friendly Interface:** The chatbot will feature an intuitive and user-friendly interface accessible via web, mobile, or messaging platforms, ensuring ease of use for patients and healthcare professionals.

Project Phases:

- 1. Research and Requirements Gathering:** Conduct a thorough analysis of user needs, existing healthcare chatbots, and technological requirements.
- 2. Design and Development:** Develop the chatbot's architecture, algorithms, user interface, and integration mechanisms, utilizing state-of-the-art AI and NLP technologies.
- 3. Testing and Validation:** Conduct extensive testing to evaluate the chatbot's accuracy, performance, usability, and security, incorporating feedback from users and healthcare professionals.
- 4. Deployment and Adoption:** Deploy the chatbot across various platforms and healthcare settings and promote user adoption through training sessions and awareness campaigns.
- 5. Maintenance and Continuous Improvement:** Monitor the chatbot's performance, gather user feedback, and implement updates and enhancements to improve its functionality and effectiveness over time.

Project Goals:

1. Improve accessibility to healthcare services by providing a convenient and user-friendly platform for symptom assessment and treatment guidance.
2. Enhance patient engagement and empowerment by offering personalized and evidence-based health recommendations.
3. Streamline healthcare processes and reduce the burden on healthcare resources by automating routine tasks and providing decision support to healthcare providers.
4. Foster collaboration and communication between patients and healthcare professionals, facilitating better-informed healthcare decisions and improved patient outcomes.

By undertaking this project, we aim to contribute to the advancement of healthcare delivery through innovative technology solutions, ultimately improving the health and well-being of individuals and communities.

Project Objective:

The primary objective of the project is to develop an advanced healthcare chatbot for disease diagnosis and treatment recommendations. The chatbot aims to provide accurate, personalized, and timely guidance to users regarding their health concerns, thereby improving healthcare accessibility, efficiency, and patient outcomes. Specifically, the project objectives include:

- 1. Symptom Assessment:** Develop algorithms to accurately assess user-reported symptoms and extract relevant medical information using natural language processing (NLP) techniques.
- 2. Disease Diagnosis:** Implement a diagnostic algorithm that matches user-reported symptoms with potential diseases or conditions based on medical knowledge and guidelines.
- 3. Treatment Recommendations:** Generate evidence-based treatment recommendations tailored to the user's individual needs, medical history, and preferences.
- 4. User Interaction:** Design an intuitive and user-friendly interface for seamless interaction with the chatbot, enabling users to input symptoms, ask questions, and receive personalized recommendations.
- 5. Integration with Healthcare Systems:** Ensure compatibility and integration with existing healthcare systems, electronic health records (EHRs), and telemedicine platforms to facilitate communication and data exchange with healthcare providers.
- 6. Accuracy and Reliability:** Ensure the accuracy and reliability of the chatbot's diagnostic and treatment recommendations through rigorous testing and validation against established medical standards.
- 7. Privacy and Security:** Implement robust security measures to safeguard sensitive medical information and ensure compliance with healthcare regulations (e.g., HIPAA).
- 8. User Adoption and Satisfaction:** Promote user adoption and satisfaction through user training, educational materials, and ongoing support, gathering feedback to continuously improve the chatbot's functionality and usability.

9. Scalability and Performance: Develop a scalable and high-performance chatbot system capable of handling a large volume of user interactions while maintaining responsiveness and reliability.

10. Contribution to Healthcare Innovation: Contribute to the advancement of healthcare delivery through the development and deployment of innovative technology solutions that improve patient care and healthcare outcomes.

By achieving these objectives, the project aims to create a valuable tool that empowers individuals to take control of their health, facilitates more informed healthcare decisions, and enhances the efficiency and effectiveness of healthcare delivery.

Project Scope:

The scope of the project encompasses the development of an advanced healthcare chatbot for disease diagnosis and treatment recommendations. The project will focus on the following key areas:

1. Symptom Assessment: The chatbot will be capable of accurately assessing user-reported symptoms through natural language processing (NLP) techniques, extracting relevant medical information, and understanding the context of user queries.

2. Disease Diagnosis: The chatbot will employ a diagnostic algorithm to match user-reported symptoms with potential diseases or conditions based on medical knowledge and guidelines. It will provide a list of likely diagnoses along with relevant information for further evaluation.

3. Treatment Recommendations: The chatbot will generate evidence-based treatment recommendations tailored to the user's individual needs, medical history, and preferences. It will consider factors such as allergies, medications, and comorbidities in providing personalized guidance.

4. User Interaction: The chatbot will feature an intuitive and user-friendly interface for seamless interaction with users, enabling them to input symptoms, ask questions, and receive personalized recommendations. It will utilize conversational AI techniques to engage users in natural language conversations.

5. Integration with Healthcare Systems: The chatbot will integrate with existing healthcare systems, electronic health records (EHRs), and telemedicine platforms to facilitate communication and data exchange with healthcare providers. It will ensure compatibility with industry standards and protocols for interoperability.

6. Accuracy and Reliability: The chatbot will undergo rigorous testing and validation to ensure the accuracy and reliability of its diagnostic and treatment recommendations. It will be validated against established medical standards and guidelines.

7. Privacy and Security: The chatbot will implement robust security measures to safeguard sensitive medical information and ensure compliance with healthcare regulations (e.g., HIPAA). It will employ encryption, access controls, and auditing mechanisms to protect user data.

8. User Adoption and Satisfaction: The project will include user training, educational materials, and ongoing support to promote user adoption and satisfaction. Feedback from users will be gathered to continuously improve the chatbot's functionality and usability.

The project scope does not include the provision of medical diagnosis or treatment by the chatbot itself. Instead, the chatbot will provide information and recommendations to assist users in making informed decisions about their health and seeking appropriate medical care from qualified healthcare professionals.

By focusing on these key areas within the defined scope, the project aims to deliver a high-quality healthcare chatbot that enhances patient care, improves healthcare accessibility, and contributes to the advancement of healthcare delivery.

CHAPTER - 2

LITERATURE SURVEY

1. LITERATURE SURVEY

TITLE: "Web-based chatbot for frequently asked queries (FAQ) in hospitals"

ABSTRACT: Objectives Local hospitals are operated by the resigned association of patients as passive communication channels. The online hospital data related to the users' queries are not transparent and reliable. Therefore, it is crucial to have an intelligent web chatbot that manages user requests and provides quick access to local hospital information. In this paper, we present a framework and functionality of a chatbot developed using web technologies. Methods The bot engine was integrated by several machine learning approaches like gradient descent (GD) and natural language processing (NLP) algorithms. The trained data entered into the bot were split into mini-word batches, and the GD algorithm was applied sequentially on each mini-batch. The NLP methods involved in converting a word to its stem with a text result less readable by humans. Results The employed ML algorithms were successfully incorporated to manage the alternative synchronisation of text and voice messages. Conclusions The proposed bot can be a better solution for data extraction from local hospitals. It is an insightful communication channel for both users and hospital staff.

TITLE: "Chatbot for disease prediction and treatment recommendation using machine learning"

ABSTRACT: Hospitals are the most widely used means by which a sick person gets medical check-ups, disease diagnosis and treatment recommendation. This has been a practice by almost all the people over the world. People consider it as the most reliable means to check their health status. The proposed system is to create an alternative to this conventional method of visiting a hospital and making an appointment with a doctor to get diagnosis. This research intends to apply the concepts of natural language processing and machine learning to create a chatbot application. People can interact with the chatbot just like they do with another human and through a series of queries, chatbot will identify the symptoms of the user and thereby, predicts the disease and recommends treatment. This system can be of great use to people in conducting daily check-ups, makes people aware of their health status and encourages people to make proper measures to remain healthy. According to this research, such a system is not widely used and people are less aware of it. Executing this proposed framework can help people

avoid the time-consuming method of visiting hospitals by using this free of cost application, wherever they are.

TITLE: "Chatbot for healthcare system using artificial intelligence"

ABSTRACT: Healthcare is very important to lead a good life. However, it is very difficult to obtain the consultation with the doctor for every health problem. The idea is to create a medical chatbot using Artificial Intelligence that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot. The chatbots are computer programs that use natural language to interact with users. The chatbot stores the data in the database to identify the sentence keywords and to make a query decision and answer the question. Ranking and sentence similarity calculation is performed using n-gram, TFIDF and cosine similarity. The score will be obtained for each sentence from the given input sentence and more similar sentences will be obtained for the query given. The third party, the expert program, handles the question presented to the bot that is not understood or is not present in the database.

TITLE: Development of artificial intelligence based chatbot using deep neural network"

ABSTRACT: No matter how well-known colleges are, there will always be concerns that people have during the application process and even after they have been accepted. The college hosts a variety of events, ranging from departmental activities to club activities. Not everyone is likely aware of all events. Chatbot bridges gap between people and information. The world is becoming more automated, and people expect services to become more automated as well. A chatbot is software that responds to user questions and provides information from a knowledge base. The purpose of this project is to create a chatbot for VNRVJIET that will answer queries raised about fests, departmental activities, events, clubs, infrastructure, placement data, admission procedure, and others. The proposed methodology consists of a chatbot built using Deep Neural Networks and speech recognition capabilities. The information is delivered in both speech and text modes using the proposed methodology. Data is collected and formatted in JSON format initially. The prepared data is preprocessed and then the bag of words algorithm is applied to it. The bag of words algorithm is most influential method for

object categorization. The key aspect of using this algorithm is for converting the word vector to a numerical data set for machine to do a deeper analysis. A deep neural network is created using tensor flow API, and the speech recognition function is defined for the input query and output response. Finally, chatbot function is defined and utilized for generating responses for any given query.

TITLE: “ELIZA- A computer program for the study of natural language communication between man and machine”

ABSTRACT: ELIZA is an early natural language processing computer program created from 1964 to 1967[1] at MIT by Joseph Weizenbaum.[2][3] Created to explore communication between humans and machines, ELIZA simulated conversation by using a pattern matching and substitution methodology that gave users an illusion of understanding on the part of the program, but had no representation that could be considered really understanding what was being said by either party.[4][5][6] Whereas the ELIZA program itself was written (originally)[7] in MAD-SLIP, the pattern matching directives that contained most of its language capability were provided in separate "scripts", represented in a lisp-like representation. The most famous script, DOCTOR, simulated a psychotherapist of the Rogerian school (in which the therapist often reflects back the patient's words to the patient),[8][9][10] and used rules, dictated in the script, to respond with non-directional questions to user inputs. As such, ELIZA was one of the first chatterbots ("chatbot" modernly) and one of the first programs capable of attempting the Turing test.

CHAPTER - 3

PROBLEM ANALYSIS

EXISTING SYSTEM

The existing system of the "Advanced Healthcare Chat Bot using Python" is a sophisticated solution designed to address the challenges of accessing timely medical advice in the modern world. The system features a user-friendly interface that can be seamlessly integrated into various platforms, such as websites or mobile applications. Leveraging Natural Language Processing (NLP), the chatbot interprets and processes user input, extracting relevant information about symptoms and medical history. The backbone of the system is a comprehensive medical knowledge base regularly updated to ensure accuracy and currency of information. Through the use of artificial intelligence algorithms, the chatbot can diagnose health issues based on user input and learn from interactions over time, continuously improving its diagnostic capabilities. The system includes a doctor recommendation mechanism that suggests users consult a healthcare professional based on the severity and nature of their symptoms. Emphasizing data privacy and security, the system encrypts user data and complies with healthcare data protection regulations. Additionally, it incorporates a feedback mechanism for users to contribute to the chatbot's ongoing improvement. The system's scalability and deployability make it adaptable for integration into existing healthcare systems or as a standalone application, promising enhanced accessibility to healthcare advice and potentially reducing medical costs.

LIMITATIONS OF EXISTING SYSTEM

Limited Diagnostic Accuracy:

The chatbot's diagnostic capabilities may have limitations in accurately identifying complex or rare medical conditions. The reliance on artificial intelligence algorithms and existing medical knowledge may result in misdiagnoses or incomplete assessments, emphasizing the importance of professional medical consultation for critical cases.

Dependency on User Input Quality:

The accuracy of the chatbot heavily relies on the quality and clarity of user input. Ambiguous or unclear information provided by users may lead to inaccurate assessments, and the system may struggle to handle nuanced or evolving symptoms that require in-depth medical expertise.

Lack of Physical Examination:

The chatbot lacks the ability to conduct physical examinations, which are crucial for certain medical diagnoses. Some health issues may require hands-on assessments, laboratory tests, or imaging studies that the chatbot cannot perform, potentially leading to incomplete or inaccurate recommendations.

Inability to Handle Emergency Situations:

The system may not effectively handle emergency medical situations that require immediate attention. In cases of severe health emergencies, users should be directed to seek urgent medical assistance rather than relying solely on the chatbot, highlighting a critical limitation in the system's scope.

Continuous Learning Challenges:

While the chatbot incorporates machine learning for continuous improvement, its learning is dependent on the volume and diversity of user interactions. In scenarios where the user base is limited or unrepresentative, the system may struggle to adapt and improve its diagnostic accuracy effectively.

PROPOSED SYSTEM

The proposed system for the "Advanced Healthcare Chat Bot using Python" aims to overcome the existing limitations while enhancing the overall functionality and user experience. Introducing advanced diagnostic modules based on deep learning and integrating more sophisticated Natural Language Processing (NLP) techniques, the system seeks to improve diagnostic accuracy, especially for complex and rare medical conditions. Additionally, the proposed system will implement mechanisms to prompt users for clearer input and context, minimizing the impact of ambiguous information on accuracy.

To address the limitation of the chatbot's inability to conduct physical examinations, the proposed system will incorporate features that guide users through self-assessment procedures, providing a more comprehensive set of data for analysis. Moreover, the system will include a priority escalation mechanism to promptly redirect users to emergency services in critical situations, ensuring a more responsible and effective response to urgent medical needs.

Continuous learning will be optimized through enhanced machine learning algorithms that actively seek diverse user interactions, allowing the system to adapt and improve its diagnostic capabilities more rapidly. Furthermore, the proposed system will explore partnerships with healthcare institutions to integrate real-time medical data, ensuring the knowledge base remains up-to-date and reflective of the latest advancements in the field.

ADVANTAGES OF PROPOSED SYSTEM

24/7 Accessibility and Availability:

The chatbot provides users with round-the-clock access to healthcare information and advice. This availability is especially beneficial for users seeking quick guidance on minor health issues outside regular office hours, promoting convenience and timely assistance.

Cost-Efficient Healthcare Support:

By offering preliminary health assessments and recommendations, the chatbot can potentially reduce unnecessary doctor visits for minor issues. This cost-efficient approach not only benefits users by saving time and money but also contributes to the optimization of healthcare resources.

User-Friendly Interface and Natural Language Processing:

The user interface is designed to be intuitive and user-friendly, allowing individuals to communicate their health concerns in natural language. The incorporation of Natural Language Processing (NLP) ensures a seamless interaction, making the chatbot accessible to a wide range of users, including those with varying levels of technical proficiency.

Continuous Learning and Improvement:

The chatbot employs machine learning algorithms to continuously learn from user interactions and feedback. This adaptive learning process enhances its diagnostic capabilities over time, ensuring that the system remains up to date with evolving medical knowledge and user preferences.

Privacy Preservation and Data Security:

The system prioritizes data privacy and security, employing robust encryption methods to safeguard user information. By adhering to healthcare data protection regulations, the chatbot instills confidence in users regarding the confidentiality of their health-related data, fostering trust in the system.

CHAPTER - 4

SYSTEM ANALYSIS

SYSTEM ANALYSIS

System analysis for an advanced healthcare chatbot involves evaluating various aspects of the system to ensure its effectiveness, efficiency, and reliability in providing disease diagnosis and treatment recommendations. Here's an overview of the key components of system analysis:

1. Requirement Analysis:

- Identify and analyze the requirements of the healthcare chatbot, including functional, non-functional, and regulatory requirements.
- Gather input from stakeholders, including patients, healthcare providers, and regulatory authorities, to define the scope and objectives of the system.

2. User Interface Analysis:

- Evaluate the user interface design of the chatbot to ensure it is intuitive, user-friendly, and accessible to users with varying levels of technical proficiency.
- Consider factors such as layout, navigation, interaction design, and support for multiple languages or accessibility features.

3. Data Analysis:

- Analyse the data requirements of the chatbot, including the types of medical data needed for symptom assessment, disease diagnosis, and treatment recommendations.
- Evaluate the quality, completeness, and relevance of available medical datasets and identify any gaps or limitations in data coverage.

4. Natural Language Processing (NLP) Analysis:

- Assess the NLP capabilities of the chatbot to understand and interpret user input in natural language.
- Evaluate the accuracy, robustness, and scalability of NLP algorithms for processing medical terminology, extracting relevant information, and generating appropriate responses.

5. Diagnostic Algorithm Analysis:

- Evaluate the diagnostic algorithm used by the chatbot to match user-reported symptoms with potential diseases or conditions.
- Assess the accuracy, specificity, and sensitivity of the algorithm in generating accurate diagnoses and prioritizing relevant medical conditions.

6. Treatment Recommendation Analysis:

- Analyze the treatment recommendation engine of the chatbot to ensure it provides evidence-based treatment recommendations tailored to individual patient needs and medical history.
- Evaluate the reliability, relevance, and effectiveness of treatment recommendations based on established medical guidelines and protocols.

7. Integration Analysis:

- Assess the integration capabilities of the chatbot with existing healthcare systems, electronic health records (EHRs), and telemedicine platforms.
- Evaluate interoperability standards, data exchange protocols, and security measures to ensure seamless integration and communication with external systems.

8. Usability Analysis:

- Conduct usability testing to evaluate the overall user experience of the chatbot, including ease of use, clarity of communication, and satisfaction with the provided services.
- Gather feedback from users and stakeholders to identify usability issues and areas for improvement in the chatbot's design and functionality.

9. Performance Analysis:

- Evaluate the performance of the chatbot in terms of response time, throughput, and scalability under different usage scenarios.

- Conduct performance testing to measure system performance under peak loads and identify any performance bottlenecks or optimization opportunities.

10. Security Analysis:

- Assess the security measures implemented in the chatbot to protect sensitive medical information and ensure compliance with healthcare regulations (e.g., HIPAA).

- Evaluate encryption, access controls, authentication mechanisms, and audit trails to mitigate security risks and safeguard patient data privacy.

11. Regulatory Compliance Analysis:

- Ensure that the chatbot complies with relevant healthcare regulations, standards, and guidelines, such as HIPAA, GDPR, and medical device regulations.

- Conduct regulatory compliance testing to verify adherence to legal requirements regarding data privacy, security, and patient confidentiality.

By conducting comprehensive system analysis across these dimensions, developers can identify strengths, weaknesses, and areas for improvement in the advanced healthcare chatbot, ultimately ensuring its effectiveness, reliability, and compliance with regulatory standards.

Functional Requirements:

- 1. Symptom Assessment:** The chatbot should be able to accurately assess user-reported symptoms by asking relevant questions and extracting necessary information to understand the user's health condition.
- 2. Disease Diagnosis:** Based on the symptoms provided by the user, the chatbot should be capable of diagnosing potential diseases or conditions using medical knowledge and algorithms and present a list of likely diagnoses.
- 3. Treatment Recommendations:** After diagnosing the condition, the chatbot should provide evidence-based treatment recommendations tailored to the user's individual needs, medical history, and preferences.
- 4. Integration with Healthcare Systems:** The chatbot should integrate seamlessly with existing healthcare systems, electronic health records (EHRs), and telemedicine platforms to facilitate communication and data exchange with healthcare providers.
- 5. User Interaction:** The chatbot should engage users in natural language conversations, allowing them to input symptoms, ask questions, receive recommendations, and provide feedback in an intuitive and user-friendly manner.
- 6. Privacy and Security:** The chatbot should ensure the confidentiality, integrity, and availability of user data by implementing robust security measures, such as encryption, access controls, and compliance with healthcare regulations (e.g., HIPAA).

Non-Functional Requirements:

- 1. Performance:** The chatbot should respond to user queries promptly and handle a large volume of concurrent users without degradation in performance or responsiveness.
- 2. Scalability:** The chatbot should be scalable to accommodate growth in user base and usage patterns, ensuring consistent performance under increasing loads.
- 3. Reliability:** The chatbot should be reliable and available 24/7, with minimal downtime or service interruptions to support users' healthcare needs at any time.
- 4. Usability:** The chatbot should be easy to use, with a user-friendly interface and intuitive navigation that allows users to interact with the system efficiently and effectively.
- 5. Accessibility:** The chatbot should be accessible to users with disabilities, complying with accessibility standards and providing features such as screen readers and keyboard navigation.
- 6. Security:** The chatbot should protect sensitive medical information and ensure compliance with healthcare regulations, employing encryption, access controls, and auditing mechanisms to safeguard user data.
- 7. Interoperability:** The chatbot should be interoperable with other healthcare systems and platforms, supporting industry standards and protocols for seamless integration and data exchange.
- 8. Compliance:** The chatbot should comply with relevant healthcare regulations, standards, and guidelines, such as HIPAA, GDPR, and medical device regulations, to ensure legal and ethical use of patient data.

1. TECHNICAL FEASIBILITY

2. OPERATIONAL FEASIBILITY

3. ECONOMIC FEASIBILITY

INTRODUCTION

A feasibility study assesses the operational, technical and economic merits of the proposed project. The feasibility study is intended to be a preliminary review of the facts to see if it is worthy of proceeding to the analysis phase. From the systems analyst perspective, the feasibility analysis is the primary tool for recommending whether to proceed to the next phase or to discontinue the project.

The feasibility study is a management-oriented activity. The objective of a feasibility study is to find out if an information system project can be done and to suggest possible alternative solutions.

Projects are initiated for two broad reasons:

1. Problems that lend themselves to systems solutions

2. Opportunities for improving through:

(a) upgrading systems

(b) altering systems

(c) installing new systems

A feasibility study should provide management with enough information to decide:

- Whether the project can be done
- Whether the final product will benefit its intended users and organization
- What are the alternatives among which a solution will be chosen
- Is there a preferred alternative?

TECHNICAL FEASIBILITY

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements.

The analyst must find out whether current technical resources can be upgraded or added to in a manner that fulfils the request under consideration. This is where the expertise of system analysts is beneficial, since using their own experience and their contact with vendors they will be able to answer the question of technical feasibility.

The essential questions that help in testing the operational feasibility of a system include the following:

- Is the project feasible within the limits of current technology?
- Does the technology exist at all?
- Is it available within given resource constraints?
- Is it a practical proposition?
- Manpower- programmers, testers & debuggers
- Software and hardware
- Are the current technical resources sufficient for the new system?
- Can they be upgraded to provide to provide the level of technology necessary for the new system?
- Do we possess the necessary technical expertise, and is the schedule reasonable?
- Can the technology be easily applied to current problems?
- Does the technology have the capacity to handle the solution?
- Do we currently possess the necessary technology?

OPERATIONAL FEASIBILITY

Operational feasibility is dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented.

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the feasibilities to gauge. In order to determine this feasibility, it is important to understand the management commitment to the proposed project. If the request was initiated by management, it is likely that there is management support, and the system will be accepted and used. However, it is also important that the employee base will be accepting of the change.

The essential questions that help in testing the operational feasibility of a system include the following:

- Does current mode of operation provide adequate throughput and response time?
- Does current mode provide end users and managers with timely, pertinent, accurate and useful formatted information?
- Does current mode of operation provide cost-effective information services to the business?
- Could there be a reduction in cost and or an increase in benefits?
- Does current mode of operation offer effective controls to protect against fraud and to guarantee accuracy and security of data and information?
- Does current mode of operation make maximum use of available resources, including people, time, and flow of forms?
- Does current mode of operation provide reliable services
- Are the services flexible and expandable?
- Are the current work practices and procedures adequate to support the new system?
- If the system is developed, will it be used?
- Manpower problems

- Labour objections
- Manager resistance
- Organizational conflicts and policies
- Social acceptability
- Government regulations
- Does management support the project?
- Are the users not happy with current business practices?
- Will it reduce the time (operation) considerably?
- Have the users been involved in the planning and development of the project?
- Will the proposed system really benefit the organization?
- Does the overall response increase?
- Will accessibility of information be lost?
- Will the system affect the customers in considerable way?
- Legal aspects
- How do the end-users feel about their role in the new system?
- What end-users or managers may resist or not use the system?
- How will the working environment of the end-user change?
- Can or will end-users and management adapt to the change?

ECONOMIC FEASIBILITY

Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

Possible questions raised in economic analysis are:

- Is the system cost effective?
- Do benefits outweigh costs?

- The cost of doing full system study
- The cost of business employee time
- Estimated cost of hardware
- Estimated cost of software/software development
- Is the project possible, given the resource constraints?
- What are the savings that will result from the system?
- Cost of employees' time for study
- Cost of packaged software/software development
- Selection among alternative financing arrangements (rent/lease/purchase)

The concerned business must be able to see the value of the investment it is pondering before committing to an entire system study. If short-term costs are not overshadowed by long-term gains or produce no immediate reduction in operating costs, then the system is not economically feasible, and the project should not proceed any further. If the expected benefits equal or exceed costs, the system can be judged to be economically feasible. Economic analysis is used for evaluating the effectiveness of the proposed system.

The economic feasibility will review the expected costs to see if they are in-line with the projected budget or if the project has an acceptable return on investment. At this point, the projected costs will only be a rough estimate. The exact costs are not required to determine economic feasibility. It is only required to determine if it is feasible that the project costs will fall within the target budget or return on investment. A rough estimate of the project schedule is required to determine if it would be feasible to complete the systems project within a required timeframe. The required timeframe would need to be set by the organization.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

MINIMUM (Required for Execution)		MY SYSTEM (Development)
System	Pentium IV 2.2 GHz	i3 Processor 5 th Gen
Hard Disk	20 GB	500 GB
RAM	1 GB	4 GB

SOFTWARE REQUIREMENTS

Operating System	Windows 10/11
Development Software	Python 3.10
Programming Language	Python
Domain	Image Processing & Cloud Computing
Integrated Development Environment (IDE)	Visual Studio Code
Front End Technologies	HTML5, CSS3, Java Script
Back End Technologies or Framework	Django
Database Language	SQL
Database (RDBMS)	MySQL
Database Software	WAMP or XAMPP Server
Web Server or Deployment Server	Django Application Development Server
Design/Modelling	Rational Rose

CHAPTER - 5

SYSTEM DESIGN

SYSTEM DESIGN

System design for an advanced healthcare chatbot involves creating a blueprint for the architecture, components, and interactions of the chatbot to ensure its functionality, usability, and performance. Here's an overview of the key aspects of system design:

1. Architecture Design:

- Define the overall architecture of the chatbot, including its components, modules, and their interactions.
- Choose an appropriate architecture pattern, such as a microservices architecture or a serverless architecture, based on scalability, flexibility, and performance requirements.

2. Component Design:

- Identify the key components of the chatbot, such as the user interface, natural language processing (NLP) engine, diagnostic algorithm, treatment recommendation engine, integration layer, and security layer.
- Define the responsibilities and interfaces of each component to facilitate modularity, reusability, and maintainability.

3. User Interface Design:

- Design an intuitive and user-friendly interface for the chatbot, ensuring ease of use and clarity of communication.
- Consider different platforms and devices where the chatbot will be deployed, such as web browsers, mobile apps, or messaging platforms, and adapt the user interface accordingly.

4. Natural Language Processing (NLP) Design:

- Implement NLP algorithms to interpret and analyse user input in natural language.
- Develop techniques for entity recognition, sentiment analysis, and context understanding to enhance the chatbot's understanding of user queries and responses.

5. Diagnostic Algorithm Design:

- Design a diagnostic algorithm to match user-reported symptoms with potential diseases or conditions.
- Incorporate medical knowledge databases, symptom databases, and machine learning models to improve diagnostic accuracy and relevance.

6. Treatment Recommendation Design:

- Develop a treatment recommendation engine to generate evidence-based treatment recommendations based on diagnosed conditions and patient-specific factors.
- Consider medical guidelines, protocols, and patient preferences in generating personalized treatment recommendations.

7. Integration Design:

- Design integration points and interfaces to connect the chatbot with external systems, such as healthcare databases, electronic health records (EHRs), and telemedicine platforms.
- Implement interoperability standards, data exchange protocols, and security measures to ensure seamless integration and communication with external systems.

8. Security Design:

- Implement robust security measures to protect sensitive medical information and ensure compliance with healthcare regulations (e.g., HIPAA).
- Apply encryption, access controls, authentication mechanisms, and auditing mechanisms to safeguard patient data privacy and security.

9. Scalability and Performance Design:

- Design the chatbot for scalability and performance to handle a large volume of user interactions and maintain responsiveness under peak loads.

- Implement techniques such as load balancing, caching, and asynchronous processing to optimize performance and resource utilization.

10. Regulatory Compliance Design:

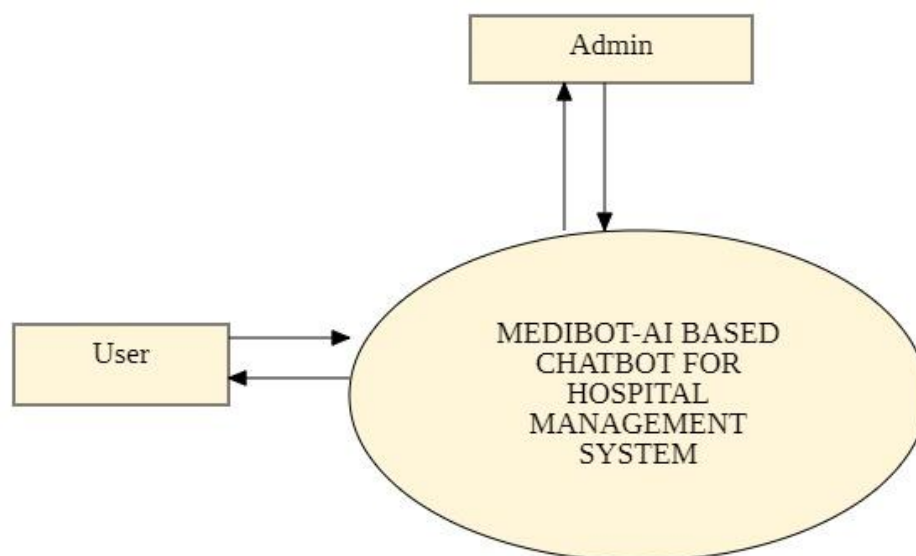
- Ensure that the chatbot complies with relevant healthcare regulations, standards, and guidelines, such as HIPAA, GDPR, and medical device regulations.

- Implement features and controls to enforce data privacy, security, and patient confidentiality in accordance with regulatory requirements.

By following a systematic approach to system design, developers can create an advanced healthcare chatbot that effectively meets the needs of users and healthcare providers, delivers accurate and personalized recommendations, and ensures compliance with regulatory standards.

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

UML represents a collection of the best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

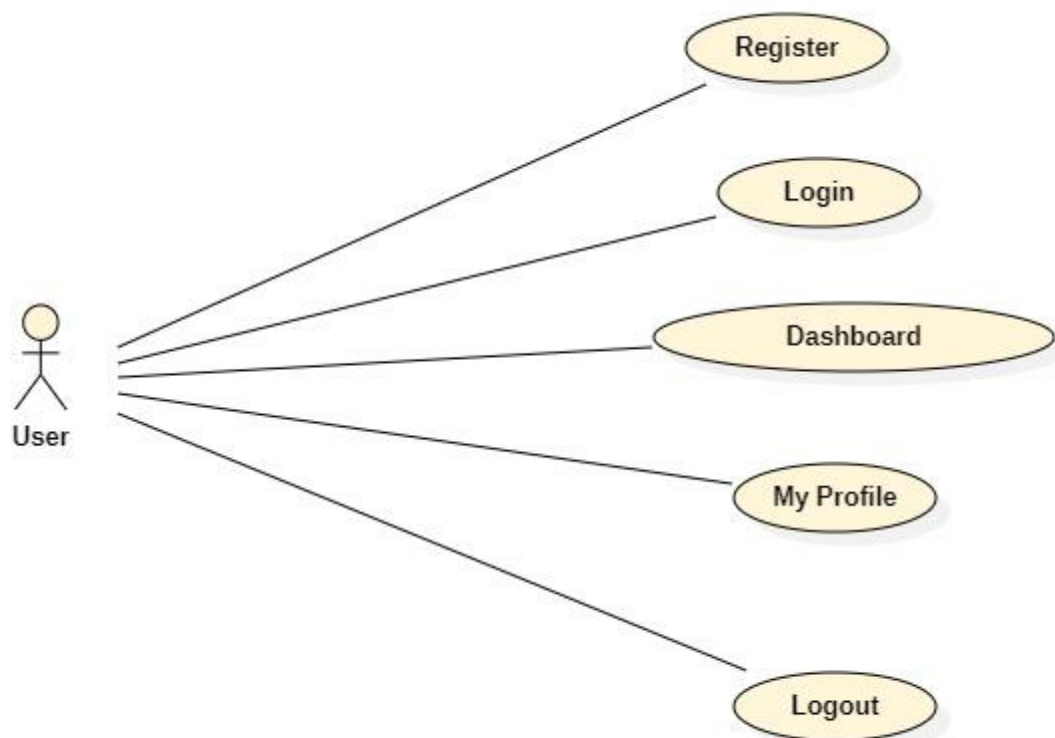
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users with a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

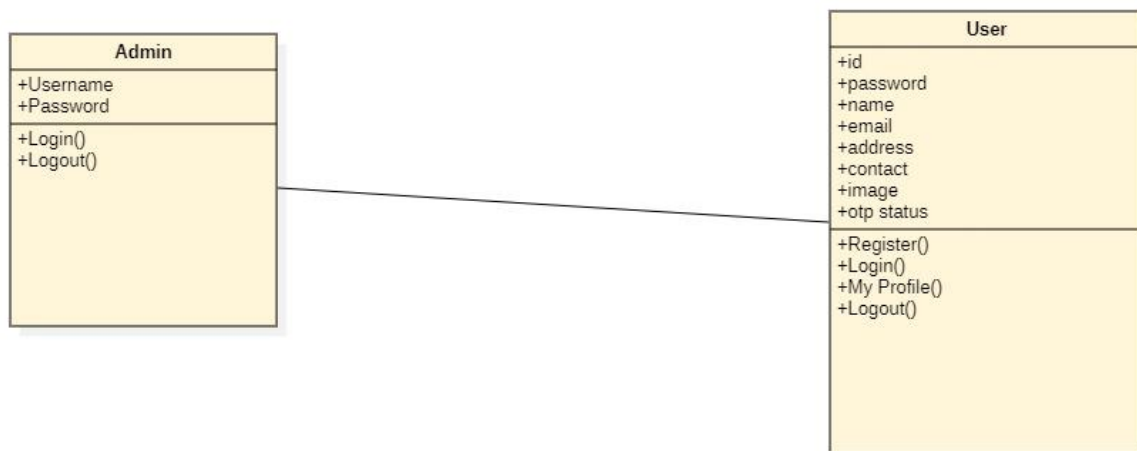
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



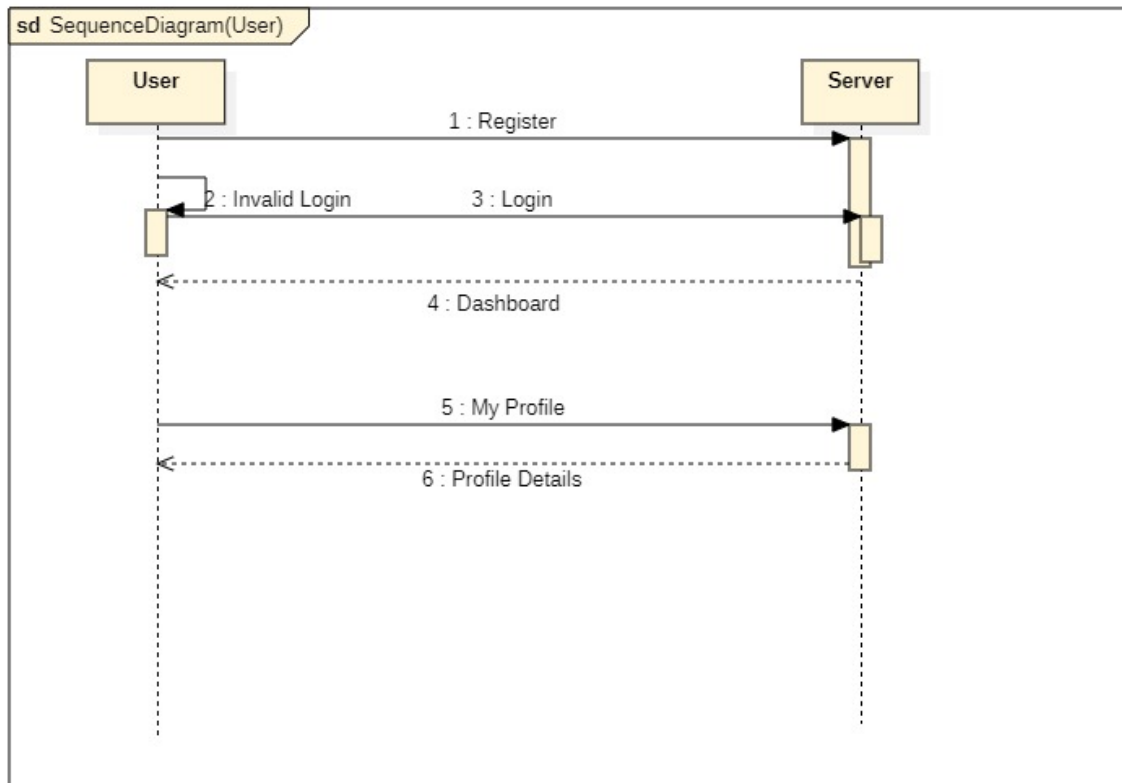
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



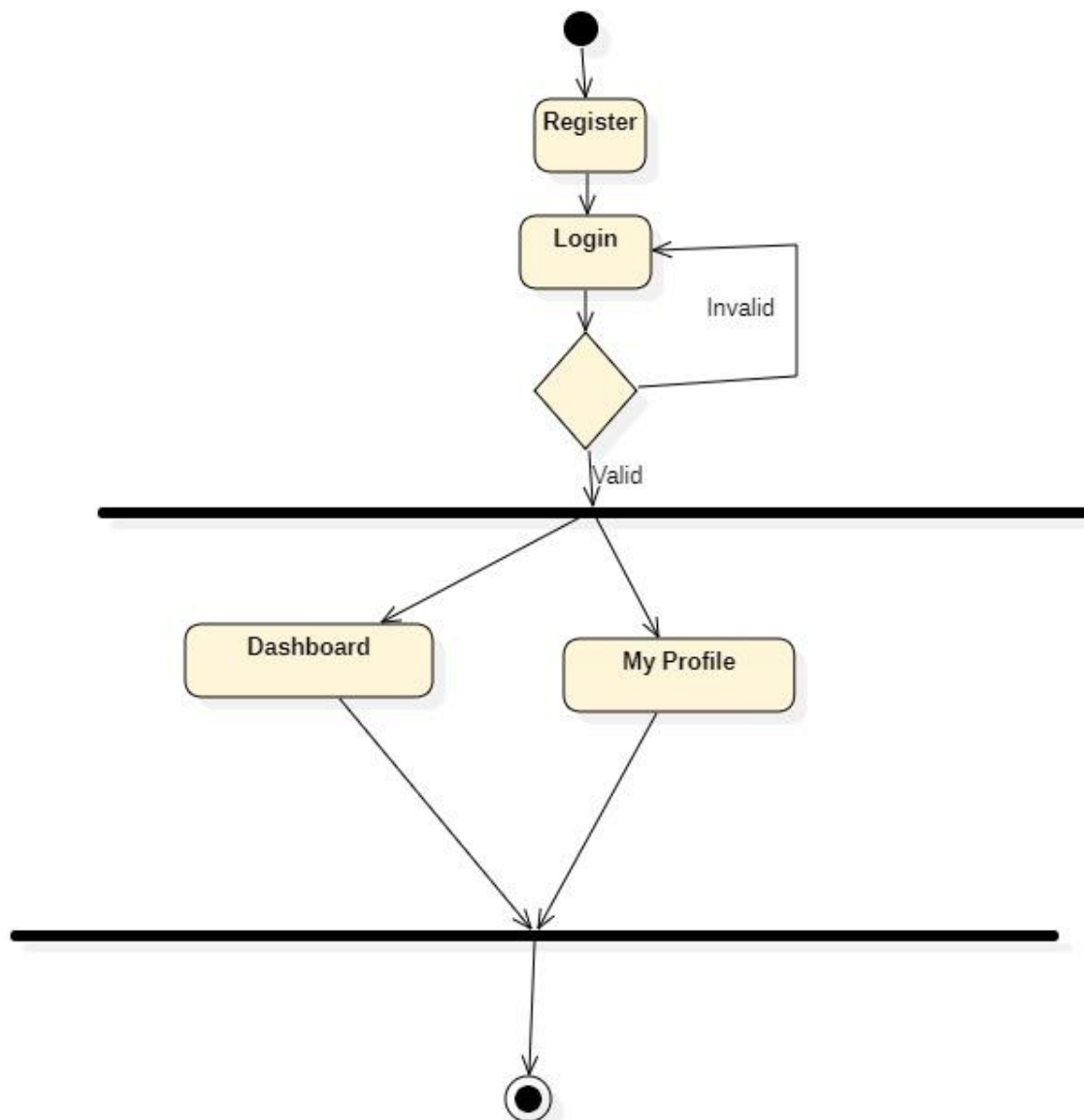
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



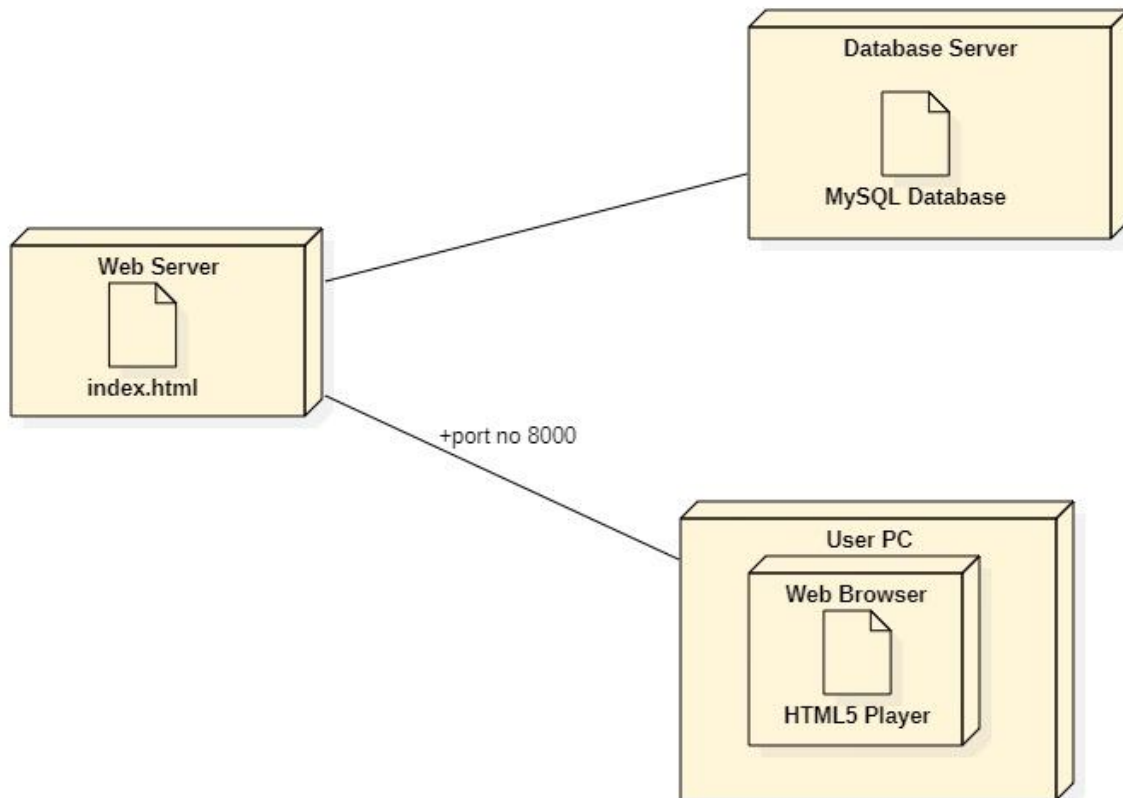
ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



DEPLOYMENT DIAGRAM:

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.



The deployment diagram maps the software architecture created in design to the physical system architecture that executes it. In distributed systems, it models the distribution of the software across the physical nodes.

The software systems are manifested using various artifacts, and then they are mapped to the execution environment that is going to execute the software such as nodes. Many nodes are involved in the deployment diagram; hence, the relation between them is represented using communication paths.

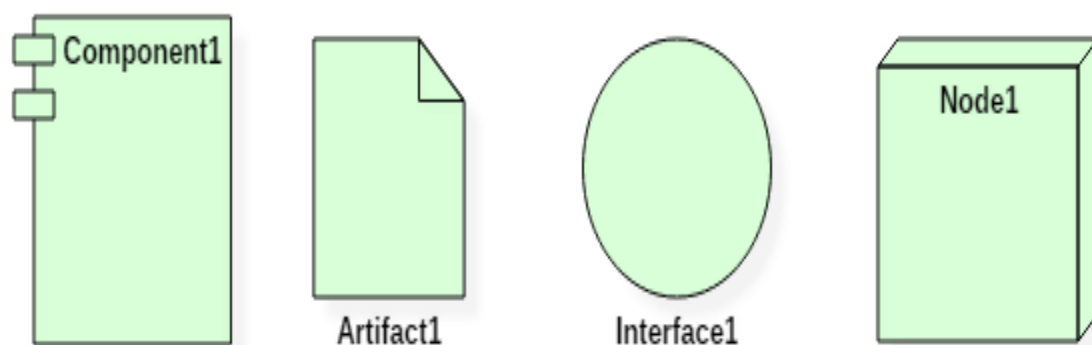
There are two forms of a deployment diagram.

- Descriptor form
 - It contains nodes, the relationship between nodes and artifacts.
- Instance form.
 - It contains node instance, the relationship between node instances and artifact instance.
 - An underlined name represents node instances.

Purpose of a deployment diagram

Deployment diagrams are used with the sole purpose of describing how software is deployed into the hardware system. It visualizes how software interacts with the hardware to execute the complete functionality. It is used to describe software to hardware interaction and vice versa.

Deployment Diagram Symbol and notations



Deployment Diagram Notations

MODULES

User Interface Module:

This module encompasses the user-facing components, including the chatbot interface integrated into websites or mobile applications. It facilitates user input, symptom description, and interaction with the chatbot, ensuring a seamless and intuitive experience.

Natural Language Processing (NLP) Module:

The NLP module is responsible for processing and understanding user input in natural language. It employs techniques such as tokenization, named entity recognition, and sentiment analysis to extract relevant information from user queries. This module enables effective communication between the user and the chatbot.

Diagnostic and Recommendation Module:

At the core of the system, this module utilizes artificial intelligence algorithms to analyze user-provided information against a comprehensive medical knowledge base. It is responsible for diagnosing health issues, offering informative overviews, and recommending users to consult with healthcare professionals based on the severity and nature of their symptoms.

Continuous Learning and Improvement Module:

The continuous learning module integrates machine learning algorithms to analyze user interactions and feedback. It enables the chatbot to learn from its experiences, improving its diagnostic accuracy over time. This module plays a crucial role in keeping the system up-to-date with evolving medical knowledge and user preferences.

Privacy and Security Module:

This module focuses on safeguarding user data and ensuring compliance with healthcare data protection regulations. It incorporates encryption techniques to protect sensitive health information, establishes secure data storage practices, and defines access controls to guarantee the privacy and security of user data throughout the interaction with the chatbot.

DATA DICTIONARY

auth_group

Table comments: auth_group

Column	Type	Null	Default
id	int(11)	No	
name	varchar(150)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	0	A	No
	BTREE	Yes	No	name	0	A	No

auth_group_permissions

Table comments: auth_group_permissions

Column	Type	Null	Default
id	bigint(20)	No	
group_id	int(11)	No	
permission_id	int(11)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	0	A	No
auth_group_permissions_group_id_permission_id_0cd325b0_uniq	BTREE	Yes	No	group_id		A	No
				permission_id	0	A	No
auth_group_permissions_group_id_b120cbf9	BTREE	No	No	group_id		A	No
auth_group_permissions_permission_id_84c5c92e	BTREE	No	No	permission_id		A	No

auth_permission

Table comments: auth_permission

Column	Type	Null	Default
id	int(11)	No	
name	varchar(255)	No	
content_type_id	int(11)	No	
codename	varchar(100)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	28	A	No
auth_permission_content_type_id_codename_01ab375a_uniq	BTREE	Yes	No	content_type_id		A	No
				codename	28	A	No
auth_permission_content_type_id_2f476e4b	BTREE	No	No	content_type_id		A	No

auth_user

Table comments: auth_user

Column	Type	Null	Default
id	int(11)	No	
password	varchar(128)	No	
last_login	datetime(6)	Yes	NULL
is_superuser	tinyint(1)	No	
username	varchar(150)	No	
first_name	varchar(150)	No	
last_name	varchar(150)	No	
email	varchar(254)	No	
is_staff	tinyint(1)	No	
is_active	tinyint(1)	No	
date_joined	datetime(6)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	0	A	No
username	BTREE	Yes	No	username	0	A	No

auth_user_groups

Table comments: auth_user_groups

Column	Type	Null	Default
id	bigint(20)	No	
user_id	int(11)	No	
group_id	int(11)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	0	A	No
auth_user_groups_user_id_group_id_94350c0c_uniq	BTREE	Yes	No	user_id		A	No
				group_id	0	A	No
auth_user_groups_user_id_6a12ed8b	BTREE	No	No	user_id		A	No
auth_user_groups_group_id_97559544	BTREE	No	No	group_id		A	No

auth_user_user_permissions

Table comments: auth_user_user_permissions

Column	Type	Null	Default
id	bigint(20)	No	
user_id	int(11)	No	
permission_id	int(11)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	0	A	No
auth_user_user_permissions_user_id_permission_id_14a6b632_uniq	BTREE	Yes	No	user_id		A	No
				permission_id	0	A	No
auth_user_user_permissions_user_id_a95ead1b	BTREE	No	No	user_id		A	No
auth_user_user_permissions_permission_id_1fbb5f2c	BTREE	No	No	permission_id		A	No

django_admin_log

Table comments: django_admin_log

Column	Type	Null	Default
id	int(11)	No	
action_time	datetime(6)	No	
object_id	longtext	Yes	NULL
object_repr	varchar(200)	No	
action_flag	smallint(5)	No	
change_message	longtext	No	
content_type_id	int(11)	Yes	NULL
user_id	int(11)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	0	A	No
django_admin_log_content_type_id_c4bce8eb	BTREE	No	No	content_type_id		A	Yes
django_admin_log_user_id_c564eba6	BTREE	No	No	user_id		A	No

django_content_type

Table comments: django_content_type

Column	Type	Null	Default
id	int(11)	No	
app_label	varchar(100)	No	
model	varchar(100)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	7	A	No
django_content_type_app_label_model_76bd3d3b_uniq	BTREE	Yes	No	app_label		A	No
				model	7	A	No

django_migrations

Table comments: django_migrations

Column	Type	Null	Default
id	bigint(20)	No	
app	varchar(255)	No	
name	varchar(255)	No	
applied	datetime(6)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	id	19	A	No

django_session

Table comments: django_session

Column	Type	Null	Default
session_key	varchar(40)	No	
session_data	longtext	No	

expire_date	datetime(6)	No	
-------------	-------------	----	--

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	session_key	2	A	No
django_session_expire_date_a5c62663	BTREE	No	No	expire_date		A	No

user details

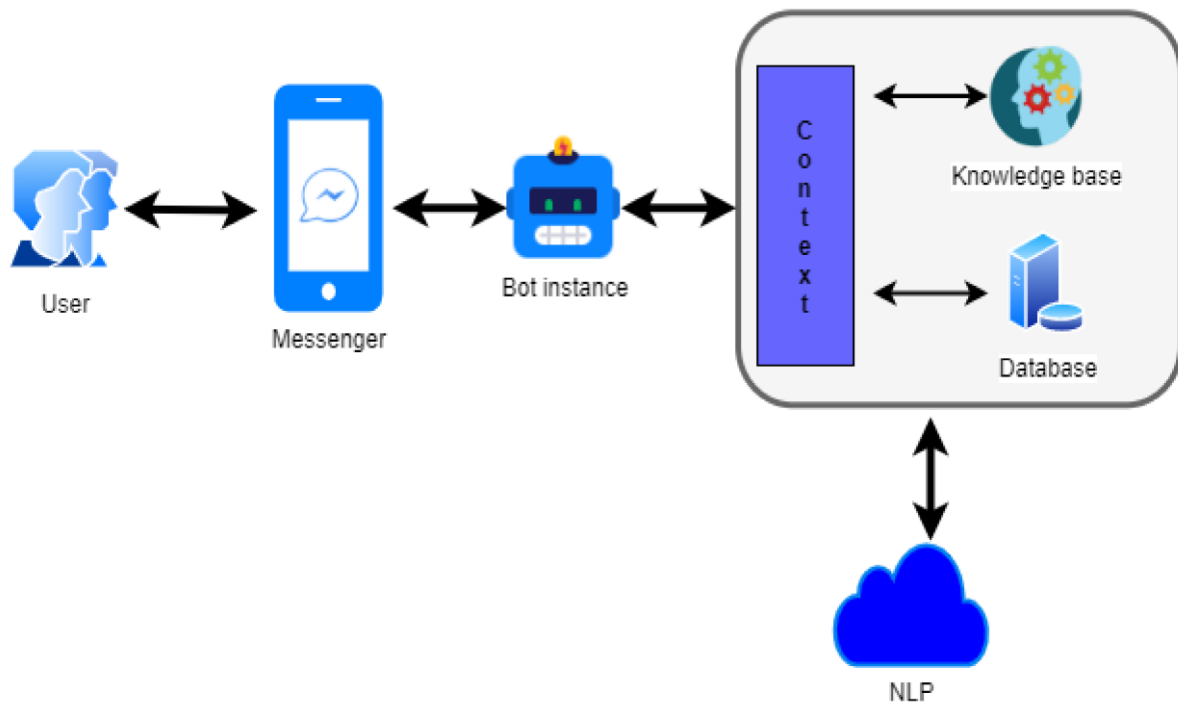
Table comments: user details

Column	Type	Null	Default
sno	int(11)	No	
name	varchar(50)	No	
email	varchar(254)	No	
Phone Number	varchar(11)	Yes	<i>NULL</i>
address	varchar(100)	Yes	<i>NULL</i>
password	varchar(20)	No	
image	varchar(100)	No	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
PRIMARY	BTREE	Yes	No	sno	2	A	No

SYSTEM ARCHITECTURE



The system architecture of an advanced healthcare chatbot for disease diagnosis and treatment recommendations typically consists of several interconnected components working together to provide a seamless user experience. Here's an overview of the key components and their interactions:

1. User Interface (UI):

- The user interface serves as the primary interaction point for users to communicate with the chatbot. It can be a web interface, mobile app, or messaging platform interface.
- The UI allows users to input symptoms, ask questions, view diagnostic results, and receive treatment recommendations in a user-friendly manner.

2. Natural Language Processing (NLP) Engine:

- The NLP engine is responsible for understanding and interpreting user input in natural language.
- It analyzes user queries, extracts relevant keywords and entities, and determines the user's intent or inquiry.
- NLP techniques such as entity recognition, sentiment analysis, and context understanding are employed to enhance understanding.

3. Diagnostic Algorithm:

- The diagnostic algorithm processes user-reported symptoms and matches them with potential diseases or conditions.
- It utilizes medical knowledge databases, symptom databases, and machine learning models to generate accurate diagnoses.

4. Treatment Recommendation Engine:

- The treatment recommendation engine generates evidence-based treatment recommendations based on the diagnosed condition and the user's individual characteristics.
- It considers factors such as medical history, allergies, medications, and treatment guidelines to provide personalized recommendations.

5. Knowledge Base:

- The knowledge base contains a comprehensive repository of medical information, including symptoms, diseases, treatments, medications, and medical guidelines.
- It serves as a reference for the diagnostic algorithm and treatment recommendation engine, providing the necessary information to make informed decisions.

6. Integration Layer:

- The integration layer facilitates communication and data exchange between the chatbot and external systems, such as healthcare databases, electronic health records (EHRs), and telemedicine platforms.
- It ensures interoperability and compatibility with industry standards and protocols, enabling seamless integration with existing healthcare infrastructure.

7. Security and Compliance Layer:

- The security and compliance layer implements robust security measures to protect sensitive medical information and ensure compliance with healthcare regulations (e.g., HIPAA).
- It includes encryption, access controls, auditing mechanisms, and regular security assessments to mitigate risks and safeguard user data.

8. Analytics and Monitoring:

- Analytics and monitoring components track user interactions, system performance, and user satisfaction metrics.
- They provide insights into chatbot usage patterns, identify areas for improvement, and support continuous optimization of the system.

Overall, the system architecture of an advanced healthcare chatbot is designed to provide accurate, personalized, and secure disease diagnosis and treatment recommendations while ensuring a seamless user experience and compliance with healthcare regulations.

CHAPTER - 6

IMPLEMENTATION

What is Python programming language?

Python is a **high-level, general-purpose, interpreted** programming language.

1) High-level

Python is a high-level programming language that makes it easy to learn. Python doesn't require you to understand the details of the computer in order to develop programs efficiently.

2) General-purpose

Python is a general-purpose language. It means that you can use Python in various domains including:

- Web applications
- Big data applications
- Testing
- Automation
- Data science, machine learning, and AI
- Desktop software
- Mobile apps

The targeted language like SQL which can be used for querying data from relational databases.

3) Interpreted

Python is an interpreted language. To develop a Python program, you write Python code into a file called source code.

To execute the source code, you need to convert it to the machine language that the computer can understand. And the Python **interpreter** turns the source code, line by line, once at a time, into the machine code when the Python program executes.

Compiled languages like Java and C# use a **compiler** that compiles the whole source code before the program executes.

Why Python

Python increases your productivity. Python allows you to solve complex problems in less time and fewer lines of code. It's quick to make a prototype in Python.

Python becomes a solution in many areas across industries, from web applications to data science and machine learning.

Python is quite easy to learn in comparison with other programming languages. Python syntax is clear and beautiful.

Python has a large ecosystem that includes lots of libraries and frameworks.

Python is cross-platform. Python programs can run on Windows, Linux, and macOS.

Python has a huge community. Whenever you get stuck, you can get help from an active community.

Python developers are in high demand.

History of Python

- Python was created by Guido Van Rossum.
- The design began in the late 1980s and was first released in February 1991.

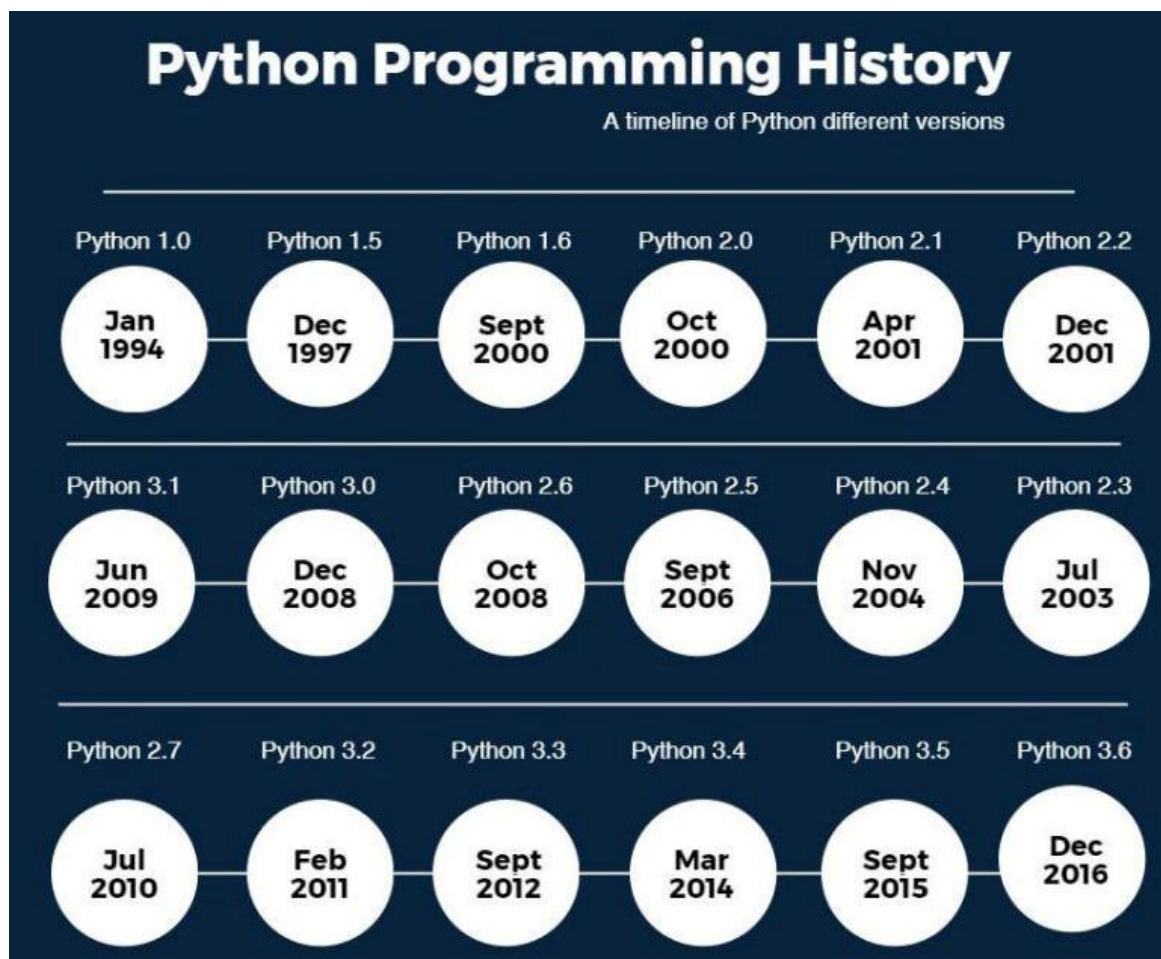
Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late 70s. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Python Version History

Implementation started - December 1989

Internal releases – 1990

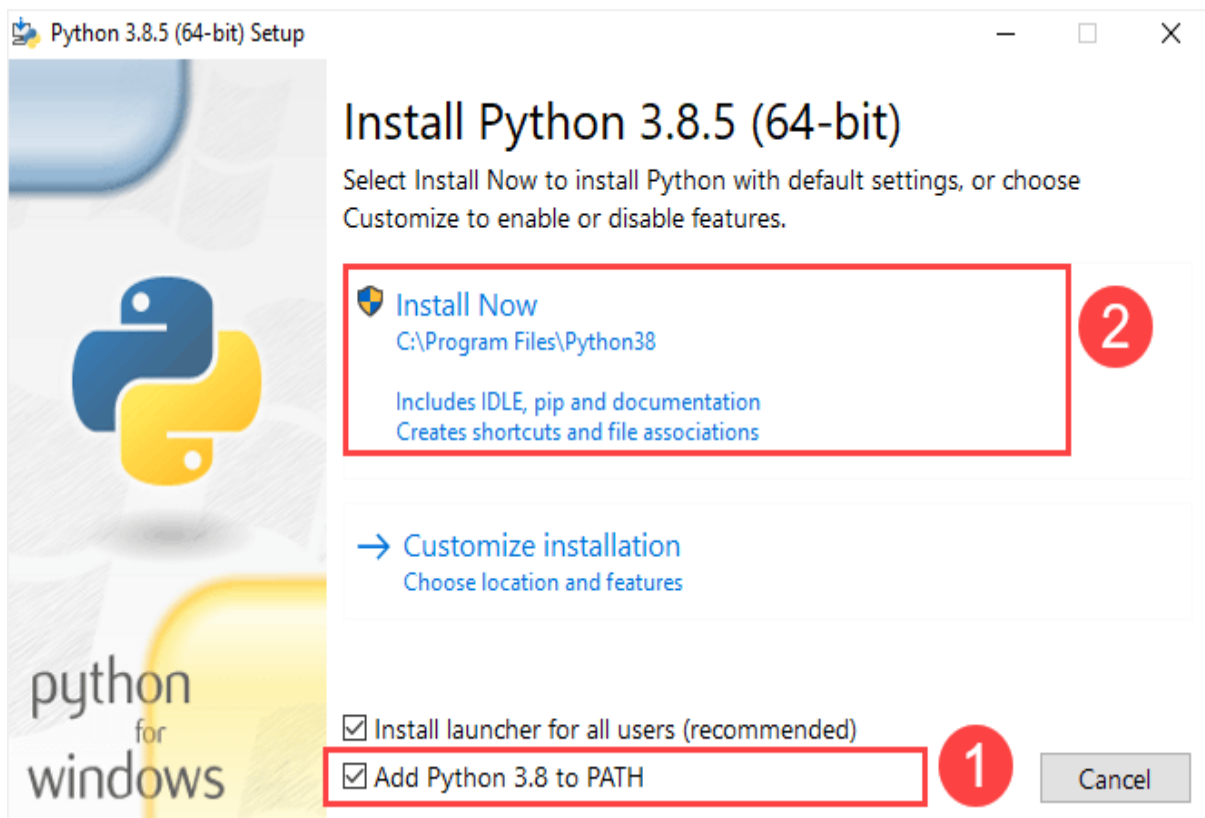


Install Python on Windows

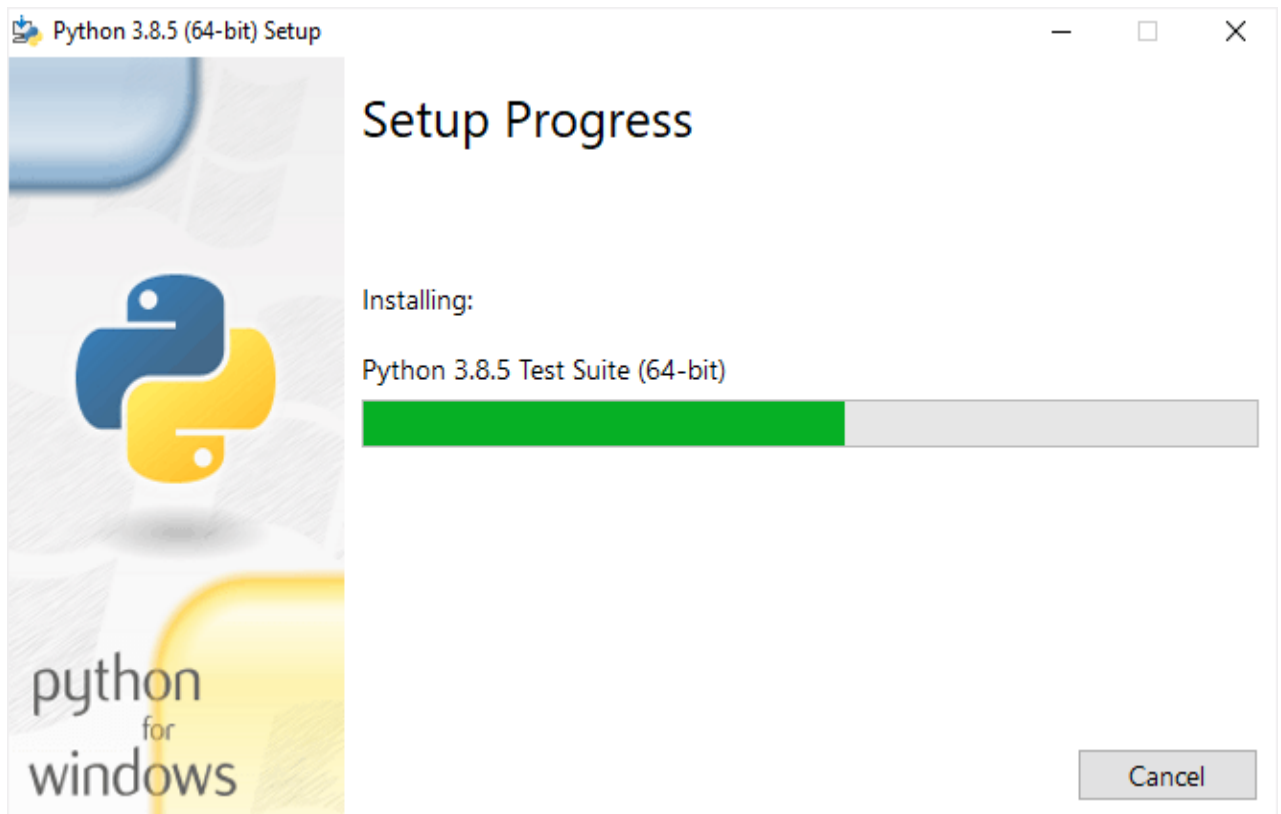
First, download the latest version of Python from the download page.

Second, double-click the installer file to launch the setup wizard.

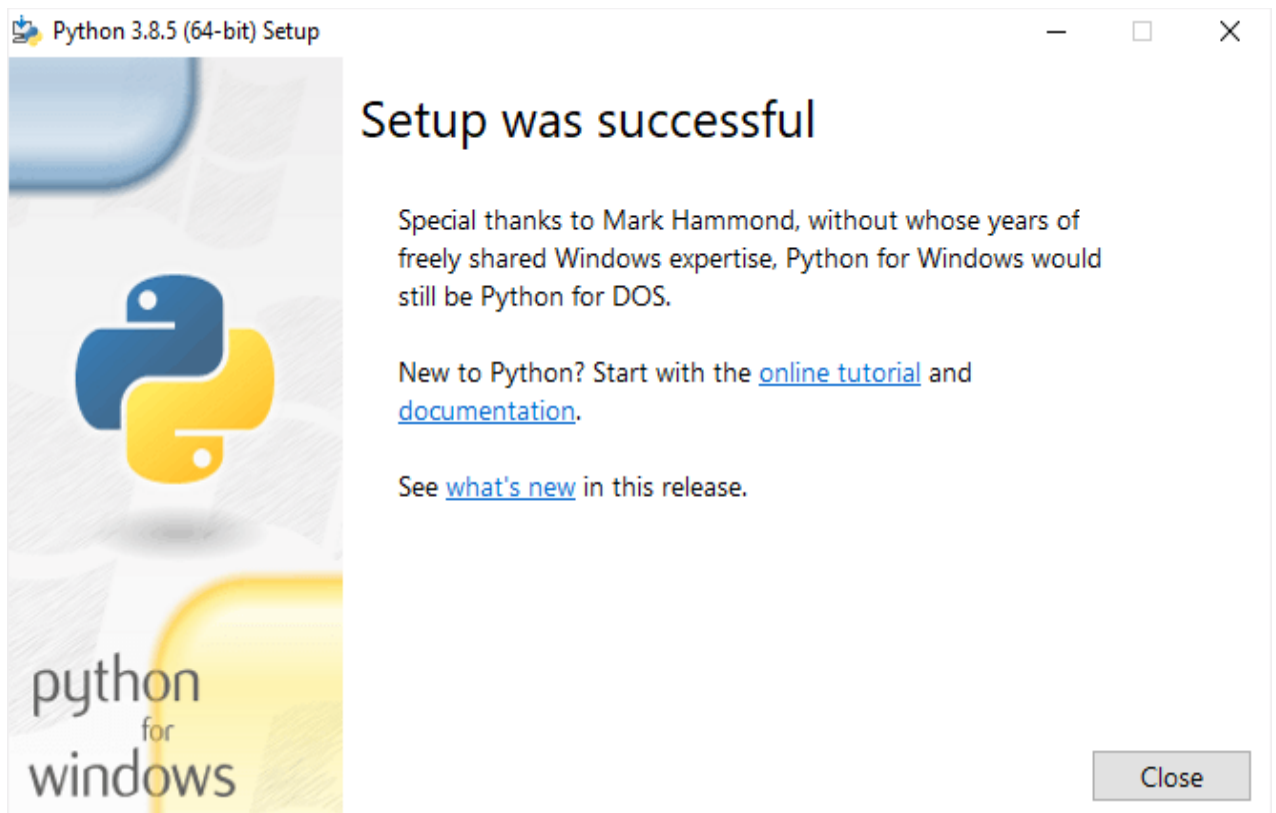
In the setup window, you need to check the **Add Python 3.8 to PATH** and click Install Now to begin the installation.



It'll take a few minutes to complete the setup.

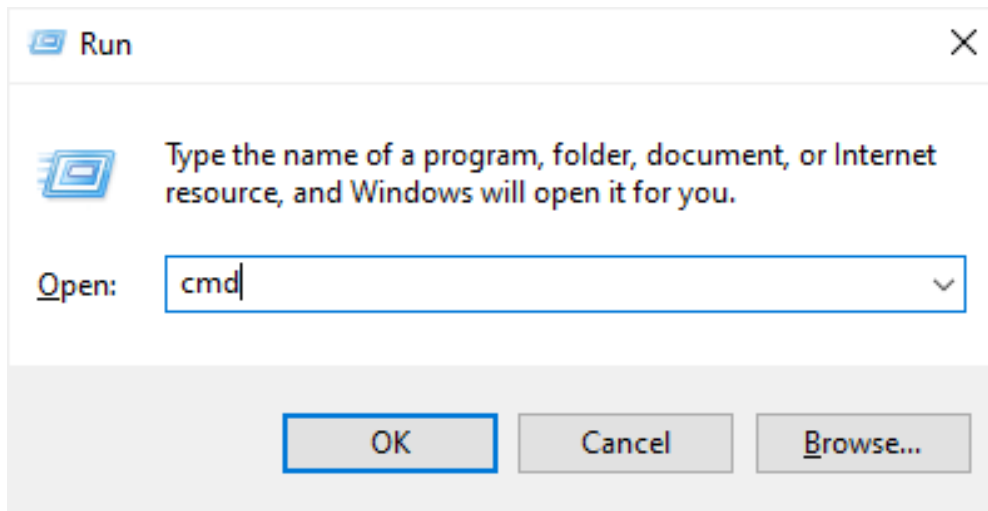


Once the setup completes, you'll see the following window:

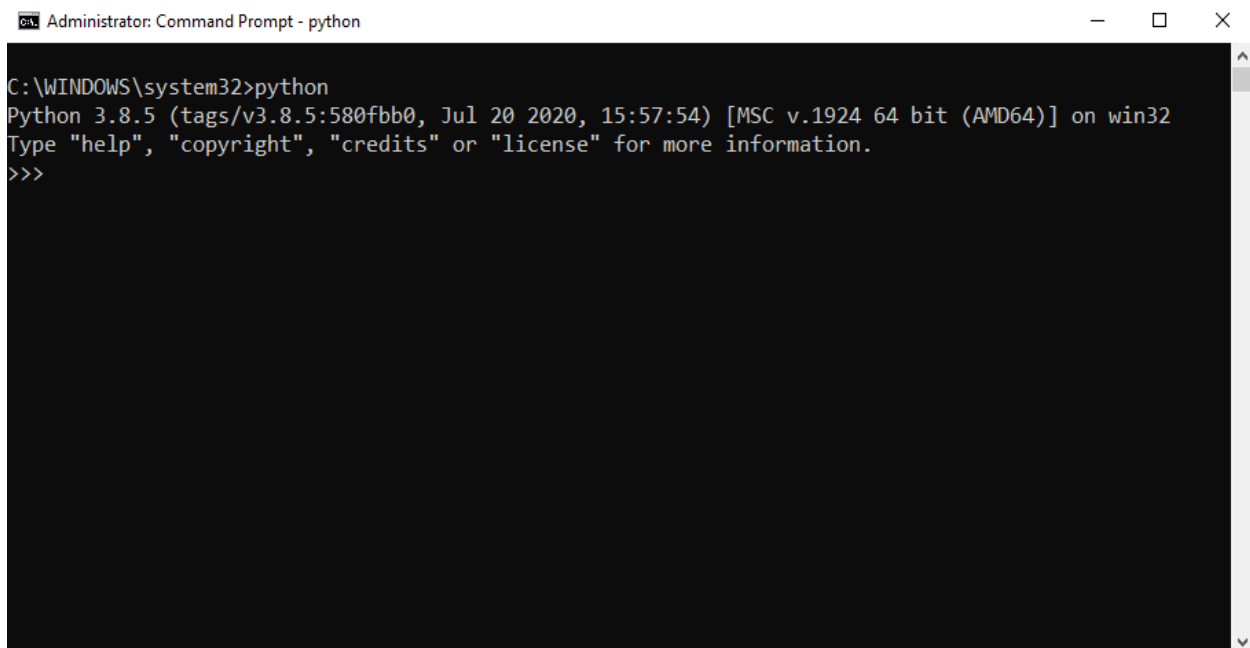


Verify the installation!

To verify the installation, you open the Run window and type cmd and press Enter:



In the Command Prompt, type python command as follows:

A screenshot of the Windows Command Prompt window titled 'Administrator: Command Prompt - python'. The command prompt shows the command 'python' being executed. The output is: 'C:\WINDOWS\system32>python', 'Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license" for more information.', and '>>>'. The prompt is currently at the '>>>' line.

If you see the output like the above screenshot, you've successfully installed Python on your computer.

To exit the program, you type Ctrl-Z and press Enter.

If you see the following output from the Command Prompt after typing the python command:

```
'python' is not recognized as an internal or external command,  
operable program or batch file.
```

Likely, you didn't check the **Add Python 3.8 to PATH** checkbox when you install Python.

To install the newer version, you replace 3.10 with that version.

A quick introduction to the Visual Studio Code

Visual Studio Code is a lightweight source code editor. The Visual Studio Code is often called VS Code. The VS Code runs on your desktop. It's available for Windows, macOS, and Linux.

VS Code comes with many features such as IntelliSense, code editing, and extensions that allow you to edit Python source code effectively. The best part is that the VS Code is open-source and free.

Besides the desktop version, [VS Code also has a browser version](#) that you can use directly in your web browser without installing it.

This tutorial teaches you how to set up Visual Studio Code for a Python environment so that you can edit, run, and debug Python code.

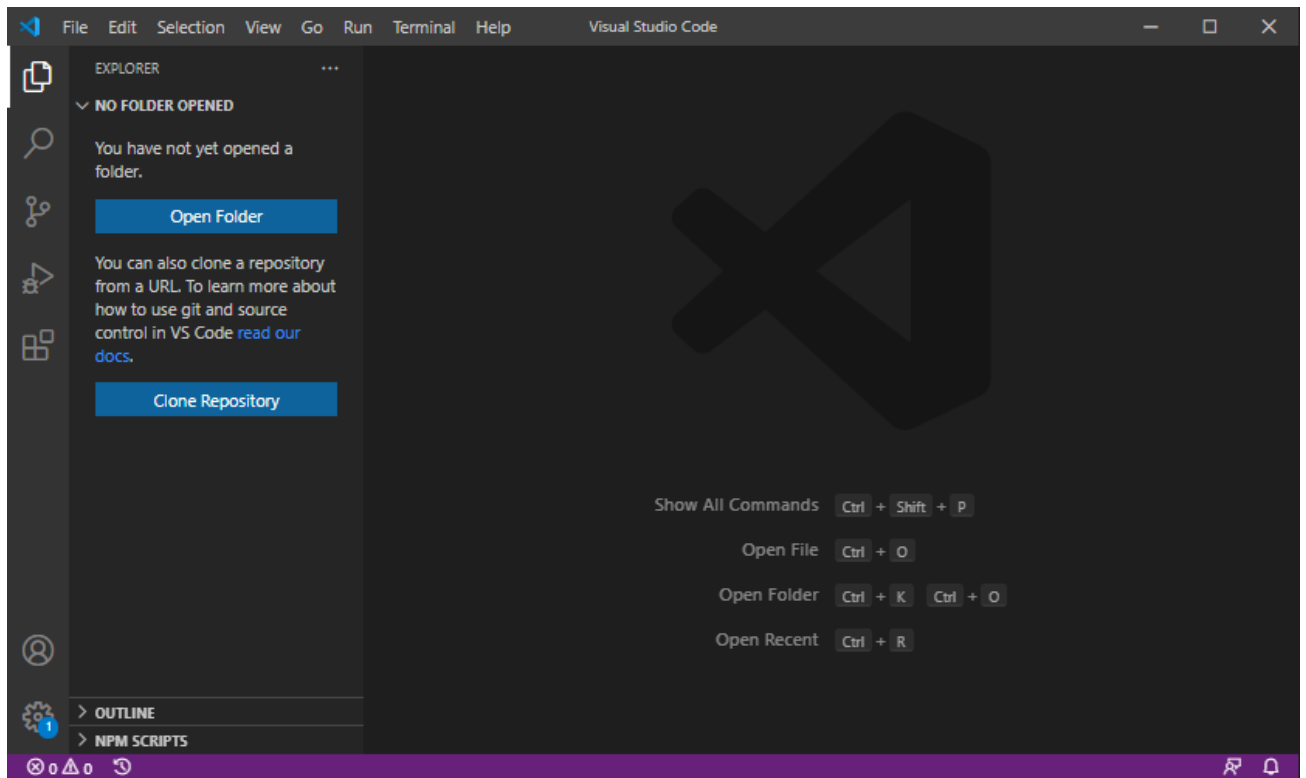
Setting up Visual Studio Code

To set up the VS Code, you follow these steps:

First, navigate to the [VS Code official](#) website and download the VS code based on your platform (Windows, macOS, or Linux).

Second, launch the setup wizard and follow the steps.

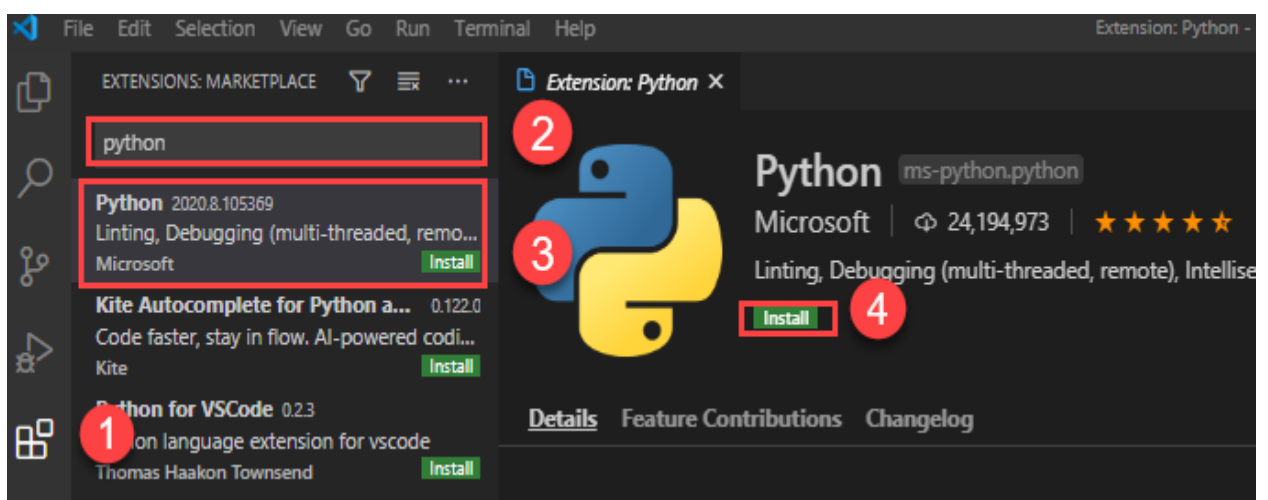
Once the installation completes, you can launch the VS code application:



Install Python Extension

To make the VS Code works with Python, you need to install the Python extension from the Visual Studio Marketplace.

The following picture illustrates the steps:



- First, click the **Extensions** tab.
- Second, type the python extension pack keyword on the search input.

- Third, click the Python extension pack. It'll show detailed information on the right pane.
- Finally, click the **Install** button to install the Python extension.

Now, you're ready to develop the first program in Python.

Creating a new Python project

First, create a new folder called helloworld.

Second, launch the VS code and open the helloworld folder.

Third, create a new app.py file and enter the following code and save the file:

```
print('Hello, World!')
```

Code language: Python (python)

The print () is a built-in function that displays a message on the screen. In this example, it'll show the message 'Hello, Word!'.

What is a function?

When you sum two numbers, that's a function. And when you multiply two numbers, that's also a function.

Each function takes your inputs, applies some rules, and returns a result.

In the above example, the print() is a function. It accepts a string and shows it on the screen.

Python has many built-in functions like the print() function to use them out of the box in your program.

In addition, Python allows you to define your functions, which you'll learn how to do it later.

Executing the Python Hello World program

To execute the app.py file, you first launch the Command Prompt on Windows or Terminal on macOS or Linux.

Then, navigate to the helloworld folder.

After that, type the following command to execute the app.py file:

```
python app.py
```

Code language: Python (python)

If you use macOS or Linux, you use python3 command instead:

```
python3 app.py
```

Code language: CSS (css)

If everything is fine, you'll see the following message on the screen:

```
Hello, World!
```

Code language: Python (python)

If you use VS Code, you can also launch the Terminal within the VS code by:

- Accessing the menu **Terminal > New Terminal**
- Or using the keyboard shortcut **Ctrl+Shift+`**.

Typically, the backtick key (`) locates under the Esc key on the keyboard.

Python IDLE

Python IDLE is the Python Integration Development Environment (IDE) that comes with the Python distribution by default.

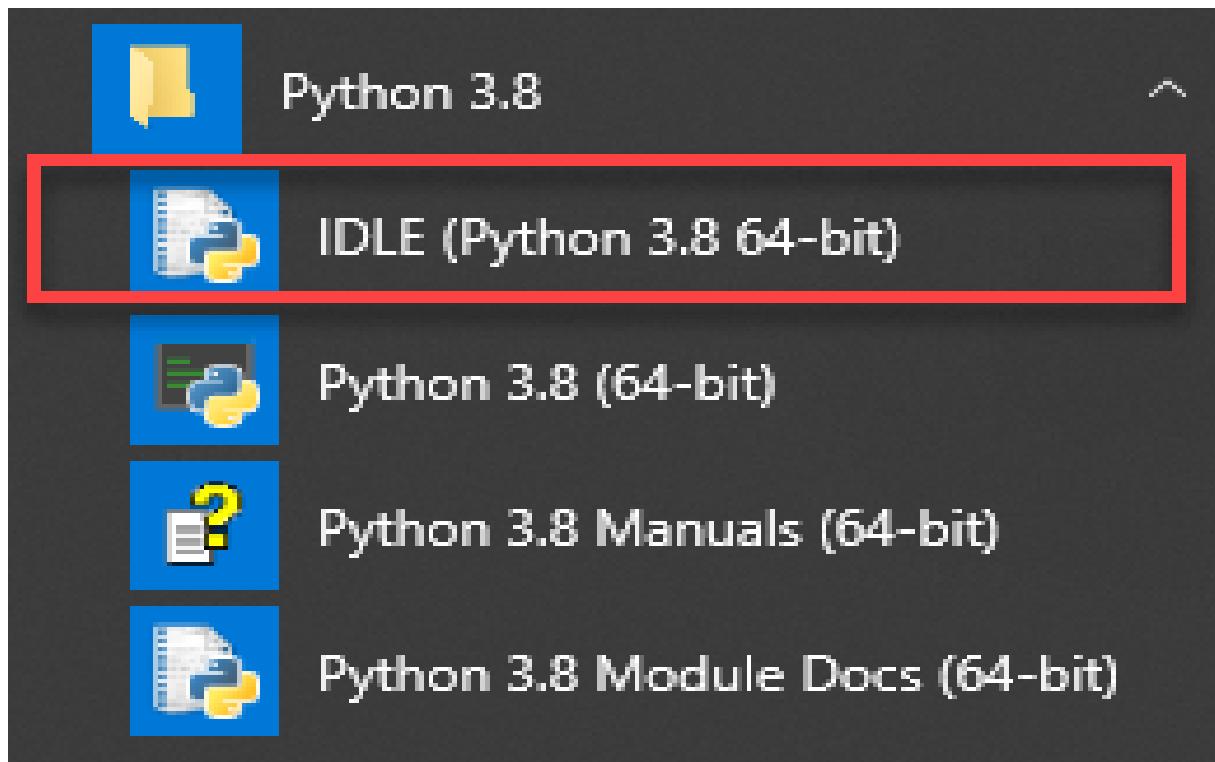
The Python IDLE is also known as an interactive interpreter. It has many features such as:

- Code editing with syntax highlighting
- Smart indenting
- And auto-completion

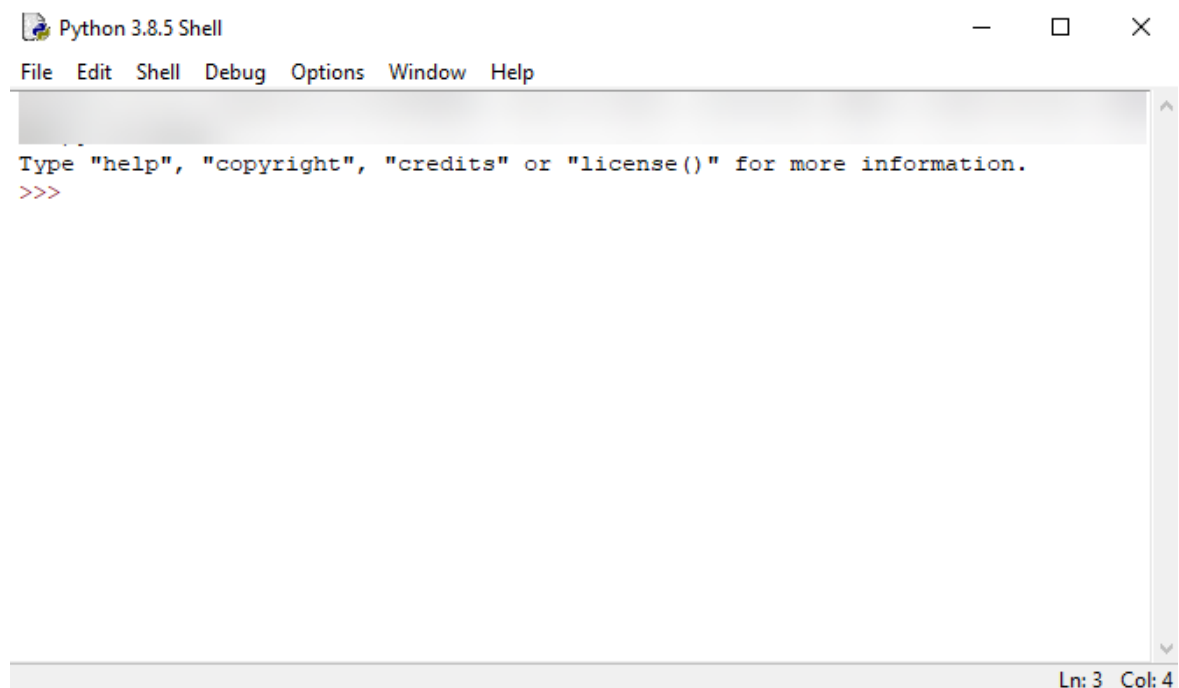
In short, the Python IDLE helps you experiment with Python quickly in a trial-and-error manner.

The following shows you step by step how to launch the Python IDLE and use it to execute the Python code:

First, launch the Python IDLE program:




A new Python Shell window will display as follows:



Now, you can enter the Python code after the cursor `>>>` and press Enter to execute it.

For example, you can type the code `print('Hello, World!')` and press Enter, you'll see the message Hello, World! immediately on the screen:

A screenshot of a Python 3.8.5 Shell window. The window has a title bar with the text "Python 3.8.5 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the items "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main area of the window is a text editor with a light gray background. It contains the following text: "Type 'help', 'copyright', 'credits' or 'license()' for more information." followed by a prompt ">>>". The user has entered the code `print('Hello, World!')` and pressed Enter. The output "Hello, World!" is displayed on the next line. The prompt ">>>" is followed by a vertical cursor. At the bottom right of the window, a status bar shows "Ln: 5 Col: 4".

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello, World!')
Hello, World!
>>> |
Ln: 5 Col: 4
```

Python Syntax

Whitespace and indentation

If you've been working in other programming languages such as Java, C#, or C/C++, you know that these languages use semicolons (;) to separate the statements.

However, Python uses whitespace and indentation to construct the code structure.

The following shows a snippet of Python code:

```
# define main function to print out something
def main():
    i = 1
    max = 10
    while (i < max):
```



```
print(i)
i = i + 1
# call function main
main()
```

The meaning of the code isn't important to you now. Please pay attention to the code structure instead.

At the end of each line, you don't see any semicolon to terminate the statement. And the code uses indentation to format the code.

By using indentation and whitespace to organize the code, Python code gains the following advantages:

- First, you'll never miss the beginning or ending code of a block like in other programming languages such as Java or C#.
- Second, the coding style is essentially uniform. If you have to maintain another developer's code, that code looks the same as yours.
- Third, the code is more readable and clearer in comparison with other programming languages.

Comments

The comments are as important as the code because they describe why a piece of code was written.

When the Python interpreter executes the code, it ignores the comments.

In Python, a single-line comment begins with a hash (#) symbol followed by the comment. For example:

```
# This is a single line comment in Python
```

Continuation of statements

Python uses a newline character to separate statements. It places each statement on one line.

However, a long statement can span multiple lines by using the backslash (\) character.

The following example illustrates how to use the backslash (\) character to continue a statement in the second line:

```
if (a == True) and (b == False) and \
    (c == True):
    print("Continuation of statements")
```

Identifiers

Identifiers are names that identify variables, functions, modules, classes, and other objects in Python.

The name of an identifier needs to begin with a letter or underscore (_). The following characters can be alphanumeric or underscore.

Python identifiers are case-sensitive. For example, the counter and Counter are different identifiers.

In addition, you cannot use Python keywords for naming identifiers.

Keywords

Some words have special meanings in Python. They are called keywords.

The following shows the list of keywords in Python:

```
False  class  finally  is  return
None   continue  for  lambda  try
True   def  from  nonlocal  while
and    del  global  not  with
as     elif  if  or  yield
assert  else  import  pass
break  except  in  raise
```

Python is a growing and evolving language. So, its keywords will keep increasing and changing.

Python provides a special module for listing its keywords called keyword.

To find the current keyword list, you use the following code:

```
import keyword  
  
print(keyword.kwlist)
```

String literals

Python uses single quotes ('), double quotes ("), triple single quotes (") and triple-double quotes (""") to denote a string literal.

The string literal need to be surrounded with the same type of quotes. For example, if you use a single quote to start a string literal, you need to use the same single quote to end it.

The following shows some examples of string literals:

```
s = 'This is a string'  
print(s)  
  
s = "Another string using double quotes"  
print(s)  
  
s = """ string can span  
multiple line """  
print(s)
```

DIALOGFLOW

- Dialogflow is a Google service which operates on a Google Cloud Platform.
- The Dialogflow is an intuitive and user-friendly tool that includes *Google's machine learning expertise* and some *Google products* such as Google Cloud Speech-to-Text.
- Dialogflow is an NLP (Natural Language Processing) platform, which is used to develop an application related to the conversations and experience for the customers of the company in different languages on numerous platforms.
- Dialogflow is mainly used to build actions for most of the Google Assistant devices.

What is Dialogflow?

Dialogflow is defined as a Google service which operates on a Google Cloud Platform. The Dialogflow is an intuitive and user-friendly tool that includes some *Google products* such as Google Cloud Speech-to-Text and *Google's machine learning expertise*. It is mainly used to build actions for most of the Google Assistant devices.

In other words, Dialogflow is defined as an NLP (Natural Language Processing) platform, which is used to develop an application related to the conversations and experience for the customers of the company in different languages on numerous platforms. By using Google-powered, products developers can design text-based and voice-conversation interfaces in order to answer the queries of customers in various languages. *For example* - various companies use Dialogflow to make messaging bots that reply to the queries of the customers on different platforms such as *Google Assistant, Slack, Facebook Messenger, Alexa Voice Search (AVS), etc.*

Features of the Dialogflow

There are various features of the Dialogflow:

1. Develop serverless apps easy.
2. Deliver automated phone service.
3. Designed for a voice-first world.
4. Reply on automatic spelling correction.
5. Improve experience with built-in analytics.
6. Understand user sentiment.
7. Deploy across platforms and languages.
8. Bring your enterprise knowledge.
9. Powered by Google machine learning.

Develop Serverless Apps Easy

There is a unified code editor offered by the Google Dialogflow through which we can easily develop natively serverless apps which are connected to our conversational interface via firebase cloud functions.

Deliver Automated Phone Service

With the help of the Dialogflow phone gateway, within a minute we can easily add the dedicated phone number into our Dialogflow agent and the users who call the agent with the new phone number will contact your Dialogflow agent directly. The Phone Gateway is based on the cost invested by Google in various things such as speech recognition, phone connectivity, speech synthesis, natural language processing, etc.

Designed for a Voice-First World

With a single [API](#) request, we are able to extend our conversational interface to identify voice communications and produce a voice reply. Dialogflow supports synchronous modes and real-time streaming, which are powered by the Google Cloud Text-to-speech and Cloud Text-to-Speech.

Reply on Automatic Spelling Correction

In the chat environments, most of the users interact in a hurry, and sometime there may be situations where the users do not emphasize on accurate spelling or grammar in their chats. So, in that situation, there is a feature provided by the Dialogflow named automatic spelling correction that helps to automatically spell correction mistakes with the help of the technology used by Google for search.

Improve Experience with Built-in Analytics

In the Dialogflow, the integrated analytics dashboard provides you with a vision for conversational communication. Thus, you can customize your bot to better understanding and reply to the user intentions.

Understand User Sentiment

For the query of each user, Dialogflow performs sentiment analysis, which is powered by the *Cloud Natural Language*. The score of sentiment analysis is used to distribute the unfulfilled users to the live agents or to gain a well knowledge about which attempts lead to the highest client feeling.

Deploy Across Platforms and Languages

It supports more than 20 languages with 14 different platforms.

Bring your Enterprise Knowledge

With the help of the Dialogflow knowledge connectors, we can easily augment data in bulk from the enterprise to the agent that contains articles based on knowledge and FAQs. To extract

the correct answer from the data corpus, the knowledge connectors of the Dialogflow use those technologies used by the Google Assistant and Google Search.

Powered by Google Machine Learning

The Dialogflow tool is powered by Google [Machine learning](#). In the Dialogflow, the understanding of the natural language identify the intent of the users and helps to extract those entities which are prebuilt like a number, date, and time. By offering a small dataset, we can also train the agent to easily recognize the customer entity types.

Advantages of Dialogflow

There are various advantages of the Dialogflow:

1. Sentiment analysis
2. User-Friendly
3. Incorporates Google features
4. Swift and more efficient coding
5. Can express itself via natural conversation
6. Can handle small talks
7. Powered by Google's machine learning.

1. Sentiment Analysis: - The Dialogflow can perform sentiment analysis for the queries of the user.

2. User-Friendly: - The Dialogflow is a user-friendly tool, and built with a serverless application structure and an integrated code editor.

3. Incorporates Google feature: - The Dialogflow includes various features of Google, such as speech-to-text and machine learning.

4. Swift and More Efficient Coding: - By simplifying the coding process, the Dialogflow helps us to save the time of the developers. The developers can efficiently perform all the tasks related to the coding because the system contains a built-in, inline code editor. By using this, their agents can be linked via Cloud functions or on-premise to their application.

5. Can Express itself via Natural Conversation: - With the help of Dialogflow, we can create a chatbot. And the chatbots can make a conversation in a natural manner. It simply means that although the customers speak to an application program or a computer, generally to ask for help or assistance, they will obtain in-context replies. When you communicate in a chatbot, then you will not feel like you are talking with a robotic or mechanical.

6. Powered by Google's Machine Learning: - In Dialogflow, the machine learning technology is used and it increase the Dialogflow's capabilities.

Why choose Dialogflow

There are various reasons for choosing Dialogflow:

1. Multi-channel easy integration
2. Price
3. Natural Language Processing

1. Multi-channel Easy Integration: - The Dialogflow offers you single-click integrations to various types of the most popular messaging applications such as *Twitter, Skype, Kik, Viber, Facebook Messenger, Telegram, Slack, Twilio*, and also for several voice assistants such as *Microsoft Cortana, Amazon Alexa*, and *Google Assistant*.

2. Price: - The free edition of Dialogflow is also available if you only learn how to create a chatbot.

3. Natural Language Processing: - The Dialogflow uses the concept of NLP (Natural Language Processing), so the Dialogflow offers a good user experience. The agents of Dialogflow are better at NLP.

Agents

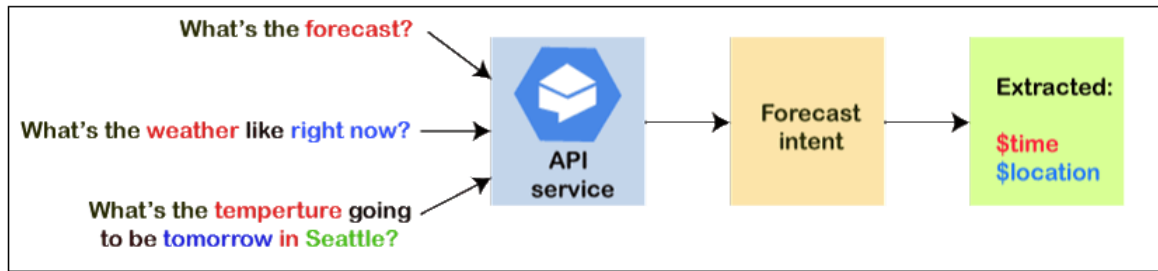
The Dialogflow agent is defined as a virtual agent whose task is to manage the end -user conversations. An agent is a module that can understand the complexities of the human language. During a conversation, the Dialogflow converts the text of end-user or audio into the structured data so that your applications can understand this. The Dialogflow agent is designed to manage the kinds of conversation your system needs.

The Dialogflow agent is the same as an agent in a human call center. Both are trained to handle the conversation scenario and don't need to be over explicit in their training,

Intents

An intent classifies the intention of an end-user for one conversation turn. For every agent, we have to define various users, and our combined agent is capable of handling an entire conversation. At the time, when the end-user says or writes something which is denoted as end-user expression, then the Dialogflow check and matches the end-user expression to your agent's best intent. Intent matching is also called *Intent Classification*.

For example: If you need to make an agent for the weather, which has the capability to identify and respond to the queries of the end-users related to the weather. Then you have to define an intent for the weather forecasting questions. When an end-user asks, "what is the forecast?" then the Dialogflow check and match the end-user expression to the intent of the forecast. If we want to obtain the essential information from the end-user expression such as location, time for the weather forecast according to our desire, then we have to define our intent. The obtained information is essential to the system for performing the queries related to the system.



The following things are comprised in the basic intent:

1. Training Phrases
2. Action
3. Parameters
4. Responses

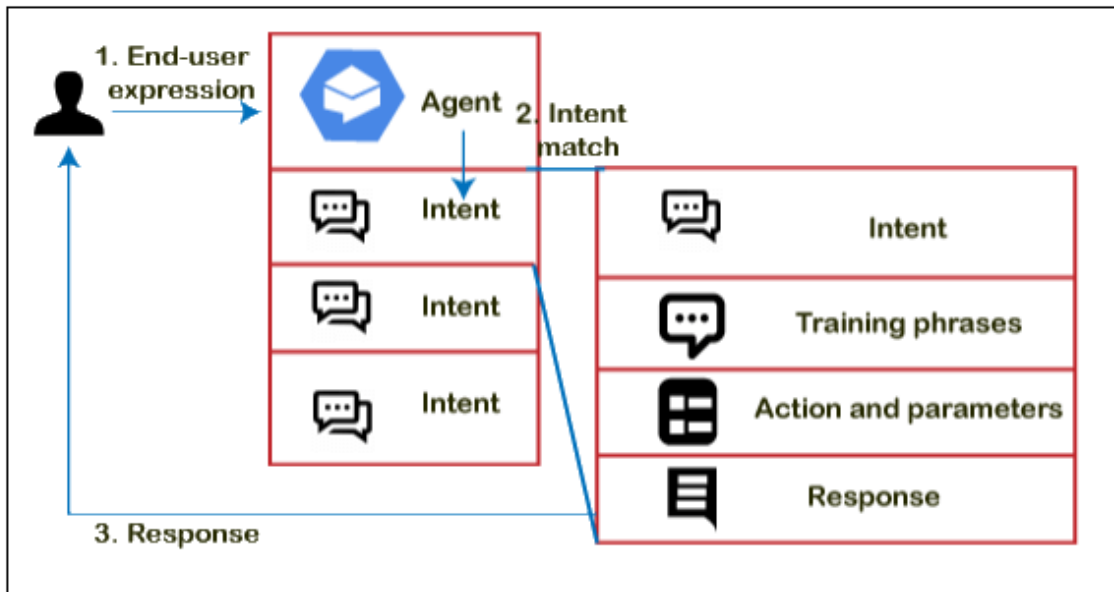
1. Training Phrases: - Training phrases means an example phrases for what the end-users can say. If one of these phrases resembles an end-user expression, then the Dialogflow matches the intent. There is no need to define each possible example as the built-in machine learning of Dialogflow expands with other related phrases on your list.

2. Action: - For the agent, we can define an action. At the time, when we match an intent, then the Dialogflow gives the actions to the system, and the action can be used to trigger various actions that are already defined in the system.

3. Parameters: - At run time, if we want to march an intent then as a parameter the Dialogflow gives the value from the expressions of the end-user. Every parameter contains a type known as an entity type, and the entity type exactly dictates how data is retrieved. The parameters are not like raw input of end-user. Parameter means structured data, which is used to perform logic or produce responses.

4. Responses: - We can define speech, visual, or text replies to return to the end-user. These are able to give the answer to the end-user and also ask for more information to the end-user and can also terminate the conversation.

The below figure shows the basic flow for intent matching and responding to the end-user.



Dialogflow Console

Dialogflow Console is defined as a web user interface which is provided by the Dialogflow. The Dialogflow console is used to build, test, and create agents. The Dialogflow console and Google Cloud Platform Console are different from each other. The main task of the Dialogflow is to control the Dialogflow agents, and the main task of the Google Cloud Platform Console (GCP) is to handle the settings of the Google Cloud Platform (GCP) Dialogflow such as billing and some other resources of the Google Cloud Platform (GCP).

With the help of Dialogflow Console, we can also create an agent.

Follow-up Intents

The **Follow-up Intents** can be used to create contexts automatically for the pairs of intents. A follow-up intent means a child of the intent of its associated parent. When the follow-up intent is created, then automatically an output context will be added to the parent intent, and in the follow-up intent, the input context having a similar name will be added. A follow-up intent is matched only in one situation, and the situation is, if the parent intent is in the turn of previous conversational. Multiple levels nested follow-up intents can also be created in the Dialogflow.

In the Dialogflow, there are various types of predefined follow-up intents for the replies of the common end-user such as cancel, yes, no, etc. To manage the responses of the customer, we can also create our own follow-up intent.

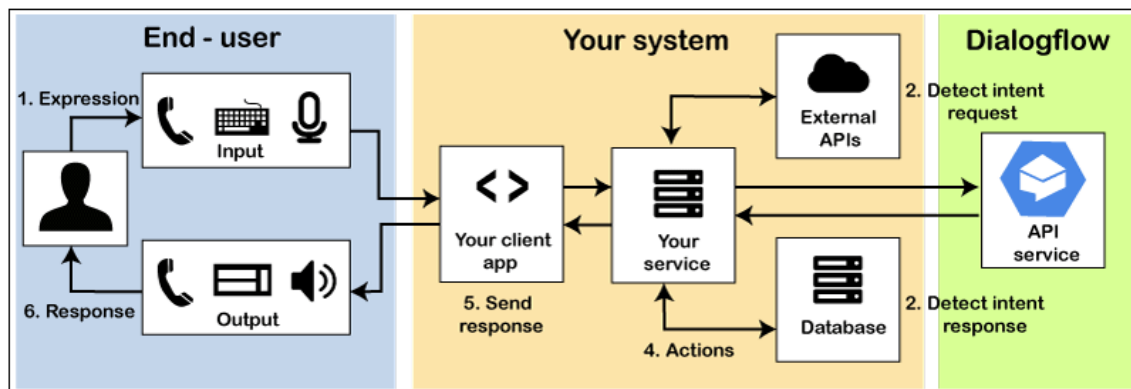
User Interactions with Integrations

There are various types of conversation platforms on which the Dialogflow can integrate, such as Facebook, Slack, Google Assistant, etc. To create an agent for any platforms, you can use one of the options from the various integrations' options. In the Dialogflow, there is a facility of handling direct end-user interactions so that your more focus is to create an agent.

User Interactions with the API

If any of the integration options you are not using, then you have to write a code that can interact with the end-user directly. For each conversational turn, you have to interact with the API of Dialogflow directly so that you can send the expression of end-user and obtain intent matches.

The below figure shows how the processing flows when interacting with the API.



1. The End-user is speaking or types the phrases.
2. In the detect intent request message, your service sends the end-user expression to the Dialogflow.
3. Then, the Dialogflow sends your service a message of intent to detect an answer. The information include in the message are related to the parameter, action, the responses and the matched intent declared for the intent.
4. Your service performs those actions which are needed, such as external API calls or Database queries.
5. Your service sends the replies to the end-user.
6. The end-user can hear or see the replies.

Components of the Dialogflow

There are various components of the Dialogflow:

1. Entity
2. Invocation
3. Fulfillment Request
4. Intents
5. Response
6. Context
7. User says

1. Entity: - Entity is defined as a knowledge repository that is used by the agent to answer the questions of the user. There are various types of system entities, such as weather, location, date, etc.

2. Invocation: - Invocation is like saying hello to a friend.

3. Fulfillment Request: - The Dialogflow send a request to retrieve the necessary data, (sent to webhook) the webhook performs the task like determine how to respond and how to send back to the Dialogflow.

4. Intents: - Intents comprises logic and elements to parse the information of the user. It helps to map what the users are saying with the responses. There are various components in the intent, such as events, response, the user says, contexts, and action.

5. Response: - The backend system will produce a set of responses, including user calls, webhook, intentions, entities, etc.

6. Context: - The context is used to store the values of the parameter for several kinds of intent. With the help of the context, the broken conversation can also be repaired.

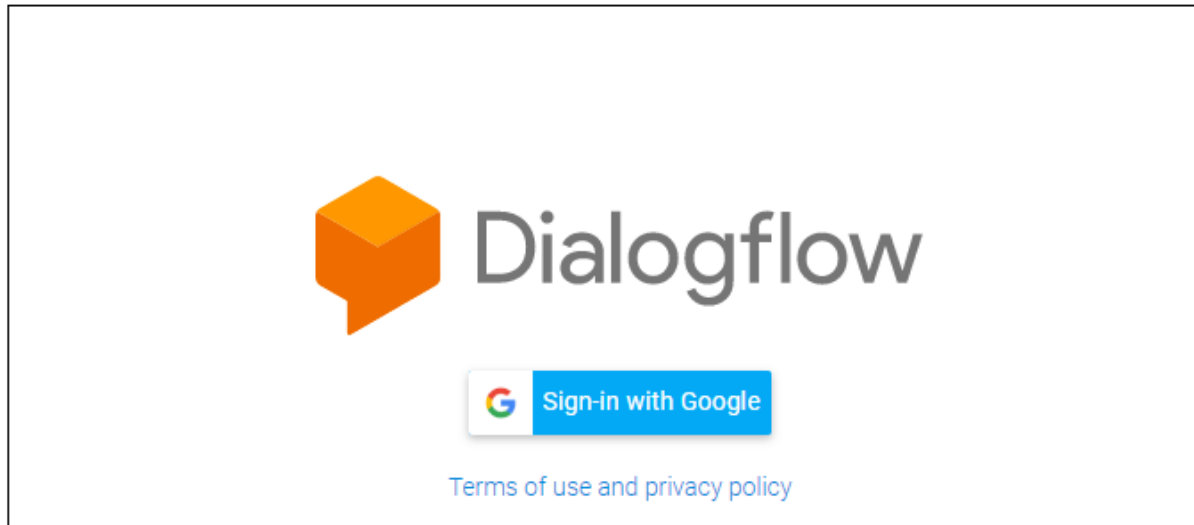
7. User Says: - User says means there are various forms of the same question that can be asked by the user. We can add more variations so that the agent can understand in a better way.

How to Create Your First Dialogflow Agent

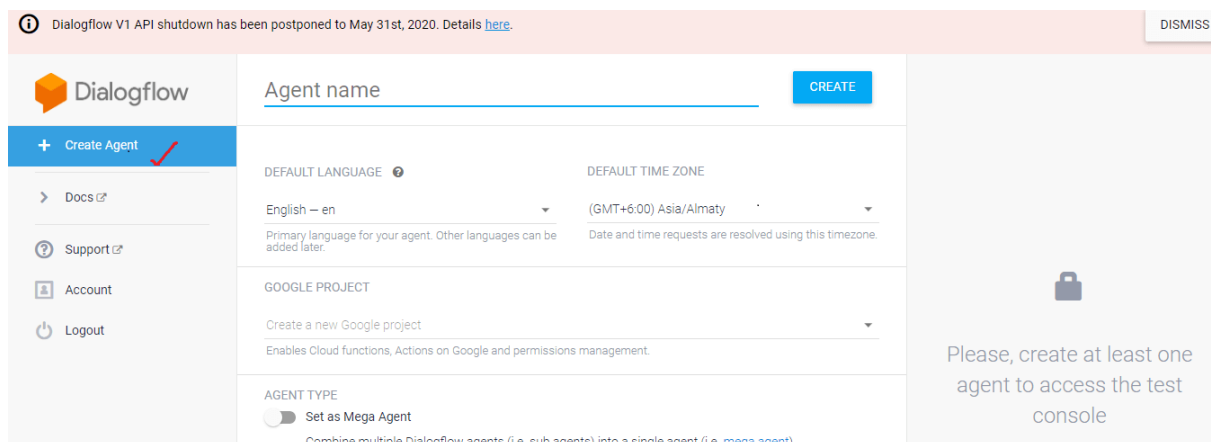
There are various steps to create your first Dialogflow agent:

1. First, we have to log in to the Dialogflow. We can log in to the Dialogflow by using the link <https://dialogflow.com/>. After login successfully, we have to click on the **Sign up for free**.

Then we have to connect with the Google account. If you want to use a Dialogflow, then it is must that you have a Google account.

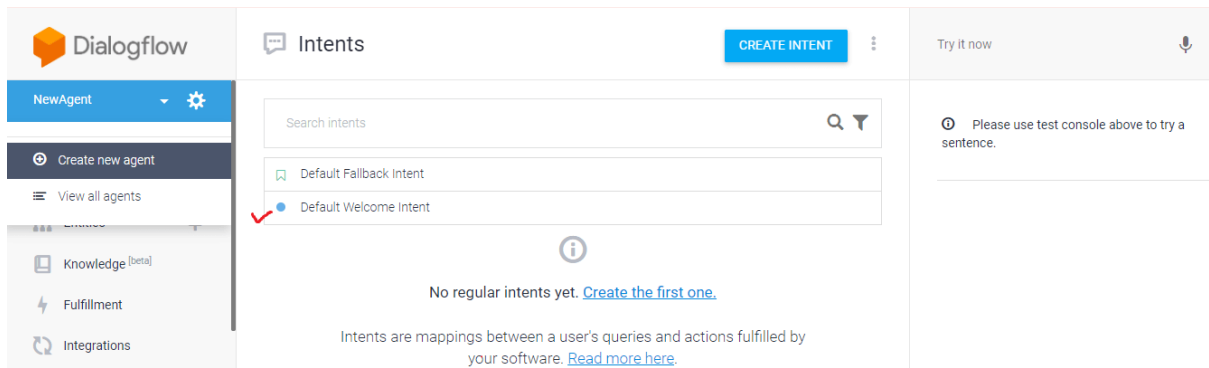


2. Next, we have to click on the **create agent** to create a new agent. For this, first select the three things: language, the default time zone, and the name for your new bot.

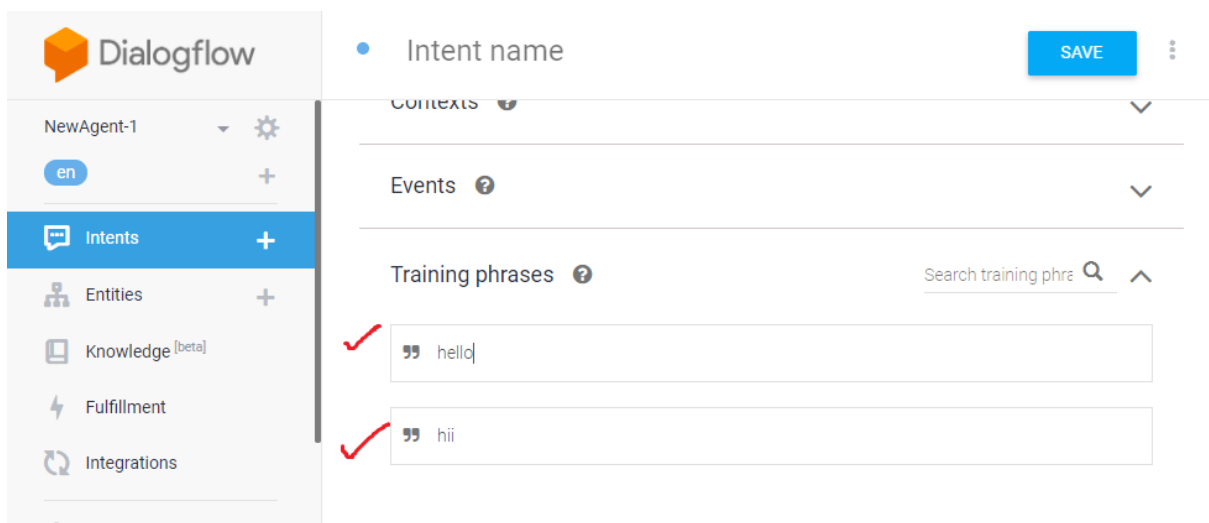
The image shows the "Create Agent" form in the Dialogflow console. At the top, there is a notification bar with an information icon, a message about the Dialogflow V1 API shutdown, and a "DISMISS" button. The form itself has a left sidebar with the Dialogflow logo and navigation links: "Create Agent" (highlighted with a red checkmark), "Docs", "Support", "Account", and "Logout". The main form area has a "CREATE" button at the top right. Below it are three sections: "DEFAULT LANGUAGE" with a dropdown set to "English - en", "DEFAULT TIME ZONE" with a dropdown set to "(GMT+6:00) Asia/Almaty", and "GOOGLE PROJECT" with a dropdown set to "Create a new Google project". At the bottom, there is an "AGENT TYPE" section with a radio button selected for "Set as Mega Agent". A right sidebar contains a briefcase icon and the text "Please, create at least one agent to access the test console".

3. Then, create the bot say hello. The bot currently doesn't know how to respond to the user input. The journey to teaching it how to behave is just beginning. First, you have to model the personality of the bot a bit and answer hello to him/her and then present yourself.

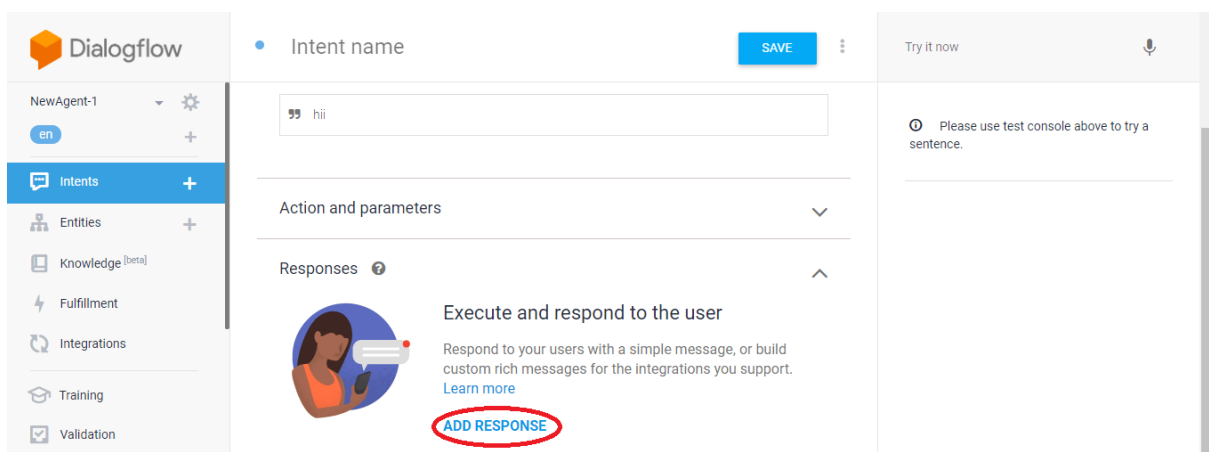
4. Click on the option named **Default Welcome Intent**.



5. Next, we have to add 'hello' and 'hi' to training phrases, and these will be in the text form and then click on the enter button.



6. Next, we have to go down on the option named **Responses** and remove or delete the existing ones.



8. Then, click on the **ADD RESPONSES**, and then we have to click on the **Text response** Then type the response which you want.

The screenshot shows the Dialogflow console interface. On the left is a sidebar with navigation options: NewAgent-1, Intents, Entities, Knowledge (beta), Fulfillment, Integrations, Training, and Validation. The main area is titled 'Intent name' with a 'SAVE' button. Below this, there are sections for 'Action and parameters' and 'Responses'. The 'Responses' section is expanded, showing a 'Text Response' tab. Under this tab, there are two response variants: 1. 'hello! my name is dimple and i am the virtual assistant of cool_name_restaurant would you like to book a table or learn more about restaurant beforehand' and 2. 'Enter a text response variant'. On the right, there is a 'Try it now' section with a microphone icon and a message: 'Please use test console above to try a sentence.'

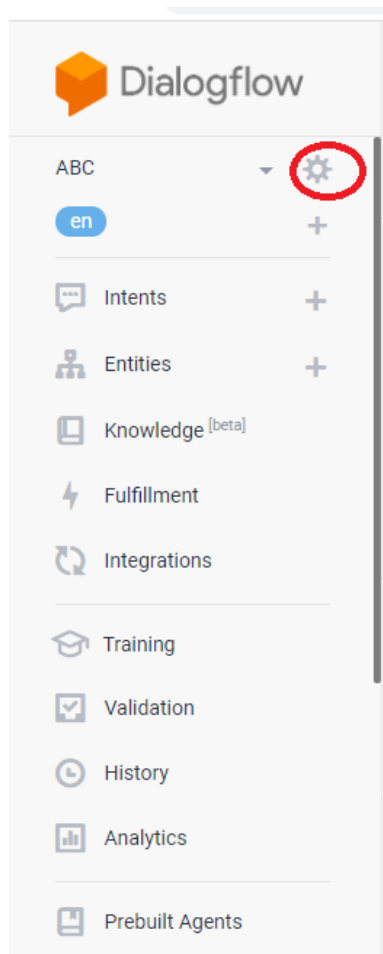
8. When you text the response appropriately, then you have to save the response by clicking on the **Save** button.

The screenshot shows the Dialogflow console interface. On the left is a sidebar with navigation options: ABC, Intents, Entities, Knowledge (beta), Fulfillment, Integrations, Training, Validation, History, and Analytics. The main area is titled 'Intent name' with a 'SAVE' button circled in red. Below this, there are sections for 'Contexts', 'Events', and 'Training phrases'. The 'Training phrases' section is expanded, showing a 'Train the intent with what your users will say' section. This section includes a description: 'Provide examples of how users will express their intent in natural language. Adding numerous phrases with different variations and parameters will improve the accuracy of intent matching. [Learn more](#)' and a button labeled 'ADD TRAINING PHRASES'. Below this, there is an 'Action and parameters' section with the text 'Extract the action and parameters'.

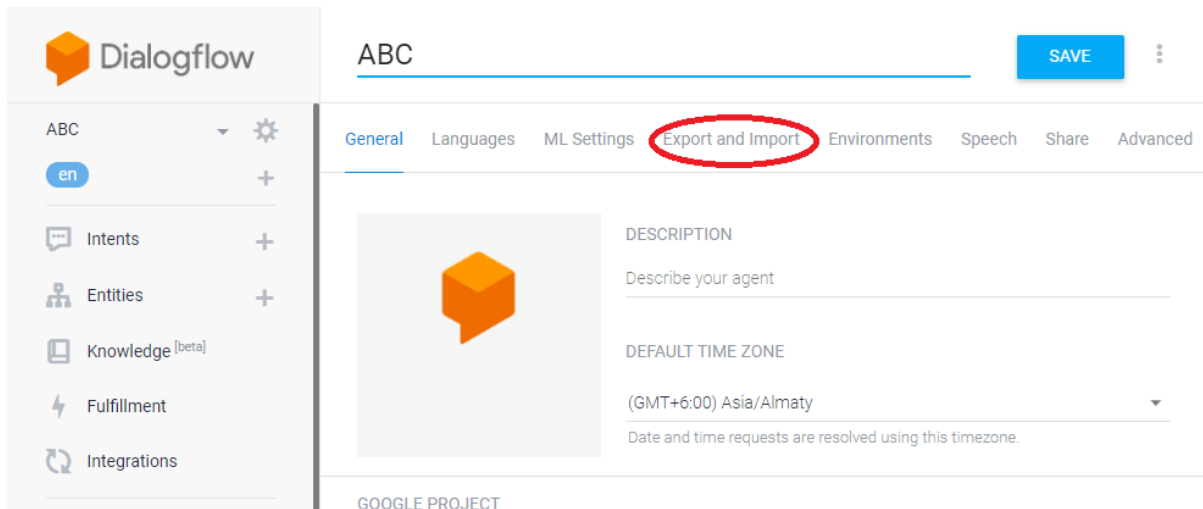
How to Import the Example File to your Agent

There are various steps to import the example file to your agent:

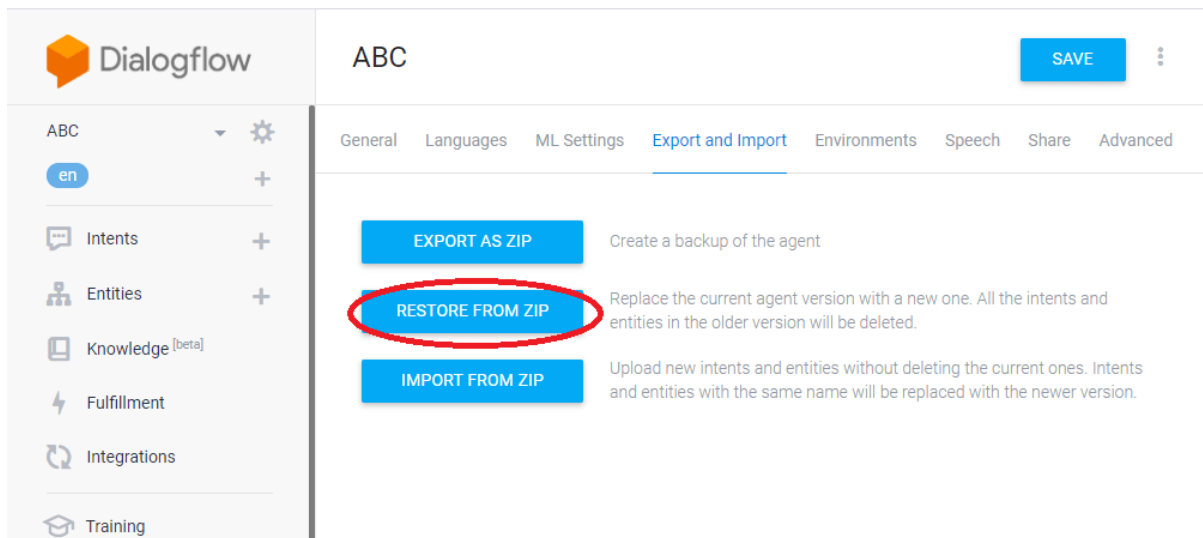
1. First, you have to download the new-agent.zip file.
2. Then go to the Dialogflow Console.
3. Then, choose an agent.
4. Next, click on the setting icon, which is present next to the agent name.



5. Then, choose the **Export and Import**



6. Last, select the option named Restore from Zip and then to restore the file which you downloaded, you have to follow some instructions on each step.

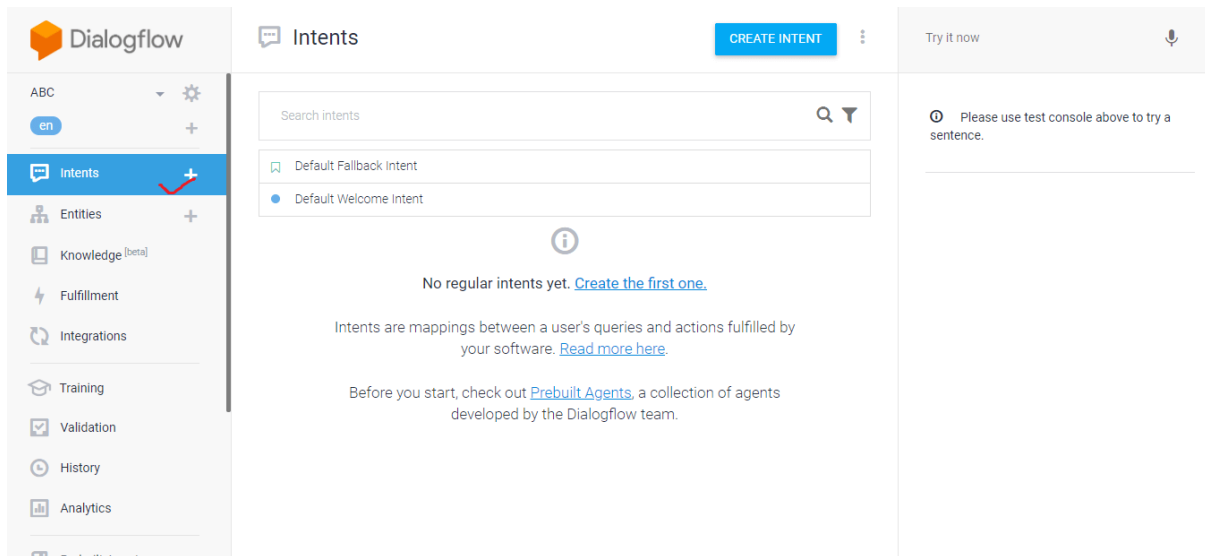


How to create a New Intent

In this section, we create an agent which is capable of answering the question such as "what is your name". for every intent we have to define various training phrases. A training phrases is also known as end-user expression. It is an example of what the possible questions the end-user can say or type to the agent. We can define any number of training phrases which offer Dialogflow, a different type of expressions which must match the intent.

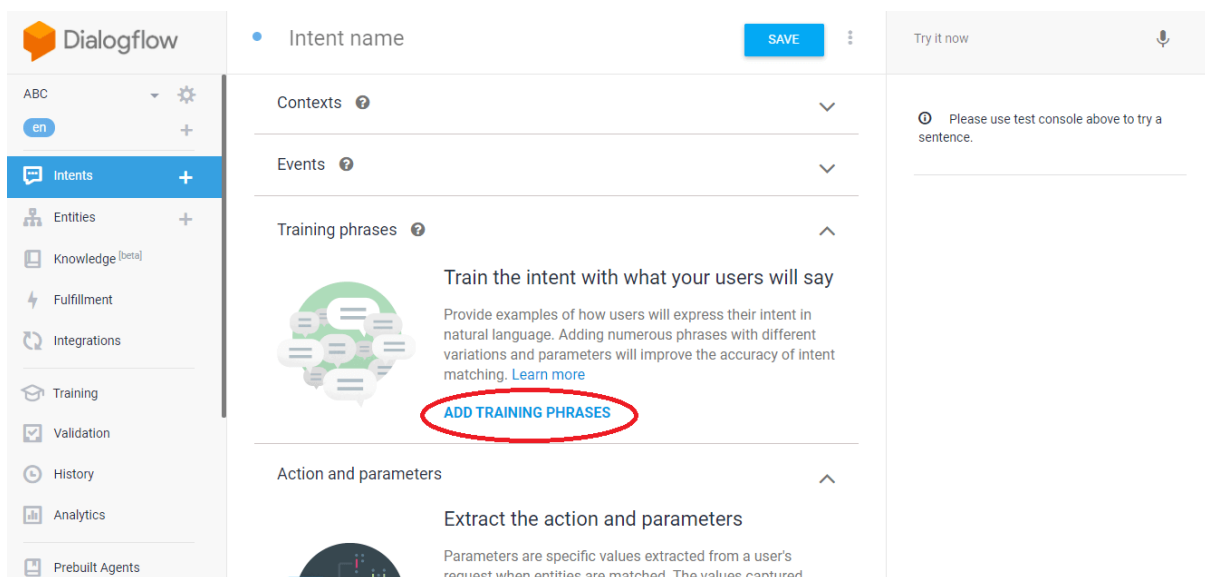
There are following steps to create a new intent:

1. First, we have to click on the add intent + button on the left sidebar menu next to the intents.



2. Next, click on the **get-agent-name**, which is present in the **Intent name**

3. Then, click on the option named **Add training phrases**, which is present in the **Training Phrases**



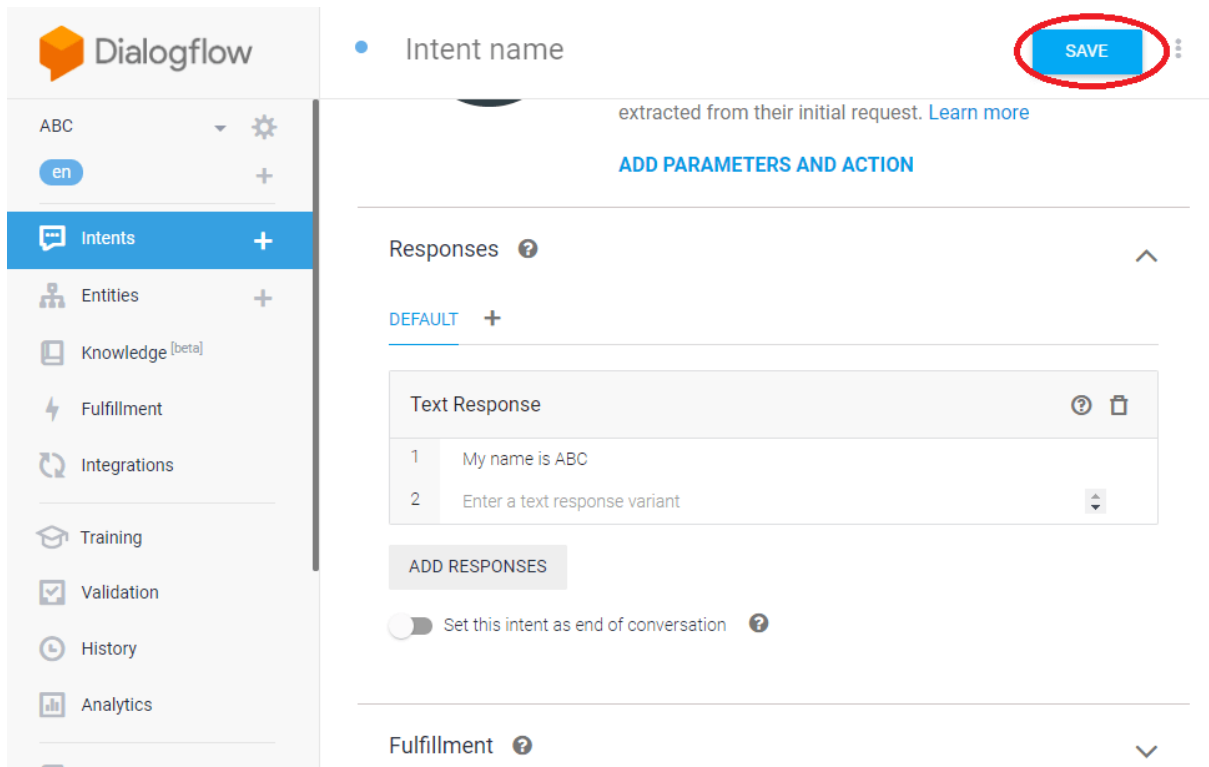
4. Next, we enter the training phrases according to the requirement of the Intent. Then we have to click enter after every entry.

- *What is your name?*
- *Tell me your name?*
- *Do you have a name?*

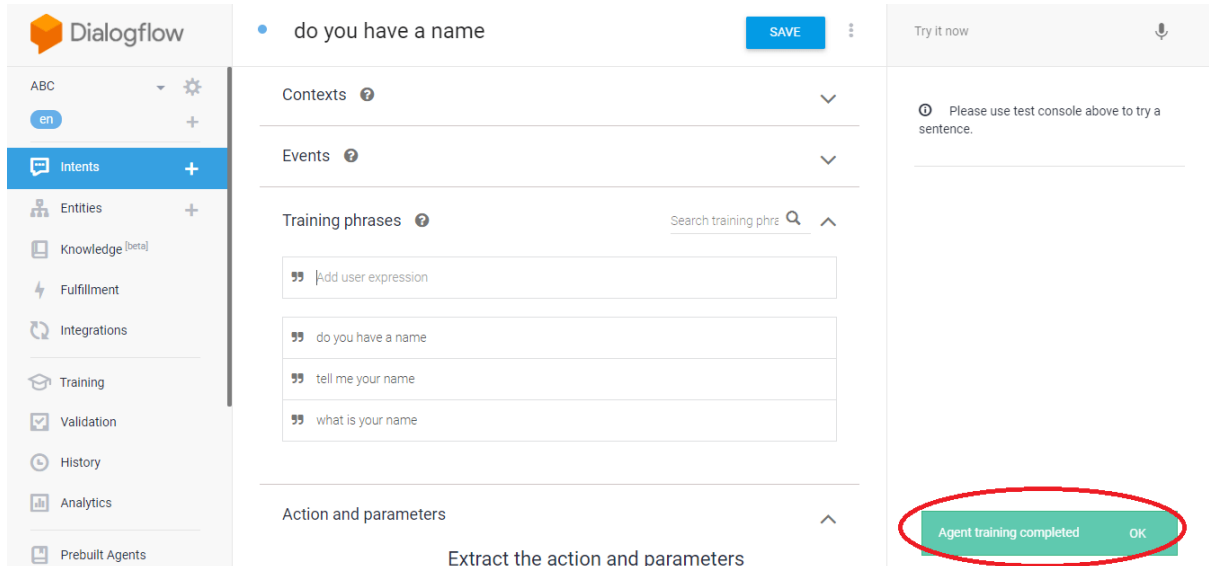
Note: In most of the cases, it is must that we have to enter at least 10-20 training phrases so that the intent matching is reliable.

- Then, in the **Response** section, we have to enter the response, such as "My name is ABC" in the **Text Response** section.

- Click on the Save button.



7. Then, we have to wait till the dialog of the Agent Training indicates that **training is complete**.



Type what is your name into the simulator? Click Enter.

Your agent gives correct responses to the expression, no matter if the expression may be a little bit dissimilar from the training phrases that you supplied.

For a machine learning model, the Dialogflow uses training phrases like an example, to match the expression of the end-user to the intents. Against each intent, in the agent, the model tests

the expressions and then provides a score to each intent, and then the intent which contains the highest score is matched.

When the intent having the highest-scoring has a very less score, then the fallback intent is matched.

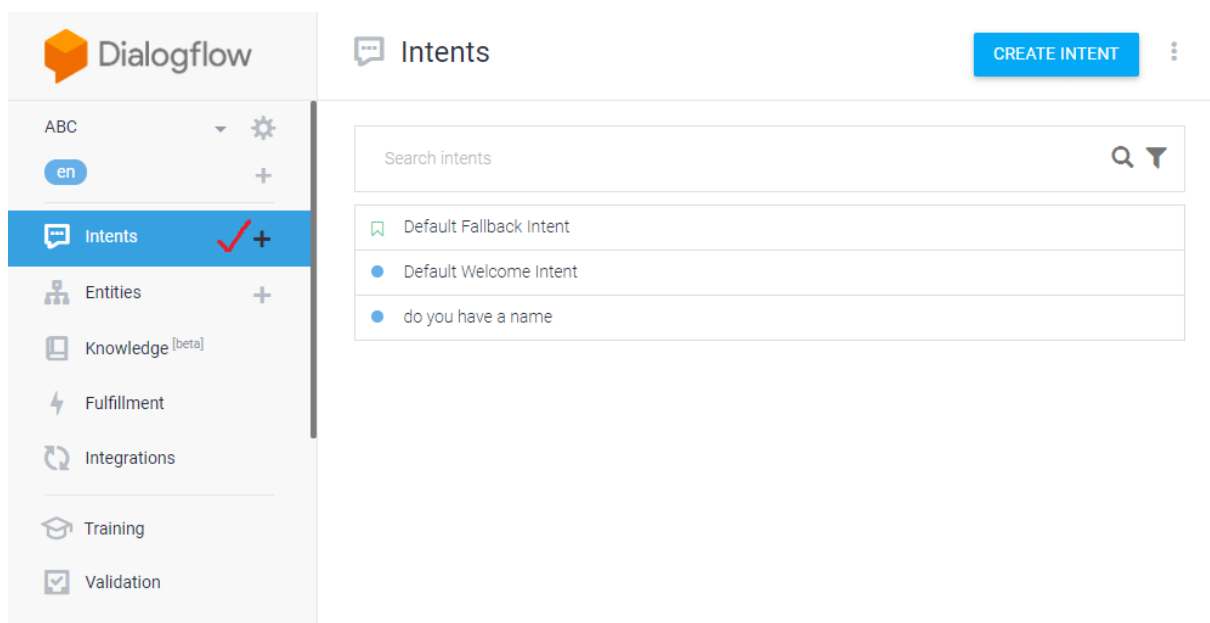
How to Create Parameters in Dialogflow

Whenever at runtime, the intent is matched, then as a parameter, the Dialogflow gives the values extracted from the end-user expression. And every parameter has a type which is known as an entity type. Type of entity dictates exactly how data is extracted.

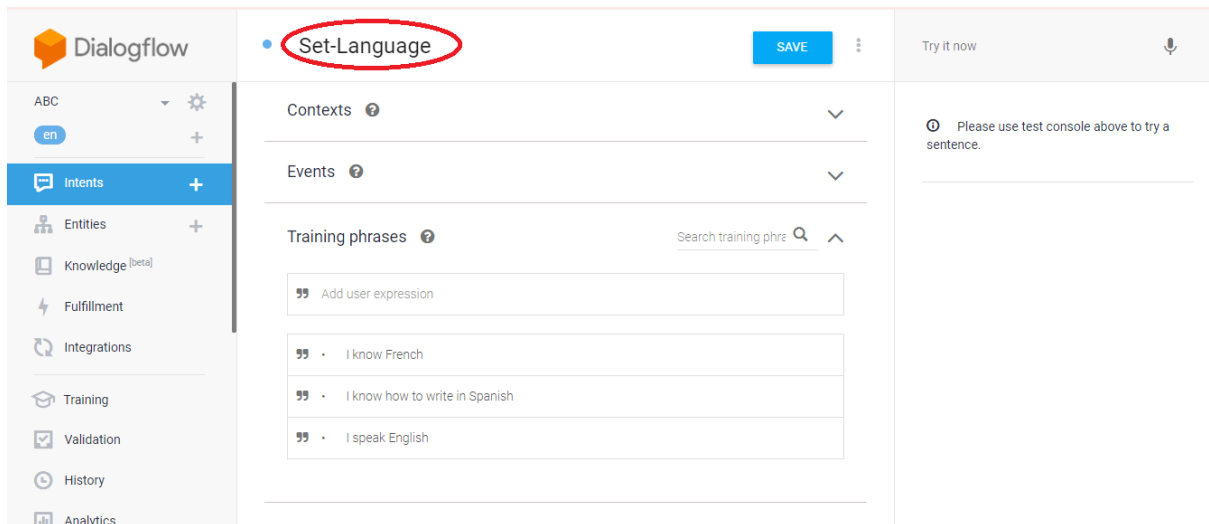
There are following steps to create a parameter:

(create a new intent with parameters)

1. First, we have to click on the add intent + button on the left sidebar menu next to the intents.

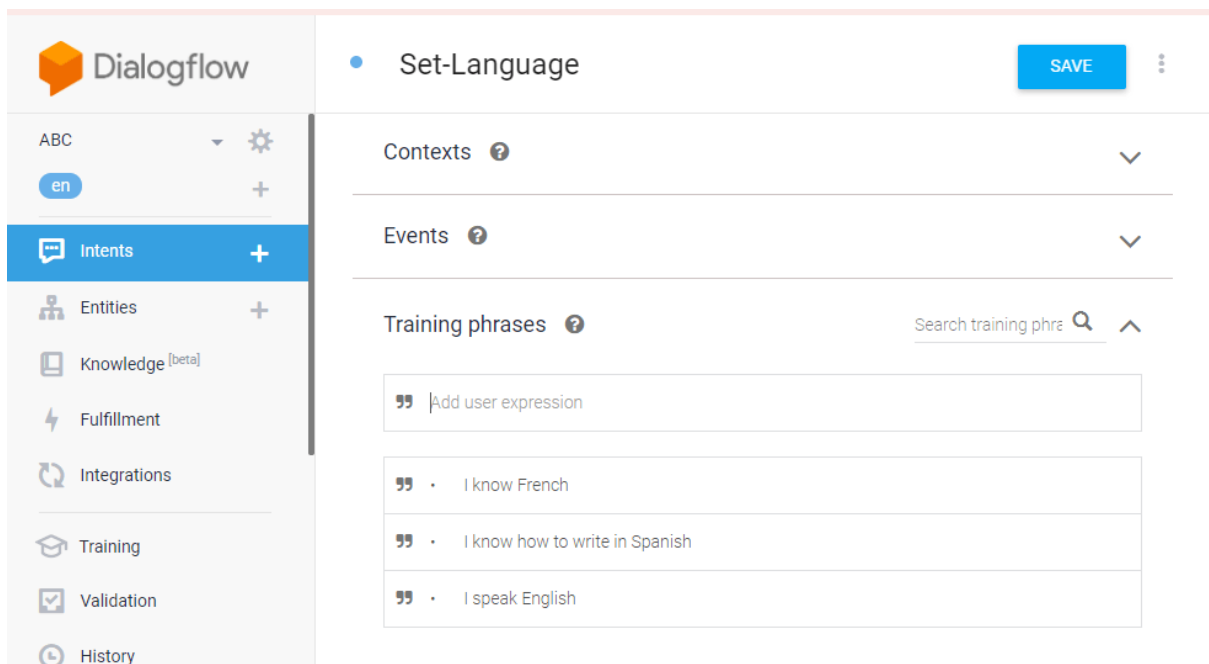


2. Then, we have to give the name to the intent, and the name is as **set-language**, which appears at the top side of the intent form.



3. Now, we have to add the following training phrases:

- I speak English
- I know how to write in Spanish
- I know French



4. Now, we click on the **Save** button, and then we have to wait for the **Agent Training** dialog indicating that **Agent training completed**.

The screenshot shows the Dialogflow 'Set-Language' configuration page. The left sidebar contains navigation options: Intents, Entities, Knowledge [beta], Fulfillment, Integrations, Training, Validation, History, and Analytics. The main area is titled 'Set-Language' and includes sections for Contexts, Events, Training phrases, and Action and parameters. The Training phrases section shows a list of phrases: 'I know French', 'I know how to write in Spanish', and 'I speak English'. A 'SAVE' button is circled in red. A green notification box at the bottom right indicates 'Agent training completed' with an 'OK' button, also circled in red.

Automatically, in the training phrases, the Dialogflow recognizes parameters which are known as system entities. These are the entities which the Dialogflow gives for various common data types like date, color, and location.

Note: We can annotate our training phrases manually if our training phrases are not annotated automatically.

In the **Action and parameters** table, the Dialogflow creates a row, which is present below the **Training phrases**.

The screenshot shows the Dialogflow 'set-language' configuration page. The left sidebar contains navigation options: Intents, Entities, Knowledge [beta], Fulfillment, Integrations, Training, Validation, History, and Analytics. The main area is titled 'set-language' and includes sections for Action and parameters and Responses. The Action and parameters section shows a table with the following data:

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
<input type="checkbox"/>	language	@sys.language	\$language	<input type="checkbox"/>

A '+ New parameter' link is visible below the table. The Responses section shows a 'DEFAULT' response.

- **Required:** - The required parameter is optional because the checkbox is not checked.

- **Parameter Name:** - The parameter name is language because it recognizes the parameter as a language.
- **Entity:** - Entity is a type of entity. The entity is recognized like **@language** system entity.
- **Value:** - Value is the identifier that you use whenever referring this parameter to its value.
- **Is List:** - The checkbox of **Is list** is not checked. Thus, the parameter is not a list.

Note: We can also manually annotate the entity in the training phrases if it is not detected automatically.

Test your Parameter.

If you want to test your parameter, then first you have to type 'I Know French' and then press enter.

The screenshot shows a chat interface with the following components:

- Try it now** button with a microphone icon.
- Agent** header.
- USER SAYS** section with the text "I know Spanish" and a **COPY CURL** link.
- DEFAULT RESPONSE** section with a dropdown arrow and the text "Slanguage-programming is a better programming language."
- CONTEXTS** section with a **RESET CONTEXTS** link and a box containing the text "set-language-followup".
- INTENT** section (partially visible).

We can easily find that the Dialog extracts the **language** parameter appropriately with the value Spanish and Spanish is appropriately inserted where the reference of the parameter was used in the response.

SOURCE CODE

USER - DASHBOARD

```
{% load static %}
<!DOCTYPE html>
<html lang="zxx">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <meta name="description" content="">
  <meta name="author" content="codebook.in">

  <title>User Dashboard</title>

  <!-- bootstrap.min css -->
  <link rel="stylesheet" href="{% static
'userapp/plugins/bootstrap/css/bootstrap.min.css' %}">
  <!-- Icon Font CSS -->
  <link rel="stylesheet" href="{% static
'userapp/plugins/icofont/icofont.min.css' %}">
  <!-- Slick Slider CSS -->
  <link rel="stylesheet" href="{% static 'userapp/plugins/slick-
carousel/slick/slick.css' %}">
  <link rel="stylesheet" href="{% static 'userapp/plugins/slick-
carousel/slick/slick-theme.css' %}">

  <!-- Main Stylesheet -->
  <link rel="stylesheet" href="{% static 'userapp/css/style.css' %}">
  <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
</head>

<body id="top">
  {% if messages %}
  {% for message in messages %}
    {% if message.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
      <script>swal({
        title: "Success!",
        text: "{{message}}",
        icon: "success",
        button: "OK",
      });
      </script>

    {% elif message.level == DEFAULT_MESSAGE_LEVELS.WARNING %}
      <script>swal({
        title: "Warning :",
        text: "{{message}}",
        icon: "warning",
```



```

        button: "OK",

    });
</script>
{% elif message.level == DEFAULT_MESSAGE_LEVELS.INFO %}
    <script>swal({
        title: "info :)",
        text: "{{message}}",
        icon: "info",
        button: "OK",

    });
</script>
{% elif message.level == DEFAULT_MESSAGE_LEVELS.ERROR %}
    <script>swal({
        title: "error :)",
        text: "{{message}}",
        icon: "error",
        button: "OK",

    });
</script>
{% endif %}
{% endfor %}
{% endif %}

<header>

<nav class="navbar navbar-expand-lg navigation mb-2" id="navbar">
    <div class="container">
        <a class="navbar-brand" href="">
            <img src="" alt="" class="img-fluid">
        </a>

        <button class="navbar-toggler collapsed" type="button" data-
toggle="collapse" data-target="#navbarmain" aria-controls="navbarmain" aria-
expanded="false" aria-label="Toggle navigation">
            <span class="icofont-navigation-menu"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarmain">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active"><a class="nav-link" href="{% url
'user_dash' %}">Dashboard</a></li>
                <li class="nav-item"><a class="nav-link" href="{% url
'user_profile' %}">My Profile</a></li>
                <li class="nav-item"><a class="nav-link" href="{% url 'home'
%}">Logout</a></li>
            </ul>
        </div>
    </div>

```

```

        </div>
    </nav>
</header>

<!-- content start -->

<section class="page-title bg-2">
    <div class="overlay"></div>

    <div class="container">

        <div class="row">
            <div class="col-md-12">
                <div class="block text-center">
                    <h1 class="text-capitalize mb-5 text-lg">User Dashboard</h1>

                </div>
            </div>
        </div>
    </div>
</section>

<section class="section service bg-gray">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-lg-7 text-center">
                <div class="section-title">
                    <h2>ADVANCED HEALTHCARE CHATBOT FOR DISEASE DIAGNOSIS AND
TREATMENT RECOMMENDATION</h2>
                    <div class="divider mx-auto mb-4"></div>
                    <p>"Introducing our cutting-edge healthcare chatbot,
revolutionizing disease diagnosis and treatment recommendations with advanced
technology and personalized care."</p>
                </div>
            </div>
        </div>

        <div class="row">
            <div class="col-lg-4 col-md-6 col-sm-6">
                <div class="service-item mb-4">
                    <div class="icon d-flex align-items-center">
                        <i class="icofont-laboratory text-lg"></i>
                        <h4 class="mt-3 mb-3">Laboratory services</h4>
                    </div>

```

```
        <div class="content">
            <p class="mb-4">"Empowering health through precision:
Our dynamic laboratory services lead the way to accurate diagnoses and
tailored treatments."</p>
        </div>
    </div>
</div>
```

```
<div class="col-lg-4 col-md-6 col-sm-6">
    <div class="service-item mb-4">
        <div class="icon d-flex align-items-center">
            <i class="icofont-heart-beat-alt text-lg"></i>
            <h4 class="mt-3 mb-3">Heart Disease</h4>
        </div>
        <div class="content">
            <p class="mb-4">"Empowering health through precision:
Our dynamic laboratory services lead the way to accurate diagnoses and
tailored treatments."</p>
        </div>
    </div>
</div>
```

```
<div class="col-lg-4 col-md-6 col-sm-6">
    <div class="service-item mb-4">
        <div class="icon d-flex align-items-center">
            <i class="icofont-tooth text-lg"></i>
            <h4 class="mt-3 mb-3">Dental Care</h4>
        </div>
        <div class="content">
            <p class="mb-4">"Empowering health through precision:
Our dynamic laboratory services lead the way to accurate diagnoses and
tailored treatments."</p>
        </div>
    </div>
</div>
```

```
<div class="col-lg-4 col-md-6 col-sm-6">
    <div class="service-item mb-4">
        <div class="icon d-flex align-items-center">
            <i class="icofont-crutch text-lg"></i>
            <h4 class="mt-3 mb-3">Body Surgery</h4>
        </div>

        <div class="content">
            <p class="mb-4">"Empowering health through precision:
Our dynamic laboratory services lead the way to accurate diagnoses and
tailored treatments."</p>
        </div>
    </div>
```

```

        </div>
    </div>

    <div class="col-lg-4 col-md-6 col-sm-6">
        <div class="service-item mb-4">
            <div class="icon d-flex align-items-center">
                <i class="icofont-brain-alt text-lg"></i>
                <h4 class="mt-3 mb-3">Neurology Sargery</h4>
            </div>
            <div class="content">
                <p class="mb-4">"Empowering health through precision:
Our dynamic laboratory services lead the way to accurate diagnoses and
tailored treatments."</p>
            </div>
        </div>
    </div>

    <div class="col-lg-4 col-md-6 col-sm-6">
        <div class="service-item mb-4">
            <div class="icon d-flex align-items-center">
                <i class="icofont-dna-alt-1 text-lg"></i>
                <h4 class="mt-3 mb-3">Gynecology</h4>
            </div>
            <div class="content">
                <p class="mb-4">"Empowering health through precision:
Our dynamic laboratory services lead the way to accurate diagnoses and
tailored treatments."</p>
            </div>
        </div>
    </div>
</div>
</div>
</section>

<section class="section team">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-lg-6">
                <div class="section-title text-center">
                    <h2 class="mb-4">TEAM MEMBERS</h2>
                    <div class="divider mx-auto my-4"></div>
                    <p>"Meet the exceptional team members driving our project
forward with expertise, dedication, and collaboration."</p>
                </div>
            </div>
        </div>
    </div>

    <div class="row">

```

```

        <div class="col-lg-3 col-md-6 col-sm-6">
            <div class="team-block mb-5 mb-lg-0">
                

                <div class="content">
                    <h4 class="mt-4 mb-0"><a href="">Mr. Lodagala
Teja</a></h4>

                    <p>Team Leader, Project Developer</p>
                </div>
            </div>
        </div>
    </div>

```

```

        <div class="col-lg-3 col-md-6 col-sm-6">
            <div class="team-block mb-5 mb-lg-0">
                

                <div class="content">
                    <h4 class="mt-4 mb-0"><a href="">Ms. Ratnala
Manasa</a></h4>

                    <p>Team Member, Front-End Developer</p>
                </div>
            </div>
        </div>
    </div>

```

```

        <div class="col-lg-3 col-md-6 col-sm-6">
            <div class="team-block mb-5 mb-lg-0">
                

                <div class="content">
                    <h4 class="mt-4 mb-0"><a href="">Ms. Vanapalli
Priyanka</a></h4>

                    <p>Team Member, Back-End Developer</p>
                </div>
            </div>
        </div>
    </div>

```

```

        <div class="col-lg-3 col-md-6 col-sm-6">
            <div class="team-block">
                

                <div class="content">
                    <h4 class="mt-4 mb-0"><a href="">Mr. Konna Ravi
Teja</a></h4>

                    <p>Team Member</p>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
</div>
</div>
</section>

<script src="https://www.gstatic.com/dialogflow-
console/fast/messenger/bootstrap.js?v=1"></script>
<df-messenger
    chat-icon="https://freemsg.org/img/1538298822.png"
    intent="WELCOME"
    chat-title="HealthCare ChatBot"
    agent-id="d7aa38e2-493e-4c03-8a43-c906773fabb0"
    language-code="en"
></df-messenger>

<!-- content end -->


<!-- footer Start -->
<footer class="footer section gray-bg">
    <div class="container">
        <div class="row">
            <div class="col-lg-4 mr-auto col-sm-6">
                <div class="widget mb-4 mb-lg-0">
                    <div class="logo mb-4">
                        
                    </div>
                    <p>Social Media Accounts</p>

                    <ul class="list-inline footer-socials mt-4">
                        <li class="list-inline-item"><a
href="https://www.facebook.com/tejalodagala480"><i class="icofont-
facebook"></i></a></li>
                        <li class="list-inline-item"><a
href="https://twitter.com/tejalodagala480"><i class="icofont-
twitter"></i></a></li>
                        <li class="list-inline-item"><a
href="https://www.linkedin.com/in/teja-lodagala-317686207"><i class="icofont-
linkedin"></i></a></li>
                    </ul>
                </div>
            </div>

            <div class="col-lg-3 col-md-6 col-sm-6">
                <div class="widget widget-contact mb-5 mb-lg-0">

```

```
<h4 class="text-capitalize mb-3">Get in Touch</h4>  
<div class="divider mb-4"></div>  
  
<div class="footer-contact-block mb-3">  
    <div class="icon d-flex align-items-center">  
        <i class="icofont-email mr-3"></i>  
        <span class="h6 mb-0">Support Available for  
24/7</span>  
    </div>  
    <h4 class="mt-2"><a href="tel:+23-345-  
67890">tejalodagala480@gmail.com</a></h4>  
</div>  
  
<div class="footer-contact-block">  
    <div class="icon d-flex align-items-center">  
        <i class="icofont-support mr-3"></i>  
        <span class="h6 mb-0">Mon to Fri : 08:30 -  
18:00</span>  
    </div>  
    <h4 class="mt-2"><a href="tel:+23-345-67890">+91  
9959353902</a></h4>  
</div>  
</div>  
</div>  
</div>  
</div>  
  
<div class="footer-btm py-2 mt-3">  
    <div class="row align-items-center justify-content-center">  
        <div class="col-lg-auto">  
            <div class="copyright">  
                &copy; Copyright Reserved to <span class="text-  
color">Lodagala Teja</span>  
            </div>  
        </div>  
    </div>  
</div>  
  
<div class="row">  
    <div class="col-lg-4">  
        <a class="backtop js-scroll-trigger" href="#top">  
            <i class="icofont-long-arrow-up"></i>  
        </a>  
    </div>  
</div>  
</div>  
</div>  
</div>  
</div>
```

```

<!--
Essential Scripts
=====-->

<!-- Main jQuery -->
<script src="{% static 'userapp/plugins/jquery/jquery.js' %}"></script>
<!-- Bootstrap 4.3.2 -->
<script src="{% static 'userapp/plugins/bootstrap/js/popper.js'
%}"></script>
<script src="{% static 'userapp/plugins/bootstrap/js/bootstrap.min.js'
%}"></script>
<script src="{% static 'userapp/plugins/counterup/jquery.easing.js'
%}"></script>
<!-- Slick Slider -->
<script src="{% static 'userapp/plugins/slick-carousel/slick/slick.min.js'
%}"></script>
<!-- Counterup -->
<script src="{% static 'userapp/plugins/counterup/jquery.waypoints.min.js'
%}"></script>

<script src="{% static 'userapp/plugins/shuffle/shuffle.min.js'
%}"></script>
<script src="{% static 'userapp/plugins/counterup/jquery.counterup.min.js'
%}"></script>

<script src="{% static 'userapp/js/script.js' %}"></script>
<script src="{% static 'userapp/js/contact.js' %}"></script>

</body>
</html>

```


CHAPTER - 7

TESTING

SYSTEM TESTING

System Testing for an Advanced Healthcare Chatbot:

System testing for an advanced healthcare chatbot involves verifying and validating the functionality, performance, and usability of the chatbot to ensure that it meets the specified requirements and quality standards. Here's an overview of the different types of system testing that can be conducted:

1. Functional Testing:

- Test the chatbot's core functionalities, including symptom assessment, disease diagnosis, and treatment recommendations.
- Verify that the chatbot accurately interprets user input, provides relevant responses, and follows the intended conversation flow.
- Conduct tests to ensure that the chatbot handles various scenarios, such as different symptom combinations, user queries, and edge cases.

2. Integration Testing:

- Test the integration of the chatbot with external systems, such as healthcare databases, electronic health records (EHRs), and telemedicine platforms.
- Verify that data exchange and communication between the chatbot and external systems are seamless and error-free.
- Conduct tests to validate interoperability and compatibility with industry standards and protocols.

3. Performance Testing:

- Measure the chatbot's performance in terms of response time, throughput, and scalability.
- Conduct load testing to simulate high-volume usage scenarios and assess how the chatbot handles concurrent user interactions.
- Verify that the chatbot can handle peak loads without degradation in performance or responsiveness.

4. Usability Testing:

- Evaluate the chatbot's user interface and interaction design for ease of use, intuitiveness, and clarity.
- Conduct user testing sessions to gather feedback on the chatbot's usability, navigation, and overall user experience.
- Identify usability issues, such as confusing prompts, unclear instructions, or difficulties in understanding user queries, and address them accordingly.

5. Security Testing:

- Assess the chatbot's security measures to ensure the confidentiality, integrity, and availability of sensitive medical information.
- Conduct vulnerability assessments and penetration tests to identify potential security vulnerabilities, such as data breaches or unauthorized access.
- Verify that encryption, access controls, and other security mechanisms are properly implemented and effective in protecting user data.

6. Regression Testing:

- Conduct regression tests to ensure that recent updates or changes to the chatbot's codebase do not introduce new defects or regressions.
- Re-run previously executed test cases to verify that existing functionalities still work as expected after modifications or enhancements.

7. User Acceptance Testing (UAT):

- Involve end users, including patients and healthcare professionals, in user acceptance testing to validate that the chatbot meets their needs and expectations.
- Gather feedback from users regarding the chatbot's accuracy, relevance, and usefulness in assisting with symptom assessment, diagnosis, and treatment recommendations.

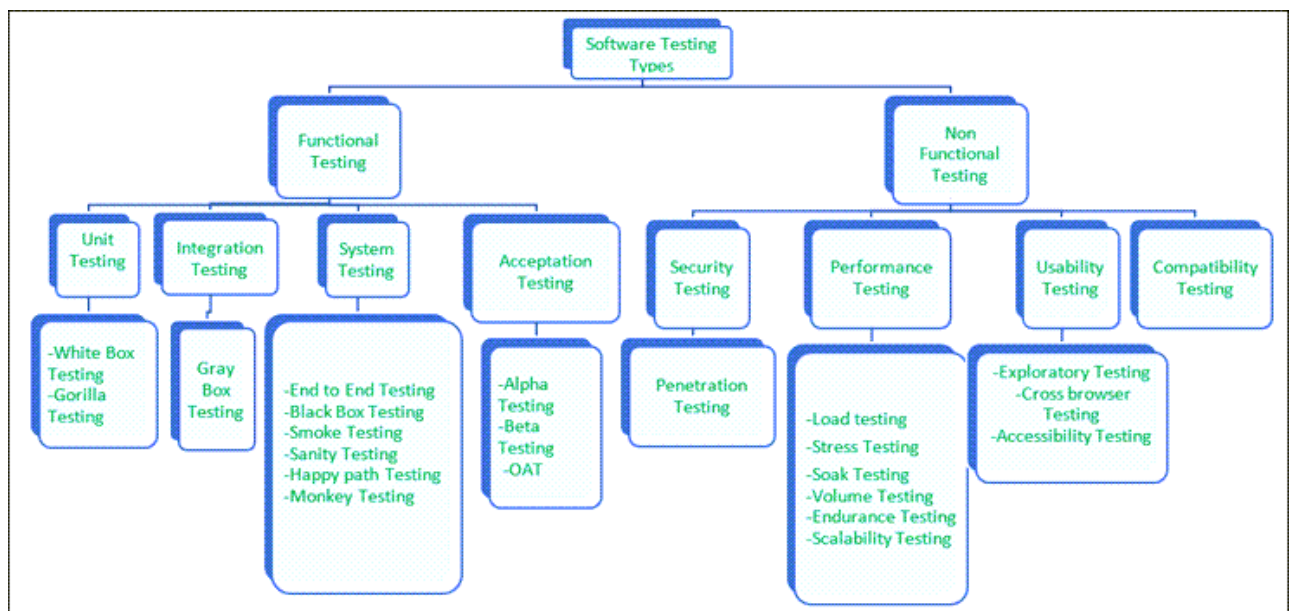
By conducting comprehensive system testing across these areas, developers can ensure that the advanced healthcare chatbot meets quality standards, performs reliably in real-world scenarios, and delivers a positive user experience for both patients and healthcare providers.

Types of Software Testing: Different Testing Types with Details

We, as testers, are aware of the various types of Software Testing like Functional Testing, Non-Functional Testing, Automation Testing, Agile Testing, and their sub-types, etc.

Each type of testing has its own features, advantages, and disadvantages as well. However, in this tutorial, we have covered mostly each and every type of software testing which we usually use in our day-to-day testing life.

Different Types of Software Testing



FUNCTIONAL TESTING

There are four main types of functional testing.

1) Unit Testing

Unit testing is a type of software testing which is done on an individual unit or component to test its corrections. Typically, Unit testing is done by the developer at the application development phase. Each unit in unit testing can be viewed as a method, function, procedure, or object. Developers often use test automation tools such as NUnit, Xunit, JUnit for the test execution.

Unit testing is important because we can find more defects at the unit test level.

For example, there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

a) White Box Testing

White box testing is a test technique in which the internal structure or code of an application is visible and accessible to the tester. In this technique, it is easy to find loopholes in the design of an application or fault in business logic. Statement coverage and decision coverage/branch coverage are examples of white box test techniques.

b) Gorilla Testing

Gorilla testing is a test technique in which the tester and/or developer test the module of the application thoroughly in all aspects. Gorilla testing is done to check how robust your application is.

For example, the tester is testing the pet insurance company's website, which provides the service of buying an insurance policy, tag for the pet, Lifetime membership. The tester can focus on any one module, let's say, the insurance policy module, and test it thoroughly with positive and negative test scenarios.

2) Integration Testing

Integration testing is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole. The focus of this type of testing is to find the defect on interface, communication, and data flow among modules. Top-down or Bottom-up approach is used while integrating modules into the whole system.

This type of testing is done on integrating modules of a system or between systems. **For example**, a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

a) Gray box testing

As the name suggests, Gray box testing is a combination of white-box testing and black-box testing. Testers have partial knowledge of the internal structure or code of an application.

3) System Testing

System testing is types of testing where tester evaluates the whole system against the specified requirements.

a) End to End Testing

It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

For example, a tester is testing a pet insurance website. End to End testing involves testing of buying an insurance policy, LPM, tag, adding another pet, updating credit card information on users' accounts, updating user address information, receiving order confirmation emails and policy documents.

b) Black Box Testing

Blackbox testing is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects.

Detailed information about the advantages, disadvantages, and types of Black Box testing can be found [here](#).

c) Smoke Testing

Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level.

Whenever a new build is provided by the development team, then the Software Testing team validates the build and ensures that no major issue exists. The testing team will ensure that the build is stable, and a detailed level of testing will be carried out further.

For example, tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the application. Smoke testing for this website verifies that all these functionalities are working fine before doing any in-depth testing.

d) Sanity Testing

Sanity testing is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test.

For example, a tester is testing a pet insurance website. There is a change in the discount for buying a policy for second pet. Then sanity testing is only performed on buying insurance policy module.

e) Happy path Testing

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on valid and positive inputs through which the application generates the expected output.

f) Monkey Testing

Monkey Testing is carried out by a tester, assuming that if the monkey uses the application, then how random input and values will be entered by the Monkey without any knowledge or understanding of the application.

The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly, no test cases are scripted, and it is not necessary to be aware of the full functionality of the system.

4) Acceptance Testing

Acceptance testing is a type of testing where client/business/customer test the software with real time business scenarios.

The client accepts the software only when all the features and functionalities work as expected. This is the last phase of testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

a) Alpha Testing

Alpha testing is a type of acceptance testing performed by the team in an organization to find as many defects as possible before releasing software to customers.

For example, the pet insurance website is under UAT. UAT team will run real-time scenarios like buying an insurance policy, buying annual membership, changing the address, ownership transfer of the pet in a same way the user uses the real website. The team can use test credit card information to process payment-related scenarios.

b) Beta Testing

Beta Testing is a type of software testing which is carried out by the clients/customers. It is performed in the **Real Environment** before releasing the product to the market for the actual end-users.

Beta Testing is carried out to ensure that there are no major failures in the software or product, and it satisfies the business requirements from an end-user perspective. Beta Testing is successful when the customer accepts the software.

Usually, this testing is typically done by the end-users. This is the final testing done before releasing the application for commercial purposes. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area.

So, the end-user uses the software and shares the feedback with the company. The company then takes necessary action before releasing the software worldwide.

c) Operational acceptance testing (OAT)

Operational acceptance testing of the system is performed by operations or system administration staff in the production environment. The purpose of operational acceptance testing is to make sure that the system administrators can keep the system working properly for the users in a real-time environment.

The focus of the OAT is on the following points:

- Testing of backup and restore.
- Installing, uninstalling, upgrading software.
- The recovery process in case of natural disaster.
- User management.
- Maintenance of the software.

NON-FUNCTIONAL TESTING

There are four main types of functional testing.

1) Security Testing

It is a type of testing performed by a special team. Any hacking method can penetrate the system.

Security Testing is done to check how the software, application, or website is secure from internal and/or external threats. This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes are.

It also checks how software behaves for any hacker's attack & malicious programs and how software is maintained for data security after such a hacker attack.

a) Penetration Testing

Penetration Testing or Pen testing is the type of security testing performed as an authorized cyberattack on the system to find out the weak points of the system in terms of security.

Pen testing is performed by outside contractors, generally known as ethical hackers. That is why it is also known as ethical hacking. Contractors perform different operations like SQL injection, URL manipulation, Privilege Elevation, session expiry, and provide reports to the organization.

Notes: Do not perform the Pen testing on your laptop/computer. Always take written permission to do pen tests.

2) Performance Testing

Performance testing is testing of an application's stability and response time by applying load. The word stability means the ability of the application to withstand in the presence of load. Response time is how quickly an application is available to users. Performance testing is done with the help of tools. Loader.IO, JMeter, LoadRunner, etc. are good tools available in the market.

a) Load testing

Load testing is testing of an application's stability and response time by applying load, which is equal to or less than the designed number of users for an application.

For example, your application handles 100 users at a time with a response time of 3 seconds, then load testing can be done by applying a load of the maximum of 100 or less than 100 users. The goal is to verify that the application is responding within 3 seconds for all the users.

b) Stress Testing

Stress testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

For example, your application handles 1000 users at a time with a response time of 4 seconds, then stress testing can be done by applying a load of more than 1000 users. Test the application with 1100,1200,1300 users and notice the response time. The goal is to verify the stability of an application under stress.

c) Scalability Testing

Scalability testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

For example, your application handles 1000 users at a time with a response time of 2 seconds, then scalability testing can be done by applying a load of more than 1000 users and gradually increasing the number of users to find out where exactly my application is crashing.

Let's say my application is giving response time as follows:

- 1000 users -2 sec
- 1400 users -2 sec
- 4000 users -3 sec
- 5000 users -45 sec
- 5150 users- crash – This is the point that needs to identify in scalability testing.

d) Volume testing (flood testing)

Volume testing is testing an application's stability and response time by transferring a large volume of data to the database. Basically, it tests the capacity of the database to handle the data.

e) Endurance Testing (Soak Testing)

Endurance testing is testing an application's stability and response time by applying load continuously for a longer period to verify that the application is working fine.

For example, car companies soak testing to verify that users can drive cars continuously for hours without any problem.

3) Usability Testing

Usability testing is testing an application from the user's perspective to check the look and feel and user-friendliness.

For example, there is a mobile app for stock trading, and a tester is performing usability testing. Testers can check the scenario like if the mobile app is easy to operate with one hand or not, scroll bar should be vertical, background colour of the app should be black and price of and stock is displayed in red or green colour.

The main idea of usability testing of this kind of app is that as soon as the user opens the app, the user should get a glance at the market.

a) Exploratory testing

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application. Test charters are used to guide the exploratory testing.

b) Cross browser testing

Cross browser testing is testing an application on different browsers, operating systems, mobile devices to see look and feel and performance.

Why do we need cross-browser testing? The answer is different users use different operating systems, different browsers, and different mobile devices. The goal of the company is to get a good user experience regardless of those devices.

Browser stack provides all the versions of all the browsers and all mobile devices to test the application. For learning purposes, it is good to take the free trial given by browser stack for a few days.

c) Accessibility Testing

The aim of Accessibility Testing is to determine whether the software or application is accessible for disabled people or not.

Here, disability means deafness, colour blindness, mentally disabled, blind, old age, and other disabled groups. Various checks are performed, such as font size for visually disabled, colour and contrast for colour blindness, etc.

4) Compatibility testing

This is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment.

DIALOGFLOW-CONSOLE

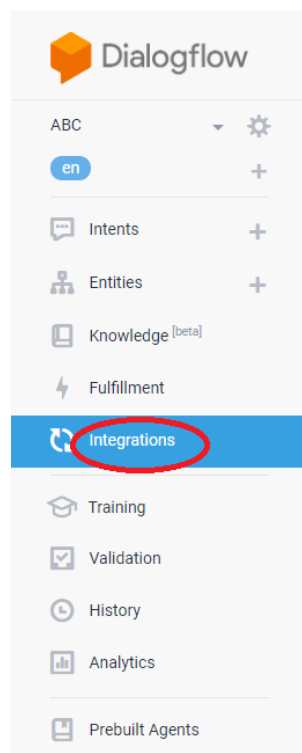
Interaction with an Integration

There are various types of conversation platforms on which the Dialogflow can integrate, such as Facebook, Slack, Google Assistant, etc. To create an agent for any platforms, you can use one of the options from the various integration options. In the Dialogflow, there is a facility of handling direct end-user interactions so that your more focus is to create an agent.

How to Enable the Integration

There are various steps to enable integration:

1. First, go to the Dialogflow Console.
2. Then, select an agent.
3. Next, click on the option named Integration present on the left sidebar menu.



4. Then, enable **Web Demo** integration.

When the web demo enables the dialog, the window offers you the following:

- It provides you the URL to a web page that hosts the integration.
- It provides HTML code so that the agent in your website can be embedded.

- It also provides a link for the agent setting, which is used to customize the webpage aspects.

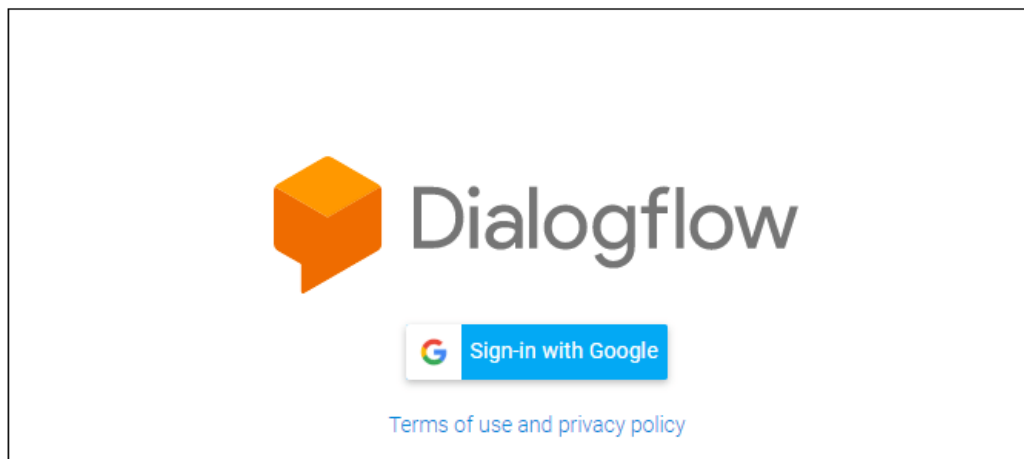
How to Build Healthcare Chatbot for Google Assistant

There are various steps to build resume chatbot for Google assistant:

Google Signup

First, log in to the Dialogflow. We can log in to the Dialogflow by using the link <https://dialogflow.com/>. After login successfully, we have to click on the **Sign up for free**.

Then we have to connect with the Google account. If we want to use a Dialogflow, then it is must that to have a Google account.



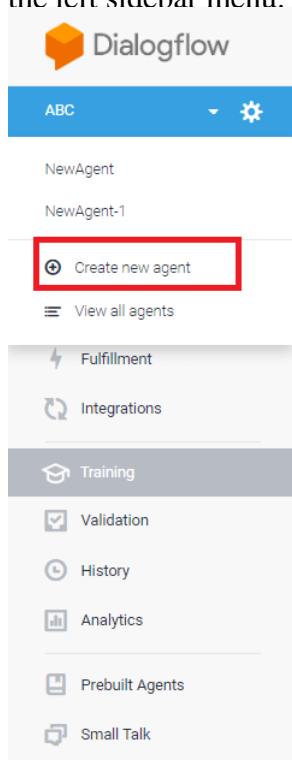
1. Create a Dialogflow Agent

If we want to build an appointment scheduler with Dialogflow then first we have to create a Dialogflow agent:

There are various steps to create a Dialogflow Agent:

- a. First, go to the **Dialogflow console**.
- b. Next, as using the Dialogflow first time, sign in with the email; otherwise, it is not required.
- c. Then accept the terms and condition, and after accepting terms and conditions, we can use the Dialogflow console.

- d. Now, we can create the agent by clicking on the **Create new agent** option, which on the left sidebar menu.



- e. Then, give the name to the agent as "Appointment Scheduler" and click on the 'CREATE' button.

A screenshot of the 'Create new agent' form in Dialogflow. The form has a header section with a text input field containing 'Appointment-Scheduler' (highlighted with a red box) and a blue 'CREATE' button (also highlighted with a red box). Below the header are four configuration sections: 'DEFAULT LANGUAGE' (set to English - en), 'DEFAULT TIME ZONE' (set to (GMT+6:00) Asia/Almaty), 'GOOGLE PROJECT' (set to Create a new Google project), and 'AGENT TYPE' (with a toggle for 'Set as Mega Agent').

In the Dialogflow, as a part of the agent, there are two types of default intents:

Search intents



🔖 Default Fallback Intent

● Default Welcome Intent

1. Default Fallback Intent: - It helps to capture those questions which the bot doesn't understand.

2. Default Welcome Intent: - The default welcome intent is matched whenever the end-user starts a conversation with your agent.

2. Test the Agent

There is a testing panel in the Dialogflow console that is used to test the agent. The testing panel appears on the right side of the Dialogflow console window.

Type "Hi" to test the agent. Then the agent will respond with the default greeting, which is defined in the default welcome intent. It will say, "Greetings! Hello! How can I help you?" we can update the response.

Try it now

Agent

USER SAYS COPY CURL

hi

DEFAULT RESPONSE
Hello! How can I help you?

INTENT
Default Welcome Intent

ACTION
input.welcome

DIAGNOSTIC INFO

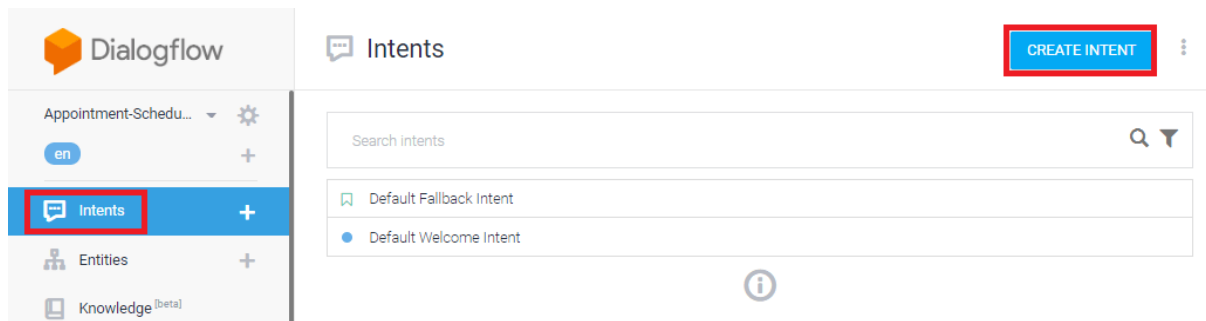
If we type "set an appointment," then the agent will not be able to give the response to this because the agent does not know what to do and so it provides the response, which is defined in the default fallback intent. Because we have not created any intent that catch this specific question.

3. Create Intent

After creating the agent, we have to create the intent.

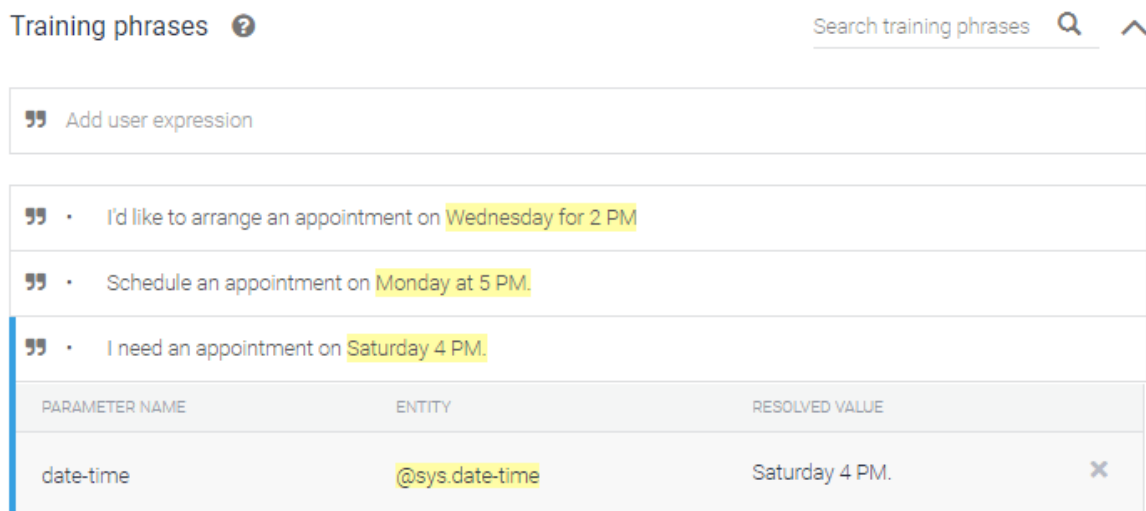
There are various steps to create the intent:

(a) First, click on the **Intent** option which is present on the left side of the Dialogflow window then click on the **CREATE INTENT**



(b) Then, click on the **Training phrases** and enter the various phrases:

- I need an appointment on Saturday at 4 PM.
- Schedule an appointment on Monday at 5 PM.
- I'd like to arrange an appointment on Wednesday for 2 PM.



4. Test your Chatbot

Now our chatbot for appointment scheduler is ready and now we test the chatbot. To test the chatbot, we have to enter the various conversations in the testing panel of the Dialogflow console.

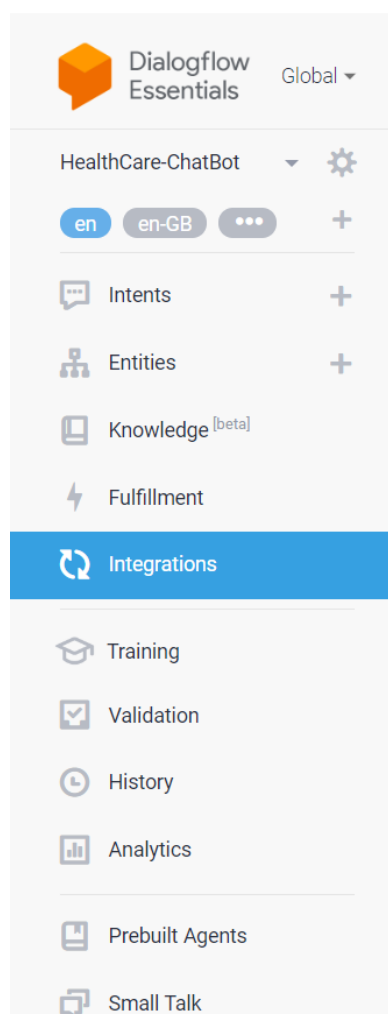
- User: "hi"
- Chatbot: "Greetings! How can I assist?"

5. Enable One-Click Web Integration

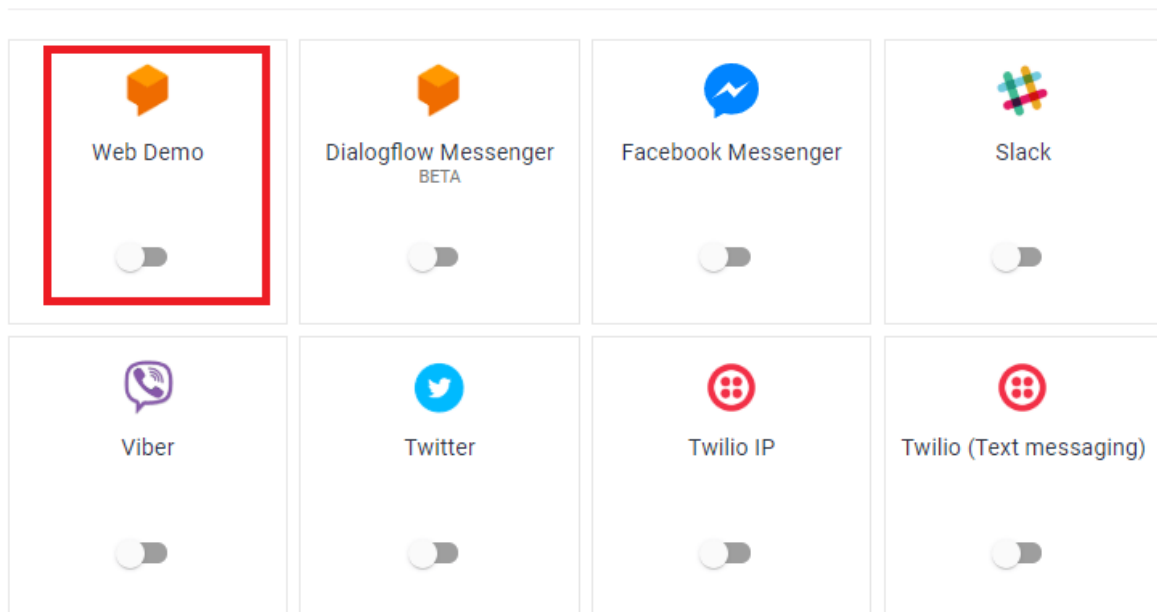
If we want to share the schedule with others, use the option named one-click we integration. In Dialogflow, there are various types of integration available for the chatbot. Look at a sample chatbot web user interface.

There are various steps to using one-click web integration:


1. In the Dialogflow, click on the




2. Then enable the Web Demo.



3. Next to launch the Web Demo, click on the URL.


Dialogflow Messenger

Dialogflow Messenger brings a rich UI for Dialogflow that enables developers to easily add conversational agents to websites. [More in documentation.](#)


 End-user interactions with the Dialogflow Messenger widget may be billed to your GCP account, depending on your Dialogflow edition.

Add this agent to your website by copying the code below

```

<script src="https://www.gstatic.com/dialogflow-console/fast/messenger/bootstrap.js?v=1"></script>
<df-messenger
  intent="WELCOME"
  chat-title="HealthCare-ChatBot"
  agent-id="d7aa38e2-493e-4c03-8a43-c906773fabb0"
  language-code="en"
></df-messenger>

```

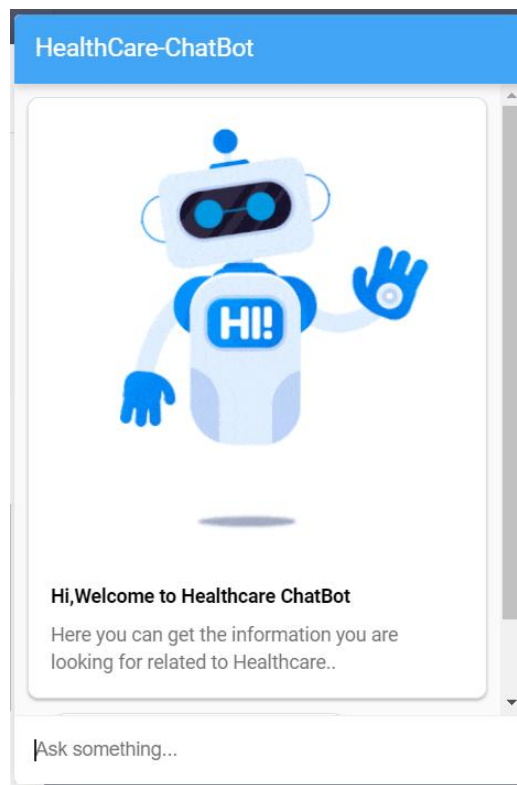
Active environment: Draft ?

CLOSE
DISABLE
TRY IT NOW

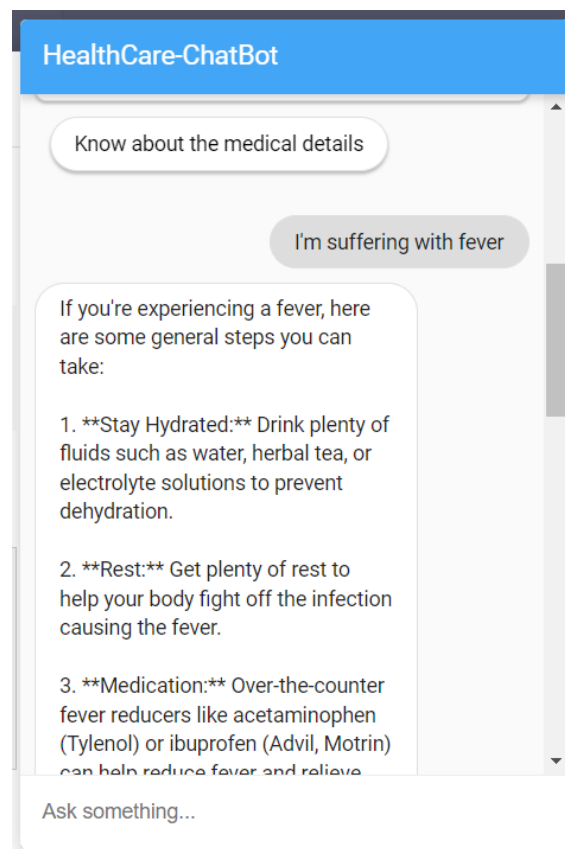
After performing the above steps, we are able to use the chat interface. We can use the chat interface by typing something where it says, 'Ask Something'.

By using the conversation below, we can begin using the chat interface.

1. Enter "Hi" when you enter "Hi" or something then the chatbot will respond you.



2. Then, enter "set an appointment for 6 PM tomorrow" and then the chatbot will respond.



CHAPTER - 8

RESULTS

We can describe the potential output screens of an advanced healthcare chatbot for disease diagnosis and treatment recommendations:

1. Welcome Screen: The welcome screen introduces the chatbot and provides a brief overview of its capabilities and how to get started.

2. Symptom Input Screen: Users can input their symptoms using natural language or select from a list of predefined options. The screen may prompt users to describe the severity, duration, and any associated factors of their symptoms.

3. Diagnostic Results Screen: After analyzing the user's symptoms, the chatbot presents a list of potential diagnoses along with relevant information about each condition. This screen may include descriptions of symptoms, possible causes, and recommended next steps for further evaluation.

4. Treatment Recommendations Screen: For each diagnosis, the chatbot provides evidence-based treatment recommendations tailored to the user's individual needs and medical history. This screen may include information about medications, lifestyle changes, and self-care tips.

5. Additional Information Screen: Users can access additional information about specific diseases, treatments, or medical guidelines. This screen may include links to reputable sources or educational resources for further reading.

6. User Interaction Screen: Throughout the conversation, the chatbot engages users in interactive dialogue to gather additional information, clarify ambiguities, and provide personalized recommendations. This screen displays the ongoing conversation between the user and the chatbot.

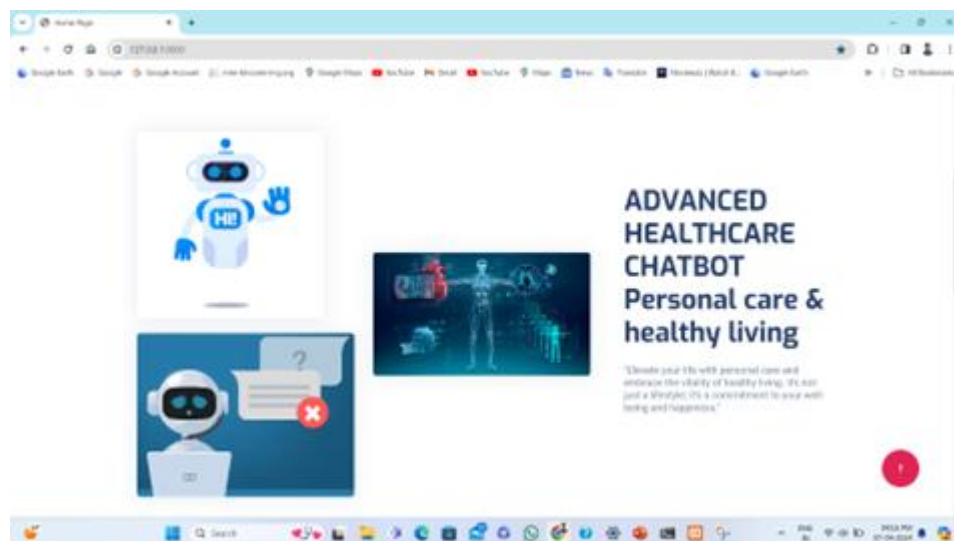
7. Integration with Healthcare Systems Screen: In some cases, the chatbot may display information retrieved from integrated healthcare systems, such as electronic health records (EHRs) or telemedicine platforms. This screen may include appointment reminders, lab results, or medication refills.

8. Privacy and Security Screen: The chatbot may include a privacy and security screen to inform users about how their data is handled and protected. This screen may include information about data encryption, access controls, and compliance with healthcare regulations (e.g., HIPAA).

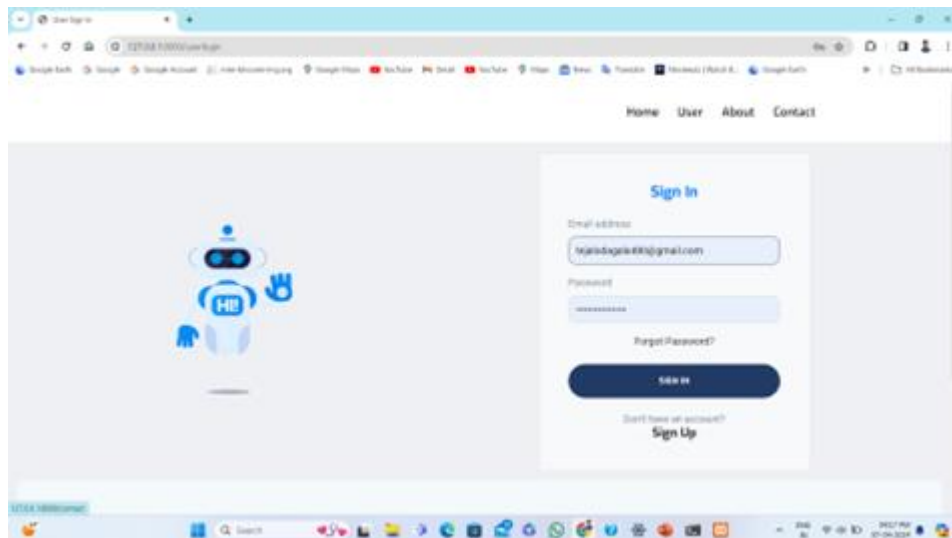
These output screens are designed to provide users with relevant information, guidance, and support throughout the symptom assessment, diagnosis, and treatment recommendation process, ultimately empowering individuals to make informed decisions about their health.



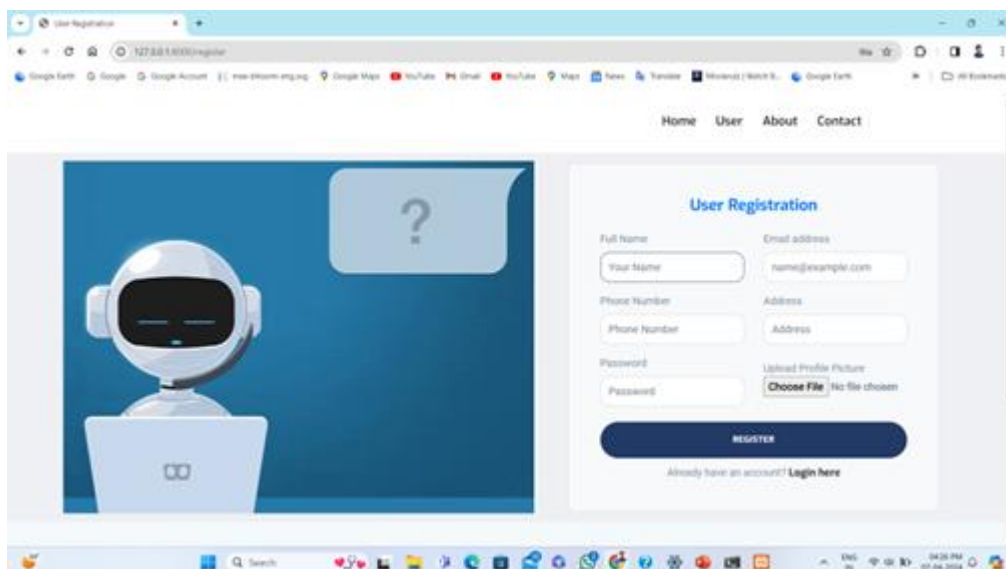
HOME PAGE



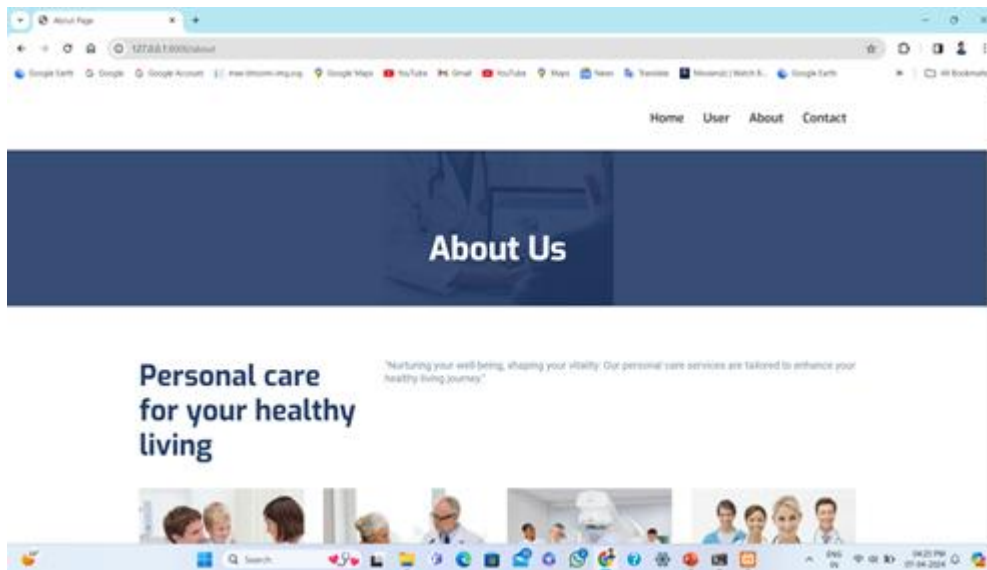
HOME PAGE



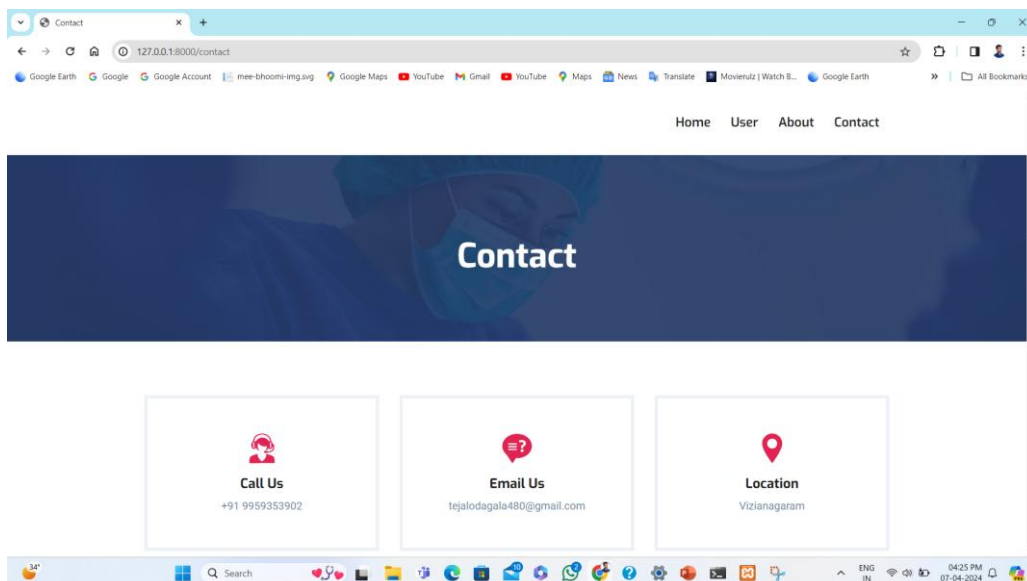
USER SIGN- IN PAGE




USER REGISTRATION PAGE




ABOUT US PAGE



CONTACT PAGE

 Dialogflow Messenger

Dialogflow Messenger brings a rich UI for Dialogflow that enables developers to easily add conversational agents to websites. [More in documentation.](#)

 End-user interactions with the Dialogflow Messenger widget may be billed to your GCP account, depending on your Dialogflow edition.

Add this agent to your website by copying the code below

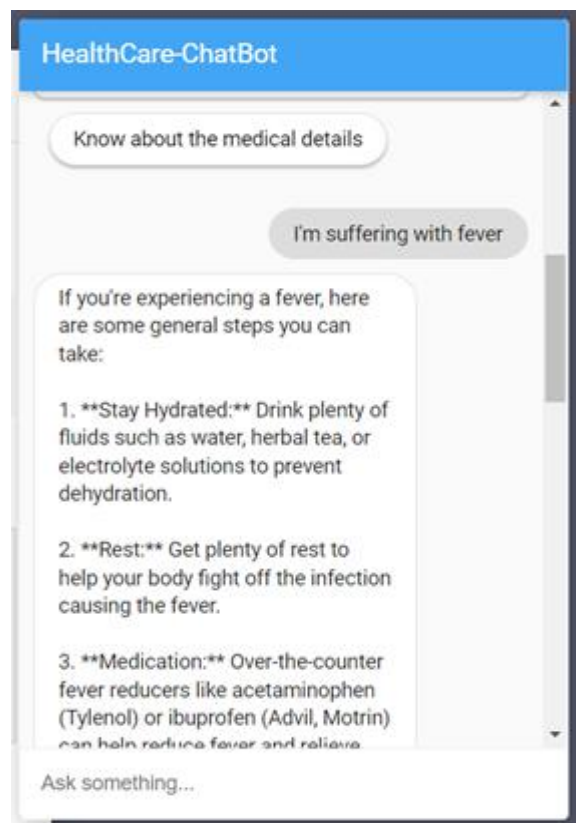
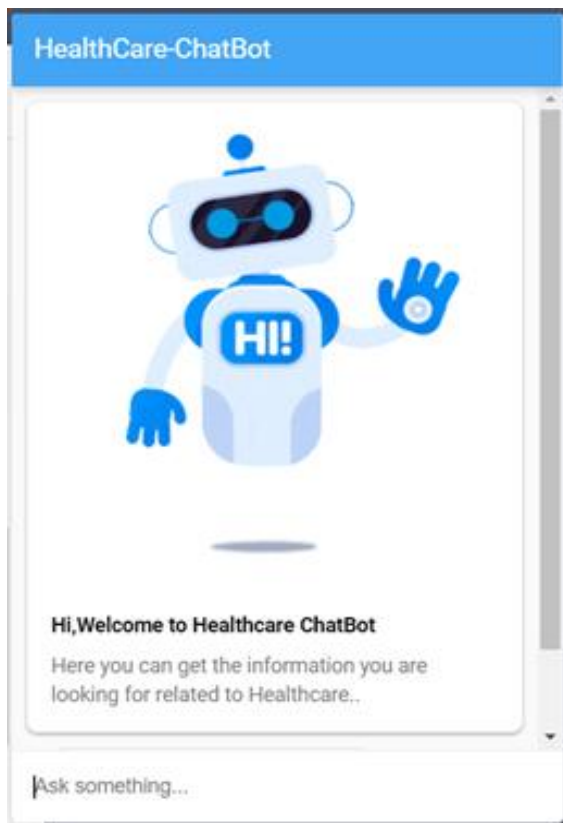
```
<script src="https://www.gstatic.com/dialogflow-console/fast/messenger/bootstrap.js?v=1"></script>
<df-messenger
  intent="WELCOME"
  chat-title="HealthCare-ChatBot"
  agent-id="d7aa38e2-493e-4c03-8a43-c906773fabb0"
  language-code="en"
></df-messenger>
```

Active environment: Draft ⓘ

CLOSEDISABLETRY IT NOW

Dialogflow Messenger

ADVANCED HEALTHCARE CHATBOT FOR DISEASE DIAGNOSIS AND TREATMENT RECOMMENDATION

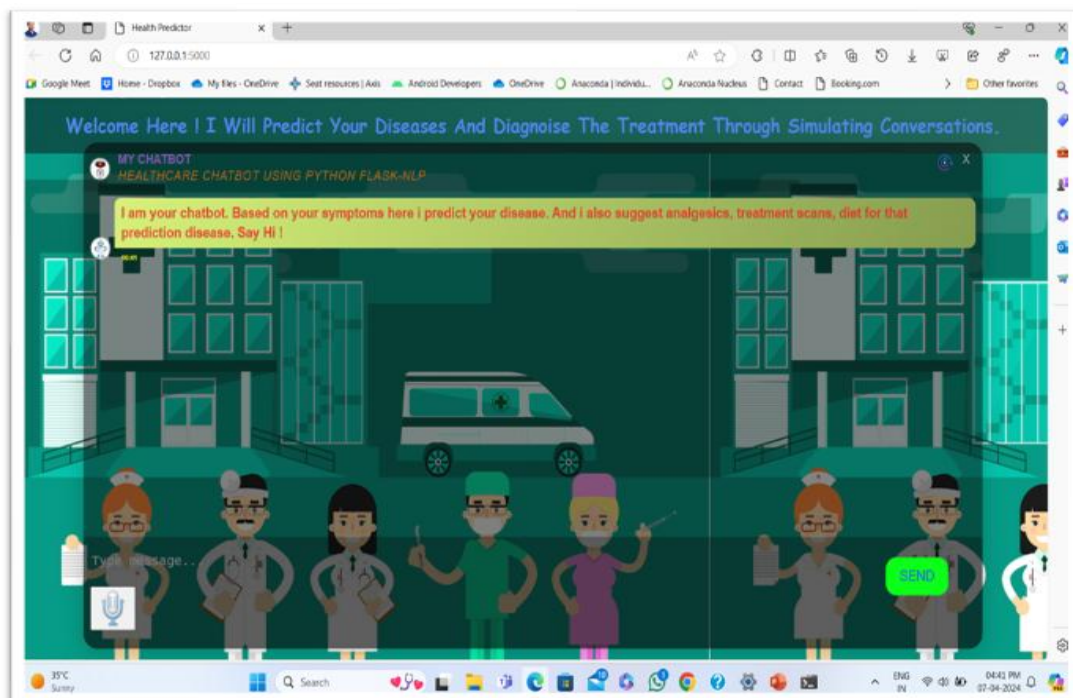


TEXT BASED WEB APPLICATION

Telegram Bot



Text- Voice Based Chatbot



CHAPTER - 9

CONCLUSION

CONCLUSION

In this project, we are implementing an AI Based Healthcare chatbot system using NLP which is easy to use and more secure than the current system it will cure the diseases and helps to maintain proper health in the current system. It will reduce the possibility of diseases. It will provide an accurate information about the health symptoms and medicines to the patients. The government will also keep the track of the medicines supplied to the medicals and hospitals. However, this system has some drawback like network issues in rural areas might cause a delay in delivering an OTP. The implementation of the system is not costly.

CHAPTER - 10

BIBLIOGRAPHY

REFERENCES

- [1] A. Sameera, Abdul-Kader, and J. Woods, "Survey on Chatbot Design Techniques in Speech Conversation Systems," IJACSA, vol. 6 Issue 7, 2015.
- [2] A. Deshpande, A. Shahane, D. Gadre, M. Deshpande, P. M. Joshi, "A survey of various chatbot implementation techniques," Vol. 11, 2017.
- [3] Real World Smart Chatbot for Customer Care using a Software as a Service (SaaS) Architecture "Godson Michael D'silva", Sanket Thakare², Shraddha More Available:<https://www.docme.ru/doc/2207164/ismac.2017.8058261>.
- [4] Divya Madhu, Neeraj Jain C. J, Elmy Sebastain, Shinoy Shaji, Anandhu Ajaya kumar," A Novel Approach for Medical Assistance Using Trained Chatbot",International Conference 2016.
- [5] YerlanJ Saurav Kumar Mishra, Dharendra Bharti, Nidhi Mishra," Dr.Vdoc: A Medical Chatbot that Acts as a Virtual Doctor", Journal of Medical Science and Technology Volume: 6, Issue 3, 2016. Available:<http://medicaljournals.stmjournals.in/index.php/RRJoMST/article/view/30>.
- [6] Pavlidou Meropi, Antonis S. Billis, Nicolas D.Hasanagas, Charalambos Bratsas, Ioannis Antoniou, Panagiotis D. Bamidis, "Conditional Entropy Based Retrieval Model in Patient-Care Conversational Cases",2017 IEEE 30th International conference on Computer-Based Medical System. Available:- <https://ieeexplore.ieee.org/abstract/document/8104260>.
- [7] Abbas Saliimi Lokman, Jasni Mohamad Zain,Fakulti Sistem Komputer, Kejuruteraan Perisian," Designing a Chatbot for Diabetic Patients", ACM Transactions on Management Information Systems (TMIS), Volume 4, Issue 2, August 2018. Available:http://ijircce.com/upload/2018/june/7_P_harmabot.pdf.

- W3 School – (<https://www.w3schools.com/python/>)
- Geek for Geeks – (<https://www.geeksforgeeks.org/python-programming-language/learn-python-tutorial/>)
- Python Official Documentation – (<https://docs.python.org/3/tutorial/>)
- Tutorials Point – (<https://www.tutorialspoint.com/python/index.htm>)
- Real Python – (<https://realpython.com/>)
- Django for Beginners – (<https://djangoforbeginners.com/introduction/>)
- Guru99 – (<https://www.guru99.com/django-tutorial.html>)
- Dialogflow Tutorial – (<https://www.javatpoint.com/dialogflow>)